Meraki - Documentación técnica

Josué Ribero Duarte - Cód. 67001295

---

1. MAPA DE ENDPOINTS

1.1 Módulo de administrador

Método	Ruta	Permisos	Descripción
PATCH	/admin/	Admin	Actualizar el nombre del administrador.
PATCH	/admin/contrasena	Admin	Actualizar la contraseña del administrador.

1.2 Módulo de autenticación

Método	Ruta	Permisos	Descripción
POST	/auth/login	Público	Inicio de sesión (cliente/admin).
GET	/auth/login	Público	Redireccionamiento si hay sesión activa.
GET	/auth/logout	Público	Cierre de sesión.

1.3 Módulo de carrito

Método	Ruta	Permisos	Descripción
POST	/carrito/agregar-producto	Cliente	Agregar producto al carrito.
POST	/carrito/pedir	Cliente	Hacer un pedido desde el carrito.
GET	/carrito/mi-carrito	Cliente	Obtener el carrito del cliente.
PATCH	/carrito/actualizar-cantidad/{productoID}	Cliente	Actualizar cantidad de un producto en el carrito.
DELETE	/carrito/{productoID}	Cliente	Eliminar producto del carrito.
DELETE	/carrito/vaciar	Cliente	Vaciar carrito.

1.4 Módulo de categorías

Método	Ruta	Permisos	Descripción
POST	/categorias/	Admin	Crear categoría.
GET	/categorias/	Público	Listar categorías activas.
GET	/categorias/todas	Admin	Ver categorías completas (incluye inactivas).
GET	/categorias/{categoriaID}	Público	Ver categoría por id.
PATCH	/categorias/{categoriaID}	Admin	Actualizar categoría por id.
PATCH	/categorias/{categoriaID}/habilitar	Admin	Reactivar una categoría.
DELETE	/categorias/{categoriaID}/deshabilitar	Admin	Desactivar categoría.

1.5 Módulo de clientes

Método	Ruta	Permisos	Descripción
POST	/clientes/registrar	Público	Registro de clientes (crea carrito y wishlist automáticamente).
GET	/clientes/mi-perfil	Cliente	Ver perfil del cliente.
GET	/clientes/	Admin	Listar clientes activos.
GET	/clientes/todos	Admin	Listar todos los clientes (activos e inactivos).
GET	/clientes/eliminados	Admin	Listar clientes eliminados (histórico).
GET	/clientes/{clienteID}	Admin	Obtener cliente por id.
PATCH	/clientes/{clienteID}	Cliente	Actualizar datos personales.
DELETE	/clientes/eliminar-cuenta	Cliente	Eliminar cuenta propia (con validaciones).

1.6 Módulo de dashboard

Método	Ruta	Permisos	Descripción
GET	/dashboard/	Admin	KPIs del sistema (renderiza HTML).
GET	/api/dashboard/resumen	Admin	Resumen de ventas mensuales (JSON).
GET	/api/dashboard/ventas-mensuales	Admin	Datos para gráfica de ventas.
GET	/api/dashboard/pedidos-recientes	Admin	Pedidos de los últimos 7 días.
GET	/api/dashboard/productos-mas-vendidos	Admin	Lista de productos más vendidos.

1.7 Módulo de detalle de carrito

Método	Ruta	Permisos	Descripción
GET	/detalleCarrito/carrito/{carritoID}	Cliente	Lista de detalles del carrito por id.
PATCH	/detalleCarrito/{detalleID}	Cliente	Actualizar detalle por id.
DELETE	/detalleCarrito/{detalleID}	Cliente	Eliminar detalle por id.

1.8 Módulo de detalle de pedido

Método	Ruta	Permisos	Descripción
GET	/pedido/{pedidoID}	Cliente	Lista de pedidos del cliente.

1.9 Módulo de dirección de envío

Método	Ruta	Permisos	Descripción
POST	/direcciones/crear	Cliente	Añadir dirección.
GET	/direcciones/mis-direcciones	Cliente	Listar mis direcciones.
PATCH	/direcciones/{direccionID}	Cliente	Editar dirección.
DELETE	/direcciones/{direccionID}	Cliente	Eliminar dirección.

1.10 Módulo de diseño personalizado

Método	Ruta	Permisos	Descripción
POST	/disenos/crear	Cliente	Crear diseño personalizado.
POST	/disenos/agregar	Cliente	Agregar diseño al carrito.
GET	/disenos/mis-disenos	Cliente	Ver diseños propios.
PATCH	/disenos/{disenoID}	Admin	Cambiar estado del diseño.

