Gestión de la Información en la Web
Curso 2017-18
Práctica Consultas MongoDB

Fecha de entrega: lunes 11 de diciembre de 2016, 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 3.5 o superior.

Calificación
Ver nota máxima en cada apartado. Además de la corrección se valorará la **concisión y claridad del código**, la incorporación de **comentarios** explicativos y la **eficiencia del código y de las consultas** tanto en tiempo como en memoria.

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 usando Python mediante la librería **pymongo**. Para ello implementaremos un servidor web sencillo que conteste a distintas peticiones **GET en el puerto 8080**, se conecte al servidor MongoDB y genera una página HTML con los resultados obtenidos.

El primer paso será descargar el fichero con la colección `usuarios.json`. Debéis importar estos datos en la colección **usuarios** dentro de la base de datos **giw** utilizando `mongoimport`. Tendréis que inspeccionar dicha colección 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 la colección, 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 6 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. `/find_users` [1pt]

Busca usuarios que tengan determinados valores en sus campos **a la vez**. Esta consulta acepta como parámetros cualquier combinación de `name`, `surname` y `birthday`; por ejemplo:

Todos los usuarios de nombre Luz:
`http://localhost:8080/find_users?name=Luz`

*Todos los usuarios de nombre Luz **Y** apellido Romero:*
`http://localhost:8080/find_users?name=Luz&surname=Romero`

Si se pasa algún argumento no contemplado se mostrará un mensaje de error indicando los argumentos inválidos, como por ejemplo en la consulta:
`http://localhost:8080/find_users?name=Luz&food=hotdog`

Si no existe ningún error el resultado será una página web que contiene una tabla. Cada fila será un usuario y sus columnas serán:

- Nombre de usuario
- e-mail
- Página web
- Tarjeta crédito (todos los datos combinados: número y fecha de expiración)
- Hash de contraseña
- Nombre
- Apellido
- Dirección (todos los datos combinados: calle, número, país, código postal)
- Aficiones (lista con todas las aficiones)
- Fecha de nacimiento

Encima de la tabla aparecerá un texto indicando el **número de resultados encontrados**, que puede ser 0.

2. `/find_email_birthdate` [1pt]

Muestra el identificador, el email y la fecha de nacimiento de todos los usuarios nacidos entre dos fechas pasadas como parámetro: `from` y `to` (**ambas fechas incluidas en el rango**). Los resultados se deben mostrar en una tabla de **únicamente 3 columnas**: *_id*, *Email* y *Fecha de nacimiento*. También se debe mostrar en la parte superior de la página el número de usuarios encontrados, justo encima de la tabla.

Usuarios nacidos entre el 1 de enero de 1973 y el 31 de diciembre de 1990:
`http://localhost:8080/find_email_birthdate?from=1973-01-01&to=1990-12-31`

3. `/find_country_likes_limit_sorted` [2pt]

Encuentra los primeros usuarios de un determinado país que tienen (al menos) una serie de aficiones que se pasan como parámetro. Estos usuarios se muestran ordenados por fecha de nacimiento según el orden especificado. Esta consulta recibe 4 parámetros:

- `country`: país seleccionado.
- `likes`: lista de aficiones separadas por comas, sin espacios.
- `limit`: número de usuarios a mostrar como máximo.
- `ord`: orden a aplicar sobre la fecha de nacimiento, que puede ser ascendente (`asc`) o descendente (`desc`).

El resultado debe ser una página web con el número de resultados seguido de una tabla con los datos de los usuarios como en el apartado 1.

*4 primeros usuarios de Irlanda a los que les gustan las películas **Y** los animales, ordenados por fecha de nacimiento ascendente:*
`http://localhost:8080/find_country_likes_limit_sorted?country=Irlanda&likes=movies,animals&limit=4&ord=asc`

4. `/find_birth_month` [2pt]

Muestra los usuarios nacidos en un mes concreto pasado como parámetro `month`. Estos resultados se deben mostrar de mayor a menor edad. El parámetro `month` puede recibir únicamente los valores `enero`, `febrero`, `marzo`, `abril`, `mayo`, `junio`, `julio`, `agosto`, `septiembre`, `octubre`, `noviembre` y `diciembre`.

El resultado debe ser una página web con el número de resultados seguido de una tabla con los datos de los usuarios como en el apartado 1.

Usuarios nacidos en abril:
`http://localhost:8080/find_birth_month?month=abril`

5. `/find_likes_not_ending` [2pt]

Muestra los usuarios que **no tienen ninguna afición que termina en el sufijo** que se pasa como parámetro `ending`. Esta consulta **no debe tener en cuenta las mayúsculas o minúsculas**, es decir, *painting* es una afición que termina en el sufijo *iNg*.

El resultado debe ser una página web con el número de resultados seguido de una tabla con los datos de los usuarios como en el apartado 1.

*Usuarios que no tienen ninguna afición que termina en **S**:*
`http://localhost:8080/find_likes_not_ending?ending=S`

6. `/find_leap_year` [2pt]

Muestra los usuarios que nacieron en un año bisiesto (`https://es.wikipedia.org/wiki/A%C3%B1o_bisiesto`) y que además su tarjeta de crédito expira en un año pasado como parámetro `exp`. Para esta consulta no se permite utilizar listados precalculados de años bisiestos sino que se debe utilizar la difinición de año bisiesto. El parámetro `exp` estará formado por 2 cifras, tal y como aparece en los documentos JSON.

El resultado debe ser una página web con el número de resultados seguido de una tabla con los datos de los usuarios como en el apartado 1.

*Usuarios que nacieron en años bisiestos y que su tarjeta caduca en el año **20**:*
`http://localhost:8080/find_leap_year?exp=20`