ChatGPT

Contexto de Anclora Nexus — Actualización Sept 2025

Visión general

Anclora Nexus es una plataforma modular de transformación de contenido que combina un **motor de conversión** con un módulo opcional de **edición y publicación de libros** (Anclora Press). Su propuesta es “reinventar tu contenido”, permitiendo a usuarios independientes transformar archivos de texto en una variedad de formatos, editar manuscritos y publicar *ebooks* o libros físicos. El proyecto se desarrolla como un monorepo con frontend React/TypeScript y backend Flask.

Módulos y funcionalidades

Núcleo de Conversión

- **Nuevos conversores**: incluye seis conversores adicionales que amplían de 4 a 10 los formatos disponibles [1] [2].
- **Capacidades**: convierte textos a HTML, DOC, Markdown, RTF, ODT y LaTeX [3], y continua añadiendo soporte para imágenes (pendiente PNG→WebP, GIF→MP4, SVG→PNG) [4].
- **Procesamiento asíncrono** y validación de entrada para manejar archivos grandes [5].
- **Normalización de codificación** a UTF-8 para asegurar consistencia [6].

Anclora Press

- **Importación flexible**: acepta .txt, .md, .doc, .docx y .pdf [7] [8].
- **Editor avanzado**: orientado a autores y creadores, con herramientas profesionales para maquetar libros [9].
- **Exportación**: soporta formatos estándar de la industria editorial y optimiza para libros digitales y físicos [10].
- **Publicación directa**: integra la subida a plataformas de publicación [10].

Sistema de créditos

- Gestiona la compra, uso y upgrades de créditos de conversión [11].
- Registra el historial de transacciones y controla el consumo de cada conversión [11].

Interfaz de usuario

- Ofrece un *dashboard* intuitivo con drag & drop de archivos, previsualización y descarga masiva [12].
- Utiliza React 18, Vite y Tailwind; las rutas se componen de `index.html → main.tsx → MainApp → pages/app(AppPage) → NewApp → MainLayout → SafeConversor` [13].

Estructura del proyecto

```
anclora-nexus/
├── frontend/    # Aplicación React/TypeScript [14]
│   ├── src/     # Componentes, páginas, servicios, utils [15]
│   ├── public/  # Recursos estáticos [16]
│   └── package.json  # Dependencias frontend [17]
├── backend/     # API Python/Flask [18]
│   ├── src/
│   │   ├── models/    # Modelos de datos [19]
│   │   ├── routes/    # Endpoints API [20]
│   │   ├── services/  # Servicios de conversión [21]
│   │   └── utils/     # Utilidades backend [21]
│   ├── tests/         # Suite de pruebas: unitarias e integración [22]
│   ├── requirements.txt  # Dependencias Python [23]
│   └── main.py       # Punto de entrada [24]
├── docs/             # Documentación [25]
├── scripts/          # Scripts de automatización [26]
└── data/             # Archivos de datos [27]
```

Instalación y ejecución

- **Prerrequisitos**: Node.js 18+, Python 3.10+ y Git [28].
- **Frontend**: `cd frontend && npm install && npm run dev` [29].
- **Backend**: `cd backend && pip install -r requirements.txt && python main.py` [30]. El puerto por defecto es 8000 y puede configurarse con la variable `PORT` [31].
- **Variables de entorno**: definir `SECRET_KEY`, `JWT_SECRET_KEY`, `ALLOWED_ORIGINS`, `FLASK_DEBUG` y `LOG_LEVEL` para el backend [32].
- **Testing**: existen suites de pruebas en backend con Pytest y en frontend con Vitest y Jest; puede ejecutarse `npm run test:all` para correr toda la suite [33].

Seguridad y monitoreo

- **Autenticación**: basada en JWT (pendiente mejorar refresh y almacenamiento seguro) [34].
- **Validación de archivos**: se valida el tipo y tamaño, aunque requiere endurecerse para mitigar ataques [34].
- **CORS**: configurable mediante `ALLOWED_ORIGINS`, pero aún sin una política unificada para Flask y SocketIO [34].
- **Métricas y logs**: la API expone métricas en `/metrics`; se utilizan logs estructurados y se planea integrar trazabilidad y request-id [31] [35].

Estado actual de la rama `augment-dev`

Al 9 de septiembre de 2025, la rama **augment-dev** incorpora los nuevos conversores y la estructura modular descrita. No obstante, persisten varias tareas en curso:

1. **Retirar archivos temporales de CI** (`temp_ci_fix.txt` y `.ci-rebuild-trigger`) [36].
2. **Implementar recuperación de contraseña en** `Login.tsx` [37].

3. **Completar ConversionEngine** con los formatos pendientes (PNG→WebP, GIF→MP4, SVG→PNG) [38].
4. **Añadir pruebas para flujos de créditos** (compra, upgrade) [39].
5. **Alinear documentación y código** (mención a FastAPI vs Flask, paths duplicados) [40].
6. **Refactorizar configuración y base de datos** para evitar archivos versionados y usar una configuración central [41].
7. **Corregir errores de codificación y limpiar logs de depuración** [42].

Roadmap resumido

El plan de mejora propuesto aborda quick wins en las primeras semanas (unificar endpoints, corregir configuración, normalizar UTF-8 y limpiar el repositorio) [43]. Posteriormente se plantean mejoras arquitectónicas (app factory y colas de trabajo), seguridad avanzada (cookies `httpOnly`, rate limiting, validación de uploads) [44], mejoras en frontend y backend (rutas protegidas, React Query, consolidación de rutas, observabilidad) [45] y una fase final dedicada a documentación OpenAPI, pruebas de contrato y herramientas de CI/CD [46].

Enlaces de referencia

- Plan de Mejora (`docs/MEJORA_ANCLORA_NEXUS.md`)
- Esquema de Arquitectura

Nota: este contexto se actualiza a septiembre 2025 y refleja el punto actual de desarrollo. Se recomienda revisarlo periódicamente para identificar progresos y nuevas necesidades.

---

1 2 3 4 5 6 7 8 9 10 11 12 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 36 37 38 39 raw.githubusercontent.com
https://raw.githubusercontent.com/ToniIAPro73/AncloraNexus/augment-dev/README.md

13 34 35 40 41 42 43 44 45 46 raw.githubusercontent.com
https://raw.githubusercontent.com/ToniIAPro73/AncloraNexus/augment-dev/docs/MEJORA_ANCLORA_NEXUS.md