Tecnologías de Bases de Datos

Grado en Ingeniería de Software

Curso 2019/20 (Evaluación Continua)

Práctica Obligatoria

El objetivo de la Práctica consiste en manejar distintas Tecnologías de Bases de Datos para solucionar un problema habitual en el contexto de gestión de datos: procesar una mezcla de fuentes de datos *distintas*, pero *relacionadas*, y de las que hay que extraer conclusiones. Es decir: existen datos que permiten llegar a esas conclusiones, pero esos datos **no** están (aún) organizados en un almacén, ni siguen un único modelo de datos.

Dicho de otro modo: la idea es que se accede a un (pequeño) *data lake*, que contiene documentos en formato semiestructurado, que proceden de orígenes diferentes, pero cuyos contenidos están directa o directamente interrelacionados. La práctica consiste en recoger esta información, seleccionar y relacionar las partes relevantes, y construir un sistema que permita realizar **consultas** integradas sobre dicha información.

El sistema final deberá tener como base central una Base de Datos NoSQL, dado que sus características las hacen especialmente adecuadas para un problema de este tipo. La Base de Datos elegida es MongoDB, ya que se adapta perfectamente a esta situación. Además, por razones obvias, MongoDB es la **única** tecnología "obligatoria" de esta Práctica; cada Grupo de Prácticas podrá elegir *libremente* qué otras tecnologías quiere utilizar.

Para no hacerlo excesivamente complicado, el *data lake* tratado en la práctica se basa en una serie de datos sobre *personajes de cómics*. Concretamente, personajes de cómic de superhéroes, con cierta preferencia por Marvel. Hay también datos mezclados de otras editoriales (sobre todo DC, pero también Wildstorm, Dark Horse, etc.).

Se parte de un *dataset* formado por datos procedentes de distinto origen (en realidad, recopilados por la misma persona, pero no elaborados por la misma persona). Además, se incluye una relación de los cómics seleccionados por el autor original, de los que proceden estos datos. Los personajes (en distintos bloques) están relacionados dentro de su universo ficticio, así que la complejidad de sus relaciones simula la de una red social. Se ha demostrado que la complejidad del Universo Marvel tiene las características de una *red libre de escala*, por lo que sirve como ejemplo.

El *dataset* está simulando (de manera **muy** simplificada) la estructura de la información que podría haber en un *data lake* (en un contexto ficticio, en este caso). Los ficheros están agrupados en bloques claramente definidos, de orígenes diferentes; las relaciones

entre los personajes simulan una red social. El listado de cómics es comparable a un conjunto concreto de transacciones dentro de un sistema en el que, lógicamente, habrá muchas más que no son listadas.

El objetivo, como se ha dicho, es **extraer** información del *dataset* de modo que pueda ser consultado. Esto se resume, sencillamente, en tres etapas:

1. **Procesar** la información inicial, cargándola de modo que sea tratable (por ejemplo, en JSON).
2. **Volcar** esta información a MongoDB, siguiendo un "esquema de datos" (flexible) previamente diseñado
3. **Construir** un sistema que permita consultar la información, siendo capaces de realizar diversas *consultas complejas*, en las que se relacionen datos procedentes de distintas colecciones. Este *sistema* se define de la manera más amplia posible, y puede estar diseñado del modo que el grupo considere más conveniente.

Descripción inicial del *Dataset*

El *dataset* original consta de *siete* ficheros de datos en formato CSV. Se trata aquí del CSV estándar, en formato internacional: los ficheros *tienen* encabezamiento y los distintos "campos" están separados por comas (¡cuidado con las comas "internas" a un campo!).

