Aclaración: aqui y en el SQL no se estan tomando en cuenta todavia lo que seria generado por los programas que usemos (como DJANGO o Three.js), solo lo que seria especifico y necesario para el proyecto independientemente del sistema, lenguajes o tecnologias aplicadas

Fecha de la Versión:

Módulo	Subprocesos	Nro.	Descripción	Nro.	Descripción	Nro.	Descripción
Gestión de Usuarios	Registro	**RF-001**	El sistema debe permitir el registro de nuevos usuarios mediante un formulario con campos: nombre completo, correo electrónico, contraseña.	**RN-001**	La contraseña debe tener mínimo 8 caracteres, incluir al menos una mayúscula, un número y un carácter especial.	**RI-001**	Entidad Usuario: ID (PK, autoincremental), usuario (varchar 100), correo (varchar 100, único), contrasena (varchar 255), estado (enum: Activo, Inactivo, Bloqueado), rol (enum: Administrador, Revisor, Usuario), fecha_registro (datetime), fecha_ultima_sesion (datetime), email_verificado (boolean).
		RF-002	El sistema debe validar las credenciales del usuario (correo y contraseña) para permitir el acceso al sistema.	**RN-002**	El correo electrónico debe ser único en el sistema, no se permiten duplicados.	**RI-002**	Entidad Sesión: ID (PK), usuario_id (FK), token_sesion (varchar 255), fecha_inicio (datetime), fecha_expiracion (datetime).
		RF-003	El sistema debe enviar un correo de verificación con enlace único al usuario registrado.	**RN-003**	El token de verificación debe ser único y tener una validez de 24 horas. Un usuario no verificado no puede acceder a funcionalidades principales. Se permite máximo 3 reenvíos de correo por hora	**RI-003**	Entidad Token_Verificacion: ID (PK, autoincremental) usuario_id (FK → Usuario.ID) token (varchar 255, UNIQUE, NOT NULL) tipo (enum: Verificacion_Email, Recuperacion_Password, Cambio_Email) fecha_creacion (datetime, NOT NULL) fecha_expiracion (datetime, NOT NULL)
		RF-004	El sistema debe permitir a los usuarios actualizar su perfil (usuario, contraseña, correo)	**RN-004**	Para cambiar correo electrónico o contraseña se requiere ingresar la contraseña actual. El cambio de email requiere verificación del nuevo correo.	**RI-004**	Entidad Cambio_Email: ID (PK, autoincremental) usuario_id (FK → Usuario.ID) email_anterior (varchar 100) email_nuevo (varchar 100) token_verificacion (varchar 255) fecha_solicitud (datetime) verificado (boolean, DEFAULT false) fecha_verificacion (datetime)
Administración de Usuarios		**RF-005**	El sistema debe permitir al administrador visualizar un listado completo de usuarios con filtros por: estado, rol, fecha de registro, email verificado.	**RN-005**	Solo usuarios con rol "Administrador" pueden acceder al módulo de administración de usuarios.	**RI-005**	Vista_Usuarios_Admin: ID, usuario, correo, estado, rol, fecha_registro, email_verificado, fecha_ultima_sesion (datetime)
		RF-006	El sistema debe permitir al administrador buscar usuarios por nombre, correo electrónico o ID.	**RN-006**	Debe existir al menos un usuario con rol "Administrador" activo en el sistema en todo momento.	**RI-006**	Índices para búsqueda: idx_nombre_completo (nombre), idx_usuario_correo (correo), idx_usuario_id (ID)"
		RF-007	El sistema debe permitir al administrador crear nuevos usuarios manualmente, asignando: nombre, correo, rol y estado inicial.	**RN-007**	Debe existir al menos un usuario con rol "Administrador" activo en el sistema en todo momento.	**RI-007**	Mismo que RI-001 (Entidad Usuario) Campos requeridos: nombre_completo, correo, rol, estado
		RF-008	El sistema debe permitir al administrador editar la información de cualquier usuario: nombre, correo, rol y estado.	**RN-008**	Todas las acciones administrativas deben quedar registradas en el log de auditoría con: usuario que ejecutó la acción, fecha/hora, acción realizada y datos modificados.	**RI-008**	Entidad Log_Auditoria: ID (PK), usuario_admin_id (FK), usuario_afectado_id (FK), accion (varchar 255), datos_anteriores (json), datos_nuevos (json), fecha_accion (datetime)"
		RF-009	El sistema debe permitir al administrador activar, desactivar o bloquear cuentas de usuario con opción de agregar un motivo.	**RN-009**	Al desactivar o bloquear una cuenta, todas las sesiones activas del usuario deben cerrarse inmediatamente	**RI-009**	Entidad Estado_Usuario: ID (PK), usuario_id (FK), estado_anterior (enum), estado_nuevo (enum), motivo (text), fecha_cambio (datetime), admin_id (FK)
		RF-010	El sistema debe permitir al administrador eliminar usuarios de forma lógica (soft delete) manteniendo su historial.	**RN-010**	Al resetear una contraseña, se debe generar una contraseña temporal aleatoria que cumpla RN-001 y enviarla por correo.	**RI-010**	Campo adicional en Usuario (RI-001): eliminado (boolean, default false), fecha_eliminacion (datetime), admin_eliminador_id (FK → Usuario.ID)

