RedES

Se desea desarrollar una red social de ámbito educativo y laboral para el mercado español que permita poner en contacto a personas conocidas o con intereses o aptitudes similares. Nosotros participaremos en este proyecto dentro del equipo de 'Data'. Este equipo será el encargado de diseñar e implementar el sistema de bases de datos que tenga la capacidad de gestionar la información necesaria para el correcto funcionamiento de la plataforma, así como el diseño e implementación de los modelos y la interfaz de mapeo que permita la comunicación con las bases de datos.

Esta plataforma se desarrollará en cuatro fases diferenciadas.

- 1ª fase: Modelos y conexión a base de datos general (Práctica 1).
- 3ª fase: Consultas/informes (Práctica 2).
- 4ª fase: Cache y sesiones (Práctica 3).
- 4ª fase: Búsqueda de conexiones (Práctica 4).

La primera fase del desarrollo de la plataforma se centrará en el modelado de las entidades persona, empresa y centro educativo. Se pide diseñar e implementar una base de datos que albergue la información asociada a dichas entidades, así como el diseño e implementación de un ODM (clases *Model* y *ModelCursor*) específico que administre la información asociada a cada entidad y la comunicación con la base de datos.

Se prevé que el listado de atributos pueda aumentar a lo largo del tiempo y sea necesario incluirlos en la base de datos. Adicionalmente, se ha de tener en cuenta que es posible que alguno de estos atributos no sea común a todos los contactos y por tanto no tenga valor para alguno de ellos. Se proporcionará un archivo de configuración YAML del ODM que permita definir de forma dinámica los modelos y especificar aquellos atributos obligatorios (que tienen que estar para cada documento), admitidos (atributos válidos/aceptados que no tienen por qué estar en cada documento) de cada modelo e índices necesarios. Para simplificar la práctica, solo se realizará la comprobación sobre variables admitidas y requeridas sobre el primer nivel de variables y no sobre variables anidadas en diccionarios o listas.

Para los atributos de tipo dirección, se ha de añadir un nuevo atributo al modelo con nombre "nombre_del_atributo_de_dirección" + "_loc" que almacene la geolocalización en formato punto de *GeoJSON*. Por ejemplo, para atributo de nombre "dirección", se añadirá un nuevo atributo llamado "direccion_loc" que albergará el geojson. Se utilizará y terminará de implementar la función *getLocationPoint* que debe devolver un punto GeoJson a partir de la dirección proporcionada. Esta función utiliza una API pública con acceso limitado y por tanto no se pueden realizar consultas de forma masiva. Consultar haciendo uso de *sleeps* y guardar los resultados.

Los atributos y datos para las persona, empresas y centros educativos no han sido informados y serán proporcionados por el estudiante según su criterio y teniendo en cuenta las consultas solicitadas en la segunda parte del enunciado. Alguna información de relevancia sería: centros educativos o empresas en los que ha estudiado o trabajado una persona, con sus respectivos periodos (años) y grados/puestos, dirección de una persona, centro educativo o empresa, descripción del perfil de una persona.

Se proporciona dos ficheros con plantillas para la práctica a desarrollar. Un primer fichero YAML con un ejemplo de definición de los modelos y sus respectivas variables admitidas y requeridas. Un segundo fichero python con la plantilla de las clases *Model*, *ModelCursor* y las funciones initApp y *getLocationPoint* que tiene que desarrollar el alumno.

Requisitos de la práctica

Diseñar e implementar el/los modelo/s que gestione la información relativa a las entidades producto, cliente, almacén y compra. Dicho ODM debe asegurar que los datos que se proveen son correctos (están los datos requeridos y no hay más datos que los admitidos).

El ODM debe contar con:

- Una clase *Model* que implemente:
  - Un método *save* que permita almacenar los datos en BBDD de modo que si es un documento nuevo lo inserte y si es un documento ya existente, solo actualice aquellos campos que hayan sido modificados. Se deberá almacenar las coordenadas junto a la dirección cuando exista el dato dirección para poder realizar consultas de geolocalización.
  - Un método *delete* que permita eliminar un documento de la base de datos.
  - Un método *find* de clase que permita realizar consultas en BBDD y devuelva un objeto de la clase *ModelCursor* que permita devolver el resultado de la consulta en formato modelo.
    - No será necesario implementar una nueva metodología de consulta, sino que el método simplemente recibirá el mismo argumento *filter* que recibe el método *find* de *pymongo*.
    - El método *find()* deberá devolver un iterador de modelos ModelCursor.
  - Un método *__init__* que permita inicializar los datos en memoria del modelo. Se deberá realizar un control sobre los datos requeridos y admitidos.
  - Un método *__setattr__* que permita modificar los datos en memoria del modelo. Se deberá realizar un control sobre los datos admitidos. Se deberá realizar un control de los atributos modificados de modo que solo se envíe la información nueva a la BBDD.
  - Un método init__class que permita inicializar los índices en caso de ser necesario.
- Una clase *ModelCursor* que implemente:
  - Un método *__iter__* que devuelva un iterador que recorre los elementos del cursor devuelto por el método *find* de *pymongo* y permita devolver los documentos en forma de objetos modelo.
    - Utilizar yield para generar el iterador

        - Utilizar la funcion next del *CommandCursor* para obtener el siguiente documento del cursor
        - Utilizar alive del *CommandCursor* para comprobar si existen mas documentos.
- Una función *initApp* que declare e inicialice tantas clases que hereden de la clase abstracta Model como colecciones sean necesarias para las entidades persona, empresa y centro educativo.
- Una función getLocationPoint que obtenga las coordenadas de una dirección en formato geojson.Point haciendo uso de la API de geopy.

Consideraciones adicionales:

- Las localizaciones deberán ser implementadas siguiendo el formato geoJSON y deberá utilizarse indexado 2dsphere.

Normativa de realización, entrega y evaluación de la práctica:

- La práctica se realizará y entregará en grupos de hasta dos integrantes.
- La práctica se realizará en python y haciendo uso de mongoDB.
- La práctica deberá ir acompañada de las pruebas necesarias para comprobar el buen funcionamiento de la funcionalidad que se pide. Estas comprobaciones se realizarán en el main de ODM.py
- La entrega se compondrá de un único fichero ZIP, que contendrá el directorio del proyecto con los archivos ODM.py, models.yml y un archivo nombre_collecion.json por cada colección en la base de datos que contengan un volcado de la base de datos generada.
- El directorio y fichero zip será renombrado con el número de la práctica P#, seguido del número del grupo G#, y finalmente seguido con el nombre y el primer apellido de los alumnos integrantes del grupo, separados mediante guiones bajos '_'.

  Ejemplo: *P1_G25_Quijote_de_la_Mancha-Sancho_Panza.zip*
- Se considerará suspensa toda práctica cuyo fichero comprimido no contenga los ficheros indicados.
- Las prácticas entregadas fuera de plazo serán calificadas sobre 9. Por cada día de retraso en la entrega se reducirá el rango de calificación en 0,2 puntos.
- Cualquier sospecha de COPIA entre dos o más prácticas o de código obtenido en internet derivará en la calificación de 0 para todos los alumnos involucrados en la evaluación en curso y la siguiente. En caso de que el alumno tenga duda de que el código pueda ser susceptible de ser entendido como copia, consultar con el profesor antes de la entrega.