1.11 Módulo de pagos

Método	Ruta	Permisos	Descripción
POST	/pagos/crear	Cliente	Crear pago de un pedido.
GET	/pagos/	Cliente	Lista de pagos del cliente.
GET	/pagos/{pagoID}	Cliente	Obtener detalle de un pago.
GET	/pagos/pedido/{pedidoID}	Admin	Obtener pago de un pedido específico.
GET	/pagos/admin/lista	Admin	Lista de pagos para admin.

1.12 Módulo de pedidos

Método	Ruta	Permisos	Descripción
GET	/pedidos/mis-pedidos	Cliente	Listar los pedidos de un cliente.
GET	/pedidos/	Admin	Listar pedidos de todos los clientes.
GET	/pedidos/admin/{pedidoID}	Admin	Pedido por id con detalles completos.
GET	/pedidos/mi-pedido/{pedidoID}	Cliente	Ver pedido de un cliente con detalles.
PATCH	/pedidos/{pedidoID}/cancelar	Cliente	Cancelar un pedido (solo estado POR_PAGAR).
PATCH	/pedidos/{pedidoID}	Admin	Actualizar estado del pedido.
PATCH	/pedidos/{pedidoID}/confirmar	Admin	Confirmar pedido y procesar puntos.

1.13 Módulo de productos

Método	Ruta	Permisos	Descripción
POST	/productos/crear	Admin	Crear producto.
GET	/productos/	Público	Listar productos activos.
GET	/productos/todas	Admin	Lista todos los productos (incluye inactivos).
GET	/productos/{productoID}	Público	Obtener producto.
PATCH	/productos/{productoID}	Admin	Actualizar un producto.
PATCH	/productos/{productoID}/habilitar	Admin	Rehabilitar un producto.
DELETE	/productos/{id}/deshabilitar	Admin	Inactivar producto.

1.14 Módulo de solicitud de recuperación de contraseña

Método	Ruta	Permisos	Descripción
POST	/recuperacion/crear	Cliente	Solicitar recuperación de contraseña.
GET	/recuperacion/validar/{token}	Cliente	Validar token de recuperación.

1.15 Módulo de transacción de puntos

Método	Ruta	Permisos	Descripción
GET	/transacciones/mis-transacciones	Cliente	Lista de transacciones de puntos.

1.16 Módulo de wishlist

Método	Ruta	Permisos	Descripción
POST	/wishlist/agregar-producto	Cliente	Agregar producto a la wishlist.
POST	/wishlist/mover-al-carrito/{productoID}	Cliente	Mover producto de la wishlist al carrito.
GET	/wishlist/mi-wishlist	Cliente	Listar wishlist.
DELETE	/wishlist/{productoID}	Cliente	Eliminar de favoritos.

2. CASOS DE USO DETALLADOS

CU-01: Registro de Cliente

Actor: Cliente
Precondición: No estar registrado previamente
Flujo Principal:

1. Cliente accede a /clientes/registrar
2. Completa formulario con nombre, email, teléfono, contraseña
3. Sistema valida formato de email y unicidad
4. Sistema valida teléfono (mínimo 7 dígitos)
5. Hashea contraseña con bcrypt
6. Crea registro Cliente
7. Crea Carrito asociado
8. Crea Wishlist asociada
9. Inicia sesión automáticamente
10. Redirige a página principal

Flujo Alternativo 1: Email existente -> HTTP 400 "Este email ya tiene una cuenta asociada"
Flujo Alternativo 2: Email en histórico -> HTTP 400 "Este email ya tuvo una cuenta asociada y no puede volver a registrarse"
Flujo Alternativo 3: Teléfono inválido -> HTTP 400 "El número de teléfono debe tener al menos 7 dígitos"
Postcondición: Cliente autenticado con carrito y wishlist creados

---

CU-02: Inicio de Sesión

Actor: Cliente / Administrador
Precondición: Estar registrado
Flujo Principal:

1. Usuario accede a /auth/login
2. Ingresa email y contraseña
3. Sistema busca en tabla Administrador
4. Si coincide, verifica hash bcrypt
5. Si es admin: crea sesión administradorID, redirige a /dashboard
6. Si no es admin: busca en tabla Cliente
7. Verifica hash bcrypt
8. Crea sesión clienteID, redirige a /
9. Genera cookie de sesión segura

