UCLM
UNIVERSIDAD DE CASTILLA-LA MANCHA

Escuela Superior de Informática

Graduado en Ingeniería Informática
Aplicaciones Distribuidas en Internet

– Práctica 2018/19: Twitter –

Puntuación por objetivos/tareas: 45 (Máximo: 25)
Fecha límite de entrega: 15 Enero 2019 (Necesaria defensa)
Método de trabajo: Individual, por parejas o por tríos

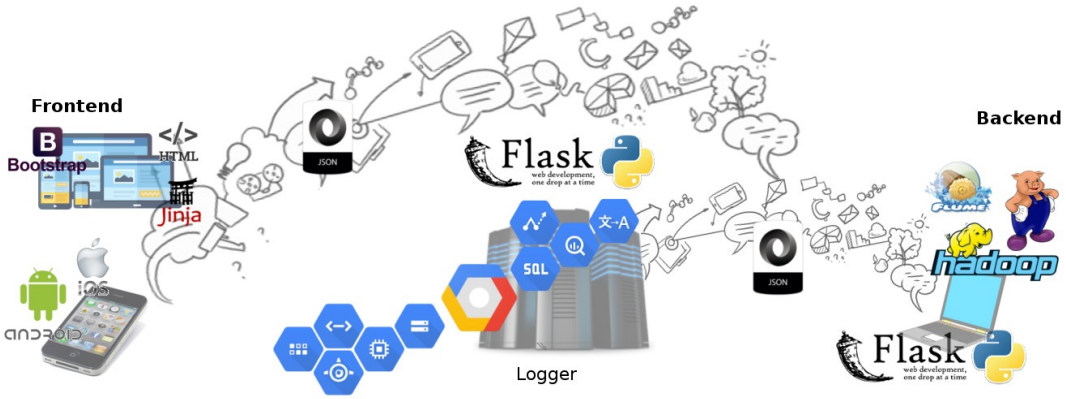
La práctica se divide en tres partes bien diferenciadas como se muestra en la figura: un backend, un logger y un frontend. Cada una de estas partes podrán ejecutarse en un nodos diferentes, por ejemplo el backend y frontend en un pc personal, mientras que el logger podría desplegarse en los servidores de Google.

- El ***backend*** realizará las tareas más costosas en términos computacionales. Este backend ofrecerá una API REST que permitirá el acceso a los servicios ofrecidos y que son detallados en este documento.

- El servidor ***logger*** será un servicio web que utilizará las tecnologías ofrecidas por Google como es el Datastore o la Memcache. También ofrecerá una API REST la cual será utilizada por el backend para anotar cierta información. También se ofrecerá en la API la posibilidad de consumir los datos de este servicio, en caso de no ofrecer esta parte, se deberá mostrar mediante algún mecanismo que los datos se han almacenado.

- El ***frontend*** será un cliente web o aplicación móvil que permitirá al usuario interaccionar con el backend para lanzar las operaciones implementadas en el backend.

En la siguiente imagen se muestra una visión general del sistema a construir. Todos los servicios serán implementados en Python haciendo uso del microframework Flask, mientras que las comunicaciones se realizarán mediante una codificación JSON, dicha codificación queda abierta a las soluciones propuestas por los/as estudiantes.



La única parte obligatoria de esta práctica es el backend, mientras que el resto queda de forma opcional para conseguir la máxima puntuación.

Backend *(Max: 22 puntos)*

El Backend albergará el procesamiento de la aplicación y por lo tanto es la parte más compleja a desarrollar. El backend desplegará Hadoop en su modalidad pseudo-distribuida o distribuida junto con Pig y/o Flume como se han visto en clase.

Para obtener los datos con los que se trabajará se proponen dos modalidades con diferente puntuación. La primera de ella es haciendo uso de Flume, que permitirá rescatar un conjunto de tweets de Twitter; la segunda consiste en hacer uso de los tweets disponibles en un fichero alojado en *campusvirtual*. En el caso de Flume, el backend ofrecerá dos operaciones: una de inicio de recolección de tweets y otra para la parada de dicha recolección.

Por otro lado, en ambas modalidades los tweets serán procesados mediante scripts PIG con el fin de obtener cierta información mediante operaciones MapReduce. Se podrán realizar entre una y tres operaciones de las listadas a continuación (ver puntuación en la rúbrica)

- Filtrar los tweets entre dos fechas.
- Listar aquellos que contengan una determinada palabra.
- Mostrar aquellos tweets que han sido retweeteados por más de Z personas.
- Mostrar usuarios que han tweeteado y la cantidad de tweets de los mismos.
- Mostrar los tweets con menos de Y likes.