Los ficheros son los siguientes, y tienen estos contenidos aproximados:

- **Characters** (csv): pares *(identificador, nombre)* para 1.169 personajes de cómic.
- **Comics** (csv): secuencias *(identificador, comic, número)* para los 41.226 cómics de la colección del autor.
- **Characters To Comics** (csv): pares *(id character, id comic)* indicando qué personajes de la primera lista aparecen en los cómics de la segunda, hasta un total de 75.257 emparejamientos.
- **Characters Info** (csv): Información detallada sobre 734 personajes, indicando características "físicas" del personaje y la editorial de origen.
- **Superheroes Power Matrix** (csv): Relación de superpoderes, en formato de matriz de ocurrencias, para 667 de estos personajes.
- **Characters Stats** (csv): Más información detallada sobre 610 de estos personajes, indicando características "cuantitativas" del personaje.
- **Marvel-DC Characters** (csv): Información detallada (estado, características físicas, universo) para 39.648 personajes de estas dos editoriales.

Estos listados no son disjuntos, están relacionados, y están parcialmente agrupados. Como puede verse, los tres primeros ficheros están relacionados y muestran información sobre los personajes presentes en los cómics del autor del *dataset*. Los tres siguientes están relacionados, y muestran información *in-universe* (dentro de su universo ficticio) para varios personajes, entrando en detalles concretos: aspecto físico, superpoderes, características cuantitativas; incluso la editorial. El último fichero es

independiente, y tiene mucha más información (tanto en variedad como en tamaño) que los dos anteriores, pero una buena parte de ella no podría relacionarse con el resto.

Debe tenerse en cuenta que los identificadores no están diseñados para ser coherentes. *Characters* y *Characters-To-Comics* utilizan los mismos, y permiten relacionarse con los *Comics*. Los identificadores de *Characters Info* no son los mismos que éstos, pero los nombres (*Name*) sí son los mismos que en *Characters Stats* y *Power Matrix*. Los identificadores de *Marvel-DC Characters* son independientes de los otros dos.

Respecto a este último, comentar que se puede realizar un tratamiento estándar:

- Todo personaje cuyo nombre termina en "*(Earth-616)*" es de Marvel, y puede sustituirse por el nombre convencional. Es decir, "*Captain America (Earth-616)*" es el mismo que "*Captain America*".
- Todo personaje cuyo nombre termina en "*(New Earth)*" es de DC, y puede sustituirse por el nombre convencional. Es decir, "*Superman (New Earth)*" es el mismo que "*Superman*".

Por el contrario, el resto de los datos entre paréntesis sí son significativos en ese fichero. Por ejemplo, "*Hawkeye (Clint Barton)*" es un personaje distinto de "*Hawkeye (Kate Bishop)*", aunque ambos están relacionados.

Finalmente, mencionar que **no todos los datos** tienen por qué ser considerados significativos para el tratamiento que desee hacer el grupo. Es decir, no es *obligatorio* que todos los datos presentes en estos siete ficheros se usen en la solución final. Esto no es casual ni forzado, sino que es parte de la realidad de un *data lake*: siempre hay datos que, siendo relevantes en su contexto, no tienen por qué serlo para el tratamiento actual. En este caso, el fichero que es aparentemente independiente de los demás puede ser la clave para relacionarlos; pero hay en éste muchos elementos que pueden descartarse.

Objetivos a realizar

Como se ha indicado, el proceso se debe realizar en tres etapas (que deben documentarse de manera suficiente; no se restringe aquí la forma en que debe documentarse, sino que cada grupo debe explicar cómo lo ha acometido, de modo que considere que ha proporcionado información *suficiente*).

Fase 1: Procesamiento

Consiste en examinar los ficheros, comprender su estructura, entender cómo se relacionan, y *decidir* qué datos se consideran relevantes y cuáles no, dependiendo de la orientación que se le quiera dar a la práctica (dependiente, sobre todo, del tratamiento que se pretenda realizar en las Fases 2 y 3).

