Informe Técnico de Auditoría

Proyecto: edf_catalogotablas
Fecha: 18/06/2025

---

1. Arquitectura General

- **Framework:** Flask (Python 3.10)
- **Base de datos:** MongoDB (remoto, acceso centralizado)
- **Autenticación:** Sistema propio (bcrypt/scrypt), sesiones en disco, roles admin/usuario.
- **Estructura modular:** Uso de blueprints para separar rutas y lógica.
- **Frontend:** Jinja2, Bootstrap, JS propio.

Blueprints principales

- `admin_bp` — Administración de usuarios, catálogos, acciones masivas.
- `maintenance_bp` — Dashboard de mantenimiento, tareas programadas.
- `auth_bp` — Registro, login, recuperación y gestión de sesiones.
- `images_bp` — Gestión de imágenes (subida, descarga, borrado).
- `scripts_tools_bp` — Utilidades y herramientas avanzadas.

---

2. Dependencias principales

- **Flask**
- **pymongo** (MongoDB)
- **Flask-Login**
- **Flask-Session**
- **bcrypt** y/o **scrypt**
- **python-dotenv**
- **openpyxl** (Excel)
- **boto3** (S3, opcional)
- **gunicorn** (despliegue)
- **pytest** (tests)

---

3. Endpoints críticos

Administración

- `/admin/` — Dashboard admin
- `/admin/verify-users` — Verificación y acciones masivas de usuarios
- `/admin/bulk_user_action` — POST para acciones masivas (verificar/borrar usuarios)
- `/admin/logs/download-multiple` — Descarga de logs

Mantenimiento

- `/admin/maintenance` — Dashboard de tareas de mantenimiento
- `/admin/maintenance/api/run_task` — POST para ejecutar tareas programadas (limpieza logs, verificación disco/MongoDB)
- `/api/truncate-logs` — Limpieza avanzada de logs (por líneas o fecha)

Autenticación

- `/auth/login`, `/auth/logout`, `/auth/register` — Gestión de sesiones y usuarios

Catálogos y datos

- `/catalogs/` — CRUD de catálogos/tablas
- `/catalogs/import` — Importación de datos desde Excel/CSV
- `/catalogs/export` — Exportación de datos

Imágenes

- `/images/upload` — Subida de imágenes
- `/images/<id>` — Acceso/descarga de imágenes

---

4. Seguridad

- Decoradores `@login_required` y `@admin_required` en rutas críticas.
- Rate limiting en endpoints de mantenimiento.
- Validación de entradas y parámetros en backend.
- Exclusión de archivos sensibles mediante `.gitignore`.
- Auditoría de acciones críticas (logs con usuario, IP, timestamp).

---

5. Buenas prácticas y mantenimiento

- `.gitignore` exhaustivo: protege entornos virtuales, logs, imágenes subidas, credenciales, archivos temporales y basura.
- Script `clean_project.py` para limpieza y revisión de archivos basura.
- Separación clara de lógica backend/frontend y utilidades.
- Documentación y comentarios en puntos críticos del código.

---

6. Estado actual

- Proyecto **estable**, sin archivos sensibles ni basura en el repo.
- Listo para nuevas features, auditoría externa o despliegue.
- Rutina de limpieza y revisión previa al commit implementada.

Contacto responsable: - Desarrollador principal: edfrutos - Repositorio:edf_catalogotablas (GitHub)
