**Documento Impact Mapping**

**“CondorShop (E-commerce)”**

**Tabla de contenido**

**Contenido**

[Datos del documento 3](#_heading=h.1fob9te)

[Impact Mapping 4](#_heading=h.tyjcwt)

**Datos del documento**

Histórico de Revisiones

|  |  |  |  |
| --- | --- | --- | --- |
| Versión | Fecha | Descripción/cambio | Autor |
| 0.1 | 11/08/2025 |  | Benjamin Huenupe |
| 0.2 | 12/08/2025 |  | Benjamin Huenupe |
| 0.3 | 18/08/2025 |  | Benjamin Huenupe |
| 1.0 | 25/08/2025 |  | Benjamin Huenupe |
| 1.1 | 01/09/2025 |  | Benjamin Huenupe |

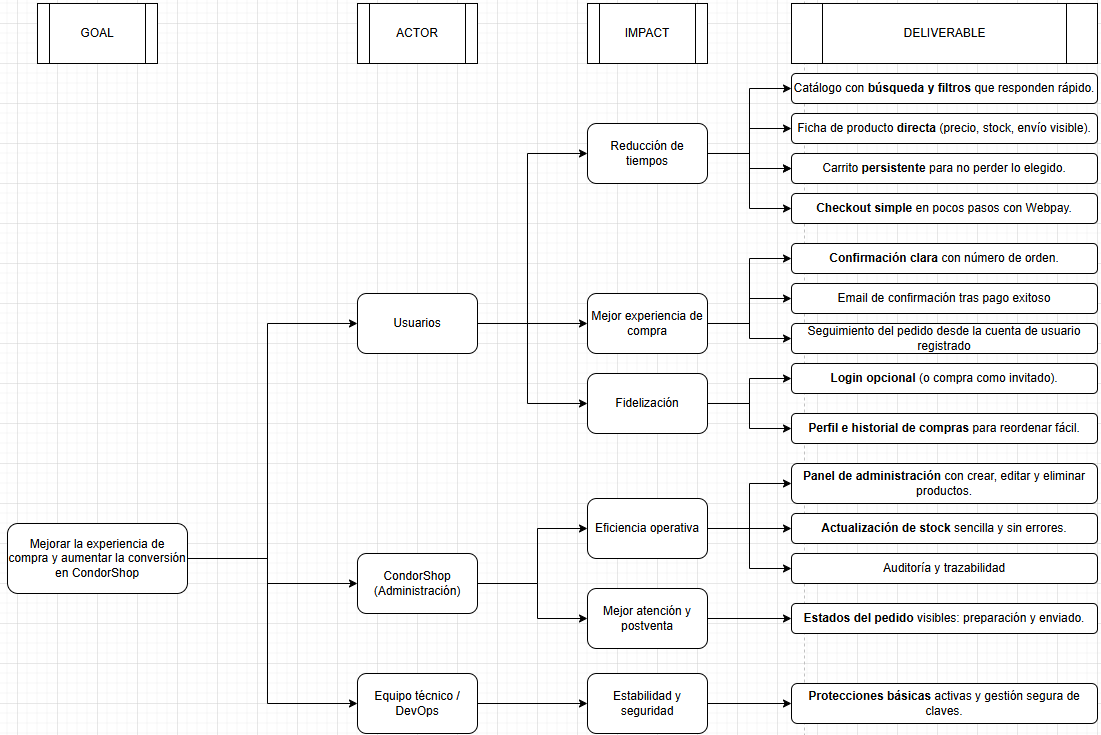
Información del Proyecto

|  |  |
| --- | --- |
| Organización | Duoc UC. Escuela de Informática y Telecomunicaciones |
| Sección | 004V |
| Proyecto (Nombre) | “CondorShop (E-commerce)” |
| Fecha de Inicio | 11/08/2025 |
| Fecha de Término | 01/09/2025 |
| 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 |

**Impact Mapping**



**Objetivo:**

Aumentar la conversión de compra y mejorar la experiencia del cliente, ofreciendo un flujo simple, rápido y confiable: descubrir productos → agregar al carrito → pagar con Webpay → recibir confirmación; con login opcional (o checkout como invitado), stock consistente, y panel de administración claro.