Flujo Alternativo 1: Credenciales inválidas -> HTTP 401 "Credenciales inválidas"
Postcondición: Sesión activa según rol

CU-03: Crear Pedido desde Carrito

Actor: Cliente
Precondición: Carrito con al menos un producto, cliente autenticado
Flujo Principal:

1. Cliente envía POST a /carrito/pedir con direccionID opcional
2. Sistema verifica carrito no vacío
3. Si direccionID presente: valida que pertenezca al cliente
4. Calcula subtotal sumando detalles
5. Si subtotal < 30000: aplica costo_envio = 8900
6. Crea registro Pedido con estado POR_PAGAR
7. Para cada detalle del carrito:
    - Verifica stock suficiente
    - Crea DetallePedido
    - Descuenta stock del producto
8. Elimina todos los detalles del carrito
9. Commit transaction
10. Retorna pedido creado

Flujo Alternativo 1: Stock insuficiente -> HTTP 400 "Stock insuficiente para {producto}"
Flujo Alternativo 2: Carrito vacío -> HTTP 400 "Carrito vacío"
Postcondición: Pedido creado, carrito vacío, stock actualizado

---

CU-04: Confirmar Pago (Admin)

Actor: Administrador
Precondición: Pedido existente con pago pendiente
Flujo Principal:

1. Admin accede a /pedidos/{pedidoID}/confirmar
2. Sistema busca pedido y pago asociado
3. Verifica que pago no esté ya confirmado
4. Actualiza estado pedido a PAGADO
5. Marca pago.confirmado = True
6. Si no pagó con puntos:
    - Calcula puntos_ganados = int(total * 0.05)
    - Acredita puntos a cliente

    - Crea transacción GANADOS
7. Commit transaction
8. Retorna pedido actualizado

Flujo Alternativo 1: Pago ya confirmado -> HTTP 400 "El pago ya está confirmado"
Postcondición: Pedido pagado, puntos acreditados (si aplica)

CU-05: Eliminar Cuenta de Cliente

Actor: Cliente
Precondición: Cliente autenticado, sin pedidos activos
Flujo Principal:

1. Cliente envía DELETE a /clientes/eliminar-cuenta
2. Sistema verifica pedidos activos (POR_PAGAR o PENDIENTE)
3. Verifica diseños en producción
4. Crea registro en ClienteHistorico con datos actuales
5. Elimina Carrito asociado
6. Elimina Wishlist asociada
7. Elimina registro Cliente
8. Limpia sesión completamente
9. Retorna mensaje de éxito

Flujo Alternativo 1: Tiene pedidos activos -> HTTP 400 "No puedes eliminar tu cuenta porque tienes pedidos activos"
Flujo Alternativo 2: Tiene diseños en producción -> HTTP 400 "No puedes eliminar tu cuenta porque tienes diseños en producción"
Postcondición: Cliente eliminado, datos en histórico, sesión cerrada

---

CU-06: Agregar a Wishlist

Actor: Cliente
Precondición: Cliente autenticado, producto activo
Flujo Principal:

1. Cliente envía POST a /wishlist/agregar-producto con productoID
2. Sistema verifica producto existe y está activo
3. Verifica si cliente tiene wishlist (si no, la crea)
4. Valida que producto no esté ya en wishlist
5. Crea WishlistItem
6. Retorna item creado

Flujo Alternativo 1: Producto ya en wishlist -> HTTP 400 "El producto ya está en tu lista de deseos"
Postcondición: Producto agregado a wishlist

CU-07: Mover Wishlist a Carrito

Actor: Cliente
Precondición: Producto en wishlist, stock disponible
Flujo Principal:

1. Cliente envía POST a /wishlist/mover-al-carrito/{productoID}
2. Sistema busca wishlist del cliente
3. Verifica producto está en wishlist
4. Obtiene carrito del cliente
5. Si producto ya en carrito: incrementa cantidad
6. Si no: crea nuevo DetalleCarrito con cantidad 1
7. Elimina producto de wishlist
8. Commit transaction
9. Retorna mensaje de éxito

Flujo Alternativo 1: Producto no en wishlist -> HTTP 404 "El producto no está en tu wishlist"
Postcondición: Producto en carrito, eliminado de wishlist

CU-08: Crear Diseño Personalizado

Actor: Cliente
Precondición: Cliente autenticado
Flujo Principal:

