Manual Técnico del Sistema "G. L. LEATHER DESIGNS"

Versión Base (Fase Parcialmente Entregada)

Este manual técnico proporciona una guía esencial para comprender la arquitectura, la implementación y el funcionamiento interno del sistema "G. L. LEATHER DESIGNS" tal como se entrega en esta fase parcial. Sirve como la documentación base que facilitará el mantenimiento, la depuración y la extensión de funcionalidades en futuras etapas del proyecto.

Arquitectura del Sistema

[Aquí podrías insertar un diagrama de alto nivel que muestre la arquitectura cliente-servidor, con el frontend, backend y la base de datos MySQL. Debe reflejar la arquitectura *actual* del proyecto.]

Tecnologías Utilizadas

- **Frontend:**
  - Next.js: Versión 14.2.3 - Framework de React para aplicaciones web.
  - React: Versión 18.2.0 - Librería para construir interfaces de usuario.
  - TypeScript: Versión 5.4.5 - Superset de JavaScript con tipado estático.
  - Tailwind CSS: Versión 3.4.3 - Framework CSS de utilidad para diseño rápido.
- **Backend:**
  - Node.js: Versión v22.16.0 - Entorno de ejecución para JavaScript en el servidor.
  - Express.js: Versión 4.19.2 - Framework web rápido y minimalista para Node.js.
  - TypeScript: Versión 5.4.5 - Superset de JavaScript con tipado estático.
  - mysql2/promise: Versión 3.9.4 - Conector MySQL para Node.js con soporte para Promesas.
  - jsonwebtoken: Versión 9.0.2 - Para generación y verificación de JWT.
  - bcryptjs: Versión 2.4.3 - Para hashing y comparación de contraseñas.
  - multer: Versión 1.4.5-lts.1- Middleware para manejo de multipart/form-data (subida de archivos).

    - cors: Versión 2.8.5 - Middleware para habilitar Cross-Origin Resource Sharing.
    - dotenv: Versión 16.4.5 - Para cargar variables de entorno desde un archivo .env.
- **Base de Datos:**
    - MySQL: Versión 8.0.36 - Sistema de gestión de base de datos relacional.

Estructura del Proyecto

El proyecto está organizado en dos directorios principales para el frontend y el backend, y un directorio para los scripts SQL:

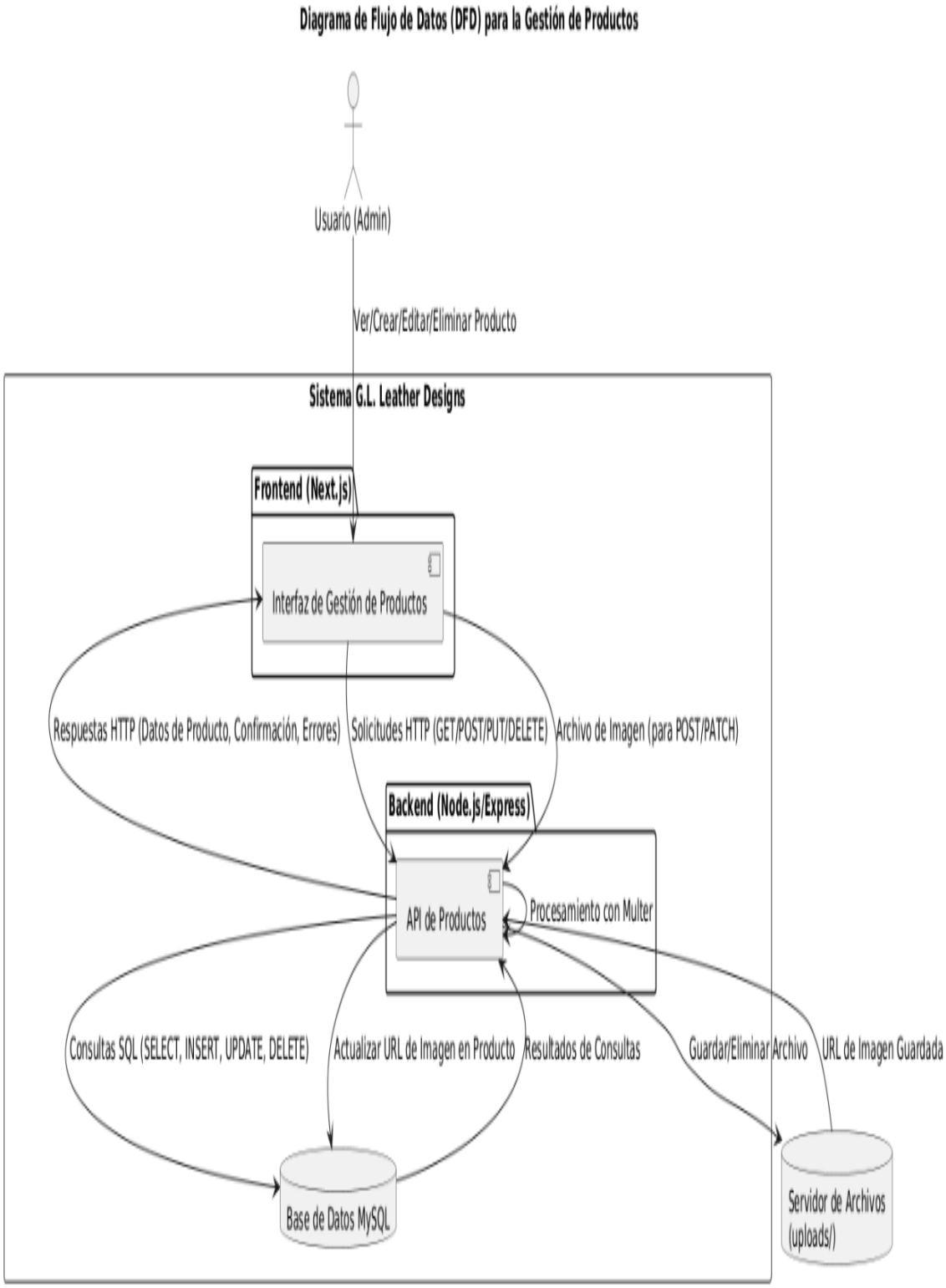
- **gl-leather-designs-final-frontend/**: Contiene la aplicación cliente desarrollada con Next.js.
    - src/app/: Contiene las rutas y páginas de la aplicación. Las rutas dinámicas como [id] se gestionan aquí.
    - src/components/: Componentes reutilizables de la interfaz de usuario.
    - src/lib/: Utilidades o funciones de ayuda específicas del frontend.
    - public/: Archivos estáticos como favicon.ico.

