PRÁCTICA 2

Extracción de datos masivos de internet

Israel Aznar Villegas

Contenido

Descripción general de la API

En esta práctica, se han utilizado los datos extraídos de consultas realizadas en la práctica anterior a RAWG. RAWG proporciona acceso a una de las bases de datos más grandes de videojuegos.

Tecnologías

MongoDB

Los datos han sido guardados en una mongodb por su sencillez y por la experiencia que he adquirido con otros proyectos.

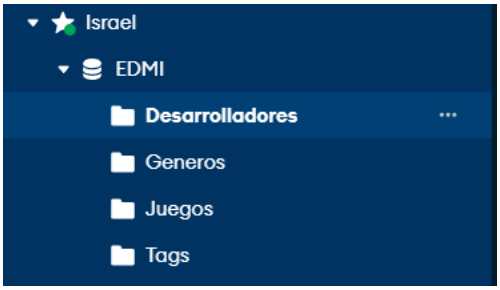

Figura 1: Base de datos creada en MongoDB

Los datos además fueron transformados ya que, el json que devuelve la API, trae información innecesaria por lo que una vez insertados, mediante un script de Python, se ha modificado valores como sustituir los valores nulos por strings vacíos o dividir los documentos con listas de objetos en, directamente, otros documentos.

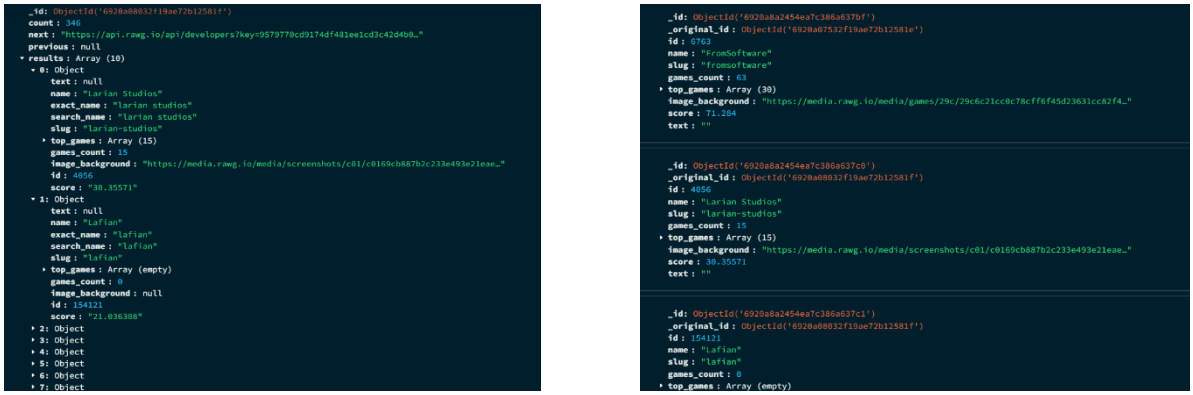

Figura 2: Ejemplo de cambio en los datos

FastAPI

Para el desarrollo de la API, se ha utilizado FastAPI, que facilita la implementación otorgando una interfaz gráfica con la que podremos interactuar de manera sencilla con esta. Para poder usarla hay que tener instaladas las siguientes librerías:

```
fastapi==0.122.0
motor==3.7.1
pydantic==2.12.4
uvicorn==0.38.0
```

Figura 3: Dependencias FastAPI

FastAPI nos recomienda usarla estructurando el código en 4 partes:

- App.py: donde se definen los endpoint y se crea la aplicación.
- Conexión.py: donde se conecta a la base de datos.
- Funciones.py: donde se crean las funciones que tratan con los datos de la base de datos.
- Modelos.py: donde se recogen los atributos que necesitamos de cada clase de los datos.

Una vez ejecutemos app.py, tendremos que acceder desde nuestro ordenador a la ruta https://localhost:8000/docs y veremos la siguiente interfaz:

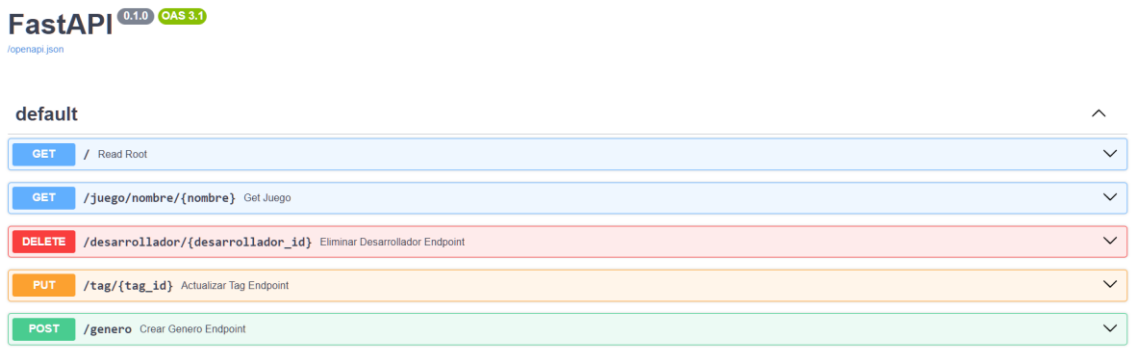

Figura 4: FastAPI

Al interactuar con ella podremos interactuar con la API creada y los datos.

