

Análisis de Requerimientos - E-commerce Modelos 3D

1. DESCRIPCIÓN GENERAL

1.1 Objetivo del Sistema

Sistema de e-commerce para venta de modelos 3D con publicación automática en MercadoLibre y entrega automatizada mediante Google Drive.

1.2 Alcance

Versión 1.0 (MVP):

- Gestión de modelos 3D por gerente
- Carga de archivos de modelos 3D al Drive
- Publicación automática en página web propia
- Publicación automática en MercadoLibre
- Procesamiento de compras
- Entrega automática vía Google Drive (compartir archivo con email del cliente)

Versión 2.0 (Futura):

- Microservicio de atención al cliente con IA
- Respuestas automáticas a mensajes de MercadoLibre

1.3 Usuarios del Sistema

- **Gerente/Administrador:** Publica y gestiona modelos 3D, sube archivos al Drive
 - **Cliente:** Compra modelos 3D
 - **Sistema:** Automatizaciones (carga a Drive, publicación, entrega)
-

2. REQUERIMIENTOS FUNCIONALES

RF-01: Gestión de Modelos 3D

Actor: Gerente

Descripción: El gerente puede crear, editar y eliminar productos (modelos 3D). Al crear un producto, el sistema automáticamente sube el archivo del modelo 3D al Google Drive.

Datos del Producto:

- Título
- Descripción
- Precio
- Nombre del Anime
- Nombre del Personaje
- Categoría opcional (keycap, Marvel, DC, etc.)
- Imágenes del modelo (múltiples)
- Archivo del modelo 3D (.stl, .obj, .fbx, etc.)
- Estado (activo/inactivo)
- **Nota:** No hay stock, son archivos digitales con ventas ilimitadas

Flujo:

1. Gerente accede al panel de administración
2. Crea nuevo producto con todos los datos
3. Gerente selecciona archivo del modelo 3D desde su computadora
4. Sistema sube el archivo automáticamente a Google Drive
5. Sistema obtiene el ID del archivo en Drive
6. Sistema valida datos
7. Sistema guarda producto en base de datos PostgreSQL (incluyendo Drive file ID)
8. Sistema publica automáticamente en:
 - Página web propia
 - MercadoLibre (vía API)

RF-02: Publicación en Página Web

Actor: Sistema

Descripción: Cuando se crea/actualiza un producto, el sistema lo muestra automáticamente en la página web.

Páginas/Vistas necesarias:

- **Catálogo de productos:** Grid/lista con productos
- **Detalle de producto:** Información completa, imágenes, botón "Comprar"
- **Carrito de compras:** Productos seleccionados
- **Checkout:** Formulario de compra y pago

RF-03: Publicación en MercadoLibre

Actor: Sistema

Descripción: Al crear un producto, el sistema lo publica automáticamente en MercadoLibre usando la API de MercadoLibre.

Datos a sincronizar:

- Título (adaptado a límites de MercadoLibre)
- Descripción
- Precio
- Imágenes
- Stock
- Categoría (mapeo con categorías de MercadoLibre)

Consideraciones:

- Obtener credenciales OAuth de MercadoLibre
- Manejar tokens de acceso
- Webhook para notificaciones de ventas

RF-04: Procesamiento de Compras

Actor: Cliente, Sistema

Flujo de Compra (Página Web):

1. Cliente selecciona producto
2. Cliente hace clic en "Comprar"
3. Sistema verifica si el cliente está autenticado:
 - **Si NO está autenticado:**
 - Sistema muestra pantalla "Regístrate para comprar"
 - Botón: "Iniciar sesión con Google"
 - Cliente es redirigido al flujo de Google OAuth (RF-07)
 - Tras autenticarse, vuelve al producto
 - **Si está autenticado:**
 - Sistema obtiene email del usuario desde la sesión
4. Sistema redirige a pasarela de pago (MercadoPago)
5. Cliente completa pago
6. Sistema recibe confirmación de pago
7. Sistema procesa entrega (RF-05)

Flujo de Compra (MercadoLibre):

1. Cliente compra en MercadoLibre
2. MercadoLibre envía webhook a nuestro sistema
3. Sistema valida compra
4. Sistema obtiene email del cliente (desde MercadoLibre)
5. Sistema procesa entrega (RF-05)

RF-05: Entrega Automática vía Google Drive

Actor: Sistema

Descripción: Al confirmar una compra, el sistema comparte automáticamente el archivo del modelo 3D almacenado en Google Drive con el email del cliente, otorgando permisos de descarga.

Flujo:

1. Sistema recibe confirmación de compra
2. Sistema obtiene:
 - Email del cliente
 - ID del archivo del modelo 3D en Google Drive asociado al producto
3. Sistema usa Google Drive API para compartir archivo:
 - Tipo: "user"
 - Role: "reader" (solo lectura/descarga)
 - Email: email del cliente
4. Sistema guarda registro de compra en base de datos:
 - Usuario (email)
 - Producto (modelo 3D)
 - Fecha
 - Monto
 - Link de descarga (opcional)
5. Sistema envía email de confirmación al cliente con:
 - Información de compra
 - Nombre del modelo 3D adquirido
 - Instrucciones de acceso al archivo en Drive
 - Link a Google Drive (opcional)

RF-06: Gestión de Ventas

Actor: Gerente

Descripción: Panel para ver historial de ventas.

Información mostrada:

- Lista de compras
- Cliente (email)
- Producto comprado
- Fecha y hora
- Monto
- Origen (página web / MercadoLibre)
- Estado

RF-07: Autenticación con Google OAuth

Actor: Cliente, Gerente

Descripción: Sistema de autenticación usando Google OAuth para todos los usuarios (clientes y administradores).

Funcionalidades:

- Login con Google OAuth 2.0
- Registro automático al iniciar sesión por primera vez
- Sesión segura (JWT generado tras autenticación exitosa)
- Logout
- Redirección a registro/login al intentar comprar

Flujo de Autenticación:

1. Usuario hace clic en "Iniciar sesión con Google"
 2. Sistema redirige a Google OAuth
 3. Usuario autoriza con su cuenta de Google
 4. Google devuelve token de acceso y datos del usuario (email, nombre, foto)
 5. Sistema verifica si el usuario existe en la BD:
 - Si existe: Actualiza last_login y genera JWT
 - Si no existe: Crea nuevo usuario en tabla `users` y genera JWT
 6. Sistema devuelve JWT al frontend
 7. Frontend almacena JWT y redirige al usuario
-

3. REQUERIMIENTOS NO FUNCIONALES

RNF-01: Performance

- Tiempo de respuesta de API: < 2 segundos
- Tiempo de publicación en MercadoLibre: < 5 segundos
- Tiempo de compartir archivo en Drive: < 3 segundos

RNF-02: Seguridad

- Conexión HTTPS obligatoria
- Autenticación JWT para panel admin
- Validación de pagos mediante webhooks seguros
- Credenciales de APIs en variables de entorno
- No exponer URLs directas de Google Drive en frontend

RNF-03: Disponibilidad

- Sistema disponible 99% del tiempo
- Manejo de errores de APIs externas (MercadoLibre, Google Drive)
- Logs de todas las operaciones críticas