- **gl-leather-designs-final-backend/**: Contiene el servidor API REST desarrollado con Node.js y Express.
    - src/: Código fuente TypeScript del backend.
        - src/app.ts: Archivo principal del servidor, donde se configuran Express, las rutas de la API, los middlewares globales (CORS, JSON, autenticación) y la conexión a la base de datos.
        - src/utils/: Funciones de utilidad auxiliares (ej. asyncHandler para manejo de promesas en rutas).
    - dist/: Código JavaScript compilado a partir de los archivos TypeScript (src/). Estos son los archivos ejecutables del backend.
    - uploads/: Directorio donde se almacenan físicamente las imágenes de productos subidas por los usuarios.

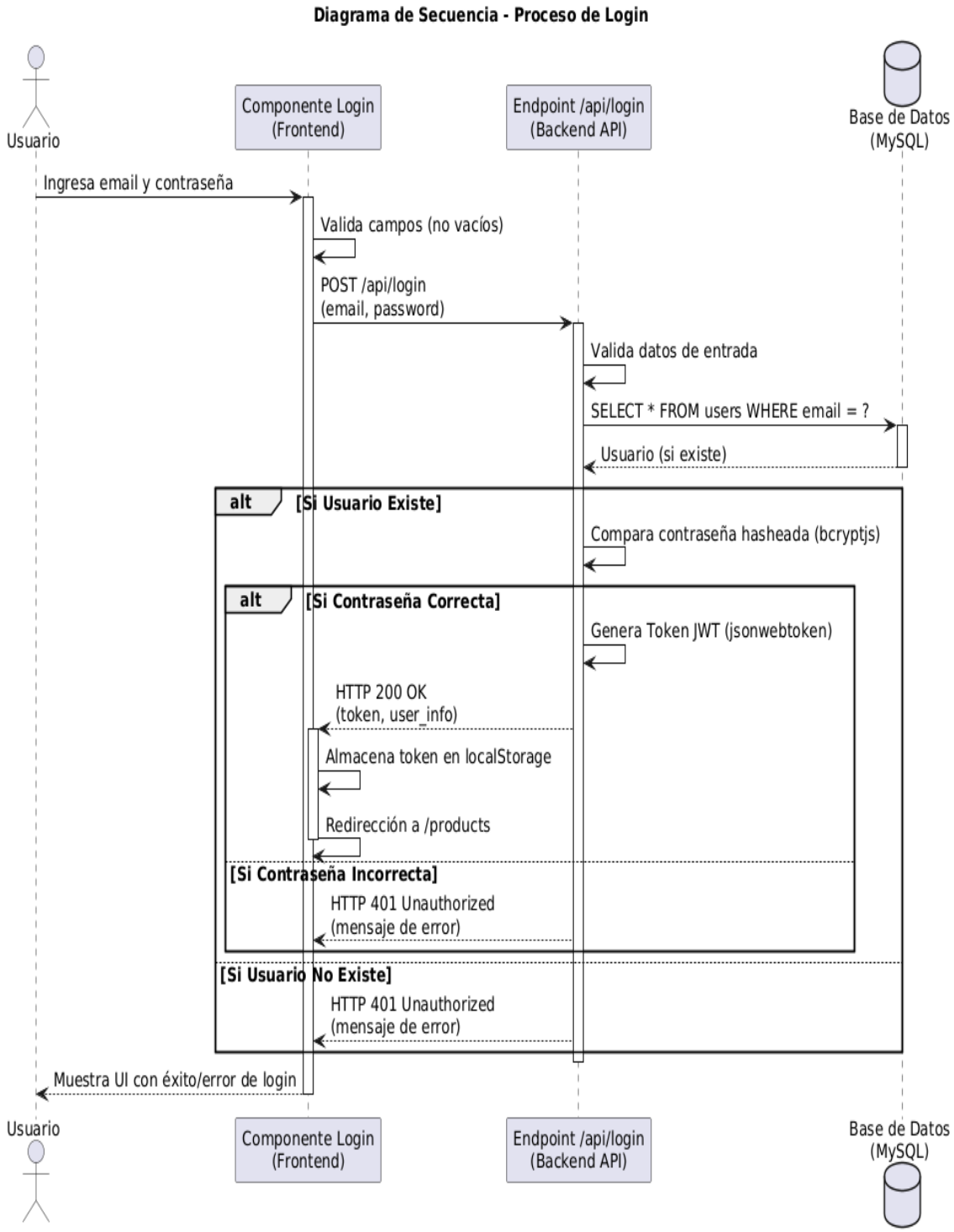
- **sql/**: Contiene los scripts SQL necesarios para la configuración de la base de datos.

Diagramas Lógicos

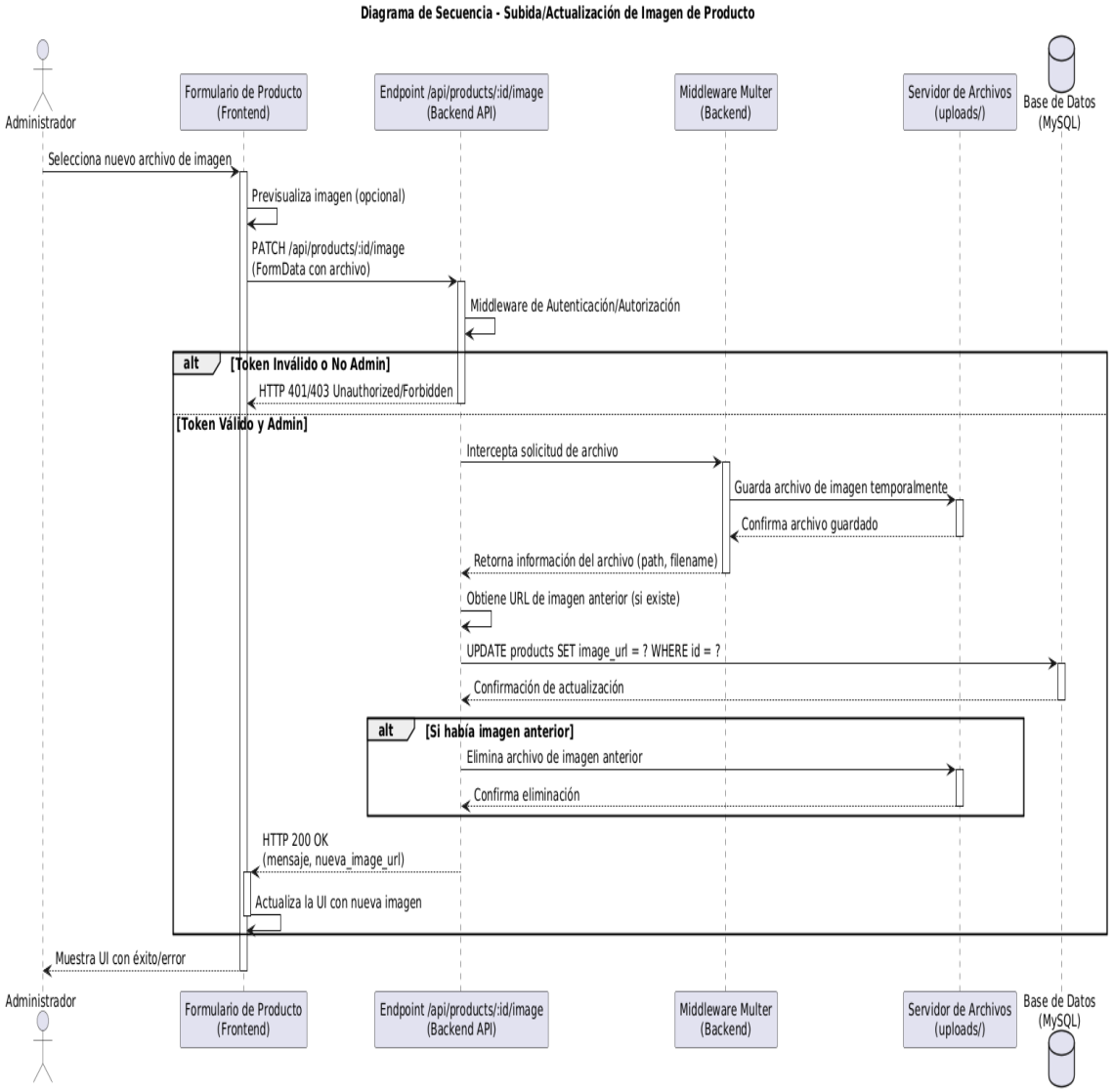
- **Diagrama de Flujo de Datos (DFD) para la gestión de productos:** Muestra cómo los datos fluyen desde el frontend (ej. formulario de edición), pasan por el backend (ruta PUT /api/products/:id) y se persisten/obtienen de la base de datos.



