**Retrospectivas de Sprint S1–S7**

**“CondorShop (E-commerce)”**

**Tabla de contenido**

**Contenido**

[Datos del documento 3](#_heading=h.30j0zll)

[Descripción del proyecto. 4](#_heading=h.1fob9te)

[Descripción de las retrospectivas de los Sprint. 4](#_heading=h.3znysh7)

[Puntos de mejoras. 6](#_heading=)

[Lecciones aprendidas. 6](#_heading=h.tyjcwt)

# Datos del documento

**Histórico de Revisiones**

|  |  |  |  |
| --- | --- | --- | --- |
| Versión | Fecha | Descripción/cambio | autor |
| 0.1 | 20/11/2025 |  | Juan Rodríguez |

**Información del Proyecto**

|  |  |
| --- | --- |
| Organización | Duoc UC. Escuela de Informática y Telecomunicaciones |
| Sección | 004V |
| Proyecto (Nombre) | “CondorShop (E-commerce)” |
| Fecha de Inicio | 20/11/2025 |
| Fecha de Término | 20/11/2025 |
| Patrocinador principal | DuocUC |
| Docente | Marcos Valenzuela |

**Integrantes**

| Rut | Nombre | Correo |
| --- | --- | --- |
| 21.343.914-6 | Benjamin Huenupe | Be.huenupe@duocuc.cl |
| 19.880.451-7 | Juan Rodríguez | Juap.rodriguez@duocuc.cl |

# Descripción del proyecto.

|  |
| --- |
| CondorShop es una tienda en línea desarrollada entre agosto-noviembre 2025 que ofrece catálogo de productos con búsqueda, filtros, carrito persistente y pago integrado mediante Transbank Webpay Plus. Implementada con Django 5.2.8 + DRF 3.16.1 + PostgreSQL 14+ en backend y React 18 + Vite + Tailwind CSS en frontend, registra órdenes y stock con transacciones ACID. Incluye sistema de perfilamiento con dos roles (Clientes/Administradores), Django Admin para gestión de productos y pedidos, autenticación JWT, rate limiting en 7 endpoints críticos, historial de precios, auditoría automática y 6 suites de tests unitarios. El sitio es responsive, maneja carritos para usuarios e invitados, aplica envío gratis sobre CLP 50.000 y permite reintentos de pagos rechazados. |

# Descripción de las retrospectivas de los Sprint.

|  |
| --- |
| **Sprint 1:**   * **Principales Problemas Detectados:** User debía usar email como USERNAME\_FIELD y requirió custom UserManager (IMP-01). Jerarquía de categorías se agregó después en migración 0002 (IMP-02). Inicialmente se usó ID; se migró a slug para SEO (IMP-03). * **Solución Adoptada:** Implementar custom UserManager desde el inicio como USERNAME\_FIELD. Crear migración 0002 para parent\_category y ajustar fixtures en categorías. Usar slugify() para generar slugs limpios y URLs limpias; sistema SEO implementado.   **Sprint 2:**   * **Principales Problemas Detectados:** Lógica de sesión y merge de carritos cuando usuario inicia sesión (IMP-04). Diferencias en cálculos de subtotal y validación en sesión usuario vs invitado (IMP-05). Validación de descuentos mutuamente excluyentes requirió clean() personalizado (IMP-04). Inicialmente se usó token básico; se migró a SimpleJWT para refresh tokens (IMP-05). Configuración SMTP en producción requiere variables de entorno adicionales (IMP-15). * **Solución Adoptada:** Implementar almacenamiento en session con @ratelimit(key='ip', rate='5/m') que retorna tokens access y refresh. Definir lógica de merge; implementar endpoints de sesión con select\_for\_update() y Race conditions. Agregar variables SMTP en settings.py (EMAIL\_BACKEND, EMAIL\_HOST, EMAIL\_PORT=587, EMAIL\_USE\_TLS=True, EMAIL\_HOST\_USER, EMAIL\_HOST\_PASSWORD, DEFAULT\_FROM\_EMAIL).   **Sprint 3:**   * **Principales Problemas Detectados:** Se agregó signals pre\_save y post\_save para detectar cambios de precio en Product (IMP-07). Se agregó código compatibilidad Race conditions con select\_for\_update() dentro de @transaction.atomic (IMP-08). Paginación de catálogo y búsqueda requirió usar useSearchParams (IMP-09). Optimistic updates en carrito: inicialmente se reconstruía sin rollback; se agregó con rollback en error (IMP-10). * **Solución Adoptada:** Implementar signals pre\_save, save() en Product. Registrar eventos clave en logging. Implementar @transaction.atomic con select\_for\_update() y probar reserve\_stock() y release\_stock() simulando compras concurrentes. Implementar useSearchParams para manage page correctamente en catálogo + limpia con botón "Limpiar filtros". Implementar rollback en caso de error; agregar toast en confirmación.   **Sprint 4:**   * **Principales Problemas Detectados:** Procesador externo (Transbank) no está integrado con credenciales en producción (IMP-11). Lógica inicial permite múltiples carritos activos; se resolvió con Cart.get\_or\_create (IMP-12). Límite de 25 caracteres para buy\_order; se ajusta el formato buy\_order (IMP-14). * **Solución Adoptada:** Solicitar credenciales Product Owner / Proveedor Webpay 5; mockear pago mientras tanto. Implementar Cart.get\_or\_create, cart\_id que maneja estado único y unificar lógica de creación/recuperación de carritos. Ajustar formato de buy\_order a 'ORD-{comic\_id}-{timestamp(u)}' para webpay\_service.create\_transaction().   **Sprint 5:**   * **Principales Problemas Detectados:** Credenciales Webpay cargadas en inicio requiere API\_KEY error en database (IMP-13). Límite 19-25 caracteres para buy\_order; formularios buy\_order múltiple (IMP-14). Configuración SMTP en producción requiere variables entorno (IMP-15). * **Solución Adoptada:** Implementar importación lazy para ordenar aplicación sin credenciales en dev. Ajustar formato de buy\_order a 'ORD-{comic\_id}-{timestamp(u)}'. Agregar variables SMTP pendientes en settings.py con validación en correo de confirmación enviado.   **Sprint 6:**   * **Principales Problemas Detectados:** Campos null=True, blank=True temporales para migración 0010; se harán obligatorios después (IMP-16). Modelo Order en Admin requiere campos shipping\_snapshot\_\_customer\_email requiere select\_related (IMP-17). * **Solución Adoptada:** Agregar campos temporales null=True en OrderShippingSnapshot para completar migración; rellenar después obligatorios. Agregar select\_related() en QuerySet Admin para optimizar consultas en pedidos con columnas.   **Sprint 7:**   * **Principales Problemas Detectados:** No se considera requerimiento herramientas del cliente (IMP-18). Falta de documentación final; podría retrasar el cierre oficial del proyecto (IMP-19). * **Solución Adoptada:** Configurar herramientas básicas de profiling y crear checklist de accesibilidad aplicados en vistas críticas. Crear DEPLOYMENT.md con pasos de despliegue, README con instrucciones claras, y documento de evidencias final. |