Una vez examinados, deben ser *procesados*, con el objetivo de volcarlos posteriormente a la Base de Datos. Cualquier sistema de procesamiento adecuado puede ser utilizado. Así, se permite:

- Transformarlos a ficheros JSON (por cualquier medio: con un programa, una transformación, tratamiento directo…)
- Cargarlos en memoria, en una estructura de datos.
- Procesarlos como *data frames* en los lenguajes de programación que los admiten.
- Volcarlos a un sistema de gestión de flujos de datos (Debezium, Elasticsearch) para volcarlos desde ahí a la Fase 2. [Aunque este enfoque es *excesivo* a todas luces.]
- Diseñar un programa que los vuelque en la base de datos a medida que los va procesando (fusionando por tanto la Fase 1 con la Fase 2).

No es *obligatorio* volcar toda la información del *dataset* en la Base de Datos. Puede descartarse la información que no se pretenda utilizar, y "filtrarse" previamente del modo que se desee.

Fase 2: Volcado

Una vez se ha diseñado la estructura (el "esquema") que se plantea para la Base de Datos, se deben volcar los datos procesados en la Fase anterior a una base de datos MongoDB.

Se creará una base de datos específica para este tratamiento, en la que se incluirán todas las *colecciones* con los distintos datos que vayan a procesarse. La base de datos se llamará como el Grupo de Prácticas.

- Es decir, si el grupo es el grupo 6, la base de datos se llamará **`grupo6`**.

Es decir, esencialmente se permite:

- Volcarlos desde ficheros JSON, utilizando la utilidad `mongoimport`.
- Volcarlos desde un programa, en el que se recoja de la fuente previamente procesada, es decir:
  - Escribirlos a MongoDB según se van leyendo de los ficheros originales.
  - Escribirlos a MongoDB según se van leyendo los ficheros JSON.
  - Escribirlos a MongoDB según se van leyendo de memoria.
  - Escribirlos a MongoDB según se van leyendo del *data frame*.
  - Etc.

Como parte de este proceso, se debe haber **diseñado un esquema** para los contenidos de las distintas colecciones (lo que puede incluir documentos incrustados, o no), y teniendo en cuenta las relaciones entre las distintas colecciones (tal como se ha mostrado en clase).

Este esquema ha de ser explicado en la documentación. Se puede usar un "diagrama" **similar** a un *diagrama relacional* típico de una Base de Datos relacional, teniendo en cuenta, *por supuesto*, que los modelos son distintos.

Lo más importante de este modelo es mostrar cómo se han considerado las relaciones (1:1, 1:N, N:M). Asimismo, en el caso de que haya documentos con distinta estructura dentro de la misma colección, debe explicarse cuál es esta estructura. Si esto se considera relevante, se debe explicar *por qué*.

Fase 3: Construcción

Esta fase es el objetivo de la Práctica. Las dos Fases anteriores se planifican en función de las necesidades de ésta.

El objetivo de esta Fase es **construir** un sistema que permita implementar estas consultas. Se permite que este sistema se construya de **cualquier** manera posible (*sin utilizar medios externos al propio sistema*). [1]

El objetivo es que el sistema sea capaz de responder a las consultas que el propio Grupo haya planteado.

Es decir, una vez construido el sistema, el usuario debería poder usarlo para resolver las consultas. Obviamente, si las consultas se hacen usando una base de datos (la propia MongoDB, por ejemplo), debe proporcionarse el código de estas consultas. Si se hace mediante un programa, debe proporcionarse el programa.

Cada consulta debe poder tratarse por separado, de modo que se vea que realmente está contestando a la pregunta que se plantea. [2]

---

[1] Con el objeto de evitar ambigüedades: "sin utilizar medios externos al propio sistema" significa que el sistema de consulta ha de ser definido en su totalidad. Es decir, que cualquier *soporte* que se quisiera usar en el sistema de consulta **debe formar parte** del mismo, y *como tal*, debe describirse como parte de la solución.

Por ejemplo, si el sistema de consulta utilizase una base de datos SQLite, debe documentarse de qué versión se trata, cómo se ha configurado, y cualesquiera otras cuestiones involucradas que sean relevantes para su uso; así como, lógicamente, proporcionar el fichero *.sqlite* correspondiente.