		RF-011	El sistema debe permitir al administrador desbloquear cuentas bloqueadas por intentos fallidos de inicio de sesión.	RN-011	La eliminación de usuarios debe ser lógica (soft delete), marcando un campo "eliminado" pero conservando todos los datos históricos.	RI-011	Actualización en Usuario (RI-001): intentos_fallidos (int, default 0), fecha_bloqueo (datetime), fecha_desbloqueo (datetime), admin_desbloqueador_id (FK → Usuario.ID)
Gestión de Productos Modulo de Admin	Crear Producto	RF-012	El sistema debe permitir al administrador crear un nuevo producto con nombre, descripción, precio base e imagen principal.	RN-012	Nombre único y requerido (máx 100 caracteres); descripción máxima 500 caracteres; precio_base > 0 con 2 decimales; al menos una imagen principal y una variante para completar creación.	RI-012	Entidad Producto: ID (PK, autoincremental), nombre (varchar 100, UNIQUE, NOT NULL), descripcion (text, máx 500), precio_base (decimal 10,2, NOT NULL, CHECK > 0), fecha_creacion (timestamp DEFAULT CURRENT_TIMESTAMP), activo (boolean DEFAULT TRUE), aprobado (boolean DEFAULT FALSE), usuario_id_creador (FK → Usuario.ID). Índice: idx_producto_nombre (nombre)
		RF-013	El sistema debe permitir al administrador subir imagen principal del producto con validación de formato.	RN-013	Formatos JPG/PNG únicamente; tamaño máximo 2MB; resolución mínima 400x400px; imagen principal obligatoria; almacenar alt_text para accesibilidad.	RI-013	Entidad Imagen_Producto: ID (PK, autoincremental), producto_id (FK → Producto.ID, NOT NULL), ruta_imagen (varchar 500, NOT NULL), alt_text (varchar 200), principal (boolean DEFAULT FALSE), orden (int DEFAULT 0), fecha_carga (timestamp DEFAULT CURRENT_TIMESTAMP). Índice: idx_imagen_principal (producto_id, principal)
		RF-014	El sistema debe permitir al administrador subir imágenes adicionales para galería del producto.	RN-014	Máximo 5 imágenes por producto; mismas restricciones formato/tamaño que principal (JPG/PNG <2MB); reordenables; al menos 1 imagen principal requerida; marcar cual es principal.	RI-014	Tabla Imagen_Producto con campo orden (int) para ordenar; CONSTRAINT UNIQUE(producto_id, principal) solo una imagen principal; CHECK al menos 1 imagen presente por producto.
		RF-015	El sistema debe permitir al administrador definir combinaciones de talla y color como variantes del producto.	RN-015	Tallas disponibles: S, M, L, XL; colores en formato HEX (#RRGGBB); stock inicial ≥0 por variante; máximo 4 tallas y 10 colores por producto; cada combinación talla+color única.	RI-015	Entidad Variante: ID (PK, autoincremental), producto_id (FK → Producto.ID, NOT NULL), talla (enum 'S','M','L','XL', NOT NULL), color_hex (varchar 7, NOT NULL, formato #RRGGBB), color_nombre (varchar 50), stock (int DEFAULT 0, CHECK ≥0), precio_variante (decimal 10,2 NULL, CHECK > 0 OR NULL), fecha_creacion (timestamp). CONSTRAINT: UNIQUE (producto_id, talla, color_hex). Índices: idx_variante_talla
		RF-016	El sistema debe permitir al administrador definir precio específico por variante diferente al precio base.	RN-016	Precio variante es opcional (NULL); si NULL, usa precio_base del producto; si definido, CHECK > 0 con máximo 2 decimales; validar coherencia (no menor a 50% del precio_base, mostrar advertencia).	RI-016	Tabla Variante: campo precio_variante (decimal 10,2 NULL, CHECK precio_variante > 0 OR precio_variante IS NULL). Validación en aplicación: si precio_variante < (precio_base * 0.5), mostrar advertencia visual al admin.
		RF-017	El sistema debe validar que el producto tenga configuración mínima requerida antes de completar la creación.	RN-017	Requerir: nombre completo, descripción, imagen principal, al menos 1 variante (talla+color+stock>0), precio_base >0; mostrar checklist visual de requisitos cumplidos.	RI-017	Checklist visual en frontend: ✓ Nombre (sí/no), ✓ Descripción (sí/no), ✓ Imagen Principal (sí/no), ✓ Variante (sí/no), ✓ Stock (sí/no). Deshabilitar botón "Crear Producto" hasta completar todos.
	Buscar/Filtrar Producto	RF-018	El sistema debe permitir al administrador buscar productos por nombre con búsqueda parcial.	RN-018	Búsqueda parcial, insensible a mayúsculas (LIKE '%query%'); resultados paginados máximo 20 por página; ordenar por fecha creación DESC por defecto; mostrar fecha en resultado.	RI-018	SQL: SELECT p.ID, p.nombre, p.precio_base, p.fecha_creacion, COUNT(v.ID) as num_variantes FROM Producto p LEFT JOIN Variante v ON p.ID=v.producto_id WHERE LOWER(p.nombre) LIKE CONCAT('%', LOWER (?), '%') AND p.activo=true GROUP BY p.ID ORDER BY p.fecha_creacion DESC LIMIT 20 OFFSET ? Índice: idx_producto_nombre (nombre)
		RF-019	El sistema debe permitir al administrador filtrar productos por estado (activo/inactivo, aprobado/no aprobado).	RN-019	Filtros combinables: activo (true/false), aprobado (true/false) con lógica AND; mostrar conteos de productos en cada opción; mostrar solo productos activos por defecto.	RI-019	Índices: idx_activo (activo), idx_aprobado (aprobado). SQL: SELECT p.* FROM Producto p WHERE p.activo=? AND p.aprobado=? ORDER BY p.fecha_creacion DESC LIMIT 20 OFFSET ?
