Catálogo de Tablas

Aplicación Flask para la gestión de catálogos de tablas con autenticación, gestión de usuarios y almacenamiento en MongoDB.

🏗️ Características Principales

- **Gestión de Usuarios**: Sistema de autenticación con roles (admin/user)
- **CRUD de Catálogos**: Creación, lectura, actualización y eliminación de catálogos
- **Importación/Exportación**: Soporte para Excel y CSV
- **Almacenamiento**: Imágenes en sistema de archivos local o S3
- **Monitoreo**: Sistema integrado de logging y monitoreo

🏗️ Estructura del Proyecto

```
.
├── app/                    # Código principal de la aplicación
│   ├── routes/             # Rutas de la aplicación
│   ├── static/             # Archivos estáticos (CSS, JS, imágenes)
│   ├── templates/          # Plantillas HTML
│   └── utils/              # Utilidades de la aplicación
├── config/                 # Archivos de configuración
├── docs/                   # Documentación del proyecto
├── models/                 # Modelos de datos
├── scripts/                # Scripts de utilidad
│   ├── db_utils/          # Utilidades de base de datos
│   ├── hooks/             # Hooks para PyInstaller
│   └── maintenance/       # Scripts de mantenimiento
├── tests/                  # Pruebas automatizadas
│   ├── integration/       # Pruebas de integración
│   └── unit/              # Pruebas unitarias
└── tools/                  # Herramientas de desarrollo
```

🏗️ Instalación

1. **Clonar el repositorio** `bash git clone [url-del-repositorio] cd edf_catalogotablas`
2. **Crear y activar entorno virtual** `bash python -m venv .venv source .venv/bin/activate # En Windows: .venv\Scripts\activate`
3. **Instalar dependencias** `bash pip install -r requirements.txt`
4. **Configurar variables de entorno** Copiar el archivo `.env.example` a `.env` y configurar según sea necesario.
5. **Iniciar la aplicación** `bash python run.py` O para producción: `bash gunicorn --bind 0.0.0.0:5000 wsgi:app`

🏗️ Dependencias Principales

- **Backend**: Flask, Flask-Login, Flask-PyMongo
- **Base de datos**: PyMongo
- **Autenticación**: bcrypt, Flask-Session
- **Procesamiento de datos**: pandas, openpyxl
- **Despliegue**: gunicorn

🏗️ Ejecución de Pruebas

Para ejecutar las pruebas unitarias:

```
pytest tests/unit/
```

🏗️ Build y despliegue de la app macOS EDFCatalogoTablas

Build local automático

1. Asegúrate de tener Python 3.10+ y[PyInstaller](PyInstaller) instalado.
2. Ejecuta el script de build: `bash ./build_mac_app.sh` Esto generará el bundle en `dist/EDFCatalogoTablas.app`.

Build automático en GitHub Actions (CI)

- Cada push a `master` lanzará un workflow que construye la app en macOS y deja el bundle como artefacto descargable.
- El workflow está en `.github/workflows/mac_build.yml`.

Instrucciones para usuarios finales

1. Descarga o clona el repositorio.
2. Si solo quieres la app, descarga `EDFCatalogoTablas.app` desde la carpeta `dist/` o desde los artefactos de GitHub Actions.
3. Haz doble clic para ejecutar. Si ves advertencia de seguridad, ve a Preferencias del Sistema → Seguridad y permite la app.
4. Para distribución profesional, firma la app con tu Apple ID.

Para pruebas de integración:

```
pytest tests/integration/
```

🛠️ Creación y distribución profesional de una app para macOS

Guía práctica y detallada para desarrollar, empaquetar, firmar, notarizar y distribuir una aplicación profesional para macOS a partir de un proyecto Python (ejemplo: EDFCatalogoTablas).

Índice

---

1. Requisitos previos

- **Apple ID** y Apple Developer Program (para firma y notarización)
- **Xcode** y Xcode Command Line Tools (`xcode-select --install`)
- **Python 3.10+**
- **Homebrew** (instalar)
- **PyInstaller** (`pip install pyinstaller`)
- (Opcional) create-dmg para crear DMG instalables

2. Preparación del entorno de desarrollo

1. **Clona el repositorio:** `bash git clone https://github.com/edfrutos/edf_catalogotablas.git cd edf_catalogotablas`
2. **Crea y activa un entorno virtual:** `bash python3 -m venv .venv source .venv/bin/activate`
3. **Instala las dependencias:** `bash pip install -r requirements.txt`
4. **Instala PyInstaller:** `bash pip install pyinstaller`