Estas operaciones pueden ser realizadas a medida o proporcionar al usuario la posibilidad de modificar los parámetros. Por ejemplo, para la última operación, el desarrollador podría fijar el valor Y, o por el contrario, proveer al usuario de los mecanismos necesarios para que éste indique el valor de Y (ver rúbrica para la puntuación). A continuación se listan algunas funcionalidades y/o características opcionales a incluir en el backend:

- *Protección de recursos:* Incluir la protección de algunos de los recursos mediante HTTPAuthBasic, OAuth o mediante OAuth de terceros (acceso mediante cuenta de Facebook, Google, …). Esta parte también puede ser implementado en la parte del logger.

- *Virtualización del Backend:* La forma de desplegar el cluster de Hadoop puede realizarse haciendo uso de la virtualización: máquinas virtuales de Vagrant o contenedores Docker.

- *Nuevos nodos de computación:* Además, se pueden incluir diferentes nodos de computación con el fin de ampliar el cluster, estos nodos pueden ejecutarse de forma virtual o en diferentes Pcs.

- *Servicio de notificaciones:* Servicio al que un conjunto de clientes pueden suscribirse con el fin de recibir información sobre las operaciones que se están realizando en el backend, es decir el servicio deberá disponer de un mecanismo capaz de enviar hooks a los clientes para informar de una nueva operación. Esos clientes pueden ser los propios logger o incluso se puede hacer uso del cliente utilizado en la receta webhook de GitHub.

Logger *(Max: 16 puntos)*

El servidor logger contendrá un registro de las operaciones realizadas por parte de los usuarios, es decir actuará de logger de toda la actividad realizada en nuestro servicio: consulta realizada, obtención de tweets, ... Para ello se implementará de nuevo un servicio web en Flask que podrá estar albergado en Google y hará uso de los servicios que éste ofrece, como es el datastore o la memcache, o bien se utilizará sqlalchemy o una lista de Python. La información almacenada constará de la operación realizada, el usuario que la realizó (si es posible), la fecha y la hora. También permitirá rescatar toda la información almacenada de las siguientes formas (implementar al menos una de ella):

- Mostrar todos los logs
- Mostrar las operaciones de un usuario determinado
- Filtrar los log por un intervalo de fechas

Frontend *(Max: 7 puntos)*

El *frontend* mostrará una interfaz funcional que permita interaccionar con el *backend* de nuestra aplicación, no es necesario "disfrazarla" o "decorarla" y se puede partir de la interfaz utilizada en el segundo trabajo (cliente twitter). En esta parte se mostrará al menos las operaciones de captura y parada de Tweets si se utiliza *Flume* y al menos una de las operaciones implementadas en el *backend*, junto a la entrada de datos necesarios para dicha operación si tal es el caso. Además, se podrán incluir mensajes flashing con el fin de informar al usuario.

Todos los servicios pueden ser testeados, pero sólo el *backend* y el *frontend* son tomados en cuenta en la rúbrica adjunta. Para esta tarea se recomienda el uso de test unitarios.

Modalidades para el desarrollo de la práctica

Si se ha elegido la modalidad por ***parejas/tríos***, éstas deberán utilizar un repositorio (GitHub o Bitbucket) para ver las actividad de cada participante. Además, se recomienda el uso de algún sistema que permita asignar responsabilidades o tareas (Trello). Para el caso ***individual*** también es recomendable el uso de repositorios.

Recomendaciones

Se recomienda enérgicamente el uso de repositorios para el desarrollo de la práctica, así como el uso de un *framework* de tests con el fin de probar aquello que ha sido implementado.

El primer paso será definir las interfaces API REST de nuestro servicio de backend y logger que vamos a ofrecer, para ello es conveniente asociar las operaciones HTTP de una determinada URL a uno de los métodos/operaciones permitidas. Seguidamente, se pasará la solución obtenida a Python mediante Flask, donde las funciones que manejan los endpoints no contendrán funcionalidad alguna, es decir lo que se crea es la estructura que tendrá el servicio web.

A la vez o en un segundo paso, se realizarán los tests correspondientes a cada uno de los *endpoints* diseñados en el paso anterior, comprobando que los códigos de estado devueltos son los correctos.

Posteriormente se implementará la lógica necesaria de cada endpoint junto con su test correspondiente. Seguidamente, se tomará la decisión sobre qué puntos adicionales serán abordados.

Nota*: Todos los miembros deberán subir un comprimible con la práctica desarrollada, así como un pequeño informe mostrando API(s) REST desarrollada(s) y el porcentaje de participación de cada integrante (un 33% indicará que todos los integrantes participaron por igual).*

Referencias:

Flask – flask.pocoo.org
Jinja2 – jinja.pocoo.org
Werkzeug – werlzeug.pocoo.org
JSON – json.org
Schema – schema.org
GCP – https://cloud.google.com/