|  |  |  |  |
| --- | --- | --- | --- |
|  | 22/05/2020 | |  |
| |  |  | | --- | --- | |  |  | |  |  | |  |  | |  |  | | | |  |
| Práctica MongoDB  *Tecnologías de Bases de Datos* | | | |
|  |  | |  |  |  | | --- | --- | --- | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |  |  |  | | |
|  |  | **Grupo6:** Raúl C0lino Singh, María Flores García, Karol Lisowski, Christian Taidi Santana | |

Práctica MongoDB

Tecnologías de Bases de Datos

Para esta práctica se nos ha pedido que, a partir de un conjunto de datos en formato .csv y aplicando los conocimientos adquiridos en la asignatura, realicemos tres procesos consistentes en; una transformación de los datos de formato .csv a un formato que pueda ser utilizado por MongoDB tal como JSON, un volcado de dichos datos en formato JSON a una base de datos MongoDB, y un diseño de 10 consultas a realizar a los datos almacenados de forma que se pueda obtener información relevante, todo esto teniendo en cuenta que se necesita diseñar una estructura para los datos en base a las consultas que se realizarán.

# Introducción

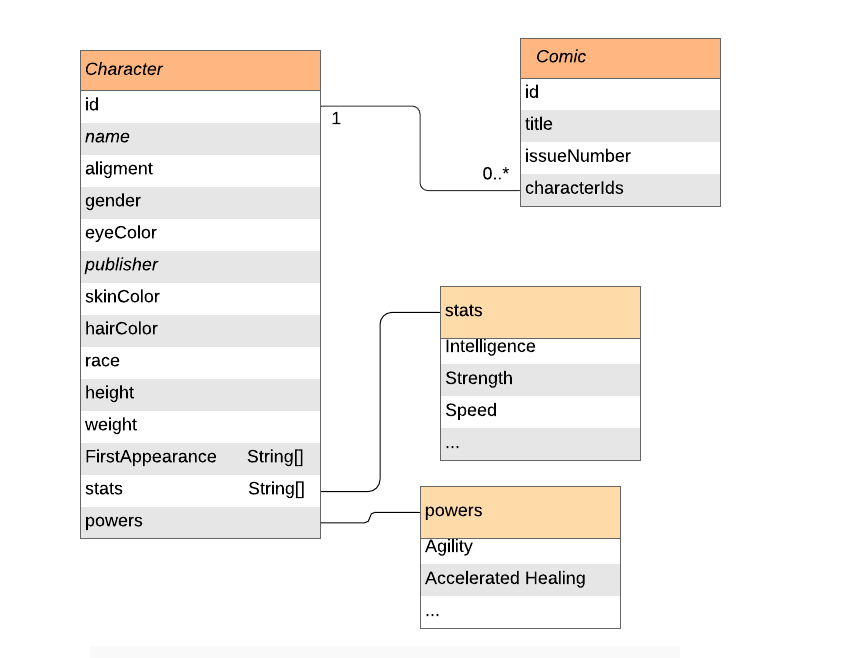
Para la realización de esta práctica hemos pasado por varios procesos creativos para converger en una idea común y desarrollar en conjunto los sistemas necesarios para completar las tareas propuestas.

En primer lugar, decidimos que la tecnología principal a utilizar sería Java para facilitarnos las tareas ya que existen librerías que permiten transformar una serie de datos en formato .csv a JSON y también para volcar datos a MongoDB, por tanto, el código que se ha usado para desarrollar las partes de la práctica está en Java.

Posteriormente determinamos qué consultas deberíamos hacer al conjunto de datos que sean interesantes. Tras proponer una serie de ideas, votamos las más apropiadas teniendo en cuenta que deberíamos elegir al menos 3 consultas complejas, con esto podemos establecer una estructura más apropiada para los datos.

Finalmente desglosamos las tareas necesarias para completar el trabajo y las realizamos.

# Estructura

Estructura de la BBDD MONGO

Character y comic son colecciones independientes, cada comic tiene una lista de ids de personaje para relacionarse con la colección Characters.

Stats y powers sonsubdocumentos embebidos en cada personaje.

Powers es una lista de super poderes que tiene el personaje, si no tiene un super poder, no aparecerá en la lista.

