RAIn 2025

Trabajo Final

Fecha de Entrega: 03/06/2025

Autor: Jorge Justino Riera **LU:** 5575 **Carrera:** Ing. Inf **Plan:** 2010

Repositorio código fuente: https://github.com/nicrom8b/crashscraper

Informe

El presente trabajo se realizó para el tema: Análisis de noticias según un tema en particular, a partir de una búsqueda, realizada sobre 10 sitios de diarios digitales. En el cual seleccionamos el tema: Accidentes de tránsito.

Diarios de Jujuy:

1. El Tribuno de Jujuy: https://eltribunodejujuy.com/
2. Todo Jujuy: https://www.todojujuy.com/
3. Somos Jujuy: https://www.somosjujuy.com.ar/
4. Jujuy al Momento: https://www.jujuyalmomento.com/
5. Jujuy Dice: https://www.jujuydice.com.ar/
6. El Pregón: https://www.pregon.com.ar/
7. El Submarino Jujuy: https://elsubmarinojujuy.com.ar/

Diarios de Salta:

1. El Tribuno (Salta): https://www.eltribuno.com
2. Informate Salta: https://informatesalta.com.ar/
3. Qué Pasa Salta: https://www.quepasasalta.com.ar

CrashScraper

CrashScraper es una aplicación web integral diseñada para recopilar, procesar, clasificar y consultar noticias sobre accidentes de tránsito en el norte de Argentina. Utiliza un conjunto de *scrapers* para extraer información de diversos diarios digitales, la almacena en una base de datos y la expone a través de una API RESTful y una interfaz web interactiva.

El proyecto integra un modelo de lenguaje grande (LLM) para permitir consultas en lenguaje natural sobre los datos recopilados.

Tecnologías: FastAPI, SQLAlchemy, MariaDB, Docker, Ollama, NLTK, spaCy.

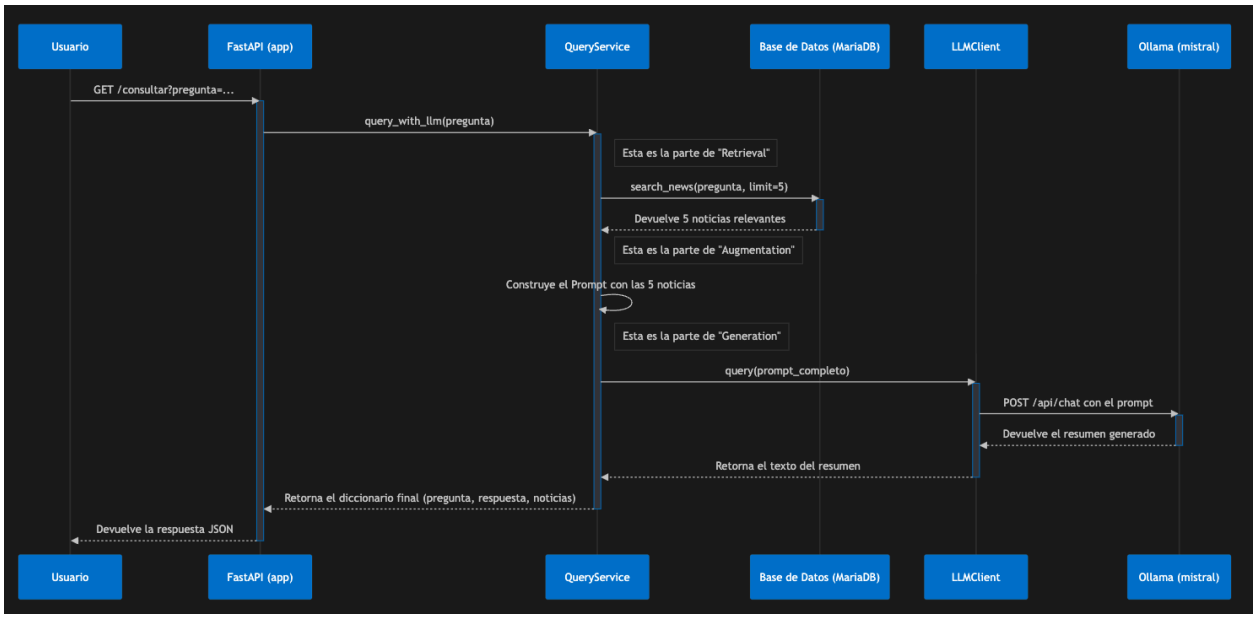
Características Principales

- **Scraping Multi-fuente:** Extrae noticias de 10 diarios digitales de Jujuy y Salta.
- **Procesamiento de Texto:** Limpia y normaliza el texto de las noticias para su análisis.
- **Clasificación Inteligente:** Utiliza un sistema de múltiples clasificadores (basados en *keywords*, *stemming*, lematización y ponderación) con un mecanismo de votación para determinar si una noticia trata sobre un accidente de tránsito.

