

ANALISIS Y DESARROLLO DE SOFTWARE

**MANUAL DE DESARROLLO/**

**DOCUMENTACION DE CODIGO**

Marzo 2025

**INDICE**

INTRODUCCIÓN

1. **Estructura del Proyecto**
2. **Tecnologías y Frameworks** utilizados.
3. **Explicación del Código** (módulos, funciones, clases).
4. **Convenciones de Código** (nomenclatura, estilos, buenas prácticas).
5. **Flujo de Datos** (cómo se comunican las partes del sistema).
6. **Ejemplos de Uso** y fragmentos de código explicativos.
7. I**nstalación y Configuración** (dependencias y entorno de desarrollo).

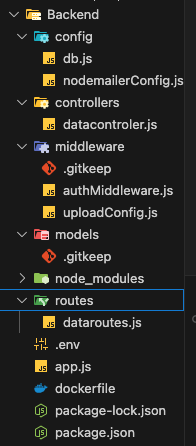
**INTRODUCCIÓN**

El Manual de Desarrollo es un documento fundamental, ya que proporciona información detallada sobre el desarrollo del software realizado. En él se describe su estructura, las tecnologías utilizadas, la funcionalidad y el manejo del aplicativo.

Este manual sirve como una guía de referencia tanto para desarrolladores como para futuros mantenedores del sistema, permitiendo comprender su funcionamiento y facilitando futuras modificaciones o mejoras. Además, incluye las instrucciones necesarias para garantizar un uso eficiente y una correcta evolución del software a lo largo del tiempo.

1. **Estructura de Proyecto:**

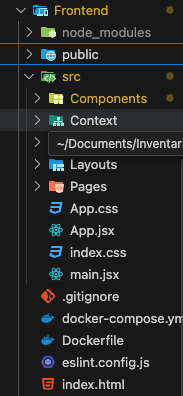
**BACKEND:**



**Backend/** → Contiene la lógica del servidor, bases de datos y API.

* Config/ → Contiene archivos de configuración, como la conexión a la base de datos y la configuración del servicio de correo electrónico para el envío de notificaciones.
* Controllers/ → Maneja las peticiones y respuestas de la API.
* Middleware/ → Contiene funciones intermedias que procesan las solicitudes antes de llegar a los controladores, como la verificación de tokens y la configuración de carga de archivos.
* Routes/ → Contiene las rutas del servidor.
* **.env** → Archivo de variables de entorno que almacena información sensible como claves secretas, configuraciones de autenticación JWT y credenciales de servicios externos (Cloudinary y Nodemailer).
* App.js → Archivo principal que inicia el servidor.

**FRONTEND**



**Frontend/** → Contiene la interfaz del usuario

* Public/ → Contiene archivos como imágenes útiles para la interfaz.
* Components/ → Componentes reutilizables como botones, formularios, etc.
* **context/** → Contiene el contexto global de la aplicación, permitiendo gestionar el estado del usuario, su autenticación y persistencia en el almacenamiento local.
* Layout / → Contiene layout que definen la estructura general de las interfaces
* pages/ → Vistas principales del aplicativo.
* Index/→ Contiene rutas de iconos y tipo de letra.
* App.jsx → Contiene todas las rutas y la protección de ellas.

**Tecnologías y Frameworks** utilizados.

1. **Tecnologías y Frameworks** **utilizados.**

**Backend:**

El backend está desarrollado con Node.js y el framework Express.js, lo que permite una gestión eficiente de rutas, peticiones y middleware. También se emplean las siguientes librerías:

* Axios: Para realizar solicitudes HTTP a servicios externos.
* Bcrypt.js: Para el cifrado de contraseñas y mejorar la seguridad.
* Cloudinary: Para el almacenamiento y gestión de imágenes en la nube.
* Cookie-parser: Para manejar cookies en las peticiones HTTP.
* CORS: Para permitir peticiones desde distintos orígenes.
* Dotenv: Para la gestión de variables de entorno y mejorar la seguridad del proyecto.
* Jsonwebtoken: Para la autenticación de usuarios mediante tokens JWT.
* Multer: Para la carga y gestión de archivos.
* Nodemailer: Para el envío de correos electrónicos en el proceso de recuperación de contraseñas.
* PG (PostgreSQL): Para la conexión y gestión de la base de datos.
* UUID: Para la generación de identificadores únicos en la aplicación.

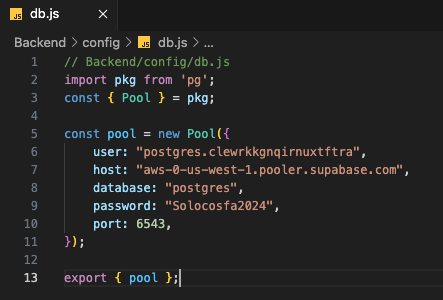
**Frontend:**

El frontend está desarrollado con React.js y utiliza Vite como herramienta de desarrollo, lo que permite una carga rápida y optimización del código. Además, se han utilizado las siguientes librerías:

* @dotlottie/player-component: Para animaciones en formato Lottie.
* Axios: Para la comunicación con el backend mediante solicitudes HTTP.
* Chart.js & react-chartjs-2: Para la visualización de gráficos en la interfaz.
* JsBarcode: Para la generación de códigos de barras.
* JSPDF & JSPDF-AutoTable: Para la generación de documentos PDF.
* Lucide-react: Para la integración de iconos en la interfaz.
* PDFMake: Para la creación y personalización avanzada de documentos PDF.
* UUID: Para la generación de identificadores únicos.
* XLSX: Para la manipulación y generación de archivos Excel

1. **Explicación del Código** (módulos, funciones, clases).

