Gestión de la Información en la Web
Curso 2016-17
Práctica *Aggregation Pipeline*

Fecha de entrega: miércoles 11 de enero de 2017, 13:55h

Entrega de la práctica
La entrega de la práctica se realizará a través del Campus Virtual de la asignatura mediante un fichero `grupoXX.zip` donde `XX` es el numero de grupo. Este fichero constará de un fichero `consultas.py` con el código del servidor web, cuyo esqueleto se puede descargar del Campus Virtual. Además del servidor web, el fichero **ZIP** contendrá las vistas necesarias para mostrar los datos adecuadamente (si las habéis utilizado).

Lenguaje de programación
Python 2.7 o **3.5**.

Calificación
Cada ruta tiene un peso del 20 %.

Declaración de autoría e integridad
Todos los ficheros entregados contendrán una cabecera en la que se indique la asignatura, la práctica, el grupo y los autores. Esta cabecera también contendrá la siguiente declaración de integridad:

> *(Nombres completos de los autores)* declaramos que esta solución es fruto exclusivamente de nuestro trabajo personal. No hemos sido ayudados por ninguna otra persona ni hemos obtenido la solución de fuentes externas, y tampoco hemos compartido nuestra solución con nadie. Declaramos además que no hemos realizado de manera deshonesta ninguna otra actividad que pueda mejorar nuestros resultados ni perjudicar los resultados de los demás.

No se corregirá ningún fichero que no venga acompañado de dicha cabecera.

En esta práctica consultaremos un servidor MongoDB desde Python mediante la librería **pymongo** y consultas avanzadas que usen el *Aggregation Pipeline*. Para ello implementaremos un servidor web que conteste a distintas peticiones **GET en el puerto 8080**, se conecte al servidor MongoDB y muestre los resultados obtenidos convenientemente formateados como página HTML.

El primer paso será descargar los ficheros con las colecciones `usuarios.json` y `pedidos.json`. Debéis importar estos datos en la colección **usuarios** y **pedidos** respectivamente dentro de la base de datos **giw**, utilizando para ello `mongoimport`. Tendréis que inspeccionar ambas colecciones para conocer el esquema de los documentos (todos siguen el mismo patrón). **Es muy importante respetar el nombre de la base de datos y de las colecciones, pues para la corrección se usarán esos nombres**.

Para realizar la práctica debéis descargar el esqueleto básico `consultas.py` del servidor web del Campus Virtual y usarlo como base. Este esqueleto incluye las 5 rutas en las que el servidor web debe responder a peticiones GET. No se permite cambiar las rutas, el puerto, el método HTTP ni añadir nuevas rutas.

1. `/top_countries` [20 %]

Recibe un parámetro `n`, y genera una página HTML mostrando los `n` países con más número de usuarios junto a dicho número de usuarios. En caso de empate a usuarios, los países se ordenan alfabéticamente. Lo resultados se deben mostrar en una tabla con 2 columnas: la primera contendrá el nombre del país y la segunda el número de usuarios. Justo debajo de la tabla debe aparecer un mensaje indicando el número de resultados devueltos por esta consulta.

Primeros 3 países según el número de usuarios:
`http://localhost:8080/top_countries?n=3`

2. `/products` [20 %]

Recibe un parámetro **min** representando un precio mínimo. Genera una página HTML que muestra, por cada producto cuyo precio es igual o superior a **min**, el número de unidades vendidas entre todos los pedidos junto con su precio unitario (que será el mismo en todos los pedidos en los que aparezca). Los resultados se muestran en una tabla de 3 columnas: nombre de producto, número de unidades vendidas y finalmente precio unitario. Justo debajo de la tabla debe aparecer un mensaje indicando el número de resultados devueltos por esta consulta.

Unidades vendidas de los productos que cuestan 2,34 *EUR o más:*
`http://localhost:8080/products?min=2.34`

3. `/age_range` [20 %]

Recibe un parámetro **min** representando un numero mínimo de usuarios. Genera una página HTML que muestra, por cada país que tiene más de **min** usuarios, el rango de edades, es decir, la diferencia entre la `edad máxima` y la

`edad mínima`. Estos resultados deben aparecer ordenados de mayor a menor rango de edades, y en caso de empate, de manera alfabética por el nombre del país. Los resultados se muestran en una tabla de 2 columnas: nombre de país y rango de edades. Bajo de la tabla debe aparecer un mensaje indicando el número de resultados devueltos por esta consulta.

Rango de edades de los países con al menos 80 *usuarios:*
`http://localhost:8080/age_range?min=80`

4. `/avg_lines` [20 %]

Genera una página HTML que muestra, por cada país, el número promedio de líneas que tienen los pedidos realizados por usuarios de dicho país. Los resultados se muestran en una tabla de 2 columnas: nombre de país y número de líneas promedio de sus pedidos. Bajo de la tabla debe aparecer un mensaje indicando el número de resultados devueltos por esta consulta.

Promedio de líneas de pedidos por países:
`http://localhost:8080/avg_lines`

5. `/total_country` [20 %]

Recibe como parámetro `c` el nombre de un país, y calcula el total de euros gastado en todos los pedidos realizados por usuarios de dicho país. Los resultados se muestran en una tabla de 2 columnas: nombre de país y total de euros gastados. Se debe mostrar el número de resultados devueltos por esta consulta bajo la tabla.

Total de euros gastados por usuarios alemanes:
`http://localhost:8080/total_country?c=Alemania`