- **Diagrama de Secuencia para el Proceso de Login:** Ilustra el orden de interacción entre el usuario, el componente de login del frontend, el endpoint /api/login del backend y la base de datos para la autenticación y generación del token JWT.



- **Diagrama de Secuencia para la Subida/Actualización de Imagen de Producto:** Detalla el flujo cuando el usuario sube una imagen, cómo Multer la procesa en el backend, cómo se guarda en uploads/ y cómo se actualiza la URL en la base de datos.



Configuración del Entorno de Desarrollo

Prerequisitos:

- Node.js (versión recomendada: v18 o v20)
- npm (gestor de paquetes de Node.js, generalmente incluido con Node.js)
- MySQL Server (versión 8.0 recomendada, o compatible)
- Un editor de código (ej. Visual Studio Code)

- Cliente Git (para clonar el repositorio)

Pasos de Instalación y Configuración:

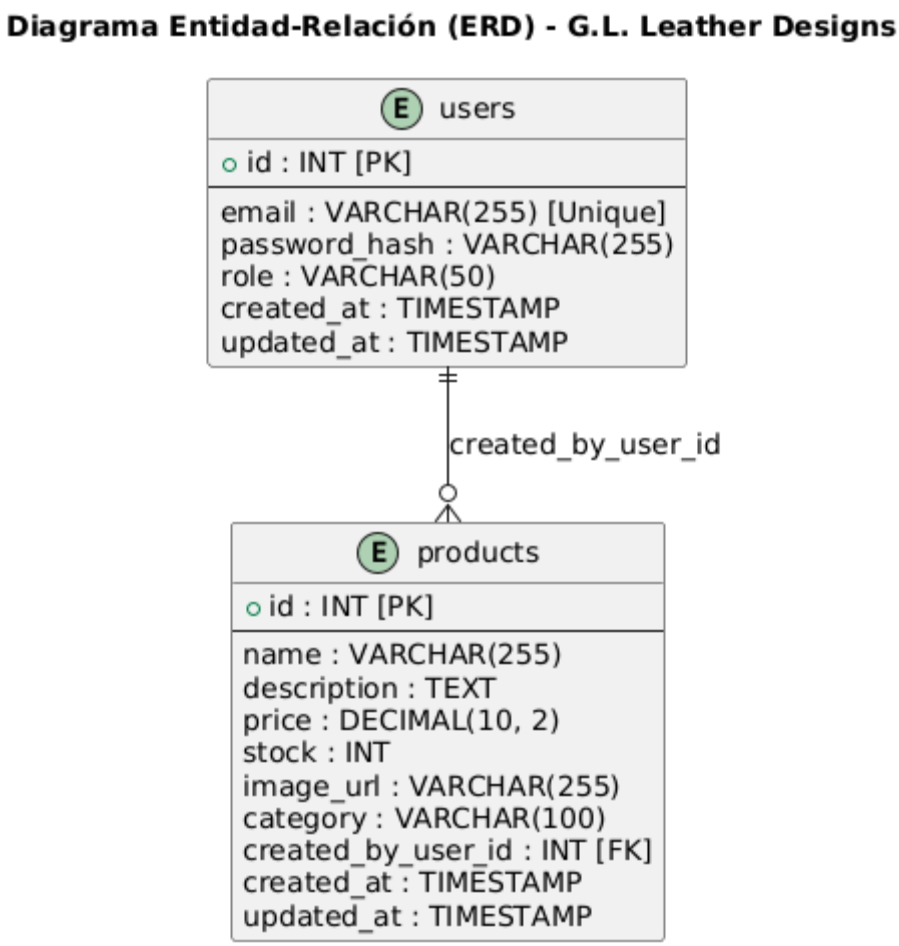
1. **Clonar el Repositorio:**
2. git clone [URL_DE_TU_REPOSITORIO]
3. cd gl-leather-designs-final # Navega a la carpeta principal del proyecto