**Backend:**

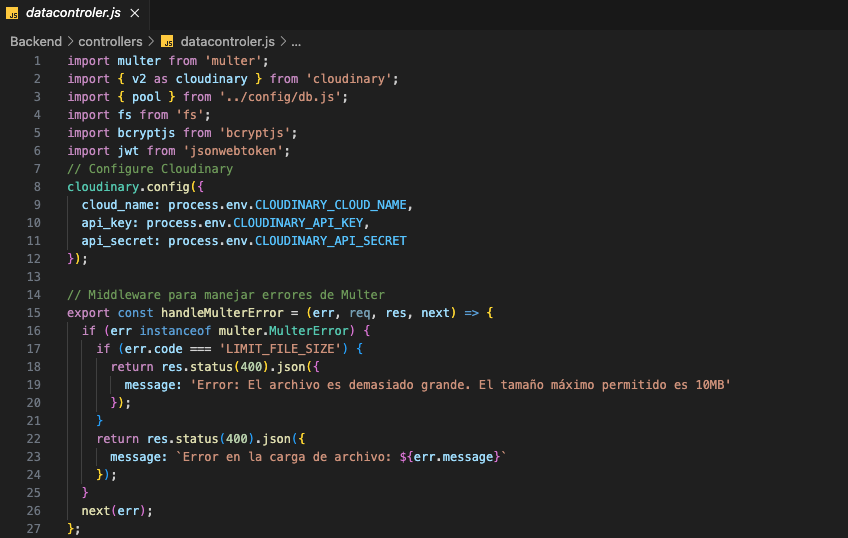
**Db.js: Conexión con la base de datos (Supabase)**

**nodemailerConfig.js:**

**Configuración** para poder enviar correos con Gmail.

****

**datacontroler.js:**Este código es un **controlador** en Node.js que maneja la lógica de errores en la carga de archivos y la consultas en una base de datos.



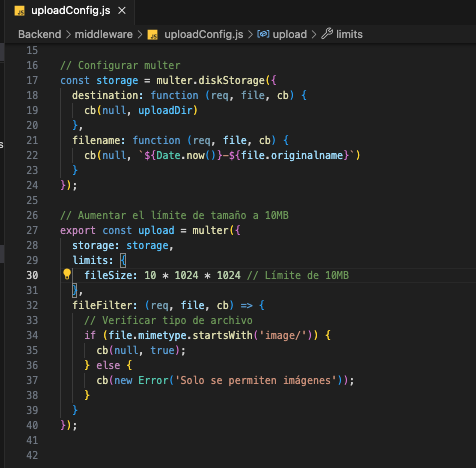
**authMiddleware.js:**

Protege rutas verificando tokens JWT.

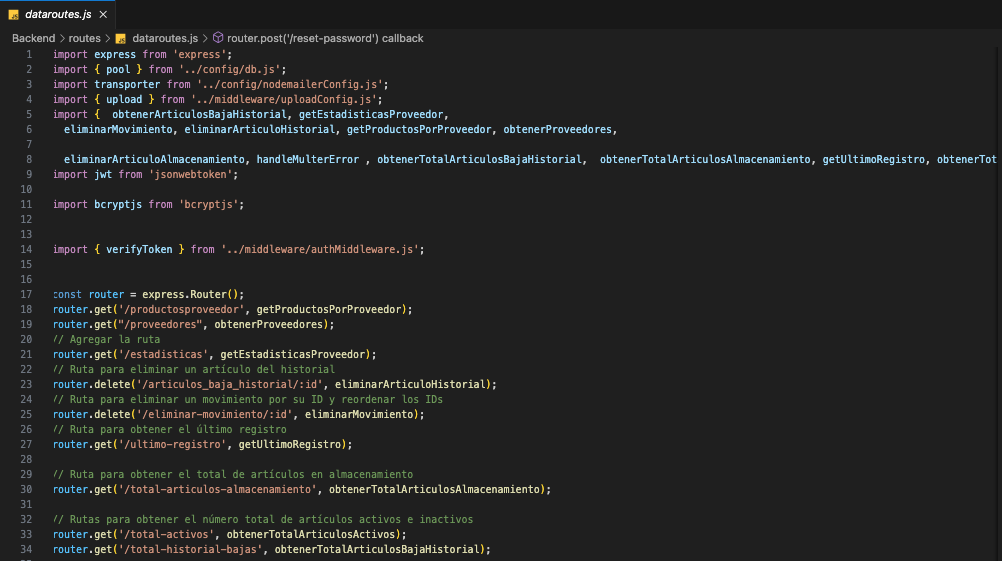


**uploadConfig.js:**

Maneja la subida de imágenes con restricciones de tamaño y formato.



**Dataroutes:**

Este archivo es un router de Express que define las rutas de la API y las conecta con los controladores correspondientes.  


**app.js**

Este archivo es el punto de entrada principal del backend. Configura el servidor Express, maneja CORS, gestiona middleware y define las rutas de la API.