1. Cliente envía POST a /disenos/crear con imagenURL y precioEstimado
2. Sistema crea DisenoPersonalizado con estado PENDIENTE
3. Asocia al cliente actual
4. Retorna diseño creado

Postcondición: Diseño guardado para revisión

CU-09: Actualizar Estado de Diseño (Admin)

Actor: Administrador
Precondición: Diseño pendiente existe
Flujo Principal:

1. Admin envía PATCH a /disenos/{disenoID}
2. Sistema verifica diseño existe
3. Actualiza estado y/o precioEstimado
4. Retorna diseño actualizado

Postcondición: Diseño actualizado con nuevo estado/precio

CU-10: Generar Reporte de Ventas

Actor: Administrador
Precondición: Existencia de pedidos pagados
Flujo Principal:

1. Admin accede a /reportes/ventas.csv
2. Sistema genera CSV con: id, fecha, cliente, total, estado
3. Retorna archivo para descarga

Postcondición: Reporte CSV generado

---

4. FLUJOS de EXCEPCIÓN Y REGLAS de NEGOCIO

Reglas de Negocio Clave

1. **Cálculo de Envío:** Si subtotal < $30.000, costo_envio = $8.900
2. **Sistema de Puntos:** 1 punto = $1. Gana 5% del total en cada compra confirmada
3. **Unicidad:** Email único en clientes y administradores. SKU único en productos
4. **Stock:** No permite agregar al carrito más de stock disponible
5. **Eliminación de Cliente:** Solo si no tiene pedidos activos ni diseños en producción
6. **Cancelación:** Solo pedidos en estado POR_PAGAR pueden ser cancelados
7. **Categorías:** No se pueden eliminar categorías con productos activos
8. **Sesiones:** Middleware maneja expiración automática de sesiones