4. **Configuración del Backend:**
5. cd gl-leather-designs-final-backend # Entra a la carpeta del backend
6. npm install # Instala las dependencias del backend
7. # Crea un archivo .env en esta carpeta (gl-leather-designs-final-backend) con la siguiente información:
8. # DB_HOST=localhost
9. # DB_USER=root
10. # DB_PASSWORD=XXX
11. # DB_NAME=gl_leather_designs_db
12. # JWT_SECRET=una_clave_secreta_fuerte_aleatoria_para_jwt
13. # FRONTEND_URL=http://localhost:3000
14. npm run start # Para compilar e iniciar el servidor backend

15. **Configuración del Frontend:**
16. cd ../gl-leather-designs-final-frontend # Vuelve a la raíz del proyecto y entra a la carpeta del frontend
17. npm install # Instala las dependencias del frontend
18. # Crea un archivo .env.local en esta carpeta (gl-leather-designs-final-frontend) con la siguiente información:
19. # NEXT_PUBLIC_API_BASE_URL=http://localhost:5000
20. npm run dev # Para iniciar el servidor frontend en modo desarrollo

Configuración de la Base de Datos

1. Asegúrate de que tu servidor MySQL esté corriendo.
2. Conéctate a tu servidor MySQL (ej. usando MySQL Workbench, DBeaver o la línea de comandos).
3. **Diagrama Entidad-Relación (ERD):**



Este diagrama visualiza la estructura de la base de datos, las tablas, sus atributos clave y las relaciones que definen el modelo de datos para los módulos implementados.

4. Ejecuta el script sql/create_database.sql para crear la base de datos y las tablas (users, products).
5. Ejecuta el script sql/initial_data.sql para insertar algunos datos de prueba y un usuario administrador (ej. admin@micorreo.com / admin123).

Estrategia de Manejo de Errores

- **En el Backend:** Se implementa un middleware de manejo de errores global para capturar y centralizar la gestión de excepciones no controladas. Se utilizan bloques try-catch en las rutas de la API y funciones asyncHandler para una gestión robusta de promesas. Los errores se loggean detalladamente en la consola del servidor para depuración y se devuelven respuestas HTTP apropiadas (400, 401, 403, 404, 500) con mensajes claros al frontend, diferenciando errores de cliente de errores internos del servidor.

- **En el Frontend:** Los componentes utilizan bloques try-catch en las llamadas a la API para interceptar errores de la red o del servidor. El estado de la interfaz de usuario se actualiza para mostrar mensajes de error amigables al usuario (ej. utilizando un estado setError y un componente de alerta visual), evitando interrupciones abruptas de la experiencia.

Validaciones de Entrada

- **En el Backend:** Se realizan validaciones estrictas de datos en las rutas de la API (ej. POST /api/products, PUT /api/products/:id, POST /api/register, POST /api/login) para asegurar que los campos obligatorios estén presentes, los tipos de datos sean correctos (ej. números para precio/stock), los formatos (ej. email) sean válidos y los valores cumplan con las reglas de negocio (ej. precio > 0). Esto es crucial para la integridad de la base de datos y la protección contra inyección SQL y otros ataques.
- **En el Frontend:** Se utilizan atributos required en los campos del formulario HTML y validación básica en el evento handleSubmit para proporcionar retroalimentación inmediata al usuario sobre la corrección de los datos antes de que la solicitud sea enviada al backend, mejorando la usabilidad.