RNF-04: Escalabilidad

- Diseño preparado para agregar microservicios
- Base de datos PostgreSQL con índices optimizados
- Arquitectura limpia (separación de responsabilidades)

4. MODELO DE DATOS (PostgreSQL)

Tabla: products

```
CREATE TABLE products (  
    id SERIAL PRIMARY KEY,  
    title VARCHAR(255) NOT NULL,  
    description TEXT,  
    price DECIMAL(10, 2) NOT NULL,  
    anime_name VARCHAR(255) NOT NULL, -- Nombre del anime  
    character_name VARCHAR(255) NOT NULL, -- Nombre del personaje  
    optional_category VARCHAR(100), -- keycap, Marvel, DC, etc. (opcional)  
    google_drive_file_id VARCHAR(255) NOT NULL, -- ID del modelo 3D en Drive  
    google_drive_file_url TEXT, -- URL del archivo en Drive  
    file_name VARCHAR(255), -- Nombre original del archivo (ej:  
    naruto_uzumaki.stl)  
    file_format VARCHAR(20), -- Formato del archivo (stl, obj, fbx, etc.)
```

```

    mercadolibre_id VARCHAR(100) UNIQUE,
    status VARCHAR(20) DEFAULT 'active', -- active, inactive
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
-- Nota: No hay campo de stock porque son archivos digitales (ventas
ilimitadas)

```

Tabla: product_images

```

CREATE TABLE product_images (
    id SERIAL PRIMARY KEY,
    product_id INTEGER REFERENCES products(id) ON DELETE CASCADE,
    image_url TEXT NOT NULL,
    display_order INTEGER DEFAULT 0,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

```

Tabla: purchases

```

CREATE TABLE purchases (
    id SERIAL PRIMARY KEY,
    product_id INTEGER REFERENCES products(id),
    user_id INTEGER REFERENCES users(id), -- Usuario que compró
    customer_email VARCHAR(255) NOT NULL, -- Email para entrega (desde
Google account)
    amount DECIMAL(10, 2) NOT NULL,
    payment_id VARCHAR(255) UNIQUE, -- ID de MercadoPago/Stripe
    source VARCHAR(50) NOT NULL, -- 'web', 'mercadolibre'
    status VARCHAR(50) DEFAULT 'completed', -- pending, completed, failed
    mercadolibre_order_id VARCHAR(100),
    drive_shared BOOLEAN DEFAULT false,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

```

Tabla: users

```

CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    google_id VARCHAR(255) UNIQUE NOT NULL, -- ID de Google OAuth
    email VARCHAR(255) UNIQUE NOT NULL,
    name VARCHAR(255),
    picture_url TEXT, -- URL de la foto de perfil de Google

```

```
role VARCHAR(50) DEFAULT 'customer', -- customer, admin
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
last_login TIMESTAMP
);
-- Todos los usuarios (clientes y admins) usan Google OAuth para
autenticación
```

Índices

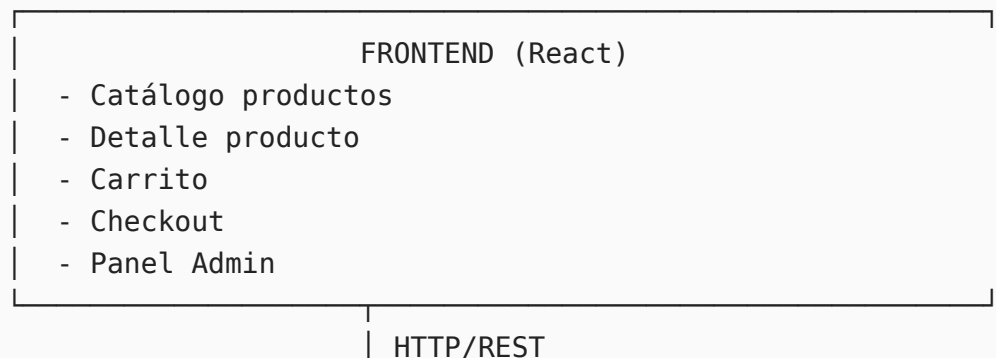
```
CREATE INDEX idx_products_status ON products(status);
CREATE INDEX idx_products_anime ON products(anime_name);
CREATE INDEX idx_products_character ON products(character_name);
CREATE INDEX idx_products_optional_category ON products(optional_category);
CREATE INDEX idx_purchases_email ON purchases(customer_email);
CREATE INDEX idx_purchases_created ON purchases(created_at DESC);
CREATE INDEX idx_mercadolibre_id ON products(mercadolibre_id);
CREATE INDEX idx_purchases_user_id ON purchases(user_id);
```

