Documentación Básica del Código

Estructura del Proyecto

El proyecto está organizado en varios directorios y archivos clave:

- `.dockerignore`, `.gitignore`, `.env.save`: Archivos de configuración para Docker y Git.
- `Dockerfile`, `docker-compose.yml`: Archivos de configuración para la contenedorización del proyecto.
- `README.md`: Documento de presentación y guía básica del proyecto.
- `config`, `data`, `docker`, `docs`, `frontend`, `notebooks`, `scripts`, `src`, `tests`: Directorios que contienen configuraciones, datos, documentación, interfaz de usuario, notebooks para experimentación, scripts, código fuente y pruebas, respectivamente.
- `pyproject.toml`, `pytest.ini`, `requirements.txt`, `setup.py`: Archivos de configuración para dependencias, pruebas y la instalación del paquete.
- `test_db_connection.py`: Script para probar la conexión con la base de datos.

Archivos Clave

`src/`: Contiene el código fuente principal del proyecto.

- `spiders/`: Contiene los spiders desarrollados para realizar el scraping.
- `items.py`: Define las estructuras de datos que se extraen.
- `middlewares.py`: Contiene middlewares personalizados para el procesamiento de solicitudes y respuestas.
- `pipelines.py`: Define los pipelines para procesar y almacenar los datos extraídos.
- `settings.py`: Archivo de configuración principal de Scrapy.

Objetivo del Proyecto

El objetivo del proyecto es desarrollar un sistema de scraping eficiente para extraer datos

relevantes de diversas fuentes web, procesarlos y almacenarlos en una base de datos para su posterior análisis.

Tecnologías Empleadas

- Lenguaje de Programación: Python
- Framework de Scraping: Scrapy
- Contenedorización: Docker
- Gestión de Dependencias: `requirements.txt`, `pyproject.toml`
- Automatización y Orquestación: `docker-compose`
- Pruebas: Pytest
- Almacenamiento de Datos: Base de datos SQL/NoSQL

Descripción Breve del Código

1. Scraping:
   - `spiders/`: Contiene las clases de los spiders que definen cómo navegar y extraer datos de las páginas web. Cada spider hereda de `scrapy.Spider` y sobrescribe el método `parse` para definir la lógica de extracción.
   - `items.py`: Define las estructuras de datos mediante clases que heredan de `scrapy.Item`, con campos definidos usando `scrapy.Field`.
   - `middlewares.py`: Middlewares personalizados para modificar solicitudes y respuestas durante el scraping.
   - `pipelines.py`: Procesa los datos extraídos y los almacena en la base de datos.

2. Configuración:
   - `settings.py`: Configuración de Scrapy, incluyendo ajustes de tiempo de espera, concurrencia, y

almacenamiento.

3. Contenedorización y Orquestación:

- Los archivos `Dockerfile` y `docker-compose.yml` permiten empaquetar la aplicación y sus dependencias en contenedores, facilitando la implementación y escalabilidad del proyecto.

4. Pruebas:

- Las pruebas automatizadas en `tests/` aseguran la funcionalidad correcta del código a lo largo del tiempo.