Manejo de Sesiones/Tokens (Flujo del JWT)

El sistema utiliza un mecanismo de autenticación basado en Token Web JSON (JWT) para la gestión de sesiones:

- Cuando un usuario inicia sesión en la ruta /api/login, el backend genera un **Token Web JSON (JWT)**. Este token está firmado con una clave secreta (definida en el .env del backend) y tiene una fecha de expiración (establecida en 24 horas para esta fase).
- Este JWT se envía al frontend en la respuesta del login exitoso. El frontend lo almacena en el localStorage del navegador.
- Para todas las solicitudes subsiguientes a rutas protegidas (ej. cualquier operación CRUD de productos), el frontend incluye este JWT en el encabezado Authorization de la solicitud HTTP, en el formato Bearer <token>.
- El middleware protect del backend intercepta estas solicitudes, verifica la validez y la firma del JWT. Si el token es válido y no ha expirado, el middleware decodifica la información del usuario y permite el acceso a la ruta

solicitada. Si el token es inválido o ha expirado, la solicitud es rechazada con un código de estado 401 Unauthorized o 403 Forbidden, lo que indica la necesidad de volver a autenticarse.

Guía de Despliegue

[Aquí se proporcionarían instrucciones generales para el despliegue de los módulos entregados.]

- **Frontend (Next.js):** Plataformas como Vercel o Netlify son ideales para desplegar aplicaciones Next.js de forma eficiente.
- **Backend (Node.js/Express):** Plataformas como Render, Heroku, o servicios de IaaS como AWS EC2/DigitalOcean Droplet, pueden ser utilizadas para desplegar el servidor backend.

Guía de Contribución

- **Control de Versiones (Git Workflow):** El proyecto sigue un flujo de trabajo basado en ramas (ej. Feature Branch Workflow). La rama main/master debe contener siempre el código estable de producción. Las nuevas funcionalidades o correcciones se desarrollan en ramas separadas (feature/nombre-funcionalidad, bugfix/descripcion-bug). Los cambios se integran a través de Pull Requests/Merge Requests que requieren revisión de código, asegurando la calidad y la colaboración ordenada.
- **Estándares de Codificación:** Se recomienda seguir las configuraciones de herramientas como ESLint y Prettier (si se usan) para mantener un estilo de código consistente y legible en todo el proyecto.
- **Cómo crear nuevas funcionalidades o rutas:** Descripción de los pasos para extender el sistema.
- **Cómo ejecutar pruebas unitarias/de integración:** Indicaciones sobre cómo ejecutar los conjuntos de pruebas (si aplican).

Consideraciones de Seguridad

Además de la autenticación JWT, se han implementado:

- Hashing de contraseñas con bcryptjs para evitar el almacenamiento de contraseñas en texto plano.

- Validaciones de entrada para prevenir ataques comunes como la inyección SQL (gracias al uso de mysql2/promise con consultas preparadas) y XSS (a través de la sanitización implícita de React/Next.js y validaciones explícitas).
- Manejo seguro de la subida de archivos para prevenir ejecución de código malicioso.

5.5. Anexo 5: URLs del Sistema (Entorno Actual)

Este archivo simple contendrá las URLs relevantes para acceder al sistema en su estado actual.

URLs del Sistema gl_leather_designs_urls.txt

URLs del Sistema 'G. L. LEATHER DESIGNS'

Ambiente de Desarrollo Local (si se ejecuta en local)
Frontend: http://localhost:3000
Backend API Base: http://localhost:5000
Verificar Conexión DB: http://localhost:5000/check-db
Gestionar Productos API: http://localhost:5000/api/products

Ambiente de Despliegue en Producción (si aplica)
Frontend URL: https://vercel.com/templates
Backend API URL: https://github.com/Gentleman-Programming/Front-End-Clean-Architecture

Credenciales de Acceso (Cuenta de Administrador de Prueba)
Email: frans@micorreo.com
Contraseña: 222222

7. Entrega de Código Fuente y Archivos Compilados

El código fuente de la aplicación (frontend y backend) y los archivos compilados (.next/ para frontend y dist/ para backend) serán entregados dentro de la estructura de carpetas detallada en el punto 8. Los archivos se entregarán en su formato nativo (TypeScript/JavaScript) y los compilados en su formato generado.