- **API RESTful:** Provee *endpoints* para buscar, filtrar y obtener estadísticas de las noticias.
- **Interfaz Web Interactiva:** Un panel de control para visualizar estadísticas, una página de acciones para ejecutar *scrapers*/clasificadores y una interfaz de consulta.
- **Consultas con LLM:** Permite realizar preguntas en lenguaje natural (ej: "¿cuántos accidentes hubo en moto el fin de semana?") que son respondidas por un modelo de lenguaje grande (Mistral) a través de Ollama.
- **Orquestación con Docker:** Todo el ecosistema (API, base de datos, Ollama) se gestiona fácilmente con `docker-compose`.

Arquitectura y Flujo de Datos

1. **Ejecución de Scrapers:** El `Scraper Runner` (accesible desde la UI o por *script*) instancia y ejecuta todos los *scrapers* configurados.
2. **Extracción y Almacenamiento:** Cada *scraper* navega por un sitio de noticias, extrae los artículos relevantes y los guarda en la base de datos MariaDB. El medio (diario) se gestiona en una tabla separada (`media`).
3. **Clasificación:** El `Classifier Runner` (también ejecutable desde la UI o *script*) busca noticias sin clasificar en la base de datos.
4. **Análisis y Votación:** Cada noticia es analizada por cuatro clasificadores distintos. El resultado final se decide por mayoría de votos.
5. **Exposición vía API:** La API de FastAPI expone *endpoints* para acceder a los datos clasificados, obtener estadísticas y realizar búsquedas.
6. **Interfaz de Usuario:** La UI, construida con HTML, CSS y JavaScript, consume la API para mostrar *dashboards*, permitir la ejecución de tareas y realizar consultas.
7. **Consultas LLM:** Las preguntas en lenguaje natural se envían al `QueryService`, que interactúa con el `LLMClient` para obtener una respuesta del modelo Ollama.



Puesta en Marcha

REQUISITOS

- Docker y `docker-compose`
- `pipenv` (para desarrollo local)
- Python 3.11

USANDO DOCKER (MÉTODO RECOMENDADO)

Este comando levantará todos los servicios necesarios: la API, la base de datos y el servidor Ollama con el modelo Mistral.

```
Unset
# Construye las imágenes y levanta los contenedores

docker-compose up --build
```

La primera vez, Docker descargará la imagen de MariaDB, Ollama y el modelo Mistral, lo cual puede tardar varios minutos.

DESARROLLO LOCAL

Para desarrollo fuera de Docker, necesitarás una instancia local de MariaDB y Ollama.

```
Unset
# 1. Instalar dependencias

pipenv install


# 2. Configurar variables de entorno si es necesario

# (ej: en un archivo .env)
```

```
# 3. Ejecutar la aplicación

pipenv run uvicorn app.main:app --reload
```

Servicios Expuestos

- **API / Interfaz Web:** http://localhost:8000
- **MariaDB:** `localhost:3306`
- **Ollama API:** http://localhost:11434

Medio	**Clase del Scraper**	**Archivo**
El Tribuno de Jujuy	ElTribunoScraper	app/scrapers/eltribuno.py
Todo Jujuy	TodoJujuyScraper	app/scrapers/todojujuy.py
Somos Jujuy	SomosJujuyScraper	app/scrapers/somosjujuy.py
Jujuy Al Momento	JujuyAlMomentoScraper	app/scrapers/jujuyalmomento.py
Jujuy Dice	JujuyDiceScraper	app/scrapers/jujuydice.py
El Pregón (Jujuy)	PregonScraper	app/scrapers/pregon.py
El Submarino (Jujuy)	ElSubmarinoJujuyScraper	app/scrapers/elsubmarinojujuy.py
El Tribuno de Salta	ElTribunoSaltaScraper	app/scrapers/eltribuno_salta.py
Informate Salta	InformateSaltaScraper	app/scrapers/informate_salta.py
Qué Pasa Salta	QuePasaSaltaScraper	app/scrapers/quepasasalta.py

Para evitar falsos positivos, una noticia se clasifica como "accidente" solo si al menos dos de los siguientes clasificadores votan `True`:

- `es_accidente_simple`: Búsqueda simple de palabras clave.
- `es_accidente_stemmer`: Aplica *stemming* (raíz de palabras) a los términos de búsqueda para una coincidencia más amplia.
- `es_accidente_lemmatizer`: Usa lematización (lema de la palabra) con spaCy para una comprensión semántica más precisa.
- `es_accidente_ml_weighted`: Un modelo simple ponderado que asigna más peso a términos clave como "choque" o "siniestro vial".

Scripts Utilitarios

Ubicados en el directorio `scripts/`. Deben ejecutarse como módulos desde la raíz del proyecto.

- **Inicializar la DB:** Crea las tablas y carga datos de ejemplo.

```
Unset
```

- `pipenv run python -m scripts.init_db`

- **Limpiar la DB:** Elimina todas las noticias, manteniendo las tablas.

```
Unset
```

- `pipenv run python -m scripts.limpiar_db`

- **Ejecutar Scrapers y Clasificadores:**

```
Unset
```

- `pipenv run python -m app.scraper_runner`