		RF-020	El sistema debe permitir al administrador filtrar productos por rango de precio (precio_base).	RN-020	Rango de precio_base entre min y max; validar min < max; mostrar precios encontrados; si >20 resultados aplicar paginación; ordenar por precio ASC.	RI-020	SQL: SELECT p.* FROM Producto p WHERE p.precio_base BETWEEN ? AND ? AND p.activo=true ORDER BY p.precio_base ASC LIMIT 20 OFFSET ? Índice: idx_producto_precio (precio_base)
		RF-021	El sistema debe permitir al administrador ordenar resultados de búsqueda por diferentes criterios.	RN-021	Opciones de ordenamiento: Nombre (A-Z), Fecha (nuevo/antiguo), Precio (menor/mayor), Stock (menor/mayor); default = Fecha DESC; seleccionable por dropdown.	RI-021	Lógica de orden en aplicación: ORDER BY campo ASC/DESC; campos disponibles: nombre, fecha_creacion, precio_base, stock_total (SUM de variantes).
	Editar Producto	RF-022	El sistema debe permitir editar nombre, descripción y precio_base de un producto existente.	RN-022	Confirmar cambios antes de guardar; nombre mantiene UNIQUE constraint; no editar nombre si producto en pedidos activos; precio_base > 0; mostrar cambios resaltados en interfaz.	RI-022	Tabla Auditoria: ID (PK), usuario_id (FK), accion (varchar, e.g. 'Edicion'), tabla_afectada (varchar, 'Producto'), registro_id (int), campo_modificado (varchar), valor_anterior (text), valor_nuevo (text), fecha (timestamp).
		RF-023	El sistema debe permitir editar stock de variantes específicas.	RN-023	Stock ≥0; notificar si cambia a 0 (stock agotado); confirmar cambios antes de guardar; advertencia si producto está en carrito de clientes activos; registrar cambio en Auditoria.	RI-023	Tabla Variante: actualizar campo stock (int, CHECK ≥0). Tabla Notificacion: ID (PK), usuario_id (FK), tipo (varchar, e.g. 'StockCero'), producto_id (FK), fecha (timestamp).
		RF-024	El sistema debe permitir editar precio de variante específica (precio_variante).	RN-024	Precio variante >0 o NULL (opcional); validar coherencia (no <50% del precio_base); confirmar cambios; mostrar diferencia visual vs precio_base.	RI-024	Tabla Variante: actualizar precio_variante (decimal 10,2 NULL, CHECK > 0 OR NULL). Validación cliente: if (precio_variante < precio_base * 0.5) { mostrarAdvertencia("Precio muy bajo") }
		RF-025	El sistema debe permitir cambiar la imagen principal del producto.	RN-025	Validar JPG/PNG <2MB, resolución ≥400x400px; actualizar marca principal; guardar versión anterior; mostrar preview antes de confirmar cambio.	RI-025	Tabla Imagen_Producto: UPDATE principal=true para nueva imagen, principal=false para imagen anterior; versioning con fecha_carga para histórico.
		RF-026	El sistema debe permitir agregar o eliminar imágenes de la galería.	RN-026	Máximo 5 imágenes total por producto; reordenar con drag-drop; eliminar requiere confirmación; validar formatos JPG/PNG únicamente.	RI-026	Tabla Imagen_Producto: DELETE si no es principal; UPDATE orden (int) para reordenar; mostrar preview en tiempo real de galería actual.

		RF-027	El sistema debe permitir agregar o eliminar variantes (combinaciones talla+color).	RN-027	No permitir eliminar variante si está en pedidos activos; confirmar cambio; máximo 4 tallas + 10 colores por producto; mostrar matriz visual talla x color.	RI-027	Tabla Variante: DELETE solo si no existen FK en Pedido_Item; INSERT/UPDATE con validaciones. Validación: SELECT COUNT(*) FROM Pedido_Item WHERE variante_id=?
		RF-028	El sistema debe mostrar advertencia si producto está en uso (carrito/pedidos activos).	RN-028	Buscar FK referencias en Pedido_Item y Carrito_Item; mostrar cantidad de clientes afectados; permitir edición con confirmación explícita del admin.	RI-028	SQL: SELECT COUNT(*) FROM Pedido_Item pi WHERE pi.variante_id IN (SELECT ID FROM Variante WHERE producto_id=?); SELECT COUNT(*) FROM Carrito_Item ...
	Eliminar Producto	RF-029	El sistema debe permitir eliminar lógicamente un producto (marcarlo como inactivo).	RN-029	Marcar activo=false (no eliminar físicamente); no permitir si variantes con stock >0 O en pedidos activos; confirmar acción; registrar en Auditoria.	RI-029	Tabla Producto: UPDATE activo=false. Validación ANTES: SELECT SUM(stock) FROM Variante WHERE producto_id=? AND stock>0; SELECT COUNT(*) FROM Pedido_Item WHERE variante_id IN (SELECT ID FROM Variante WHERE producto_id=?); Tabla Auditoria con accion='Desactivacion'.