[2] Por supuesto, esto **no** excluye que se haga un único sistema o programa que resuelva todas las consultas. Lo único que indica esta instrucción es que las consultas deben poder *realizarse* por separado, y deben poder *mostrar sus resultados* por separado. Si se describen como consultas aisladas, esto se consigue directamente. Si se hace en un programa, este debería permitir alguna forma de separarlas; por ejemplo, haciendo una pausa entre una y otra, o invocándolas a través de un menú; etc.

Consultas

El objetivo de la Práctica es que cada grupo diseñe **diez** consultas que tengan sentido en el contexto de este *dataset*.

Estas consultas deben enumerarse en la Memoria de manera totalmente cualitativa, es decir, diciendo qué se pretende en cada una de ellas. Para después resolverla en la Fase 3.

De las **diez** consultas al menos **tres** deberían ser razonablemente *complejas*; es decir, su complejidad es tal que expresar la pregunta correspondiente (en español) podría requerir alguna explicación.

La Memoria debe indicar explícitamente cuáles de las diez consultas se consideran *complejas*. Sólo se piden **tres**, pero obviamente pueden ser más.

Ejemplo 1:

Por ejemplo, una consulta *sencilla* sería decir, simplemente:

- *¿En cuántos de los comics del dataset aparece el Capitán América?*

Y esto es *todo* lo que habría que indicar sobre ella (y cualquiera de las otras nueve) en la **primera parte** de la Memoria.

Pero luego, en la Fase 3, debería resolverse la consulta, y aquí ya había que entrar en detalles.

Así, ***por ejemplo:***

En MongoDB, asumiendo una colección análoga a los ficheros correspondientes (es decir, si no se han combinado previamente de algún modo), habría que hacer, como mínimo, dos consultas, a saber:

```
db.characters.find("name": "Captain America")
```

De la que se obtendría el *id* correspondiente (en este caso, *1009220*), para después:

```
db.charactersToComics.find("characterID": "1009220").count()
```

Pero la respuesta correcta en la Fase 3 **no sería** indicar estas dos consultas, ya que son dos consultas *independientes*, y por tanto **no resuelven** la consulta. La respuesta correcta debería *combinar* estas dos consultas,

- Sea haciendo un programa que recoge el *characterID* de la primera y lo utiliza en la segunda,
- Sea haciendo una consulta compuesta mediante el *aggregation framework* para hacer este *join*,

- Sea usando cualquier otro medio de los que el propio MongoDB (o el sistema construido en la Fase 3) hace posibles.

Ejemplo 2:

Un ejemplo de consulta *compleja* podría ser:

Decimos que dos personajes se conocen si han aparecido en el mismo cómic. Por ejemplo, el Capitán América y el Soldado de Invierno (Winter Soldier) se conocen, porque ambos aparecen en el comic Cap Transport (2005) #12.

- *Para cada uno de los personajes incluidos en characters, ¿**cuántos** conocidos tiene cada uno de ellos?*

Como puede verse, la pregunta necesita una (breve) explicación previa para poder formularse.

En este caso, la respuesta no es necesariamente mucho más difícil que la anterior, a pesar de las apariencias, salvo por el hecho de que no se refiere a un solo personaje, sino que debe aplicarse a todos ellos.

Todas las consultas que se consideren *complejas* deben implicar, como mínimo, a más de una colección.

Igualmente, se valorará positivamente que varias de las consultas complejas incluyan agrupamientos (y lógicamente, funciones agregadas).

Tecnología

Se puede usar **cualquier** tecnología que el grupo de prácticas conozca y considere oportuna. Se puede elegir desde el lenguaje de programación (Java, Javascript, Python…) a cualquier otro sistema auxiliar (por ejemplo, una librería concreta). Como ya se ha indicado, la única tecnología obligatoria es MongoDB.

