Backend Flask: AI Document Processing

Este archivo explica en detalle el funcionamiento del backend desarrollado con Flask para procesar documentos PDF, generar descripciones tipo ficha bibliográfica y extraer etiquetas clave utilizando modelos de lenguaje.

Propósito

Este backend permite recibir un archivo PDF desde el frontend, procesarlo mediante NLP y modelos LLM, y almacenar la información extraída para construir una base documental consultable.

Componentes principales

1. Importaciones clave

Se importan módulos para:

- Manejar la API REST (Flask)
- Recibir archivos (`request`, `secure_filename`)
- Permitir comunicación con el frontend (CORS)
- Llamar funciones auxiliares del módulo `processing.py`

```
from app.utils.processing import (
    extraer_texto_pdf,
    dividir_en_chunks,
    generar_embeddings,
    consultar_resumen,
    consultar_etiquetas
)
```

2. Rutas de almacenamiento

Se definen carpetas para almacenar archivos y datos:

- `data/docs/`: PDFs subidos
- `data/embeddings/`: chunks, embeddings, metadata, resumenes

3. Carga de datos previos

Al iniciar, el backend intenta cargar los datos previos guardados (si existen). Esto permite mantener una colección documental acumulativa persistente.

chunks = pickle.load(...) if os.path.exists(...) else []

4. Inicialización de la app Flask

app = Flask(__name__)
CORS(app)

5. Ruta `/procesar_pdf` (POST)

Esta es la ruta principal del backend.

Flujo completo:

1. Recibe el archivo PDF
2. Lo guarda en `data/docs/`
3. Extrae texto página por página
4. Divide el texto en chunks
5. Crea metadata por chunk (documento + página)
6. Genera embeddings con `SentenceTransformer`
7. Consulta un modelo (Ollama o Bedrock) para obtener:
    - Una descripción tipo ficha bibliográfica
    - 3 etiquetas clave
8. Almacena todo en memoria y disco:
    - `chunks.pkl`, `metadata.pkl`, `embeddings.npy`, `resumenes.json`
9. Devuelve JSON al frontend con:
    - nombre del documento
    - descripción
    - etiquetas
    - fuente de las etiquetas (ej. `bedrock`, `tf-idf`)

6. Ejecución del servidor

```
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5001, debug=True)
```

Lanza el backend accesible por `localhost:5001` o desde red local.

Integración con el frontend

El frontend envía un archivo PDF y recibe:

```
{
  "documento": "ejemplo.pdf",
  "resumen": "...",
  "etiquetas": ["agua", "proyecto", "canal"],
  "fuente_etiquetas": "bedrock"
}
```

Archivos generados

- `chunks.pkl`: lista de texto por fragmentos
- `metadata.pkl`: documento y página por chunk
- `embeddings.npy`: vectores de cada chunk
- `resumenes.json`: resumen y etiquetas por documento

Modelos utilizados

- **Embeddings**: `all-MiniLM-L6-v2`
- **LLM**: configurable entre:
    - Ollama (`llama3.1` local)
    - Amazon Bedrock (`meta.llama3-1-8b-instruct-v1:0`)

Última actualización: integrada con Bedrock y TF-IDF como respaldo.