**Problemas que solucionar**

* En tiendas pequeñas suele haber fricción para encontrar productos, carritos que se pierden, pagos que fallan o confirmaciones poco claras. Eso baja la conversión y la confianza.
* Falta de consistencia en el stock durante compras concurrentes.
* Ausencia de herramientas administrativas eficientes para gestionar catálogo y pedidos.

**Necesidad a satisfacer**

Una experiencia fluida y agradable: catálogo optimizado, carrito persistente sin pérdida de datos, checkout sin trabas (invitado o con cuenta), pago seguro con Transbank Webpay Plus, y gestión transparente de pedidos con estados visibles.

**Actores**

* Clientes nuevos (no registrados): llegan, descubren productos y deciden si comprar o no.
* Clientes recurrentes (registrados): vuelven, se loguean y repiten compra con datos guardados.
* Administrador de tienda: mantiene catálogo/stock y gestiona pedidos en tiempo real.
* Equipo técnico / DevOps: asegura seguridad, integridad de datos y cumplimiento de estándares.

**Impactos esperados:**

**A) CLIENTES NUEVOS**

**Impacto A1 — Encuentran rápido lo que buscan**

KPI implementado:

* Paginación de catálogo con 20 productos por página
* Queries optimizadas con select\_related('category') y prefetch\_related('images')
* 8 índices en PostgreSQL: slug, price, stock\_qty, created\_at, category, name, active
* Búsqueda por nombre/descripción con filtros por categoría

Deliverables:

* Catálogo con búsqueda y filtros funcionales (/api/products/)
* Ficha de producto con slug único (/api/products/{slug}/)
* 8 índices en modelo Product para consultas rápidas
* Paginación configurada (PAGE\_SIZE=20)

**Impacto A2 — Compran sin fricción aunque no tengan cuenta**

KPI implementado:

* Checkout funcional para invitados sin registro obligatorio
* Carrito persistente en BD con session\_token (no expira automáticamente)
* Formulario de checkout con 6 campos obligatorios (email, nombre, apellido, calle, ciudad, región)

Deliverables:

* Modelo Cart con soporte para session\_token (invitados) y user (registrados)
* Endpoint /api/orders/create acepta órdenes sin autenticación
* OrderShippingSnapshot guarda datos del cliente sin requerir cuenta
* Rate limiting: 10 órdenes/hora por usuario para prevenir abuso
* Email de confirmación activo con Gmail SMTP tras pago exitoso

**B) CLIENTES RECURRENTES**

**Impacto B1 — Se loguean sin freno y usan datos guardados**

KPI implementado:

* Modelo Address para guardar direcciones con campo is\_default
* Pre-llenado automático en checkout si usuario está autenticado
* Endpoint /api/checkout/mode retorna direcciones guardadas del usuario

Deliverables:

* Modelo Address con campos: label, street, number, apartment, city, region, postal\_code
* Relación user.addresses (OneToMany)
* Frontend pre-llena datos desde Address
* Rate limiting en login: 5 intentos/minuto por IP

**Impacto B2 — Recompran y consultan su historial**

KPI implementado:

* Endpoint funcional para historial de pedidos (/api/orders/)
* Visualización completa de: ID pedido, estado, monto total, fecha, items con snapshots de precios
* Historial de cambios de estado con timestamps en OrderStatusHistory

Deliverables:

* Endpoint /api/orders/ lista pedidos del usuario autenticado
* Endpoint /api/orders/{id}/ muestra detalle completo de pedido
* Modelo OrderStatusHistory registra cambios automáticamente
* 7 estados de pedido: PENDING, PAID, FAILED, CANCELLED, PREPARING, SHIPPED, DELIVERED