En la **Fase 1**, los ficheros deberán ser procesados utilizando *cualquier medio* que el grupo estime oportuno, de entre las muchas opciones disponibles:

- Programáticamente (por ejemplo, en Python), en distintos grados de detalle;
- Usando aplicaciones capaces de transformar datos (por ejemplo, OpenRefine);
- O aplicaciones capaces de interpretar el formato (por ejemplo, Excel, aunque *con cuidado*);
- Sistemas de captura de datos (CDC), aunque en este contexto son sin duda excesivos.
- Etc.

Se permite el uso de aplicaciones *online* (durante el proceso de transformación), pero se debe considerar que *(a)* es la peor solución posible, ya que no da ningún control; y *(b)* la mayoría tienden a perder mucha información

Por supuesto, se prefieren las dos primeras alternativas con mucha diferencia. Respecto a la primera, prácticamente cualquier lenguaje de programación capaz de manejar texto es perfectamente capaz de leer ficheros CSV y manejar ficheros JSON. Aunque las muchas librerías de Python y Javascript hacen esto especialmente fácil, no hay en realidad preferencia por ningún lenguaje en concreto. Cada Grupo debería utilizar su favorito, aunque el único motivo para elegirlo fuera la comodidad.

En la **Fase 2** se asume el uso de MongoDB Community Edition, en su versión actual (se permite el uso de versiones antiguas, siempre que se indique de manera explícita; y se agradecería un breve comentario sobre el motivo de esta elección).

Igualmente, en la **Fase 3** se pueden usarse los medios que se consideren convenientes, en este caso con mucha más libertad.

- Se permite, por supuesto, realizar las consultas directamente en MongoDB o con cualquiera de las aplicaciones de su ecosistema.
- Se permite, también, utilizar cualquiera de los *conectores* de MongoDB con aplicaciones externas (siempre que estas aplicaciones externas sean accesibles para los estudiantes y los profesores, tanto en la fase de desarrollo como en la de evaluación).
    - Por ejemplo, se permite de manera explícita el uso del *MongoDB Connector for Business Intelligence*. En tal caso, ha de documentarse explícitamente cómo debe ser usado.
    - Igualmente, en ese caso, debe indicarse de manera explícita qué solución de BI (Power BI, Tableau…) se utilizará para su procesamiento.
        - Obviamente, no hace falta señalar que dicha solución de BI deberá poder ser usada *legalmente*. Lógicamente, no se considera el uso de versiones de prueba.
    - Igualmente, en ese caso, debe indicarse de manera explícita cómo se enlaza dicha solución de BI con el citado conector de BI.
    - Igualmente, y asumiendo que las consultas se realizan desde el sistema de BI, dichas consultas deberán documentarse como si éste fuera el sistema final.
- Se permite, también, utilizar una base de datos auxiliar. Es decir, si se plantean consultas muy complejas y se prefiere utilizar una base de datos relacional, se puede *trasvasar* la información de MongoDB a la citada base de datos, y realizar las consultas en esta última, pero:
    - Deberá describirse en detalle el modelo de datos de la base de datos de destino, justificando su correspondencia con el original.

  - Deberá describirse también el método (o programa) de transvase, justificando cualquier pérdida de datos (o no) durante este proceso de migración.
  - Lógicamente, deberán documentarse todas las consultas en la sintaxis.
  - Igualmente, como se ha indicado más arriba, deberá proporcionarse un volcado de la base de datos de destino (por ejemplo, el fichero *.sqlite* correspondiente).
- Se permite, también, el uso de sistemas de "trasvase automático" de MongoDB a otra base de datos, en un sentido similar a lo mencionado en el punto anterior.

  - Por ejemplo, MoSQL (código abierto, en Github) trasvasa datos de MongoDB a PostgreSQL, permitiendo hacer las consultas sobre esta última. Una solución de este tipo sería válida.