		RF-030	El sistema debe mostrar modal de confirmación con validaciones previas antes de eliminar.	RN-030	Modal debe mostrar: nombre producto, variantes activas, pedidos en proceso; advertencia de irreversibilidad (lógica); permitir revertir (reactivar) después.	RI-030	Modal UI con detalles del producto a desactivar; mostrar FK dependencias; botones: Desactivar / Cancelar; post-desactivación mostrar opción "Reactivar".
	Aprobar/Desaprobar	RF-031	El sistema debe permitir aprobar un producto para que sea visible en catálogo público.	RN-031	Requiere: ≥1 variante, ≥1 imagen principal, descripción, precio_base>0; solo usuarios con rol Admin/Revisor; mostrar checklist pre-validación; registrar en Auditoria.	RI-031	Tabla Producto: UPDATE aprobado=true, fecha_aprobacion=NOW(), usuario_id_aprobador=?. Validación: SELECT EXISTS (SELECT 1 FROM Imagen_Producto WHERE producto_id=? AND principal=true); SELECT COUNT(*) FROM Variante WHERE producto_id=?; Tabla Auditoria: accion='Aprobacion'.
		RF-032	El sistema debe permitir desaprobar un producto aprobado con motivo documentado.	RN-032	Requiere motivo (máx 200 caracteres); solo Admin/Revisor; producto se oculta del catálogo; registrar en Auditoria; notificar al creador del producto.	RI-032	Entidad Motivo_Desaprobacion: ID (PK), producto_id (FK → Producto.ID), motivo (varchar 200, NOT NULL), usuario_id_revisor (FK → Usuario.ID), fecha (timestamp). Tabla Producto: UPDATE aprobado=false.
		RF-033	El sistema debe mostrar historial de aprobaciones y desaprobaciones de un producto.	RN-033	Timeline visual de cambios de estado; usuario responsable; fecha; motivo si existe; permitir revertir si desaprobado (re-aprobar).	RI-033	SQL: SELECT * FROM Motivo_Desaprobacion WHERE producto_id=? ORDER BY fecha DESC; SELECT * FROM Auditoria WHERE tabla_afectada='Producto' AND registro_id=? AND accion IN ('Aprobacion','Desaprobacion') ORDER BY fecha DESC
	Gestión de Categorías	RF-034	El sistema debe permitir al administrador crear una nueva categoría para el catálogo.	RN-034	Categorías únicas por nombre; máximo 100 caracteres; requerir descripción.	RI-034	Entidad Categoria_Diseno: ID (PK, autoincremental), nombre (varchar 100, unique), descripcion (text), activo (boolean).
		RF-035	El sistema debe permitir al administrador editar o eliminar categorías existentes en el catálogo.	RN-035	No eliminar si hay diseños asociados; migrar diseños a "Sin Categoría" si se elimina.	RI-035	Actualización en Categoria_Diseno: Campos editables: nombre, descripcion; chequeo FK en Diseno_Publico.
Catálogo Vista de Cliente	Visualización	RF-036	El sistema debe mostrar grid de productos aprobados y activos en la página de catálogo.	RN-036	Solo productos con aprobado=true AND activo=true; carga inicial 50 productos; grid responsive (3-4 cols desktop, 2 tablet, 1 mobile); lazy loading imágenes; skeleton loaders durante carga.	RI-036	SQL: SELECT p.ID, p.nombre, p.precio_base, MIN(v.precio_variante) AS precio_min, i.ruta_imagen FROM Producto p LEFT JOIN Variante v ON p.ID=v.producto_id LEFT JOIN Imagen_Producto i ON p.ID=i.producto_id WHERE p.aprobado=true AND p.activo=true AND i.principal=true GROUP BY p.ID ORDER BY p.fecha_creacion DESC LIMIT 50 OFFSET ? Índice: idx_aprobado_activo (aprobado, activo)
		RF-037	El sistema debe mostrar información básica en cards de producto (nombre, precio, imagen, stock).	RN-037	Nombre máx 60 caracteres (truncar con ...); mostrar precio mínimo de variantes; imagen 200x200px thumbnail; alt_text para accesibilidad; badge rojo "Stock bajo" si <5 unidades.	RI-037	Mostrar en card: nombre (SUBSTR(nombre, 1, 60)), precio (COALESCE(MIN(precio_variante), precio_base)), imagen thumbnail, stock agregado; badge rojo si SUM (stock) < 5.
		RF-038	El sistema debe permitir paginación en vista de catálogo (siguiente/anterior).	RN-038	50 productos por página; mostrar página actual y total de páginas; botones siguiente/anterior; opción ir a página específica; mantener filtros activos al cambiar página.	RI-038	Parámetro OFFSET en SQL; mostrar navegación paginada: [1][2][3]...[Siguiente]; mostrar "X-Y de Z productos encontrados" en interfaz.
		RF-039	El sistema debe permitir cargar más productos (infinite scroll) como alternativa a paginación.	RN-039	Botón "Cargar más" al final de página; carga siguiente 50 productos sin recargar página; indicador loading visible; mantener filtros aplicados.	RI-039	AJAX/fetch con LIMIT 50 OFFSET (pagina*50); append resultados al DOM sin recargar; spinner visual durante carga.