**C) ADMINISTRADOR DE TIENDA**

**Impacto C1 — Mantiene catálogo/stock sin planillas ni demoras**

KPI implementado:

* CRUD completo de productos en Django Admin
* Gestión de stock transaccional con select\_for\_update() en reserve\_stock(), release\_stock(), confirm\_sale()
* Historial de precios automático (ProductPriceHistory) con signals

Deliverables:

* Django Admin configurado con filtros por: categoría, estado activo, fecha
* ProductAdmin con inline para imágenes y historial de precios
* Métodos transaccionales: @transaction.atomic() en creación de órdenes
* InventoryMovement registra movimientos de stock con tipos: SALE, RETURN, ADJUSTMENT, RESERVATION

**Impacto C2 — Gestiona pedidos con claridad**

KPI implementado:

* Visualización completa de pedidos en Django Admin con inline de items
* Información de pago Webpay: buy\_order, token, monto, últimos 4 dígitos de tarjeta
* Cambio manual de estados con registro automático en OrderStatusHistory

Deliverables:

* OrderAdmin con filtros por: estado, fecha, método de pago
* Inline de OrderItem con snapshots de precios (preserva precio histórico)
* PaymentTransaction relacionado visible con datos de Webpay
* 7 estados de pedido gestionables desde el admin
* Notificación por email al cliente cuando se confirma el pago (send\_order\_confirmation\_email activa)

**Impacto C3 — Auditoría y trazabilidad de operaciones**

KPI implementado:

* Sistema de auditoría automática con AuditMiddleware activo
* Registro de acciones en tablas críticas: users, products, orders
* Modelo AuditLog con: usuario, acción, tabla, record\_id, valores anteriores/nuevos, IP, timestamp

Deliverables:

* AuditMiddleware activo en MIDDLEWARE (línea 70 settings.py)
* AuditLog visualizable en Django Admin con filtros
* Índices en: user, table\_name, action, created\_at
* Acciones registradas: CREATE, UPDATE, DELETE, PASSWORD\_RESET\_REQUEST

**D) EQUIPO TÉCNICO / DEVOPS**

**Impacto D1 — Estabilidad y seguridad operativa**

KPI implementado:

* Configuración de seguridad para producción activa cuando DEBUG=False:
  + SECURE\_SSL\_REDIRECT=True
  + SESSION\_COOKIE\_SECURE=True
  + CSRF\_COOKIE\_SECURE=True
  + SECURE\_HSTS\_SECONDS=31536000 (1 año)
  + X\_FRAME\_OPTIONS='DENY'
* Rate limiting activo en 6 endpoints críticos
* django-environ para manejo seguro de secretos (.env)
* Logs estructurados en consola y archivo (logs/django.log) con nivel INFO

Deliverables:

* 6 endpoints con rate limiting: login (5/min), registro (5/min), checkout (10/h), shipping-quote (20/min), password-reset (3/h), password-reset-confirm (5/h)
* django-cors-headers configurado con CORS\_ALLOWED\_ORIGINS
* Variables sensibles en .env: SECRET\_KEY, DATABASE\_URL, WEBPAY\_API\_KEY, WEBPAY\_COMMERCE\_CODE
* Logging con StreamHandler (consola) y FileHandler (archivo logs/django.log)
* Protección CSRF/XSS por defecto en Django

**Impacto D2 — Calidad de código y testing**

KPI implementado:

* 6 suites de tests unitarios con pytest
* Cobertura de flujos críticos: checkout, autenticación, serialización, historial de precios

Deliverables:

* test\_checkout.py: 8 casos (órdenes autenticadas/invitados, cotización envío, snapshots)
* test\_serializers.py: validación de datos de productos
* test\_price\_history.py: tracking automático de cambios de precio
* test\_auth.py: registro, login, recuperación de contraseña
* test\_api\_contract.py: smoke tests de endpoints principales
* test\_utils.py: funciones auxiliares del sistema