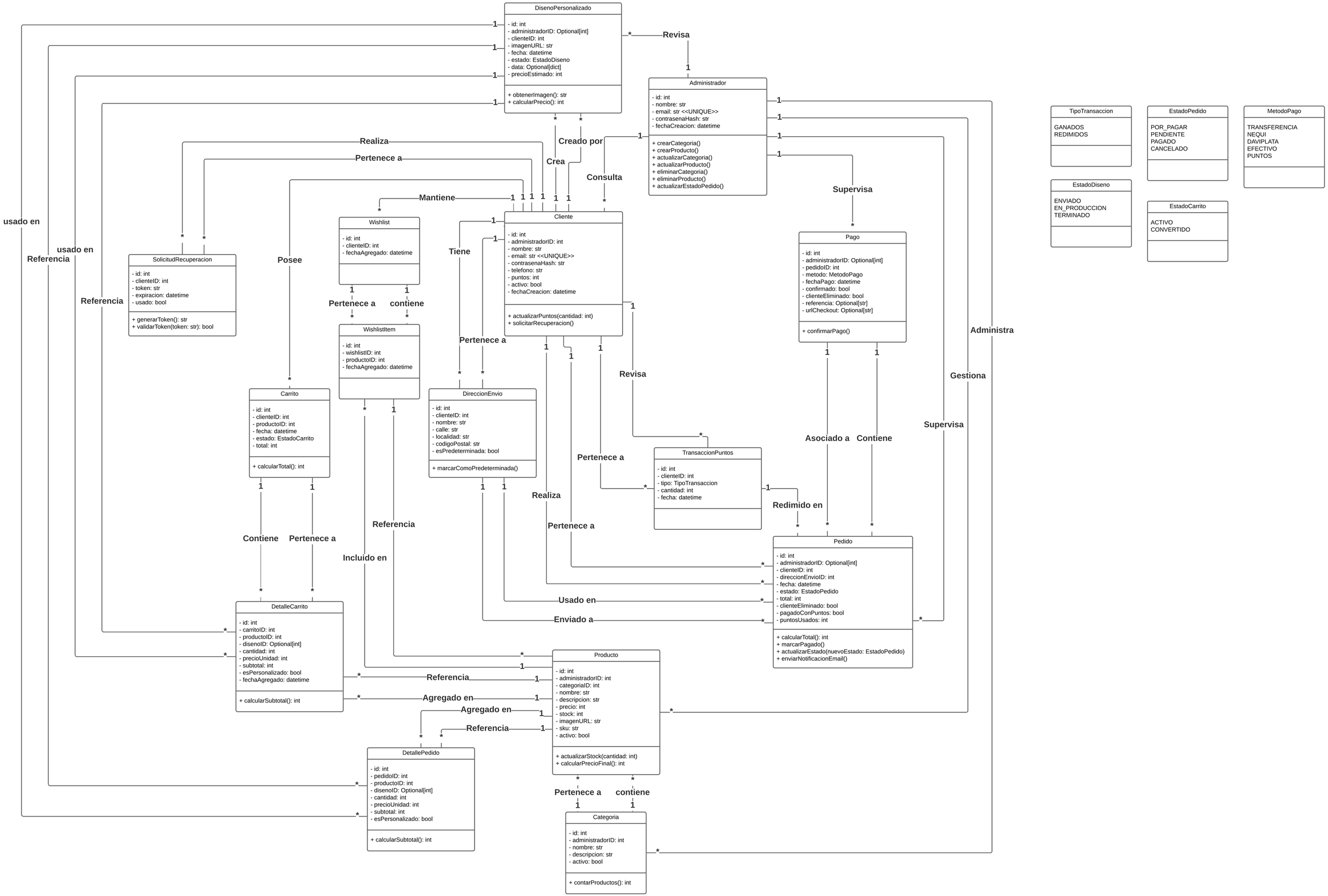
Manejo de Errores

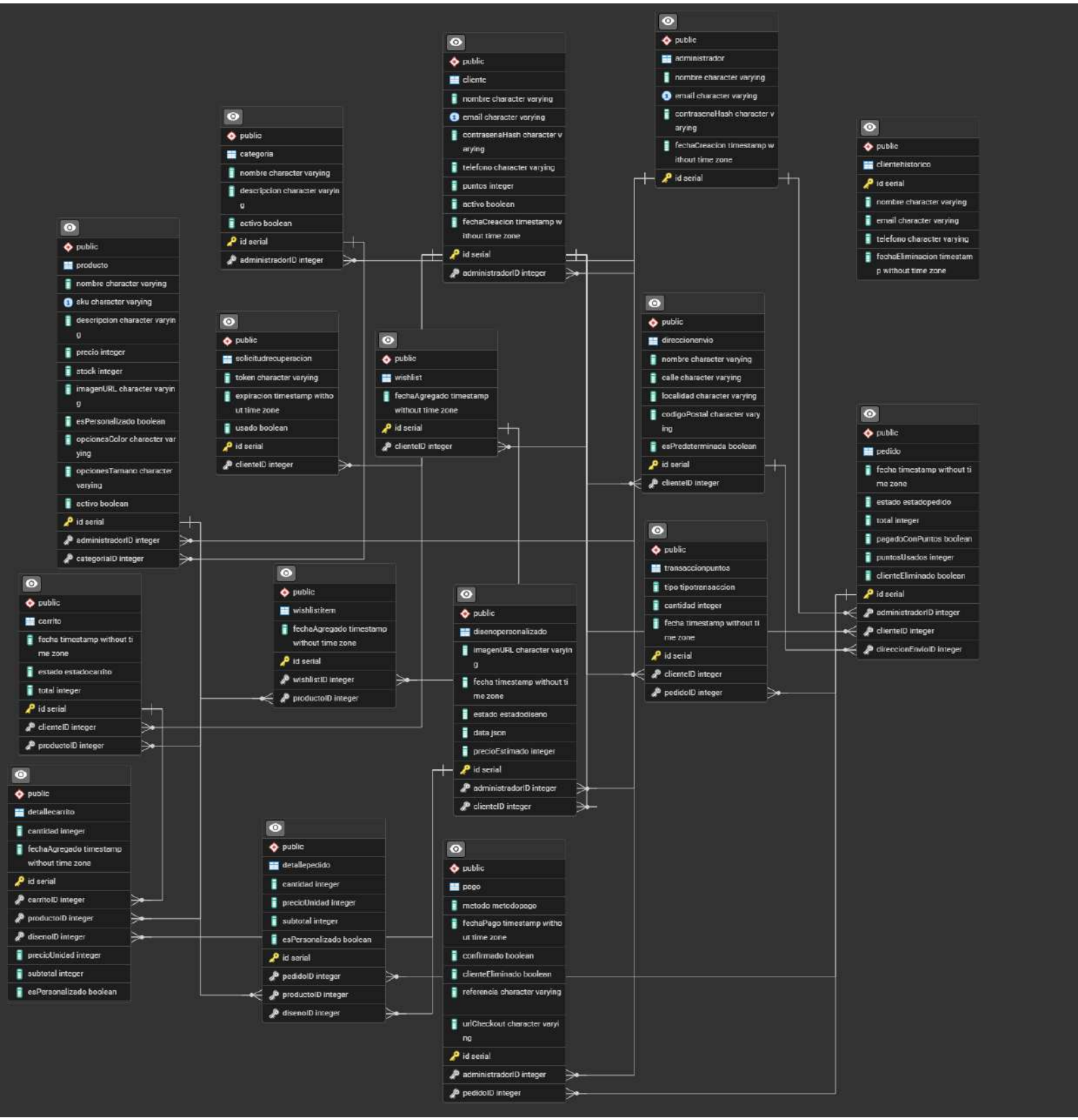
- **400 Bad Request:** Validación de datos, stock insuficiente, reglas de negocio
- **401 Unauthorized:** Credenciales inválidas, token expirado
- **403 Forbidden:** No tiene permisos para la operación
- **404 Not Found:** Recurso no encontrado
- **409 Conflict:** Email duplicado, SKU duplicado, producto ya en wishlist
- **500 Internal Server Error:** Error inesperado del servidor