		RF-040	El sistema debe registrar visualización de catálogo por cliente logueado para análisis.	RN-040	Registrar: fecha/hora acceso, usuario_id, cantidad productos vistos en sesión; opcional para análisis; no registrar si usuario no logueado.	RI-040	Entidad Sesion_Catalogo: ID (PK), usuario_id (FK → Usuario.ID NULL), fecha_acceso (timestamp DEFAULT CURRENT_TIMESTAMP), productos_vistos (int DEFAULT 0), duracion_sesion (int segundos NULL).
	Filtros	RF-041	El sistema debe permitir filtrar productos por rango de precio.	RN-041	Rango min-max con slider o rangos predefinidos (0-50k, 50-100k, 100k+); filtro combinable con otros; paginación si >20 resultados; mostrar precios encontrados.	RI-041	SQL con WHERE: precio_base (o MIN(variante.precio_variante)) BETWEEN ? AND ? Índice: idx_producto_precio (precio_base). Frontend: slider o input number min/max dinámico.
		RF-042	El sistema debe permitir filtrar por talla disponible (S, M, L, XL).	RN-042	Mostrar solo tallas con stock>0; listar en orden: S, M, L, XL; mostrar cantidad de variantes por talla; filtro combinable; actualizar dinámicamente.	RI-042	SQL: SELECT DISTINCT talla FROM Variante WHERE producto_id IN (SELECT ID FROM Producto WHERE aprobado=true AND activo=true) AND stock>0 ORDER BY FIELD(talla,'S','M','L','XL') Índice: idx_variante_talla (talla)
		RF-043	El sistema debe permitir filtrar por color disponible.	RN-043	Mostrar solo colores con stock>0; mostrar como swatches HEX (cuadrados de color); nombres descriptivos; filtro combinable; actualizar dinámicamente.	RI-043	SQL: SELECT DISTINCT color_hex, color_nombre FROM Variante WHERE producto_id IN (...) AND stock>0 Índice: idx_variante_color (color_hex). Frontend: divs con background-color: {color_hex}.

		RF-044	El sistema debe permitir buscar productos por palabras clave (nombre y descripción).	RN-044	Búsqueda parcial en nombre+descripción; insensible mayúsculas; resultados paginados máx 20; mostrar coincidencias resaltadas; opcional: autocomplete sugerencias.	RI-044	Índice fulltext: FULLTEXT (nombre, descripcion). SQL: SELECT * FROM Producto WHERE MATCH(nombre, descripcion) AGAINST(? IN BOOLEAN MODE) AND aprobado=true AND activo=true ORDER BY fecha_creacion DESC LIMIT 20
		RF-045	El sistema debe permitir combinar filtros (precio AND talla AND color AND búsqueda).	RN-045	Lógica AND para todos filtros activos; mostrar filtros activos como chips removibles; botón "Limpiar todo"; actualizar resultados en tiempo real; mostrar conteo.	RI-045	SQL con WHERE múltiple: precio BETWEEN ? AND ? AND talla IN (...) AND color_hex IN (...) AND MATCH (nombre)... Frontend: chips filtros con X para remover; actualizar con cada filtro.
		RF-046	El sistema debe mostrar opciones de ordenamiento en dropdown.	RN-046	Opciones: Relevancia, Precio (menor/mayor), Más vendidos, Más recientes, Mejor calificación; default=Relevancia; dropdown selector; actualizar resultados al seleccionar.	RI-046	ORDER BY: MATCH() relevancia, precio_base ASC/DESC, ventas DESC, fecha_creacion DESC, calificacion_promedio DESC. Dropdown con opciones clickeables.
	Interacción Cards	RF-047	El sistema debe permitir clic en card para ir a vista detalle del producto.	RN-047	Clic abre vista detalle; pasar producto_id en URL (/producto?id=X); registrar visualización si logueado; mantener scroll anterior al volver a catálogo.	RI-047	Entidad Historial_Visualizacion: ID (PK), usuario_id (FK → Usuario.ID NULL), producto_id (FK → Producto.ID), fecha_visualizacion (timestamp), duracion_segundos (int NULL). Breadcrumb: Catálogo > Nombre_Producto.
		RF-048	El sistema debe permitir agregar al carrito desde card de catálogo con mini-modal.	RN-048	Botón "Agregar carrito" en card; mini-modal para seleccionar talla+color+cantidad; validar stock disponible; actualizar carrito sin recargar página; mostrar confirmación (toast).	RI-048	Mini-modal: SELECT tallas/colores con stock>0; input cantidad con max=stock disponible; INSERT Carrito_Item; mostrar toast "Agregado al carrito"; botones: Ver carrito / Continuar.
		RF-049	El sistema debe mostrar stock disponible agregado en card.	RN-049	Mostrar "Stock: X unidades" o "Agotado"; calcular SUM(variantes.stock) agregado; actualizar con filtros; si logueado, permitir notificación de restock.	RI-049	SUM(v.stock) AS stock_total FROM Variante WHERE producto_id=?; mostrar en card; indicador visual rojo si stock_total < 5.
		RF-050	El sistema debe mostrar rating en card si producto tiene calificaciones.	RN-050	Mostrar estrellas (1-5) con promedio; cantidad de calificaciones entre paréntesis; solo si ≥3 calificaciones registradas en sistema.	RI-050	SQL: AVG(estrellas) AS cal_promedio, COUNT(*) AS num_calificaciones FROM Calificacion WHERE producto_id=? AND estrellas IS NOT NULL Mostrar solo si num_calificaciones >= 3.