# Consultas

1. Lista de comics en los que aparece un personaje.

Muestra una lista con todos los nombres de personajes y recibe como parámetro el nombre de un personaje (introducido por teclado) y devuelve todos los cómics en los que aparece. El nombre debe coincidir al completo.

1. Lista de frecuencias de color de ojos.

Devuelve una lista ordenada descendentemente por frecuencia de aparición del color de ojos de todos los personajes.

1. Cómic con más personajes.

Devuelve el título del comic con más personajes y el número de personajes que aparecen en él.

1. Editorial con más personajes calvos.

Devuelve el nombre de la editorial con más personajes calvos y el número de personajes calvos que contiene.

1. Ratio Hombres/Mujeres en un universo.

Muestra una opción por pantalla, si se introduce el número “1”, se escoge como universo “Marvel” y para cualquier otro valor se escoge “DC Comics” y devuelve el número de hombres, el número de mujeres y el ratio entre ellos.

1. Personaje más odiado.

El personaje más odiado es el más antiguo que aparece menos veces y está muerto.

1. Personaje más alto de un cómic/saga.

Recibe como parámetro el nombre de un cómic (introducido por teclado) y devuelve el personaje más alto. El título del cómic tiene que coincidir.

1. Comparativa de inteligencia media entre dos razas.

Muestra todas las razas existentes y recibe como parámetro dos razas devolviendo la que tiene más inteligencia media con dicho valor, esta consulta se considera compleja ya que se tiene que hacer una separación por raza, y calcular la media de inteligencia de cada una en base al subdocumento *“stats”* de cada personaje y el número de personajes de cada raza.

1. Primer personaje con superpoderes.

Devuelve el primer personaje que aparece en un cómic con superpoderes, esta consulta es compleja por tener que unir las colecciones de *“comic“* y *“character”*  y comprobar por cada personaje que va apareciendo en los cómics, cuál tiene el subdocumento *“powers”* mediante la operación *“lookup”.*

1. Personajes que aparecen en una colección de cómics.

Una colección de cómics es una serie de números distintos con un mismo nombre (Ej. *Jessica Jones* tendría *Jessica Jones (2016) #1*, *Jessica Jones (2016) #2* …), recibe como parámetro el nombre de la colección y devuelve los personajes que aparecen en todos los cómics que la componen, esta consulta se considera compleja debido a que una saga puede estar compuesta por varios cómics y puede que haya personajes que no salgan en todos los cómics de una saga, por tanto se busca en todos los cómics de una saga uniendo la lista de personajes de cada comic con el personaje al que corresponde, tras esto se agrupa por nombre de personaje y se devuelven únicamente los nombres de los personajes que aparecen.

# Fase 1

Para nosotros la Fase 1 y la Fase 2 han ido de la mano ya que todos los datos pasan por *Java* desde el .csv hasta *Mongo*, hemos desarrollado un código encargado de que durante la transformación se asignen los valores adecuados a cada entidad, la parte menos eficiente quizás sea la asignación de *ids* de personajes a cada comic, aunque en todos los casos hemos hecho uso de *Parallel Streams* para aprovechar la concurrencia a la hora de procesar los datos, por tanto no hacemos el procesamiento en los archivos *JSON* sino que directamente lo hacemos en memoria, lo que agiliza el proceso al evitar accesos a disco o base de datos.

Por tanto, existe un fragmento de código que se encarga de transformar los datos de .csv a objetos java, y de objetos java a *JSON* tras haber sido procesados, tras la transformación a JSON se añaden a la base de datos.

# Fase 3

Para la Fase 3 hemos utilizado el mismo programa desarrollado para la Fase 1 al que hemos incluido un menú por entrada de texto en el que se pueden elegir las distintas consultas e introducir los distintos valores necesarios para realizarlas, también hay una opción del menú que realiza las fases 1 y 2 llamada *“Data Dump”* para no realizar el volcado cada vez que se ejecute el programa, de esta forma solo se ejecuta cuando es necesario.

Para realizar todas las operaciones hemos creado una clase encargada de realizar los accesos a base de datos y de construir las consultas para lanzarlas a MongoDB, cada método ejecutará una consulta de las propuestas en el menú, con una operación extra que muestra todas las razas o nombres de personaje existentes en el conjunto de datos (para la consulta de calcular que raza tiene mayor inteligencia media y los comics en los que aparece un personaje respectivamente).