8. Estructura General de Entrega (Archivo Comprimido)

La entrega final se organizará en un único archivo comprimido (.zip o .rar) siguiendo la siguiente estructura de carpetas:

/gl_leather_designs_entrega_final/

```
├── /gl-leather-designs-final-frontend/
# Código fuente del frontend (Next.js)
│   ├── src/
│   ├── .next/# Archivos compilados del frontend (generados por `next build`)
│   ├── package.json
│   └── ... (otros archivos del frontend)
├── /gl-leather-designs-final-backend/    # Código fuente del backend
(Node.js/Express)
│   ├── src/
│   ├── dist/                    # Archivos compilados del backend (generados por
`tsc`)
│   ├── uploads/                   # Carpeta para imágenes (si se desea incluir algunas
de prueba)
│   ├── package.json
│   ├── .env.example
│   └── ... (otros archivos del backend)
├── /sql/                        # Scripts SQL de la base de datos
│   ├── create_database.sql            # Script para crear la base de datos y tablas
│   └── initial_data.sql               # Script para insertar datos de prueba
├── /documentacion/                  # Carpeta para todos los documentos anexos (en
formato PDF)
│   ├── 1. Acta_Aprobacion_Requerimientos.pdf
│   ├── 2. Documentacion_Modulos.pdf
│   ├── 3. Pruebas_Realizadas.pdf
│   ├── 4. Manual_Tecnico.pdf
│   └── urls_sistema.txt               # Archivo de texto simple con URLs
└── README.md                        # Archivo principal con instrucciones de alto
nivel
```

9. Próximos Módulos y Funcionalidades a Desarrollar

Como parte de la visión estratégica y la evolución planificada del sistema "G. L. LEATHER DESIGNS", el equipo de desarrollo propone la implementación de los siguientes módulos y funcionalidades en las próximas fases del proyecto. Estos complementarán la base actual y transformarán la aplicación en una solución de ventas integral y aún más robusta, alineada con las necesidades de crecimiento del negocio.

9.1. Sistema de Ventas y Carrito de Compras

Este módulo central permitirá a los clientes interactuar directamente con el catálogo de productos, gestionar sus selecciones y completar transacciones.

- **Carrito de Compras Inteligente:**
    - **Funcionalidad:** Los usuarios podrán añadir, eliminar y ajustar la cantidad de productos en un carrito virtual antes de proceder a la compra.
    - **Persistencia:** El contenido del carrito se mantendrá guardado incluso si el usuario cierra la sesión o el navegador, garantizando una experiencia de compra fluida y conveniente.
    - **Previsualización Detallada:** Se ofrecerá un resumen claro y conciso de los productos seleccionados, cantidades, precios unitarios, subtotales e impuestos aplicables.
- **Flujo de Checkout Optimizado:**
    - Se implementará un proceso de compra guiado, intuitivo y paso a paso, abarcando la recopilación de información de envío, la selección de métodos de pago y la confirmación final del pedido.
- **Historial de Pedidos para Clientes:**
    - Cada cliente podrá acceder a un registro detallado de sus compras anteriores, incluyendo el estado actual del pedido y todos los detalles relevantes para su seguimiento.
- **Gestión de Estados de Pedidos (Administración):**
    - Se implementará un sistema de estados (ej. Pendiente de Pago, Procesando, Enviado, Entregado, Cancelado) que permitirá al administrador una gestión eficiente y transparente del ciclo de vida de los pedidos.

9.2. Integración de Formas de Pago

Para una experiencia de compra completa y segura, se integrarán diversas opciones de pago.

- **Pasarelas de Pago Seguras:**
    - Implementación de integración con pasarelas de pago líderes del mercado (ej. Stripe, PayPal, o soluciones locales relevantes), ofreciendo transacciones seguras y eficientes.
- **Diversidad de Métodos de Pago:**
    - Soporte para tarjetas de crédito/débito, y si la estrategia comercial lo requiere, opciones como transferencias bancarias directas o pago contra entrega.
- **Garantía de Seguridad en Transacciones:**
    - Se asegurará que todas las transacciones cumplan con los estándares de seguridad más exigentes, protegiendo la información financiera del cliente y la integridad de las operaciones.

9.3. Facturación Electrónica

La automatización de la facturación es crucial para la eficiencia administrativa y el cumplimiento legal.

- **Generación Automática de Facturas:**
    - Creación automatizada de facturas a partir de pedidos completados, reduciendo la carga manual y el margen de error operativo.
- **Cumplimiento Legal y Normativo:**
    - Se asegurará que las facturas generadas cumplan con la normativa legal y fiscal vigente del país (formato, requisitos fiscales), posiblemente a través de la integración con un proveedor de servicios de facturación electrónica local.
- **Distribución y Acceso al Cliente:**
    - Se ofrecerá la opción de enviar facturas por correo electrónico al cliente y permitir su descarga en formato PDF desde su historial de pedidos.

9.4. Gestión de Inventarios

Un control de inventarios preciso es vital para la operación eficiente del negocio, minimizando pérdidas y optimizando la disponibilidad de productos.