    - No obstante, en primer lugar, debe lograrse que esta solución *funcione* como se espera de ella.
      - Por ejemplo, el proyecto MoSQL no está activo (*discontinued*), pero se permite utilizar versiones "antiguas" si de este modo se logra que funcionen.
    - En segundo lugar, se debe documentar adecuadamente su instalación, despliegue y uso.
    - En tercer lugar, obviamente, también se debe documentar adecuadamente qué versión de la base de datos destino (PostgreSQL, en este ejemplo) se está usando, y cómo se configura.
    - En cuarto lugar, y de manera análoga, también se debe proporcionar el volcado de la base de datos (de nuevo, PostgreSQL, en este ejemplo) generada a partir de MongoDB.
- Se permite, también, como ya se ha indicado en apartados anteriores, construir un programa (en cualquier lenguaje de programación) que utilice la interfaz de programación de MongoDB para acceder a la base de datos y realizar las consultas descritas.

  - Lógicamente, se permite utilizar cualquier soporte auxiliar (propio del lenguaje de programación o mediante una librería) que facilite esta tarea al programa.
  - Lógicamente, si se utiliza un soporte de este tipo, debe explicarse de qué modo se utiliza y cuál es su objetivo.
  - Lógicamente, y como también se ha indicado más arriba, se debe proporcionar el código fuente completo de este programa (preferiblemente agrupado en su propio directorio).
- Se permite, en resumen, cualquier otro medio análogo a estos, o incluso alguna combinación de varios.

El objetivo de este último apartado es permitir el máximo margen de maniobra. Se valora tanto la originalidad como el sentido común: aunque pueda ser muy original usar

un sistema de gestión de flujos de datos (y se admite), probablemente no es el medio más sensato, porque complica el sistema excesivamente. No obstante, esta elección podría tener sentido, de nuevo, si es adecuadamente justificada (y documentada).

Nota: Cualquier duda al respecto sobre si determinada tecnología es válida o no (la respuesta, en principio, es *sí*, siempre que no se sobrepasen unos límites razonables) debe ser consultada con los profesores de la asignatura.

Normas de Entrega

Fechas

La fecha límite de entrega de la Práctica será el **viernes, 22 de Mayo** de 2020, a las 23:55 horas.

La entrega se realizará a través de la tarea habilitada a tal efecto en el Aula Virtual (pestaña de "Evaluación" de la asignatura). Se admiten entregas desde el mismo momento de publicación de esta tarea.

Sólo será necesaria la entrega por parte de **uno** de los miembros del Grupo de Prácticas.

En caso de haber más de una entrega por parte del mismo Grupo, se tendrá en cuenta únicamente la **última** de ellas, y todas las demás se considerarán *no válidas*.

Contenido

La entrega consistirá en un **único fichero comprimido** (se admiten los formatos ZIP, 7Z y TGZ; se prefiere el primero), que habrá de contener los siguientes elementos:

1. Una Memoria de la Práctica (**en PDF**), en la que se resumen los aspectos conceptuales del trabajo realizado en la misma.

2. Un directorio **`.\mongo`**, que contendrá un volcado de la Base de Datos.

3. Un directorio **`.\codigo`**, que contendrá todo el código relativo a la Fase 3 de la práctica. Por ejemplo, si se ha optado por instrumentarlas como consultas, contendrá un *script* o serie de *scripts* incluyendo las consultas. Si se ha optado por hacer un programa, contendrá el código (en su propio directorio, adecuadamente estructurado), así como todos los elementos adicionales necesarios para que éste funcione. [3]

---

[3] Si el programa utiliza alguna librería externa, no es necesario incluirla si está fácilmente disponible localizable en Internet. Sí debería incluirse en el caso de que para su descarga sea necesario realizar algún tipo de suscripción o registro previo (siendo ambos extremos que se desaconsejan). No es necesario incluir el *software* propio de

4. *Opcionalmente*, un directorio **`.\files`**, que contendrá la transformación de los ficheros CSV procesados en la Fase 1. Por ejemplo, si se ha optado por generar ficheros JSON, estos deberán incluirse en este directorio.