# Puntos de mejoras.

|  |
| --- |
| * SEO: URLs implementadas pero pendientes meta tags y sitemap.xml. * Exportación: No hay exportación CSV desde Django Admin. * Testing: 6 suites unitarias completadas pero faltan tests E2E. * Backups: Manuales documentados pero sin automatización. * Producción: Pendiente documentación de despliegue y credenciales Webpay reales. * CI/CD: GitHub Actions no implementado. * Seguridad: django-csp marcado como mejora futura. * Observabilidad: Logging implementado pero sin métricas ni alertas. |

# Lecciones aprendidas.

|  |
| --- |
| * **Diseño Inicial vs Refactorización:** Definir custom UserManager con email como USERNAME\_FIELD, jerarquías de categorías con parent\_category y slugs para SEO desde el inicio evita migraciones costosas posteriores (IMP-01, IMP-02, IMP-03). * **Autenticación y Sesiones:** Migrar de tokens básicos a SimpleJWT con refresh tokens y diseñar explícitamente el merge de carritos anónimo-autenticado requiere planificación temprana; @ratelimit protege endpoints críticos (IMP-04, IMP-05). * **Consistencia de Datos:** Implementar @transaction.atomic() con select\_for\_update() es esencial para evitar race conditions en operaciones de stock concurrentes; signals pre\_save/post\_save permiten auditoría desacoplada (IMP-07, IMP-08). * **Frontend Reactivo:** useSearchParams en React Router v6 maneja paginación correctamente; optimistic updates mejoran UX pero requieren rollback robusto en errores (IMP-09, IMP-10). * **Integraciones Externas:** Transbank requiere solicitar credenciales productivas tempranamente; Cart.get\_or\_create() evita múltiples carritos activos; formato buy\_order debe ajustarse a límites de 19-25 caracteres ('ORD-{id}-{timestamp}') (IMP-11, IMP-12, IMP-14). * **Configuración por Entornos:** django-environ y lazy imports permiten arrancar sin credenciales en dev; variables SMTP deben documentarse en .env.example desde el inicio (IMP-13, IMP-15). * **Optimización de Queries:** select\_related() en Django Admin reduce consultas N+1; campos null=True temporales en migraciones facilitan transiciones pero deben hacerse obligatorios después (IMP-16, IMP-17). * **Documentación Final:** Herramientas de profiling, checklist de accesibilidad, DEPLOYMENT.md y README claros evitan retrasos en cierre del proyecto (IMP-18, IMP-19). |