Documentación por Módulo

Funcionalidades Entregadas en la Fase Parcial

Este documento detalla los módulos y sus funcionalidades que han sido completamente desarrollados y entregados en esta fase parcial del proyecto "G. L. LEATHER DESIGNS". Se proporciona una descripción técnica de los datos de entrada, procesos internos y datos de salida para cada módulo.

Módulo: Autenticación (Login y Registro)

- **Nombre del Archivo(s) Relevante(s):**
    - Frontend: gl-leather-designs-final-frontend/src/app/login/page.tsx, gl-leather-designs-final-frontend/src/app/register/page.tsx
    - Backend: gl-leather-designs-final-backend/src/app.ts (rutas /api/login y /api/register)
- **Descripción:** Este módulo permite a los usuarios gestionar su acceso al sistema, incluyendo el registro de nuevas cuentas y el inicio de sesión para usuarios existentes. Es la puerta de entrada a las funcionalidades protegidas.
- **Datos de entrada:**
    - email: Texto (string), formato de correo electrónico.
    - password: Texto (string), contraseña segura.
- **Proceso:**

1. **Validación de Credenciales en Frontend:** La interfaz de usuario valida inicialmente que los campos de email y password no estén vacíos y cumplan con formatos básicos antes del envío.
2. **Comunicación con Backend:** Las credenciales se envían al endpoint correspondiente del backend (/api/login o /api/register).
3. **Procesamiento en Backend:**
    - **Registro:** Se verifica la unicidad del email, se hashea la password utilizando bcryptjs para almacenamiento seguro y se inserta el nuevo usuario en la base de datos con un rol por defecto.
    - **Login:** Se busca el usuario por email, se compara la password proporcionada con el hash almacenado. Si coinciden, se genera un Token Web JSON (JWT) firmado con una clave secreta (process.env.JWT_SECRET) y con una fecha de expiración, conteniendo la información básica del usuario (ID, email, rol).

        - Se implementa manejo de errores para credenciales inválidas o emails ya registrados.

4. **Respuesta al Frontend:** El backend responde con el JWT y la información del usuario (en caso de login exitoso) o mensajes de error.
5. **Gestión de Sesión en Frontend:** Si el login es exitoso, el frontend almacena el JWT de forma segura en localStorage y redirige al usuario a la página de productos o dashboard.

- **Datos de salida:**
    - **Acceso Exitoso:** Redirección interna en el frontend.
    - token: Texto (string, el JWT generado).
    - user: Objeto JSON con id (numérico), email (texto) y role (texto) del usuario.
    - message: Texto (string, mensaje de éxito o error detallado).
    - **Códigos de Estado HTTP:** 200 (OK), 201 (Created), 400 (Bad Request), 401 (Unauthorized), 409 (Conflict), 500 (Internal Server Error).

Módulo: Gestión de Productos (CRUD)

- **Nombre del Archivo(s) Relevante(s):**
    - Frontend: gl-leather-designs-final-frontend/src/app/products/page.tsx (listado), gl-leather-designs-final-frontend/src/app/products/[id]/page.tsx (edición/detalle), gl-leather-designs-final-frontend/src/app/products/create/page.tsx (creación)
    - Backend: gl-leather-designs-final-backend/src/app.ts (rutas /api/products, /api/products/:id, /api/products/:id/image)
- **Descripción:** Este módulo permite al administrador realizar el ciclo completo de vida de los productos (Crear, Leer, Actualizar, Eliminar). Incluye la gestión de datos textuales del producto y la asociación/actualización de imágenes.
- **Datos de entrada:**
    - id_producto: Numérico (para consulta, edición y eliminación de un producto específico).
    - name: Texto (string), nombre del producto.
    - description: Texto (string, opcional), descripción detallada.
    - price: Numérico (decimal), precio del producto.
    - stock: Numérico (entero), cantidad disponible en inventario.

- category: Texto (string), categoría a la que pertenece el producto.
- image: Archivo (tipo File), el archivo de imagen a subir.
- Authorization: Encabezado HTTP con el JWT (Bearer <token>) para la autenticación y autorización del usuario.

- **Proceso:**

1. **Autenticación y Autorización en Backend:** Para todas las operaciones de gestión (POST, PUT, PATCH, DELETE), el backend utiliza middlewares (protect, authorize(['admin'])) para asegurar que solo usuarios autenticados con rol de admin puedan acceder a estas funcionalidades.
2. **Operaciones en Frontend:**
    - **Listado (GET /api/products):** Realiza una petición para obtener todos los productos y los muestra en una tabla/lista.
    - **Detalle/Edición (GET /api/products/:id):** Carga los datos de un producto específico para su visualización o edición.
    - **Creación (POST /api/products):** Envía los datos del nuevo producto, incluyendo el archivo de imagen (usando FormData), al backend.
    - **Actualización de Datos (PUT /api/products/:id):** Envía los datos textuales actualizados del producto en formato JSON.
    - **Actualización de Imagen (PATCH /api/products/:id/image):** Envía solo el nuevo archivo de imagen (usando FormData), con lógica en el backend para reemplazar y eliminar la imagen antigua.
    - **Eliminación (DELETE /api/products/:id):** Envía una solicitud para borrar un producto específico y su imagen asociada.
    - Gestiona los estados de carga, muestra mensajes de error y redirige al usuario tras operaciones exitosas.
3. **Procesamiento en Backend:**
    - **Validación de Entrada:** Se realizan validaciones robustas para asegurar que los datos recibidos son válidos (ej. tipo de dato correcto, campos obligatorios presentes, formato de archivo de imagen).

        - **Manejo de Archivos:** Utiliza multer para procesar la subida de imágenes, guardándolas en el directorio uploads/ del servidor y generando una URL relativa.
        - **Interacción con DB:** Ejecuta las consultas SQL correspondientes (INSERT, SELECT, UPDATE, DELETE) en la base de datos MySQL para manipular los registros de productos.
        - **Limpieza:** En operaciones de actualización de imagen o eliminación de producto, se incluye lógica para borrar los archivos de imagen obsoletos o no deseados del servidor, liberando espacio.
        - Manejo de errores específicos (producto no encontrado, datos inválidos) y errores internos del servidor.
- **Datos de salida:**
    - product_object: Objeto JSON (información detallada de un producto) o un array de objetos (listado de productos).
    - message: Texto (string, mensaje de éxito o error detallado).
    - imageUrl: Texto (string, URL de la nueva imagen subida/actualizada).
    - **Códigos de Estado HTTP:** 200 (OK), 201 (Created), 400 (Bad Request), 401 (Unauthorized), 403 (Forbidden), 404 (Not Found), 500 (Internal Server Error).