








Perfecto, Adolfo. A continuación te expongo un conjunto de **tareas siguientes sugeridas** para mejorar, consolidar y expandir el proyecto `SQLitePlus Enhanced` de forma profesional, robusta y escalable:





✓ 1. Consolidación de arquitectura

-  **Reorganizar módulos** para mayor claridad (`auth`, `database`, `schemas`, `utils`, etc.).
 -  Separar lógica de autenticación (`jwt`, seguridad) en `sqliteplus/security.py`.
 -  Agregar `schemas.py` con modelos Pydantic para validación de entradas/salidas.
-




☐ 2. Aumentar cobertura de tests

-  Tests unitarios de funciones internas (ej. `generate_jwt`, `verify_jwt`, `SQL helpers`).
 -  Tests de errores de autenticación (token inválido, expirado, etc.).
 -  Tests con entradas inválidas (datos mal formateados, columnas incorrectas, etc.).
 -  Test de concurrencia con múltiples operaciones asincrónicas (stress test con `pytest-asyncio`).
-

🔧 3. Mejora funcional

-  Añadir **borrado de filas** por ID (`DELETE FROM {table} WHERE id = ?`).
 -  Añadir **edición/actualización** de datos (`UPDATE`).
 -  Permitir **filtros** en las consultas: `/fetch?filter=msg='hola'`.
 -  Exportar tablas como CSV o JSON descargables.
-

🛡️ 4. Seguridad avanzada

-  Protección contra SQL Injection: sanitizar bien `table_name`.
-  Añadir usuarios reales y roles (`admin` / `read-only`).
-  Soporte para refrescar tokens JWT y logout.

📖 5. Documentación y dev tools

- 📄 Mejorar `README.md` con ejemplos de uso real (`curl`, Python, Postman).
- 📄 Añadir `docs/` con guía avanzada de API y estructura del proyecto.
- 📄 Activar `pre-commit` para `black`, `isort`, `flake8`.
- 📄 Añadir badge de cobertura de tests (`pytest-cov` + `codecov`).

📦 6. Packaging profesional

- 📄 Añadir `__version__` dinámico desde `sqliteplus/__init__.py`.
- 📄 Generar `.whl` y `.tar.gz` automáticos con `build`.
- 📄 Publicación a PyPI estable (`v1.0.0` cuando esté completa la suite de funciones).
- 📄 Subir a GitHub (si lo deseas), con CI/CD para tests automáticos.

🔖 7. Extensiones futuras (opcional)

- 📄 Soporte para MySQL o PostgreSQL (opcional, con SQLAlchemy).
- 📄 Interfaz web visual con Flet o Tauri.
- 📄 Panel admin en FastAPI para gestionar usuarios, logs y bases.
- 📄 Añadir `WebSocket` para notificaciones o cambios en tiempo real.

¿Quieres que empecemos a trabajar ya en alguno de estos bloques? ¿O prefieres marcar prioridades?