Proyecto Big Data: Wailor

Descripción General

Este proyecto de Big Data tiene como objetivo crear una base de datos realmente grande para poder realizar consultas con grandes cantidades de datos y medir su rendimiento.

Utiliza Python, MongoDB y Docker para procesar y almacenar gran cantidad de datos extraidos en gran parte de PokeApi y en menor escala de ChatGPT(entrenadores, gimnasios y profesores).

Requisitos

- pymongo
- requests

Estructura del Proyecto

- `src/`
  - `validators/`
    - `pokemon.json` : Validación de la colección pokemon.
  - `sprites/` sprites estraidos de mongoDB con formato .png, .gif o .svg
  - `data/` datos iniciales añadidos a MongoDB.
    - `*_api_url.json` : contienen la lista de urls a las que hacer fetch para descargar los datos de PokeApi.
    - `*_data.json` : datos completos de extraidos de las urls anteriores.
    - `gyms.json` : conjunto de gimnasios extraidos junto con su especilidad y lider.
    - `professors.json` : conjunto de profesores extraidos junto con su especialidad.
    - `types.json` : tipos extraidos junto con sus debilidades.
    - `trainers.json` : conjunto de entrenadores extraidos.
    - `villains.json` : conjunto de villanos de la saga pokemon extraidos.
  - `data_extraction/`
    - `extrac_{collection}_api_url.py` : extraer de PokeApi todos las urls que seguidamente usaremos para mayor comodidad.
    - `get_all_{collection}_data.py` : extraer todos los datos de PokeApi de las urls dadas.
    - `get_villaions_from_trainers.py` : "extraer" de los trainers aquellos que son villanos para su propia colección.
  - `data_insertion/`
    - `insert_mongo.py` : Inserción de datos en MongoDB, insertar a traves de una función las colecciones que deseemos de los datos extraidos.
  - `relationships/`
    - `update__{collection}_{collection}.py` : Creación de las relaciones entre dos colecciones dadas.
  - `images/`
    - `read_imgs_pokemon.py` : Buscamos los sprites de cada pokemon de la colección sprites y los descargamos localmente con gridfs.
    - `read_imgs_sprites.py` : Descargamos localmente los sprites de gridfs.
    - `extrac_and_update_pokemon_sprites.py` : Buscamos los links de descarga de los sprites que se encuentran en cada pokemon de la colección pokemon(urls), los descargamos y los guardamos con gridfs en mongo db, despues de esto creamos una relación entre los pokemons y los sprites que acabamos de guardar en MongoDB
- `output/` : Resultados generados.