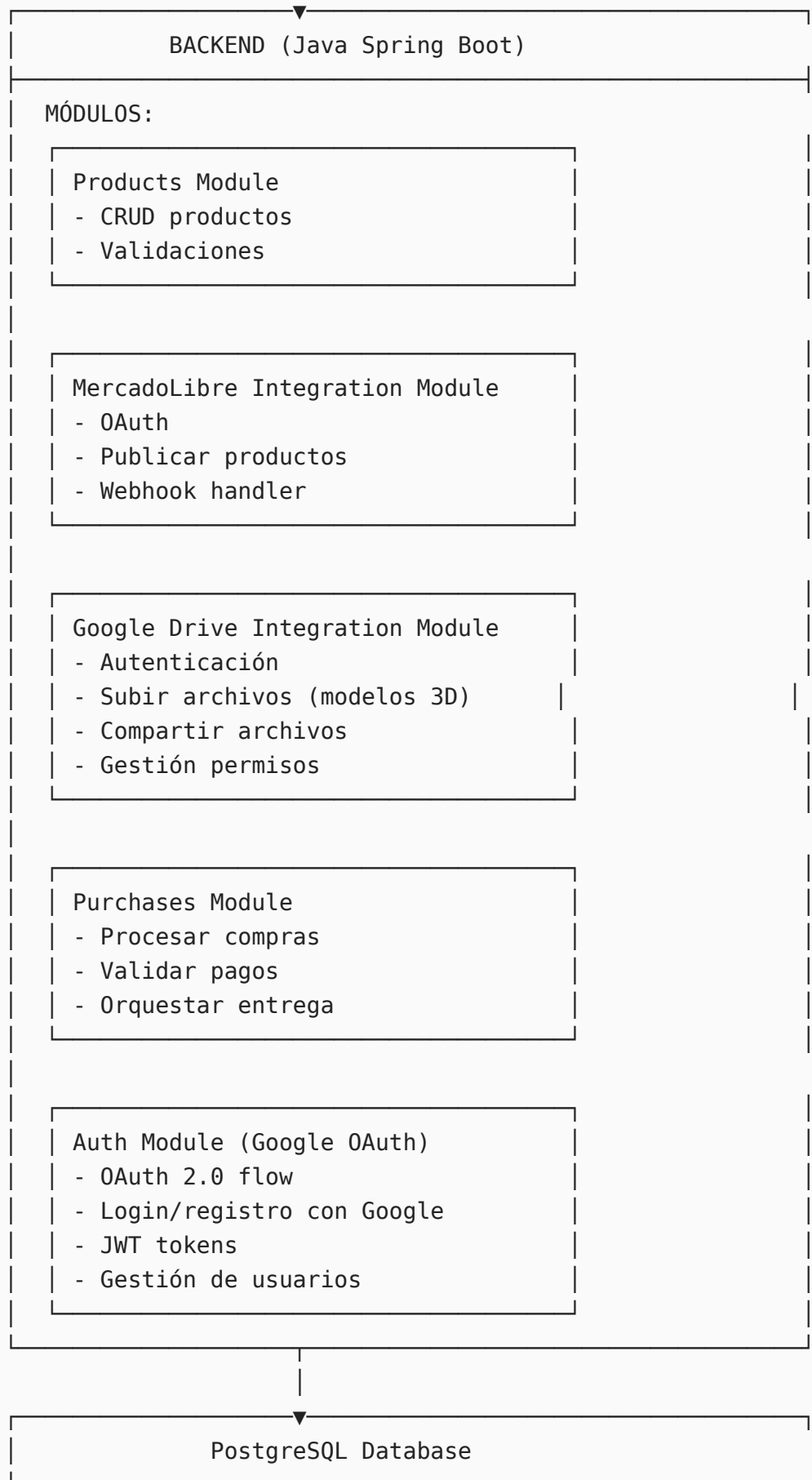
5. ARQUITECTURA DEL SISTEMA

5.1 Stack Tecnológico

- **Backend:** Java Spring Boot
- **Base de datos:** PostgreSQL
- **Frontend:** React / Angular / Vue (por definir)
- **Integraciones:**
 - Google Drive API
 - MercadoLibre API
 - Pasarela de pagos (MercadoPago recomendado)

5.2 Arquitectura Versión 1.0 (Monolito Modular)





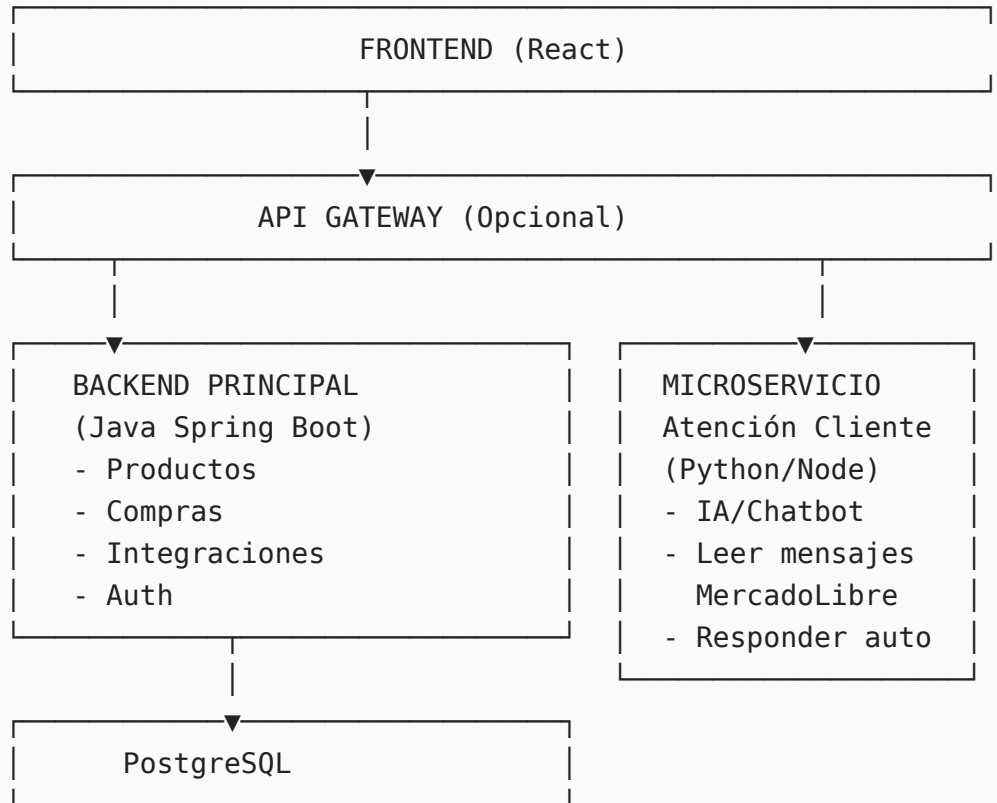
INTEGRACIONES EXTERNAS:

Google Drive
API

MercadoLibre
API

MercadoPago
(Pagos)

5.3 Arquitectura Versión 2.0 (Con Microservicio)



6. CASOS DE USO DETALLADOS

CU-01: Publicar Producto

Actores: Gerente, Sistema

Precondiciones:

- Gerente autenticado en panel admin
- Archivo del modelo 3D disponible en la computadora del gerente

Flujo Principal:

1. Gerente accede a "Nuevo Producto"

2. Gerente completa formulario:
 - Título: "Figura Naruto Uzumaki"
 - Descripción: "Modelo 3D de alta calidad de Naruto Uzumaki, listo para imprimir..."
 - Precio: \$29.99
 - Anime: "Naruto"
 - Personaje: "Naruto Uzumaki"
 - Categoría opcional: [vacío] o "keycap" o "Marvel", etc.
 - Imágenes: [sube 3 imágenes del render del modelo]
 - Archivo 3D: [selecciona archivo naruto_uzumaki.stl]
3. Gerente hace clic en "Publicar"
4. Sistema valida datos
5. Sistema sube archivo .stl a Google Drive
6. Sistema obtiene ID del archivo en Drive: "1abc123xyz..."
7. Sistema guarda producto en PostgreSQL (con Drive file ID)
8. Sistema publica en MercadoLibre:
 - Llama API de MercadoLibre
 - Obtiene ID de publicación
 - Guarda `mercadolibre_id` en BD
9. Sistema muestra mensaje: "Producto publicado exitosamente"
10. Producto aparece en:
 - Página web (catálogo)
 - MercadoLibre

Nota: No hay límite de stock porque es un archivo digital.

