ENTREGA 2: Arquitectura y Modelado en Objetos - Parte II

Objetivos de la entrega

- Diseñar e implementar, de manera incremental, las nuevas funcionalidades.
- Incorporar nociones de ejecuciones de tareas asincrónicas y/o calendarizadas.
- Familiarizarse con otros componentes dentro de la arquitectura del Sistema y la orientación a servicios.
- Familiarizarse con el concepto de APIs como mecanismo de integración y su consumo.

Unidades del Programa Vinculadas

- Unidad 2: Herramientas de Concepción y Comunicación del Diseño
- Unidad 3: Diseño con Objetos
- Unidad 6: Diseño de Arquitectura
- Unidad 8: Validación del Diseño

Alcance

- Fuentes dinámicas.
- Fuentes proxy (intermediarias). Integración con sistemas externos
  - de otras ONGs,
  - de las propias instancias de MetaMapa.
- Servicio de agregación.
- Refresco de Colecciones.
- Rechazo de solicitudes de eliminación por tratarse de *spam* en forma automática.

Dominio

Fuentes dinámicas

En esta iteración, se implementará la funcionalidad que permitirá a los ***contribuyentes*** del Sistema subir ***hechos*** en forma anónima o registrada. Retomando el diagrama de arquitectura provisto en el Contexto general, este requerimiento deberá satisfacerse en el ***servicio de fuentes dinámicas***[1]. Si la persona no está registrada en la plataforma, podrá subir hechos sin posibilidad de edición posterior. En cambio, si los sube en forma registrada podrá realizar modificaciones al mismo en caso de que lo necesitara, pero solo en el plazo de una semana. En ambos casos podrán estar sujetos a revisión por parte de las personas ***administradoras,*** que podrán aceptar, aceptar con sugerencia de cambios o rechazar la información.

Se desarrollarán los mecanismos necesarios para gestionar la subida de información en formato tanto de texto como multimedia.

[1] En el diagrama de despliegue, se indica que el servicio de fuentes dinámicas y el servicio agregador tienen una base de datos de Hechos y Solicitudes (***cada uno, una distinta***). Como aún no tratamos bases de datos en la cursada, recomendamos implementar un Patrón Repositorio que cumpla esa función y mantenga las entidades necesarias en memoria.

Fuentes proxy (intermediarias)

Existen sistemas externos provistos por otras organizaciones no gubernamentales que proveen hechos de generación propia mediante APIs y/o bibliotecas de diferente tipo. Tal como se observa en el diagrama de arquitectura, esto se realizará a través de ***servicios*** y ***fuentes proxy*** (intermediarias). Es posible (y deseable, como se verá en futuras entregas) que más de una fuente retorne el mismo hecho, es decir, con iguales o muy similares atributos.[2] En particular, en esta iteración nos estaremos integrando a una API que será provista por la Cátedra, cuya estructura será documentada y provista próximamente.

Fuente MetaMapa:
Por otro lado, el Sistema deberá proveer la capacidad de comunicarse con otras instancias de MetaMapa. Esto lo haremos a través de una interfaz API REST, siguiendo la siguiente estructura tentativa (con posibles modificaciones en el futuro):

Endpoint	**Descripción**
GET /hechos	Esta ruta expone todos los hechos del sistema y los devuelve como una lista en formato JSON. La misma acepta parámetros para filtrar los resultados: categoría, fecha_reporte_desde, fecha_reporte_hasta, fecha_acontecimiento_desde, fecha_acontecimiento_hasta, ubicacion.
GET /colecciones	Esta ruta expone todas las colecciones disponibles en esta instancia de MetaMapa, independientemente del origen de sus fuentes.
GET /colecciones/:identificador/hechos	Esta ruta, similar a la anterior, permite obtener los hechos asociados a una colección. Acepta los mismos parámetros y devuelve los resultados en el mismo formato.
POST /solicitudes	Permite crear solicitudes de eliminación, enviando los datos de la solicitud como un JSON a través del cuerpo *(body)* de la misma.