---

5. DICCIONARIO de DATOS

Entidad	Atributo	Tipo	Validaciones
Cliente	email	str	Único, formato email
Cliente	telefono	str	Mínimo 7 dígitos
Cliente	contrasenaHash	str	Hash bcrypt
Producto	sku	str	Único
Producto	precio	int	Positivo
Producto	stock	int	No negativo
Pedido	subtotal	float	Calculado automáticamente
Pedido	costoEnvio	float	8900 si subtotal < 30000
Pedido	total	float	subtotal + costoEnvio
Pago	referencia	str	UUID único
TransaccionPuntos	cantidad	int	Positivo
WishlistItem	productoID + wishlistID	int	Único compuesto


DisenoPersonalizado
- id: int
- administradorID: Optional[int]
- clienteID: int
- imagenURL: str
- fecha: datetime
- estado: EstadoDiseno
- data: Optional[dict]
- precioEstimado: int
+ obtenerImagen(): str
+ calcularPrecio(): int
Revisa
Administrador
- id: int
- nombre: str
- email: str <<UNIQUE>>
- contrasenaHash: str
- fechaCreacion: datetime
+ crearCategoria()
+ crearProducto()
+ actualizarCategoria()
+ actualizarProducto()
+ eliminarCategoria()
+ eliminarProducto()
+ actualizarEstadoPedido()
TipoTransaccion
GANADOS
REDIMIDOS
EstadoPedido
POR_PAGAR
PENDIENTE
PAGADO
CANCELADO
MetodoPago
TRANSFERENCIA
NEQUI
DAVIPLATA
EFECTIVO
PUNTOS
EstadoDiseno
ENVIADO
EN_PRODUCCION
TERMINADO
EstadoCarrito
ACTIVO
CONVERTIDO
Realiza
Pertenece a
Creado por
Crea
Consulta
Supervisa
Mantiene
Cliente
- id: int
- administradorID: int
- nombre: str
- email: str <<UNIQUE>>
- contrasenaHash: str
- telefono: str
- puntos: int
- activo: bool
- fechaCreacion: datetime
+ actualizarPuntos(cantidad: int)
+ solicitarRecuperacion()
Wishlist
- id: int
- clienteID: int
- fechaAgregado: datetime
Pertenece a
contiene
WishlistItem
- id: int
- wishlistID: int
- productoID: int
- fechaAgregado: datetime
Pago
- id: int
- administradorID: Optional[int]
- pedidoID: int
- metodo: MetodoPago
- fechaPago: datetime
- confirmado: bool
- clienteEliminado: bool
- referencia: Optional[str]
- urlCheckout: Optional[str]
+ confirmarPago()
usado en
Referencia
Posee
Tiene
Administra
Gestiona
SolicitudRecuperacion
- id: int
- clienteID: int
- token: str
- expiracion: datetime
- usado: bool
+ generarToken(): str
+ validarToken(token: str): bool
Carrito
- id: int
- clienteID: int
- productoID: int
- fecha: datetime
- estado: EstadoCarrito
- total: int
+ calcularTotal(): int
DireccionEnvio
- id: int
- clienteID: int
- nombre: str
- calle: str
- localidad: str
- codigoPostal: str
- esPredeterminada: bool
+ marcarComoPredeterminada()
Revisa
Pertenece a
TransaccionPuntos
- id: int
- clienteID: int
- tipo: TipoTransaccion
- cantidad: int
- fecha: datetime
Realiza
Redimido en
Asociado a
Contiene
Supervisa
Contiene
Pertenece a
Referencia
Incluido en
Pedido
- id: int
- administradorID: Optional[int]
- clienteID: int
- direccionEnvioID: int
- fecha: datetime
- estado: EstadoPedido
- total: int
- clienteEliminado: bool
- pagadoConPuntos: bool
- puntosUsados: int
+ calcularTotal(): int
+ marcarPagado()
+ actualizarEstado(nuevoEstado: EstadoPedido)
+ enviarNotificacionEmail()
Usado en
Enviado a
DetalleCarrito
- id: int
- carritoID: int
- productoID: int
- disenoID: Optional[int]
- cantidad: int
- precioUnidad: int
- subtotal: int
- esPersonalizado: bool
- fechaAgregado: datetime
+ calcularSubtotal(): int
Producto
- id: int
- administradorID: int
- categoriaID: int
- nombre: str
- descripcion: str
- precio: int
- stock: int
- imagenURL: str
- sku: str
- activo: bool
+ actualizarStock(cantidad: int)
+ calcularPrecioFinal(): int
Referencia
Agregado en
Agregado en
Referencia
DetallePedido
- id: int
- pedidoID: int
- productoID: int
- disenoID: Optional[int]
- cantidad: int
- precioUnidad: int
- subtotal: int
- esPersonalizado: bool
+ calcularSubtotal(): int
Pertenece a
contiene
Categoria
- id: int
- administradorID: int
- nombre: str
- descripcion: str
- activo: bool
+ contarProductos(): int