Productos Modulo Cliente - Vista de Detalles de Productos	Mostrar Detalles	RF-051	El sistema debe mostrar galería de imágenes ampliada con zoom.	RN-051	Imagen principal grande (500x500px mínimo); thumbnails 100x100px; zoom al pasar mouse (100%-200%); navegación flechas/teclado; lazy loading imágenes.	RI-051	SQL: SELECT * FROM Imagen_Producto WHERE producto_id=? ORDER BY principal DESC, orden ASC Implementar con librería (Lightbox2, Fancybox o similar); mostrar principal al cargar; permitir click en thumbnail.
		RF-052	El sistema debe mostrar descripción completa del producto.	RN-052	Descripción completa (hasta 500 caracteres); si >300 chars, mostrar preview + botón "Leer más" expandible; formatear saltos de línea.	RI-052	SQL: TRUNCATE(descripcion, 300) si LENGTH > 300 sino completo Frontend: mostrar 300 chars + "Leer más" link que expande descripcion completa.
		RF-053	El sistema debe mostrar información de precio y stock disponible.	RN-053	Mostrar precio_base o mínimo variante en grande; stock agregado (SUM variantes); desglose stock por variante; mostrar "Agotado" si SUM=0.	RI-053	SQL: SUM(v.stock) AS stock_total, MIN(COALESCE(v.precio_variante, p.precio_base)) AS precio_min FROM Producto p LEFT JOIN Variante v WHERE p.ID=? Mostrar: "Precio: $X" (grande), "Stock disponible: Y unidades".
		RF-054	El sistema debe mostrar calificación promedio y reseñas de clientes.	RN-054	Mostrar estrellas promedio (1-5); número de reseñas; solo si existen calificaciones; mostrar primeras 3-5 reseñas con opción "Ver más".	RI-054	SQL: SELECT AVG(estrellas) AS promedio, COUNT(*) AS total FROM Calificacion WHERE producto_id=?; SELECT TOP 5 ... FROM Calificacion WHERE producto_id=? ORDER BY fecha DESC
	Seleccionar Variante	RF-055	El sistema debe permitir seleccionar talla disponible.	RN-055	Listar solo tallas con stock>0; orden: S, M, L, XL; botones/radio para seleccionar; requerir antes de agregar carrito; actualizar colores disponibles al cambiar talla.	RI-055	SQL: SELECT DISTINCT talla FROM Variante WHERE producto_id=? AND stock>0 ORDER BY FIELD (talla,'S','M','L','XL') Frontend: botones con talla; onclick actualiza colores disponibles para esa talla.
		RF-056	El sistema debe permitir seleccionar color disponible.	RN-056	Mostrar como swatches HEX (cuadrados de color); solo con stock>0 para talla seleccionada; actualizar precio/stock dinámicamente; mostrar nombre color al pasar mouse (hover).	RI-056	SQL: SELECT DISTINCT v.color_hex, v.color_nombre FROM Variante v WHERE v.producto_id=? AND v.talla=? AND v.stock>0 Frontend: divs con background-color: color_hex; onclick actualiza precio mostrado.
		RF-057	El sistema debe mostrar stock de variante seleccionada en tiempo real.	RN-057	Si stock=0, mostrar "Agotado" y deshabilitar agregar carrito; si stock<5, mostrar "Últimas unidades"; actualizar dinámicamente al cambiar talla/color.	RI-057	SQL: SELECT stock, precio_variante, p.precio_base FROM Variante v JOIN Producto p WHERE v.producto_id=? AND v.talla=? AND v.color_hex=?
		RF-058	El sistema debe actualizar precio dinámicamente según variante seleccionada.	RN-058	Mostrar: precio_variante si existe, sino precio_base; actualizar en tiempo real al cambiar talla/color; mostrar diferencia si hay sobrecosto respecto a base.	RI-058	Frontend: precio_mostrado = variante.precio_variante ? variante.precio_variante : producto.precio_base; actualizar onclick color/talla en tiempo real.
	Carrito e Interacción	RF-059	El sistema debe permitir agregar producto con variante al carrito.	RN-059	Requerir talla, color, cantidad válida; cantidad máxima = stock disponible; validar antes de insertar; mostrar confirmación/toast; actualizar carrito real-time.	RI-059	Tabla Carrito_Item: carrito_id (FK), variante_id (FK), cantidad (int CHECK > 0 AND <= stock), precio_unitario (decimal), subtotal (GENERATED), fecha_agregado (timestamp). Validación: SELECT stock FROM Variante; INSERT con CHECK cantidad <= stock.
		RF-060	El sistema debe permitir ingresar cantidad a comprar con validación.	RN-060	Spinner o input numérico; validar cantidad > 0 y <= stock disponible; actualizar precio total dinámicamente; botones menos/más para incrementar/decrementar.	RI-060	Frontend: input type=number, min=1, max=variante.stock; mostrar precio_total = cantidad * precio_variante; botones +/- para incrementar cantidad.

		RF-061	El sistema debe permitir acceder al editor 3D desde detalle producto.	RN-061	Botón "Personalizar en 3D"; cargar modelo basado en variante seleccionada; solo si logueado o permitir preview; parámetros en URL (/editor-3d?variante_id=X&producto_id=Y).	RI-061	Botón onclick redirige a /editor-3d?variante_id=X&producto_id=Y; validar WebGL en navegador; si no soporta, mostrar mensaje de fallback.