Endpoints implementados

Los endpoints implementados son los siguientes:

- Raíz (/). Este endpoint solo sirve para dar un mensaje al entrar en la dirección.
- Juego (juego/nombre/{nombre}). Este endpoint sirve para acceder a los juegos según el nombre que indiquemos.
- Desarrollador (desarrollador/{desarrollador_id}). Este endpoint sirve para acceder a los desarrolladores según el id que indiquemos
- Tag (tag/{tag_id}). Este endpoint sirve para acceder a los tags según el id que indiquemos
- Género (genero). Este endpoint sirve para crear nuevos géneros.

Explicación requisitos cumplidos

A continuación, se expondrá aquellos aspectos cumplidos que se pedían en la práctica:

Requisitos	¿Dónde se cumplen?
Al menos 3 endpoints	En total hay 4 endpoints más el raíz
Al menos uno de cada tipo de petición	Se usan los cuatros tipos. Una para cada endpoint
Persistencia en BBDD	La información se mantiene en el servidor
Api usa HTTPS	Se utilizan certificados para poder acceder
Autorización	Se ha creado un endpoint extra para la autorización
Paginación	Con el get se puede elegir la página y el tamaño de la página
Rate limit	Se ha establecido que no se puede usar más de 10 peticiones en 10 segundos

Además, se ha desarrollado algunos puntos extras como son:

- La autenticación se usa mediante una prueba de identidad y devuelve un token por cada usuario.
- Para el https se crea un certificado una vez que sirve para cada vez que se inicia.

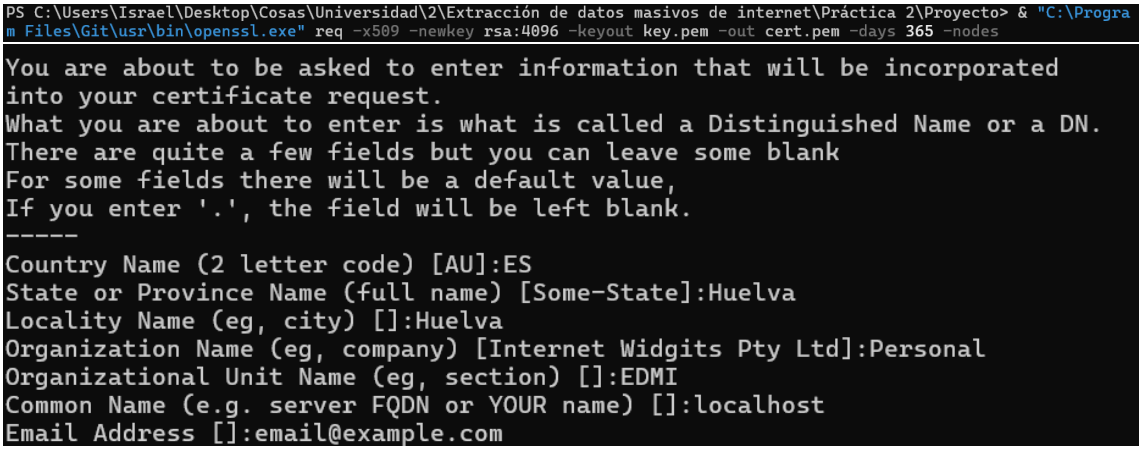

Figura 5: Creando el certificado

Figura 6: FastAPI iniciada con HTTPS

- Se devuelve códigos de error según el problema que haya.

Decisiones de diseño/retos

Durante el desarrollo de la API se tomaron varias decisiones de diseño importantes. Se eligió FastAPI por su rapidez y tipado fuerte además de la experiencia usándola, y MongoDB por la flexibilidad de su modelo documental. Para garantizar seguridad se implementó autenticación basada en tokens y almacenamiento de contraseñas mediante hashing bcrypt. También se

añadió paginación para limitar las respuestas, conexiones HTTPS para proteger las comunicaciones y un sistema de rate limiting para evitar abusos. Además, se diseñó un mecanismo para mantener persistencia y coherencia entre las distintas colecciones relacionadas en la base de datos. Los principales retos fueron los problemas de compatibilidad con bcrypt, la actualización coherente de datos relacionados y la correcta configuración de HTTPS y del limitador de peticiones.

Conclusión

En conclusión, la implementación de esta API ha permitido integrar de forma segura y eficiente el acceso a los datos, aplicando buenas prácticas como autenticación, paginación, uso de HTTPS y control de peticiones. El uso de FastAPI y MongoDB ha facilitado un desarrollo ágil y flexible, manteniendo un diseño claro y escalable. A pesar de algunos retos técnicos, especialmente en la gestión de seguridad y compatibilidad de dependencias, el sistema final ofrece una solución sólida, fiable y preparada para ampliaciones futuras.