- **Control de Stock en Tiempo Real:**

    - Actualización automática de los niveles de stock con cada venta, devolución o ajuste de inventario, proporcionando información precisa al momento.
- **Alertas Inteligentes de Bajo Stock:**
    - Notificaciones automatizadas al administrador cuando el stock de un producto baja de un umbral predefinido, facilitando la reposición oportuna y evitando roturas de stock.
- **Registro de Movimientos de Inventario:**
    - Funcionalidad para registrar manualmente entradas (compras a proveedores) y salidas (ajustes, pérdidas, etc.) de inventario, manteniendo un registro auditable.
- **Reportes Exhaustivos de Inventario:**
    - Generación de informes detallados sobre niveles de stock, movimientos de inventario y valorización, ofreciendo una visión clara del capital inmovilizado y la rotación de productos.

9.5. Gestión de Proveedores

Este módulo optimizará la relación y las transacciones con los proveedores, facilitando la cadena de suministro.

- **Base de Datos Centralizada de Proveedores:**
    - Un sistema para registrar y gestionar la información de contacto, términos y condiciones, y documentos clave de cada proveedor.
- **Gestión de Órdenes de Compra:**
    - Creación y seguimiento de órdenes de compra enviadas a proveedores, incluyendo productos, cantidades, precios de adquisición y fechas de entrega esperadas.
- **Control de Recepción de Mercancía:**
    - Proceso formal para registrar la entrada de bienes al almacén y su impacto inmediato en el inventario.
- **Historial de Transacciones con Proveedores:**
    - Registro y acceso a todas las compras y transacciones realizadas con cada proveedor, facilitando la auditoría y la gestión de la relación.

9.6. Mejoras en la Experiencia de Usuario y Analíticas

Para hacer el sistema aún más amigable, potente y estratégico tanto para el cliente final como para la administración.

- **Panel de Administración (Dashboard Ejecutivo):**
    - Un panel de control intuitivo y personalizable para el administrador, presentando métricas y estadísticas clave en tiempo real (ventas diarias/mensuales, productos más vendidos, stock bajo, margen de beneficio), permitiendo una toma de decisiones informada y ágil.
- **Búsqueda y Filtrado Avanzado de Productos (Frontend):**
    - Mejoras significativas en la experiencia de navegación del cliente con opciones de búsqueda avanzada, filtros por categoría, rango de precio, stock, y diversas opciones de ordenamiento (por popularidad, precio ascendente/descendente, etc.).
- **Reportes y Analíticas Detalladas:**
    - Generación de informes personalizables sobre ventas (por producto, categoría, fecha, cliente), desempeño de inventario (rotación, obsolescencia) y comportamiento de proveedores.
- **Personalización y Fidelización del Cliente:**
    - Implementación de funcionalidades como listas de deseos, recomendaciones de productos basadas en el historial de navegación/compra, y un módulo CRM básico para gestionar la relación con los clientes y campañas de marketing.

10. Conclusión y Próximos Pasos

La entrega de esta fase inicial del sistema "G. L. LEATHER DESIGNS" marca un hito importante en el desarrollo del proyecto. Hemos logrado establecer una base sólida y funcional, que incluye la autenticación de usuarios y la gestión completa de productos, demostrando la viabilidad técnica y el cumplimiento de los requerimientos de esta etapa.

El proceso de desarrollo, aunque con sus desafíos, ha validado la robustez de la arquitectura seleccionada y la efectividad de las tecnologías implementadas. Los problemas técnicos encontrados fueron superados mediante un enfoque metódico de depuración y la aplicación de buenas prácticas de ingeniería de software.

Próximos Pasos:

- **Validación del Cliente:** Se espera la revisión y validación exhaustiva de los módulos entregados en esta fase por parte del cliente. Su retroalimentación es crucial para confirmar que la solución cumple con sus expectativas y es adecuada para los procesos de negocio.
- **Planificación Detallada de Fases Futuras:** Una vez aprobada esta fase, se procederá con la planificación detallada de los módulos restantes delineados en la sección "9. Próximos Módulos y Funcionalidades Para Desarrollar". Esto incluirá la definición de requerimientos específicos, cronogramas y asignación de recursos.
- **Compromiso con la Evolución Continua:** El equipo de desarrollo reitera su compromiso de continuar la evolución del proyecto, asegurando la entrega de una solución integral y de alta calidad que se adapte y crezca con las necesidades de "G. L. LEATHER DESIGNS". Se mantendrá un enfoque en la optimización del rendimiento, la seguridad y la experiencia de usuario a medida que el proyecto avance.

Estamos seguros de que esta entrega parcial será el cimiento para un proyecto exitoso y que los módulos futuros complementarán y expandirán significativamente las capacidades operacionales y estratégicas de "G. L. LEATHER DESIGNS".