3. Empaquetado de la app (PyInstaller)

- Asegúrate de tener un archivo `.spec` bien configurado o usa el comando básico: `bash pyinstaller --onefile --windowed app.py`
- Para proyectos complejos, usa el archivo `edf_catalogotablas.spec` y ejecuta: `bash pyinstaller edf_catalogotablas.spec`
- El resultado estará en `dist/EDFCatalogoTablas.app`

Recursos estáticos y plantillas

- Asegúrate de incluir todos los archivos necesarios (templates, static) en el `.spec` usando `datas=[...]`.
- Documentación oficial PyInstaller

4. Automatización del build

Script Bash (`build_mac_app.sh`)

```
#!/bin/bash
set -e

echo "Limpiando builds anteriores..."
rm -rf build dist EDFCatalogoTablas.app

echo "Instalando dependencias..."
pip install -r requirements.txt

echo "Ejecutando PyInstaller..."
pyinstaller edf_catalogotablas.spec

echo "Build completado. El bundle está en dist/EDFCatalogoTablas.app"
```

Automatización en CI (GitHub Actions)

- `.github/workflows/mac_build.yml`:

```
name: Build macOS App

on:
  push:
    branches: [ master ]

jobs:
  build:
    runs-on: macos-latest
    steps:
      - uses: actions/checkout@v3
      - name: Set up Python
        uses: actions/setup-python@v4
        with:
          python-version: '3.10'
      - name: Install dependencies
        run: pip install -r requirements.txt pyinstaller
      - name: Build App
        run: pyinstaller edf_catalogotablas.spec
      - name: Upload artifact
        uses: actions/upload-artifact@v3
        with:
          name: EDFCatalogoTablas.app
          path: dist/EDFCatalogoTablas.app
```

5. Firma de la app

a) Solicita tu certificado

- Desde Apple Developer Certificates, crea un certificado **Developer ID Application** y añádelo a tu llavero (Keychain Access).

b) Firma la app

```
codesign --deep --force --verify --verbose \
  --sign "Developer ID Application: Tu Nombre (XXXXXXXXXX)" \
  dist/EDFCatalogoTablas.app
```

c) Verifica la firma

```
codesign --verify --deep --strict --verbose=2 dist/EDFCatalogoTablas.app
```

- Más info: Apple Code Signing Guide

6. Notarización con Apple

a) Comprime la app

```
cd dist
zip -r EDFCatalogoTablas.zip EDFCatalogoTablas.app
```

b) Sube a notarización

```
xcrun notarytool submit EDFCatalogoTablas.zip \
```

```
  --apple-id "tu-email@apple.com" \
  --team-id "XXXXXXXXXX" \
  --password "app-specific-password" \
  --wait
```

- Genera una contraseña específica de app.

c) Adjunta la notarización (“staple”)

```
xcrun stapler staple EDFCatalogoTablas.app
```

- Más info: Apple Notarization Guide

7. Creación de un instalador DMG

- Instala `create-dmg`: `bash brew install create-dmg`
- Crea el DMG: `bash create-dmg dist/EDFCatalogoTablas.app`
- Guía create-dmg

8. Distribución y buenas prácticas

- Comparte el `.dmg` o `.zip` firmado y notarizado.
- Incluye el archivo `LICENSE` (MIT, GPL, etc.) en el paquete.
- Proporciona instrucciones para usuarios sobre cómo permitir la app en Preferencias del Sistema si Gatekeeper la bloquea.
- Si distribuyes a gran escala, considera automatizar la subida a tu web o a servicios como AWS S3.

9. Recursos y enlaces útiles

- Apple Developer Program
- Documentación oficial notarización
- PyInstaller Docs
- create-dmg
- Automatización con GitHub Actions
- Soporte Apple: Contraseñas específicas de app

Autor: Eugenio De Frutos Sanchez — 2025
Licencia: MIT

Guía de Contribución

1. Haz un fork del proyecto
2. Crea una rama para tu característica (`git checkout -b feature/AmazingFeature`)
3. Haz commit de tus cambios (`git commit -m 'Add some AmazingFeature'`)
4. Haz push a la rama (`git push origin feature/AmazingFeature`)
5. Abre un Pull Request

Licencia

Este proyecto está bajo la Licencia MIT - ver el archivoLICENSE para más detalles.

Agradecimientos

- A todos los contribuyentes que han ayudado a mejorar este proyecto.
- A la comunidad de código abierto por las herramientas y bibliotecas utilizadas.