Si bien en próximas iteraciones trabajaremos en exponer nuestros propios servicios utilizando esta interfaz, **por ahora se desea que nuestras fuentes proxy también sean capaces de consumir este tipo de API.** Este será consumido en tiempo real y se espera que sus resultados sean siempre actuales.

Servicio de agregación

Para esta iteración, se requiere modelar el ***servicio de agregación*** descripto en el diagrama de despliegue inicial. Esto permitirá que las personas visitantes y/o contribuyentes de la plataforma accedan a hechos de una colección provenientes de ***todas las fuentes***. Por lo tanto, se permite ahora que las colecciones contengan hechos de distintas fuentes, sean estas ***estáticas***, ***dinámicas*** o provenientes de servicios externos (fuentes ***proxy***). Además, para posibilitar la futura exposición REST de las mismas, se incorporará un atributo textual identificador ***(handle)*** que será un string alfanumérico, sin espacios, único entre todas las colecciones.

[2] Queda por fuera del alcance de esta entrega realizar algún tipo de procesamiento adicional sobre los hechos “repetidos”.

Refresco de Colecciones

Se pide que, una vez por hora, el ***servicio de agregación*** actualice los hechos pertenecientes a las distintas colecciones, en caso de que las fuentes hayan incorporado nuevos hechos. Se excluye de este requerimiento la actualización de los hechos provenientes de ***fuentes proxy MetaMapa***, que deben ser obtenidos en tiempo real en todos los casos.

Solicitudes de eliminación y rechazo por spam en forma automática

En la anterior entrega, se contempló la posibilidad de que una persona (registrada o no) pueda solicitar la eliminación de un hecho que haya sido subido por otro usuario. En este caso, se solicita extender la funcionalidad a que pueda solicitar la eliminación de un hecho de cualquier fuente. Es decir, el ***servicio de agregación*** deberá contemplar las solicitudes de eliminación y, en caso de que sean aceptadas, deberá revisar su origen e impedir que se lo pueda mostrar en la colección.

Para simplificar esta tarea, se cuenta con un componente ya desarrollado que permite la detección de casos de spam evidentes. Si éste marca a la solicitud como *spam*, la solicitud deberá ser rechazada automáticamente.

```
interface DetectorDeSpam {
   boolean esSpam(String texto)
}
```

Requerimientos detallados

1. El Sistema debe permitir la ***creación de un hecho a partir de una fuente dinámica*** por parte de los contribuyentes.
2. El Sistema debe permitir la ***integración con las fuentes proxy/intermediarias necesarias*** para recolectar hechos.
1. El Sistema debe permitir la obtención de todos los hechos de las ***diferentes fuentes proxy tipo MetaMapa*** en tiempo real.
3. El Sistema debe permitir que las colecciones tengan hechos de cualquier tipo de fuente, a partir de la ***agregación de fuentes.***
4. El Sistema debe permitir el rechazo de solicitudes de eliminación en forma automática cuando se detecta que se trata de spam. Se puede investigar sobre *el algoritmo TF-IDF para la detección de spam.* Utilizar filtros de spam básicos de forma local o utilizando un servicio externo.

Consideraciones para la Entrega

- A partir de esta entrega, para el desarrollo del TPA se deberá incorporar el uso de Git Flow en base a las pautas establecidas en este documento.

Entregables

1. **Modelo del Dominio:** diagrama de clases que contemple las funcionalidades requeridas.
2. **Justificaciones de Diseño**: Documento y Diagramas Complementarios que considere el equipo.
3. **Implementación** de requerimientos de la Entrega Actual. Favorecer la reutilización del código de lo ya desarrollado, pero identificando correctamente los servicios involucrados.
4. **Diagrama de arquitectura**.
5. **Bonus.** Implementación de un filtro de spam básico de forma local o utilizando un servicio externo. Analizar Ventajas y Desventajas apoyándose en Atributos de Calidad.