5. *Opcionalmente*, un directorio **`.\extra`**, que contendrá cualquier contenido adicional que el Grupo considere necesario.

Respecto a cada uno de ellos:

- La Memoria tendrá las siguientes partes diferenciadas (no se fija extensión mínima ni máxima para ninguna de ellas):

  - *Introducción*. Breve sección que describe el enfoque general de la práctica, resumiendo lo que se ha hecho durante las tres Fases. No tiene que ser muy larga: pero debe indicar con claridad qué es lo que debe esperarse encontrar en el resto del trabajo.

  - *Estructura*. Descripción de la estructura de los datos que se almacenarán en la base de datos `grupoX` en MongoDB. En resumen, el "esquema de datos".

    - Como se ha indicado, se puede usar un diagrama similar a un diagrama relacional (teniendo en cuenta las diferencias implícitas a ambos tipos de modelos).
    - Se puede optar también por describirlas simplemente como texto.

    En ambos casos resulta especialmente significativo mencionar **(a)** las relaciones entre colecciones de datos (1:1, 1:N, N:M), y en particular qué opción se ha preferido en cada caso para almacenarlas (documentos incrustados, referencias, referencias cruzadas, etc.); y **(b)** qué datos han decidido descartarse y por qué, en el caso de que se haya hecho.

  - *Listado de Consultas*. Consiste **únicamente** en el listado de consultas, enumeradas como una secuencia de *preguntas*, acompañadas si es necesario de una breve introducción/explicación. Las consultas deben estar *numeradas*, y debe indicarse cuáles se consideran complejas y cuáles no.

  - *Fase 1*: explicación de cómo se pasa de los datos de los ficheros CSV al esquema descrito en la sección de *Estructura*. Es decir, desarrollar las *correspondencias* entre los ficheros de entrada y el esquema incluido en Mongo. **No es necesario ser exhaustivo**: pero la transición de los datos debería quedar clara. Si se utilizan ficheros intermedios (por ejemplo, ficheros JSON), debería mostrarse cómo se estructuran estos ficheros, y/o cómo éstos ayudan a la transformación.

MongoDB, aunque sí será necesario indicar qué versión *exacta* del mismo se está utilizando. En caso contrario, se asumirá la última existente.

- *Fase 3*: explicación de cómo se construye el sistema de consulta.
    - Si consta simplemente de consultas MongoDB, deberá explicar brevemente aquéllas consultas que merezcan explicación.
    - Si consiste en un programa, debería describirse brevemente el enfoque seguido para su desarrollo. Se deben explicar aquí los aspectos que merezcan comentario, a criterio del Grupo.
    - Si se usan otros medios (SQLite, Connector for BI, MosQL, etc.), aquí debería describirse someramente el enfoque utilizado, y los detalles que resulten relevantes.
    - Si el sistema es lo bastante complejo como para merecerlo, debería describirse su arquitectura.

    En general, el enfoque es bastante sencillo: si el Grupo quiere que se valore algún aspecto específico de su trabajo, no sólo debe *construirlo* (implementarlo), sino que también debe describirlo, de modo que el evaluador pueda ser consciente de su existencia y pueda valorar su complejidad.

- Con respecto al volcado de la Base de Datos mencionado en el punto 2 (directorio **mongo**), se realizará del siguiente modo:
    - Para cada una de las colecciones contenidas en la base de datos (`grupoX`), se exportará a JSON usando la instrucción siguiente, asumiendo (por ejemplo) la existencia de una colección de personajes (*characters*):

```
mongoexport --collection=characters --db=grupoX --out=characters.json
```

- Respecto a los aspectos relativos a la Fase 3; al disponerse de mayor margen de maniobra, se incluirán (y documentarán, en su caso) del modo que el Grupo considere más adecuado.

En general, no se pretende que cada Grupo tenga que proporcionar una Memoria muy extensa: simplemente, el objetivo es que no haya ningún aspecto *significativo* de la práctica que no sea mencionado, para que pueda ser adecuadamente valorado.