Flujos Alternativos:

- 4a. Datos inválidos: Sistema muestra errores de validación
- 6a. Error en MercadoLibre: Sistema guarda producto pero marca error, permite reintentar publicación

CU-02: Compra en Página Web

Actores: Cliente, Sistema

Precondiciones:

- Producto activo y disponible
- Pasarela de pagos configurada

Flujo Principal:

1. Cliente navega catálogo en página web
2. Cliente hace clic en producto "Figura Naruto Uzumaki"
3. Sistema muestra detalle del producto
4. Cliente hace clic en "Comprar"
5. Sistema verifica si el cliente está autenticado:
 - Cliente NO está autenticado
 - Sistema muestra pantalla "Regístrate para comprar"
 - Mensaje: "Inicia sesión con Google para continuar"
 - Botón: "Iniciar sesión con Google"
6. Cliente hace clic en "Iniciar sesión con Google"
7. Sistema redirige a Google OAuth
8. Cliente autoriza con su cuenta de Google: "cliente@gmail.com"
9. Google devuelve datos: email, nombre, foto
10. Sistema crea usuario nuevo en tabla `users` (o actualiza si existe):
 - `google_id`: "1234567890"
 - `email`: "cliente@gmail.com"
 - `name`: "Juan Pérez"
 - `picture_url`: "https://..."
 - `role`: "customer"
11. Sistema genera JWT y lo devuelve al frontend
12. Sistema redirige de vuelta al producto
13. Cliente hace clic nuevamente en "Comprar" (ahora autenticado)
14. Sistema redirige a MercadoPago con:
 - Producto
 - Precio
 - Email del usuario autenticado
15. Cliente completa pago en MercadoPago
16. MercadoPago envía webhook a nuestro backend
17. Sistema valida webhook (firma, autenticidad)
18. Sistema crea registro en tabla `purchases` :
 - `product_id`: 1
 - `user_id`: 5
 - `customer_email`: "cliente@gmail.com"
 - `amount`: 29.99
 - `payment_id`: "MP123456"

- source: "web"
- status: "completed"

19. Sistema llama Google Drive API:

- Obtiene archivo ID del modelo 3D
- Comparte archivo con "cliente@gmail.com"
- Rol: "reader"

20. Sistema marca `drive_shared = true` en purchase

21. Sistema envía email de confirmación al cliente:

- "Gracias por tu compra de Naruto Uzumaki"
- "Accede al archivo en tu Google Drive"
- Link opcional al Drive

22. Cliente recibe email

23. Cliente accede a Google Drive y descarga el archivo del modelo 3D

Flujos Alternativos:

- 10a. Webhook inválido: Sistema rechaza, no procesa compra
- 12a. Error al compartir Drive: Sistema reintenta, registra error, alerta admin

CU-03: Compra en MercadoLibre

Actores: Cliente, Sistema, MercadoLibre

Precondiciones:

- Producto publicado en MercadoLibre

Flujo Principal:

1. Cliente busca producto en MercadoLibre
2. Cliente compra producto en MercadoLibre
3. MercadoLibre procesa pago
4. MercadoLibre envía webhook a nuestro sistema:
 - Tipo: "order_created"
 - Order ID
 - Buyer email
 - Product ID (MercadoLibre)
5. Sistema recibe webhook
6. Sistema valida webhook
7. Sistema busca producto por `mercadolibre_id`
8. Sistema crea registro en `purchases` :

- customer_email: (desde webhook)
- product_id
- amount
- source: "mercadolibre"
- mercadolibre_order_id

9. Sistema ejecuta entrega (compartir Drive)

10. Sistema actualiza stock si es necesario

11. Cliente recibe acceso al archivo

Flujos Alternativos:

- 6a. Webhook inválido: Rechazar
- 7a. Producto no encontrado: Registrar error, alertar admin

7. INTERFACES DE USUARIO

7.1 Panel de Administración

Pantalla: Login / Registro

Elementos:

- Logo o imagen
- Título: "Iniciar sesión"
- Botón grande: "Continuar con Google" (con logo de Google)
- Texto pequeño: "Al continuar, aceptas los términos y condiciones"

Nota: No hay formulario de usuario/contraseña. Todo se maneja con Google OAuth.

Pantalla: Dashboard

Elementos:

- Menú lateral:
 - Productos
 - Ventas
 - Configuración
- Tarjetas resumen:
 - Total productos
 - Ventas del mes

- Ventas pendientes

Pantalla: Lista de Productos

Elementos:

- Botón: "+ Nuevo Producto"
- Tabla de productos:
 - Imagen miniatura
 - Título
 - Anime
 - Personaje
 - Categoría opcional
 - Precio
 - Estado (activo/inactivo)
 - Publicado en ML (sí/no)
 - Acciones (Editar, Eliminar)
- Filtros: Estado, Anime, Categoría opcional
- Búsqueda por título, anime o personaje

Pantalla: Crear/Editar Producto

Elementos:

- Campo: Título (texto) - Ej: "Figura Naruto Uzumaki"
- Campo: Descripción (textarea)
- Campo: Precio (número)
- Campo: Anime (texto con autocompletado) - Obligatorio
- Campo: Personaje (texto) - Obligatorio
- Campo: Categoría opcional (select): [Ninguna, Keycap, Marvel, DC, Otros]
- Upload: Archivo del Modelo 3D (.stl, .obj, .fbx, etc.) - El sistema lo sube automáticamente a Drive
- Upload: Imágenes del modelo (múltiples - renders o previews)
- Toggle: Estado (activo/inactivo)
- Toggle: Publicar en MercadoLibre
- Botón: "Guardar"
- Botón: "Cancelar"

Nota: No hay campo de stock (son archivos digitales con ventas ilimitadas)

Pantalla: Ventas

Elementos:

- Tabla de ventas:
 - ID compra
 - Fecha
 - Cliente (email)
 - Producto
 - Monto
 - Origen (web/ML)
 - Estado Drive (compartido/pendiente)
- Filtros: Fecha, Origen
- Búsqueda por email

7.2 Página Web Pública

Pantalla: Catálogo (Home)

Elementos:

- Header:
 - Logo
 - Barra búsqueda (por título, anime o personaje)
 - Botón: "Iniciar sesión" (si no está autenticado)
 - Avatar del usuario (si está autenticado) con menú dropdown:
 - Mi cuenta
 - Mis compras
 - Cerrar sesión
- Grid de productos (figuras de anime):
 - Imagen del render del modelo
 - Título
 - Anime - Personaje (ej: "Naruto - Naruto Uzumaki")
 - Categoría opcional (si aplica)
 - Precio
 - Botón "Ver más"
- Filtros laterales:
 - **Anime** (lista con autocompletado)
 - **Personaje** (filtro de texto)

- **Categoría opcional** (Todos, Keycap, Marvel, DC, Otros)
- Rango precio

Pantalla: Detalle de Producto

Elementos:

- Galería de imágenes del modelo (carousel con renders)
- Título del modelo 3D
- Precio (destacado)
- Descripción completa del modelo
- Botón: "Comprar ahora" (grande, destacado)
- Información adicional:
 - Entrega: "Acceso inmediato vía email"
 - Tipo: "Modelo 3D"
 - Formato: "STL / OBJ / FBX" (según corresponda)

Modal/Pantalla: Regístrate para Comprar (si no está autenticado)

Elementos:

- Título: "Inicia sesión para comprar"
- Mensaje: "Necesitas una cuenta de Google para realizar la compra. El archivo se compartirá con tu email."
- Botón grande: "Continuar con Google" (con logo de Google)
- Link: Términos y condiciones
- Botón: "Cancelar"

Pantalla: Checkout (si está autenticado)

Elementos:

- Resumen producto:
 - Imagen del modelo
 - Título
 - Anime - Personaje
 - Precio
- Información del usuario:
 - Email: [email del usuario desde Google]
 - Nombre: [nombre del usuario]
- Mensaje: "El archivo será compartido con tu cuenta de Google"