		RF-062	El sistema debe mostrar botón compartir producto en redes sociales.	RN-062	Botones: Facebook, Twitter, WhatsApp, email; incluir URL producto y descripción corta; generar preview en redes sociales.	RI-062	Implementar con APIs de redes sociales o librería share.js; compartir: URL producto + nombre + descripción corta (máx 200 chars).
		RF-063	El sistema debe mostrar productos relacionados al final de detalle.	RN-063	Máximo 4-6 productos relacionados; misma categoría u otro criterio; excluir el producto actual; mostrar grid similar al catálogo.	RI-063	SQL: SELECT p.* FROM Producto p WHERE p.aprobado=true AND p.activo=true AND p.ID != ? AND p.ID IN (SELECT DISTINCT producto_id FROM Variante WHERE talla IN (...talla actual...)) LIMIT 6
Personalización 3D de Camisetas (Editor 3D y Productos 3D fusionados en uno solo)	1. Carga y Renderizado del Modelo 3D	RF-064	El sistema debe renderizar un modelo 3D de la camiseta con el diseño aplicado en tiempo real.	RN-064	Usar motor como Three.js; soportar formatos GLTF; actualizar en <1s por cambio.		
	2. Interacción Básica	RF-065	El sistema debe permitir al usuario rotar el modelo 3D de la camiseta para ver desde diferentes ángulos.	RN-065	Rotación libre en ejes X/Y.		
	3. Personalización del Diseño	RF-066	El sistema debe permitir cambiar el colorde la camiseta	RN-066	Solo colores en formato HEX o RGB predefinidos en paleta.		
		RF-067	El sistema debe permitir agregar imágenes o logos al modelo 3D.	RN-067	Solo PNG/JPG; tamaño máximo 2MB; resolución mínima 1024x1024 px.		
		RF-068	El sistema debe permitir agregar texto personalizable sobre la camiseta.	RN-068	Máximo 30 caracteres; fuentes del sistema; tamaño y color ajustables.		
	4. Guardado y Restauración	RF-069	El sistema debe permitir guardar configuraciones personalizadas del diseño.	RN-069	Máximo 10 configuraciones por usuario; nombre obligatorio.		
		RF-070	El sistema debe permitir restaurar la camiseta al diseño original por defecto.	RN-070	La restauración elimina todos los cambios no guardados.		
	5. Vista Previa y Exportación	RF-071	El sistema debe generar una vista previa en 3D del diseño personalizado antes de agregar al carrito.	RN-071	Vista previa refleja cambios en tiempo real; exportable como PNG.		
		RF-072	El sistema debe permitir capturar una vista previa en imagen del diseño 3D.	RN-072	Captura en resolución 1080x1080 píxeles.		
		RF-073	El sistema debe permitir exportar el diseño en formato de imagen o archivo 3D (.obj, . fbx).	RN-073	Archivo exportado incluye texturas, colores y elementos aplicados.		
	6. Integración con Catálogo	RF-074	El sistema debe integrar diseños del catálogo en el modelo 3D para visualización personalizada.	RN-074	Solo diseños públicos aprobados; validar compatibilidad de tamaño con el modelo.		

		RF-075	El sistema debe permitir al administrador editar o eliminar modelos 3D existentes.	RN-075	No eliminar si hay diseños activos asociados; notificar usuarios afectados.		
Pedido		RF-076	El sistema debe permitir al usuario agregar productos al carrito desde el catálogo o el editor 3D.	RN-076	No se podrá agregar más unidades que el stock disponible para el producto.	RI-076	Entidad Carrito: carrito_id (PK), usuario_id (FK), fecha_creacion (datetime), estado (enum: Activo, Pendiente, Procesado, Vacio).
		RF-077	El usuario debe poder aumentar o disminuir la cantidad de productos dentro del carrito.	RN-077	La cantidad mínima permitida es 1 unidad por producto.	RI-077	Entidad Carrito_Item: id (PK), carrito_id (FK), producto_id (FK), cantidad (int), precio_unitario (decimal 10,2), subtotal (decimal 10,2).
		RF-078	El usuario debe poder eliminar un producto del carrito.	RN-078	Solo se podrá eliminar ítems del carrito que esten antes del pagar.	RI-078	Carrito_Item utiliza su propio ID para la eliminación directa estado_item (enum: Activo, Pendiente, Eliminado).
		RF-079	El usuario debe poder vaciar completamente el carrito con un solo botón.	RN-079	No se podrá vaciar un carrito si ya está en estado "Procesado".	RI-079	El carrito cambia su estado a "Vacio" y todos los Carrito_Item se eliminan.
		RF-080	El sistema debe mostrar al usuario los productos en su carrito, incluyendo imagen, color, talla, precio y subtotal.	RN-080	El carrito debe calcular totales automáticamente cuando se modifique una cantidad.	RI-080	Vista Carrito-Detalle: lista de Carrito_Item + total_general (decimal 10,2).
		RF-081	El usuario debe poder confirmar su compra y enviar el pedido a la pasarela de pago Wompi.	RN-081	Solo se permite procesar carritos con al menos un ítem y datos completos de envío.	RI-081	Entidad Pedido: pedido_id (PK), usuario_id (FK), carrito_id (FK), direccion_envio (varchar 255), metodo_pago (varchar 50), total (decimal), fecha_pedido (datetime), estado (enum: Pendiente, Pagado, Cancelado).