public
producto
nombre character varying
sku character varying
descripcion character varying
precio integer
stock integer
imagenURL character varying
esPersonalizado boolean
opcionesColor character varying
opcionesTamano character varying
activo boolean
id serial
administradorID integer
categoriaID integer
public
categoria
nombre character varying
descripcion character varying
activo boolean
id serial
administradorID integer
public
cliente
nombre character varying
email character varying
contrasenaHash character varying
telefono character varying
puntos integer
activo boolean
fechaCreacion timestamp without time zone
id serial
administradorID integer
public
administrador
nombre character varying
email character varying
contrasenaHash character varying
fechaCreacion timestamp without time zone
id serial
public
clientehistorico
id serial
nombre character varying
email character varying
telefono character varying
fechaEliminacion timestamp without time zone
public
solicitudrecuperacion
token character varying
expiracion timestamp without time zone
usado boolean
id serial
clienteID integer
public
wishlist
fechaAgregado timestamp without time zone
id serial
clienteID integer
public
direccionenvio
nombre character varying
calle character varying
localidad character varying
codigoPostal character varying
esPredeterminada boolean
id serial
clienteID integer
public
pedido
fecha timestamp without time zone
estado estadopedido
total integer
pagadoConPuntos boolean
puntosUsados integer
clienteEliminado boolean
id serial
administradorID integer
clienteID integer
direccionEnvioID integer
public
carrito
fecha timestamp without time zone
estado estadocarrito
total integer
id serial
clienteID integer
productoID integer
public
wishlistitem
fechaAgregado timestamp without time zone
id serial
wishlistID integer
productoID integer
public
disenopersonalizado
imagenURL character varying
fecha timestamp without time zone
estado estadodiseno
data json
precioEstimado integer
id serial
administradorID integer
clienteID integer
public
transaccionpuntos
tipo tipotransaccion
cantidad integer
fecha timestamp without time zone
id serial
clienteID integer
pedidoID integer
public
detallecarrito
cantidad integer
fechaAgregado timestamp without time zone
id serial
carritoID integer
productoID integer
disenoID integer
precioUnidad integer
subtotal integer
esPersonalizado boolean
public
detallepedido
cantidad integer
precioUnidad integer
subtotal integer
esPersonalizado boolean
id serial
pedidoID integer
productoID integer
disenoID integer
public
pago
metodo metodopago
fechaPago timestamp without time zone
confirmado boolean
clienteEliminado boolean
referencia character varying
urlCheckout character varying
id serial
administradorID integer
pedidoID integer