- Botón: "Proceder al pago"
 - Link: Términos y condiciones
-

8. INTEGRACIONES

8.1 Google OAuth 2.0 (Autenticación)

Configuración necesaria:

- Crear proyecto en Google Cloud Console
- Habilitar Google OAuth 2.0 API
- Crear credenciales OAuth 2.0 (Client ID y Client Secret)
- Configurar redirect URI (ej: <https://tudominio.com/auth/google/callback>)
- Configurar consent screen

Flujo de autenticación:

1. Usuario hace clic en "Iniciar sesión con Google"
2. Frontend redirige a:

```
https://accounts.google.com/o/oauth2/v2/auth?
client_id=YOUR_CLIENT_ID&
redirect_uri=YOUR_REDIRECT_URI&
response_type=code&
scope=openid email profile&
access_type=offline
```

3. Usuario autoriza la aplicación
4. Google redirige a `redirect_uri` con código de autorización
5. Backend intercambia código por access token:

```
POST https://oauth2.googleapis.com/token
{
  "code": "auth_code",
  "client_id": "YOUR_CLIENT_ID",
  "client_secret": "YOUR_CLIENT_SECRET",
  "redirect_uri": "YOUR_REDIRECT_URI",
  "grant_type": "authorization_code"
}
```

6. Google devuelve:
 - `access_token`
 - `id_token` (contiene email, nombre, foto)

- refresh_token

7. Backend decodifica id_token para obtener información del usuario

8. Backend crea/actualiza usuario en BD y genera JWT

Datos obtenidos de Google:

- Email (para compartir archivos en Drive)
- Nombre
- Foto de perfil
- Google ID (para identificar usuario)

8.2 Google Drive API

Configuración necesaria:

- Crear proyecto en Google Cloud Console
- Habilitar Google Drive API
- Crear credenciales OAuth 2.0
- Descargar archivo `credentials.json`

Operaciones:

1. Autenticación:

- Usar Service Account para acceso servidor-a-servidor
- O OAuth 2.0 si requiere autorización de usuario admin

2. Subir archivo del modelo 3D:

```
// Pseudocódigo
DriveService drive = getDriveService();
File fileMetadata = new File();
fileMetadata.setName("silla_moderna.stl");
fileMetadata.setMimeType("application/octet-stream");

java.io.File filePath = new java.io.File("/temp/uploaded_file.stl");
FileContent mediaContent = new FileContent("application/octet-stream",
filePath);

File file = drive.files().create(fileMetadata, mediaContent)
    .setFields("id, webViewLink")
    .execute();

String fileId = file.getId(); // Guardar este ID en la BD
```

3. Compartir archivo:

```
// Pseudocódigo
DriveService drive = getDriveService();
Permission permission = new Permission()
    .setType("user")
    .setRole("reader")
    .setEmailAddress(customerEmail);

drive.permissions()
    .create(fileId, permission)
    .setSendNotificationEmail(false)
    .execute();
```

8.2 MercadoLibre API

Configuración necesaria:

- Crear aplicación en MercadoLibre Developers
- Obtener `client_id` y `client_secret`
- Implementar OAuth 2.0 para obtener access token

Operaciones:

1. Autenticación OAuth:

- Redirect user a MercadoLibre para autorizar
- Recibir código de autorización
- Intercambiar código por access token
- Guardar access token y refresh token

2. Publicar producto:

```
// POST https://api.mercadolibre.com/items
{
  "title": "Figura Naruto Uzumaki - Modelo 3D STL",
  "category_id": "MLA1953",
  "price": 29.99,
  "currency_id": "ARS",
  "available_quantity": 9999, // Número alto porque es digital (sin límite real)
  "buying_mode": "buy_it_now",
  "listing_type_id": "gold_special",
  "condition": "new",
  "description": "Modelo 3D de Naruto Uzumaki del anime Naruto. Archivo STL"
```

```
    listo para imprimir...",
    "pictures": [
      {"source": "https://..."}
    ]
  }
}
```

3. Webhook de notificaciones:

- Configurar URL webhook en MercadoLibre
- Recibir notificaciones de:
 - Nuevas ventas
 - Preguntas
 - Mensajes

Endpoint webhook:

```
POST /api/webhooks/mercadolibre
```

8.3 Pasarela de Pagos (MercadoPago)

Configuración:

- Crear cuenta en MercadoPago
- Obtener credenciales (public_key, access_token)
- Configurar webhooks

Flujo de pago:

1. Usuario autenticado hace clic en "Comprar"
2. Frontend llama a backend con JWT
3. Backend obtiene email del usuario autenticado
4. Backend genera preferencia en MercadoPago:

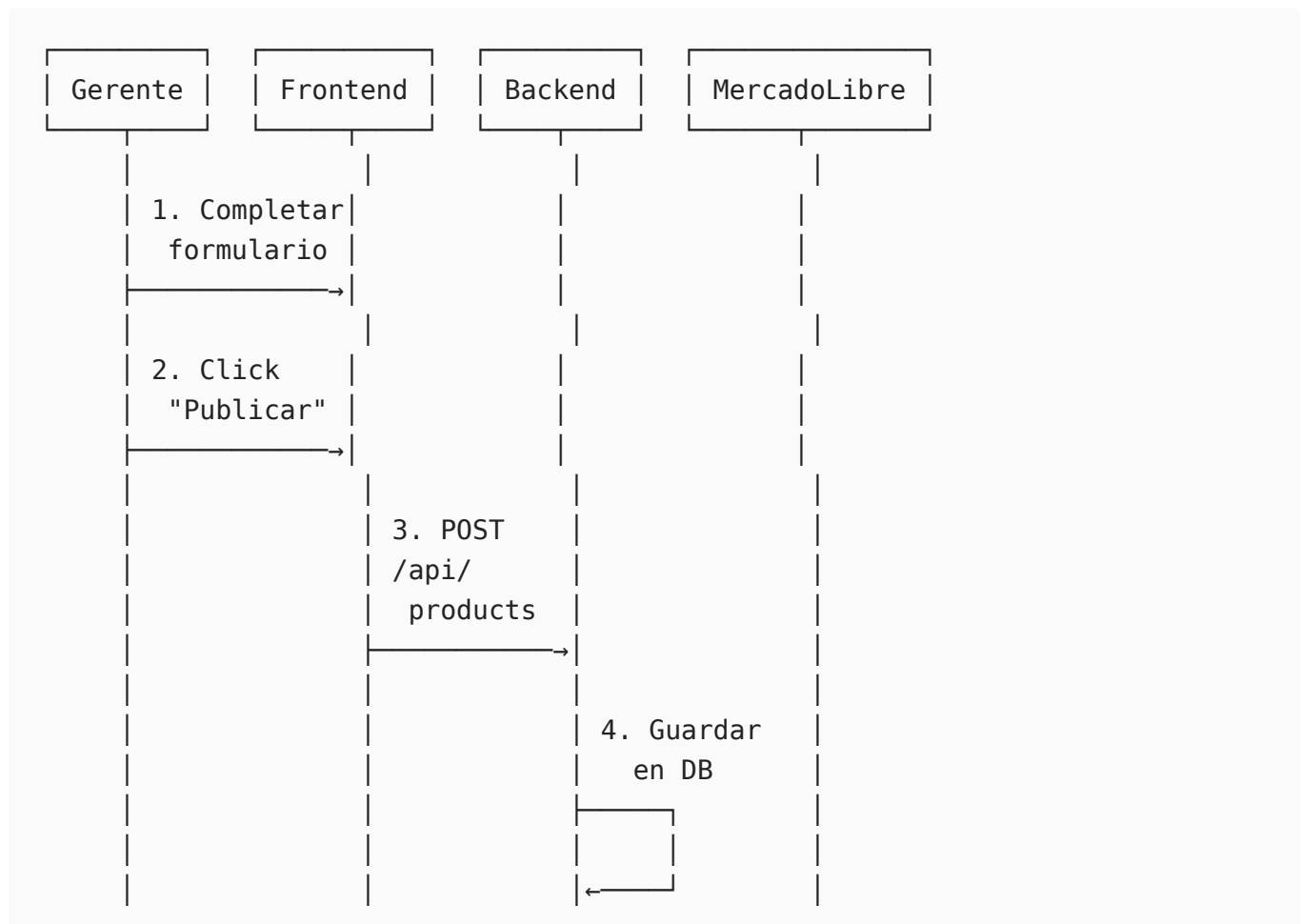
```
// POST https://api.mercadopago.com/checkout/preferences
{
  "items": [
    {
      "title": "Figura Naruto Uzumaki - Modelo 3D",
      "quantity": 1,
      "unit_price": 29.99
    }
  ],
  "payer": {
```

```
"email": "cliente@gmail.com" // Email del usuario autenticado con Google
},
"back_urls": {
  "success": "https://tudominio.com/payment/success",
  "failure": "https://tudominio.com/payment/failure",
  "pending": "https://tudominio.com/payment/pending"
},
"notification_url": "https://tudominio.com/api/webhooks/mercadopago"
}
```