		RF-082	El sistema debe recibir la confirmación del pago desde Wompi y actualizar el estado del pedido.	RN-082	Si Wompi rechaza el pago, el pedido cambia a estado "Cancelado".	RI-082	Entidad Transaccion: id (PK), pedido_id (FK), referencia_wompi (varchar 255), estado_pago (enum), fecha (datetime), valor_pagado (decimal).
Pedido Admin		RF-083	El administrador debe poder consultar la lista de pedidos realizados por los usuarios.	RN-083	El administrador solo puede ver pedidos, no modificarlos si están en estado "Pagado".	RI-083	Vista Administrativa Pedido: pedido_id, usuario, total, estado, fecha.
		RF-084	El administrador debe poder actualizar el estado del pedido (Pendiente, Producción, Enviado, Cancelado).	RN-084	Solo usuarios con rol Administrador pueden cambiar estados.	RI-084	Entidad Pedido incluye campo estado (enum: Pendiente, Producción, Enviado, Cancelado).
		RF-085	El administrador debe poder visualizar el detalle del carrito asociado a un pedido.	RN-085	Ningún administrador puede modificar cantidades una vez el pedido fue pagado.	RI-085	Carrito y Carrito_Item se asocian mediante carrito_id.
		RF-086	El sistema debe permitir generar una factura simple del pedido.	RN-086	La factura simple no será equivalente a factura DIAN.	RI-086	Entidad Factura_Simple: factura_id (PK), pedido_id (FK), fecha_generada, total_final.
Checkout		RF-087	El sistema debe permitir al usuario ingresar los datos de envío: nombre, correo, teléfono, dirección, ciudad y código postal.	RN-087	Todos los campos de envío son obligatorios para continuar al pago.	RI-087	Datos_Envio: id (PK), pedido_id (FK), nombre, correo, telefono, direccion, ciudad, codigo_postal.
		RF-088	El usuario debe visualizar un resumen del pedido con productos, cantidades, subtotales y total final.	RN-088	El total debe calcularse únicamente en backend para evitar alteraciones.	RI-088	Pedido_Resumen: lista de Carrito_Item + total_general + descuentos.
		RF-089	El sistema debe mostrar Wompi como único método de pago disponible.	RN-089	No se permite seleccionar otro método distinto a Wompi.	RI-089	Pago: metodo_pago (enum: Wompi), estado_pago (enum: Pendiente, Pagado, Fallido).
		RF-090	El sistema debe redirigir al usuario a Wompi con los datos del pedido para procesar el pago.	RN-090	El monto enviado debe coincidir exactamente con el total calculado por backend.	RI-090	Transaccion: id (PK), pedido_id (FK), referencia_wompi, valor_pagado, estado, fecha.

		RF-091	El sistema debe recibir la confirmación del pago desde Wompi y actualizar el estado del pedido.	RN-091	El pedido solo pasa a "Pagado" si Wompi lo confirma.	RI-091	Pedido.estado: Pendiente, Pagado, Cancelado.
		RF-092	El sistema debe generar un número único de pedido al finalizar el pago.	RN-092	El número de pedido debe ser incremental y no repetirse.	RI-092	Pedido.numero_pedido: varchar(30), único.
		RF-093	El sistema debe mostrar una página de confirmación de compra con detalles del pedido.	RN-093	La confirmación solo se muestra si el estado del pedido es "Pagado".	RI-093	Pedido_Confirmacion: numero_pedido, usuario, fecha, total, productos
Checkout Admin		RF-094	El administrador debe poder ver la lista de pagos procesados por Wompi.	RN-094	Solo usuarios con rol Administrador pueden acceder al historial de pagos.	RI-094	Admin_Pagos: referencia_wompi, pedido_id, valor, estado, fecha.
		RF-095	El sistema debe registrar todo intento de pago (exitoso o fallido) para auditoría.	RN-095	Ningún registro de pago puede ser eliminado por un administrador.	RI-095	Log_Pagos: id, pedido_id, estado, descripcion_evento, fecha.
		RF-096	El administrador debe poder consultar la lista de pedidos pagados para continuar con la producción.	RN-096	Solo pedidos con estado "Pagado" pueden pasar a producción.	RI-096	Pedido.estado: Pendiente, Pagado, Producción, Enviado, Cancelado.
Gestión de Contacto	Procesar consulta de contacto	RF-67	El sistema debe permitir a los usuarios enviar un formulario de contacto con campos: nombre, correo electrónico y mensaje.	RN-67	Todos los campos (nombre, correo electrónico, mensaje) son obligatorios	RI-67	Entidad Contacto: ID (PK, autoincremental), nombre (varchar 100), correo_electronico (varchar 100), mensaje (text), fecha_envio (datetime), estado (enum: Pendiente, Respondido, Archivado)"
		RF-68	El sistema debe confirmar al usuario el envío exitoso del mensaje mediante un mensaje en pantalla.	RN-68	El correo electrónico debe tener un formato válido (ej: usuario@dominio.com).	RI-68	"Entidad Configuracion_Contacto: ID (PK), email_destino_notificaciones (varchar 100), asunto_predeterminado (varchar 255), activo (boolean)"
		RF-69	El sistema debe almacenar los datos del formulario en la base de datos para su posterior procesamiento.	RN-69	El mensaje debe tener una longitud mínima de 10 caracteres y máxima de 1000 caracteres.	RI-69	
		RF-70	El sistema debe permitir a los administradores visualizar las consultas de contacto desde un panel de administración.	RN-70	Las consultas de contacto deben ser accesibles solo para usuarios con rol de Administrador.	RI-70	