**Aplicación 3**

**Software de Conversión de Archivos**

Adicional a lo que está en la documentación, esta app incluye:

Automatizar la ingesta, validación y transformación a esquema relacional de PostgreSQL.

**Resumen de pasos**

* Crear la estructura de carpetas de la data a partir del “protocolo del tesla suit”.
* Crear esquema base en PostgreSQL con las tablas arriba descritas.
* Crear scripts (Python + SQLAlchemy / psycopg2) para cargar los CSV y relacionar datos.
* Implementar backend CRUD para metadatos.
* Diseñar front web (React/Streamlit) para consulta y edición de metadatos.
* Conectar todo en la app web general que unifique protocolos y datos.

**Sugerencias para mejorar la organización:**

* Si se espera que los datos crezcan mucho, considerar añadir subcarpetas adicionales para segmentar por fecha o sesión, para no saturar un solo directorio.
* La carpeta base data puede cambiarse a un nombre más genérico como raw\_data o input\_data para mayor claridad.
* Para ambientes más grandes o distribuidos, esta estructura local puede migrar a un almacenamiento centralizado o usar bases de datos.

**Validaciones adicionales que podrías considerar para robustez y calidad:**

**Validaciones y control de errores**

* Validación de columnas y tipos en CSV
  + Verificar que cada CSV tenga las columnas esperadas antes de procesar.
  + Validar que los datos numéricos no tengan valores nulos o fuera de rango esperado.
* Integridad de archivos
  + Detectar y descartar archivos incompletos o corruptos (e.g., tamaño cero o CSV malformado).
  + Comprobar duplicados por escena, peso, tipo y fecha.
* Logs detallados
  + Mejorar el nivel y detalle del logging para facilitar debugging (por ejemplo, agregar timestamps y niveles separados para info, warning, error).
* Manejo de excepciones específicas
  + Capturar errores más específicos para saber si falló la lectura, escritura o formato.
* Arquitectura y calidad de código
* Separación en capas o módulos
  + Separar lógica de I/O, procesamiento de datos y validaciones en módulos o clases independientes.
  + Facilita tests unitarios y mantenimiento.
* Tests automatizados
  + Escribir pruebas unitarias para funciones clave (validación, conversión, fusión).
  + Tests de integración para verificar flujo completo.
* Configuración y parámetros
  + Externalizar rutas, patrones y parámetros a archivos de configuración o variables de entorno.
  + Facilita adaptación a diferentes entornos.

**Escalabilidad y microservicios**

* Si se piensa en un entorno distribuido:
  + Separar el proceso en microservicios:
    - Servicio para ingestión y validación de CSV
    - Servicio para conversión y enriquecimiento a XLSX
    - Servicio para fusión y exportación final
  + Esto permitiría paralelizar procesamiento y escalar según demanda.
* Uso de colas o pipelines para procesar datos de forma asíncrona.

**Seguridad**

* Control de acceso
  + Si se ejecuta en un entorno compartido o servidor, restringir permisos en carpetas para evitar accesos no autorizados.
* Validación de entrada
  + Evitar inyección de código o path traversal si en algún momento los nombres provienen de usuarios externos.
* Integridad de datos
  + Verificar firmas o hashes de archivos originales para garantizar no fueron modificados maliciosamente.

**Resumen**

Tu estructura y lógica actual son una base sólida para el procesamiento local de los archivos biométricos. Para escalar y robustecer el proyecto en un contexto más profesional, considera:

* Añadir validaciones de esquema y contenido de los CSV.
* Modularizar el código y externalizar configuraciones.
* Incorporar pruebas automáticas.
* Preparar para procesamiento distribuido o por microservicios si la carga o el equipo crecen.
* Implementar controles de seguridad y permisos.