3. MercadoPago devuelve `init_point` (URL de pago)
4. Frontend redirige a `init_point`
5. Cliente paga
6. MercadoPago envía webhook de confirmación
7. Backend procesa entrega

9. DIAGRAMAS DE SECUENCIA

Diagrama 1: Publicar Producto



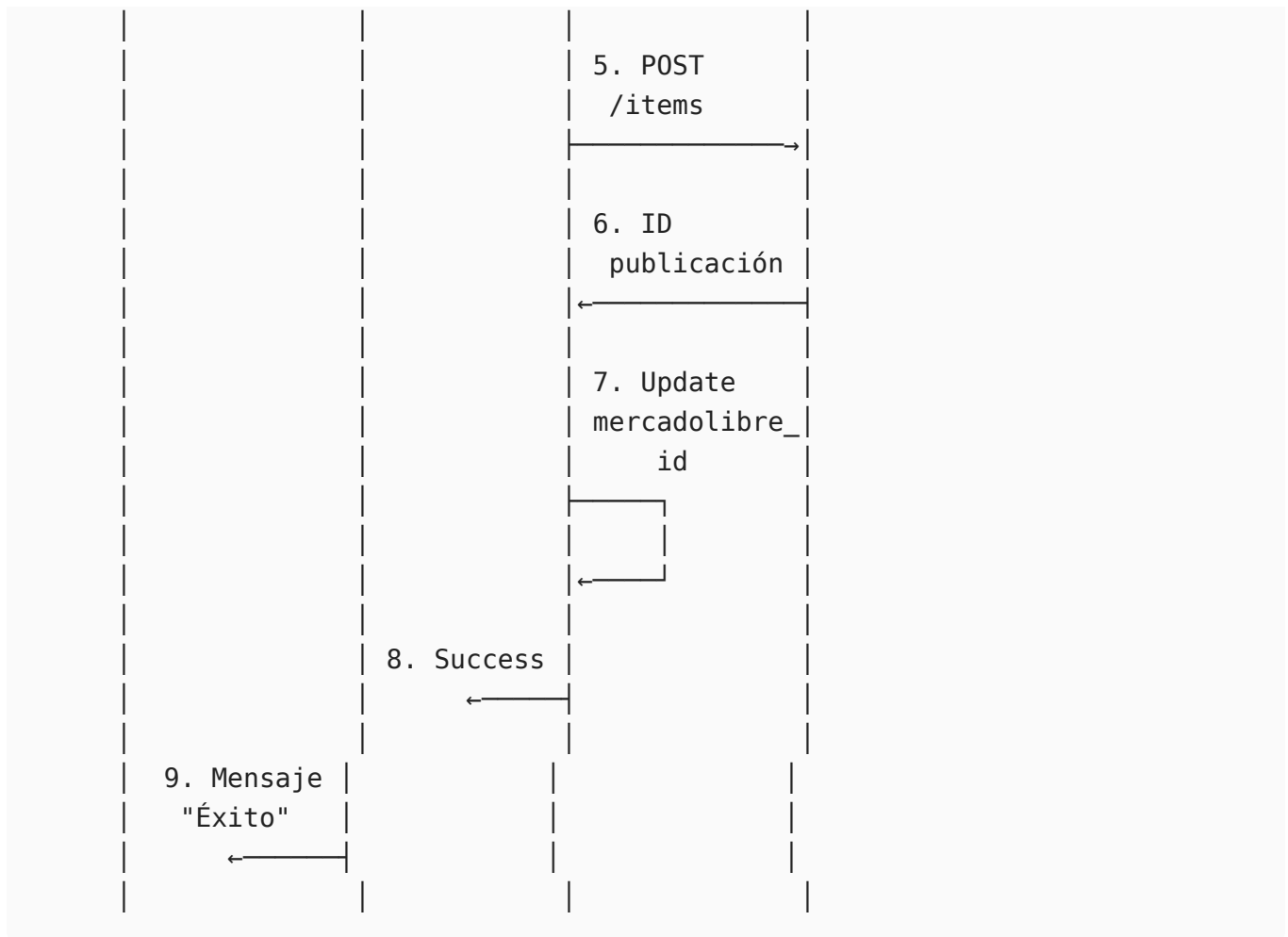
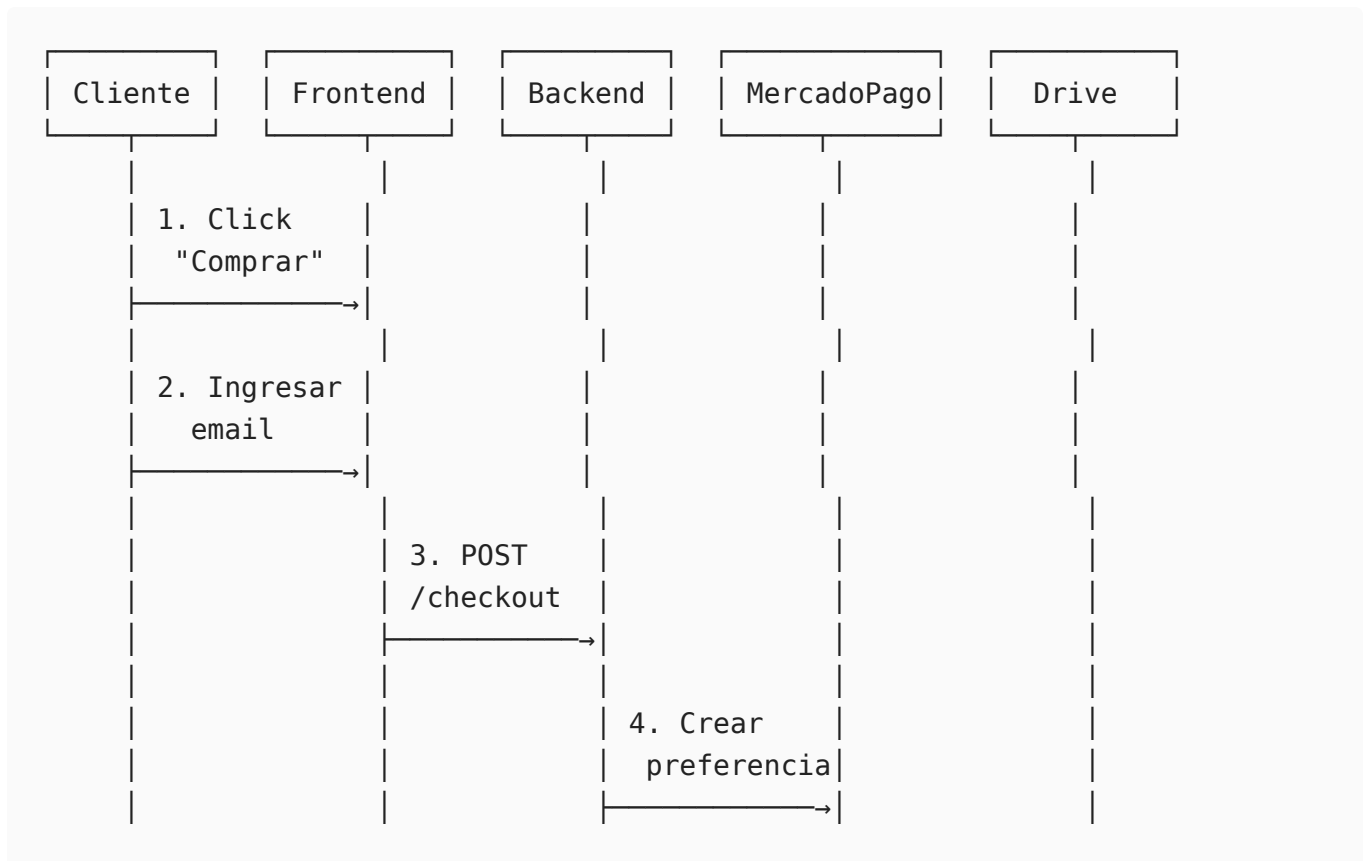
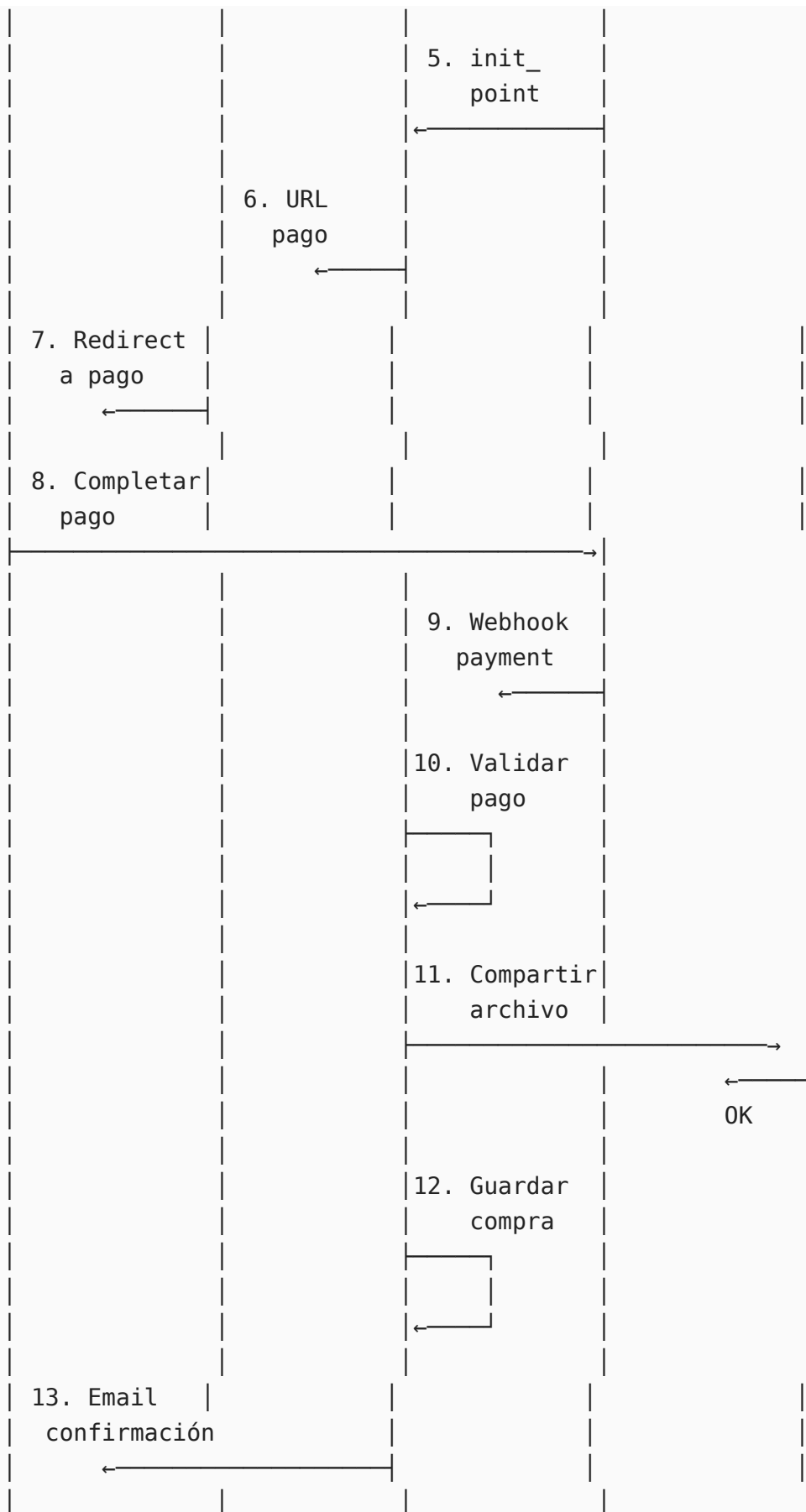


Diagrama 2: Compra en Página Web





10. ENDPOINTS API (Backend)

Autenticación (Google OAuth)

GET	/api/auth/google	# Inicia flujo OAuth (redirige a Google)
GET	/api/auth/google/callback	# Callback de Google OAuth
POST	/api/auth/logout	# Cerrar sesión
GET	/api/auth/me	# Obtener usuario actual (requiere JWT)
POST	/api/auth/refresh	# Renovar JWT

Productos

GET	/api/products	# Listar todos (público) # Query params: ? anime=Naruto&character=Naruto&category=keycap&price_min=10&price_max=100
GET	/api/products/:id	# Detalle (público)
GET	/api/products/anime/list	# Listar animes disponibles (público)
POST	/api/products	# Crear (admin, requiere JWT)
PUT	/api/products/:id	# Actualizar (admin, requiere JWT)
DELETE	/api/products/:id	# Eliminar (admin, requiere JWT)
POST	/api/products/:id/publish-ml	# Republicar en ML (admin, requiere JWT)

Compras

POST	/api/checkout	# Iniciar compra (requiere JWT - usuario autenticado)
GET	/api/purchases	# Listar todas las compras (admin, requiere JWT)
GET	/api/purchases/my	# Listar mis compras (usuario, requiere JWT)
GET	/api/purchases/:id	# Detalle compra (admin o propietario, requiere JWT)

Webhooks

POST	/api/webhooks/mercadopago	# Notificaciones pagos
POST	/api/webhooks/mercadolibre	# Notificaciones ML

Imágenes

POST /api/images/upload

Subir imagen (admin)

11. VARIABLES DE ENTORNO

```
# Database
DATABASE_URL=postgresql://user:password@localhost:5432/ecommerce
DATABASE_USERNAME=ecommerce_user
DATABASE_PASSWORD=supersecret

# JWT
JWT_SECRET=your-jwt-secret-key
JWT_EXPIRATION=86400000

# Google OAuth 2.0 (Autenticación)
GOOGLE_OAUTH_CLIENT_ID=your-oauth-client-id
GOOGLE_OAUTH_CLIENT_SECRET=your-oauth-client-secret
GOOGLE_OAUTH_REDIRECT_URI=http://localhost:8080/api/auth/google/callback

# Google Drive
GOOGLE_DRIVE_CLIENT_ID=your-client-id
GOOGLE_DRIVE_CLIENT_SECRET=your-client-secret
GOOGLE_DRIVE_REDIRECT_URI=http://localhost:8080/oauth2callback
GOOGLE_DRIVE_CREDENTIALS_FILE=/path/to/credentials.json

# MercadoLibre
MERCADOLIBRE_CLIENT_ID=your-app-id
MERCADOLIBRE_CLIENT_SECRET=your-secret
MERCADOLIBRE_REDIRECT_URI=http://localhost:8080/ml-callback
MERCADOLIBRE_ACCESS_TOKEN=your-access-token
MERCADOLIBRE_REFRESH_TOKEN=your-refresh-token

# MercadoPago
MERCADOPAGO_ACCESS_TOKEN=your-access-token
MERCADOPAGO_PUBLIC_KEY=your-public-key

# Email (opcional, para enviar confirmaciones)
SMTP_HOST=smtp.gmail.com
SMTP_PORT=587
SMTP_USER=your-email@gmail.com
SMTP_PASSWORD=your-app-password

# Application
```

```
PORT=8080
FRONTEND_URL=http://localhost:3000
```

12. TAREAS DE IMPLEMENTACIÓN

Fase 1: Setup Inicial

- ☐ Configurar proyecto Spring Boot
- ☐ Configurar PostgreSQL
- ☐ Crear esquema de base de datos
- ☐ Configurar variables de entorno
- ☐ Setup proyecto frontend

Fase 2: Módulo de Autenticación

- ☐ Implementar entidad User
- ☐ Crear AuthController
- ☐ Implementar JWT
- ☐ Crear pantallas login/logout

Fase 3: Módulo de Productos

- ☐ Implementar entidad Product
- ☐ CRUD de productos (backend)
- ☐ Subir imágenes
- ☐ Pantallas admin: lista y formulario
- ☐ Catálogo público (frontend)
- ☐ Detalle de producto (frontend)

Fase 4: Integración Google Drive

- ☐ Configurar Google Cloud project
- ☐ Implementar autenticación OAuth
- ☐ Crear servicio para compartir archivos
- ☐ Probar compartir archivo

Fase 5: Integración MercadoLibre

- ☐ Configurar app MercadoLibre

- ☐ Implementar OAuth
- ☐ Crear servicio publicación
- ☐ Implementar webhook handler
- ☐ Probar flujo completo

Fase 6: Módulo de Compras

- ☐ Implementar entidad Purchase
- ☐ Integrar MercadoPago
- ☐ Crear flujo checkout (frontend)
- ☐ Webhook handler MercadoPago
- ☐ Orquestación de entrega automática
- ☐ Pantalla de ventas (admin)

Fase 7: Testing y Deploy

- ☐ Tests unitarios
- ☐ Tests de integración
- ☐ Configurar CI/CD
- ☐ Deploy en servidor

Fase 8: Versión 2.0 (Futuro)

- ☐ Diseñar microservicio atención cliente
- ☐ Implementar IA/chatbot
- ☐ Integrar con mensajes MercadoLibre
- ☐ Deploy microservicio

13. CONSIDERACIONES ADICIONALES

13.1 Seguridad

- Implementar rate limiting en APIs
- Validar todos los webhooks con firma/token
- Nunca exponer credenciales en frontend
- Sanitizar inputs de usuario
- Implementar CORS correctamente

13.2 Manejo de Errores

- Logs centralizados (ELK, CloudWatch, etc.)
- Alertas para errores críticos:
 - Fallo al compartir Drive
 - Fallo al publicar en MercadoLibre
 - Pagos no procesados
- Sistema de reintentos para operaciones críticas

13.3 Testing

- Unit tests para servicios
- Integration tests para endpoints
- Tests E2E para flujos críticos
- Tests de webhooks con mocks

13.4 Monitoreo

- Métricas de ventas
- Tasa de éxito de entrega (Drive)
- Errores de integración
- Uptime del sistema

14. GLOSARIO

- **Modelo 3D:** Archivo digital tridimensional (.stl, .obj, .fbx, etc.) que representa un objeto 3D
- **Producto:** En este sistema, se refiere a un modelo 3D disponible para venta
- **Google Drive File ID:** Identificador único del archivo en Google Drive
- **MercadoLibre Item ID:** ID de publicación en MercadoLibre
- **Webhook:** Notificación HTTP automática de eventos
- **OAuth:** Protocolo de autenticación y autorización
- **JWT:** JSON Web Token para sesiones

15. APROBACIONES Y CAMBIOS

Versión	Fecha	Autor	Cambios
1.0	2025-12-05	Sistema	Documento inicial

Versión	Fecha	Autor	Cambios
2.0	2025-12-05	Sistema	Actualización mayor: <ul style="list-style-type: none">- Cambio de productos genéricos a figuras de anime- Eliminación de stock (archivos digitales)- Implementación de Google OAuth para autenticación- Nuevos campos: anime_name, character_name, optional_category- Pantalla de registro/login con Google- Filtros por anime y personaje

Siguiente paso: Revisar y aprobar este documento antes de comenzar el diseño de arquitectura e interfaces.