Facultad de Ingeniería - Universidad de la República

Sistema de Monitoreo Ambiental

SIMONA

Informe de Proyecto de Grado

Montevideo, Uruguay

Junio 2017

Estudiantes

Gonzalo Antúnez

Marcelo Piñeiro

Tutores

Laura González

Bruno Rienzi

Usuario responsable

Federico Herrera

SIMONA

Resumen

En los últimos años, los gobiernos han llevado a cabo varias iniciativas, con el fin de controlar y disminuir la contaminación ambiental en el mundo. Sin embargo, el control de su cumplimiento presenta algunos inconvenientes, por ejemplo, la necesidad de trasladarse a zonas de difícil acceso.

Por otro lado, los avances tecnológicos en las áreas de Sistemas de Información Geográfica (Geographic Information System, GIS), plataformas sociales y sensores han permitido la generación y procesamiento de gran cantidad de información. También existen herramientas, en el contexto empresarial, como Enterprise Service Bus (ESB) que facilitan la integración de servicios y Complex Event Processing (CEP) que soluciona el procesamiento masivo de datos en tiempo real.

El presente proyecto de grado propone como solución una plataforma GIS para lograr monitorear en tiempo real el cumplimiento de leyes medioambientales, mediante el uso de las tecnologías ESB y CEP. La plataforma propuesta obtiene datos de sensores y plataformas sociales, y permite la creación de alertas para notificar a los usuarios en caso de incumplimiento de una ley medioambiental. De esta forma, es posible actuar de manera rápida para mitigar los riesgos que esto conlleva.

Para validar funcionalmente la solución, se planteó un caso de estudio en el que organizaciones nacionales utilizan la plataforma implementada para monitorear leyes medioambientales vigentes en el río Santa Lucía.

Palabras claves

Leyes medioambientales, sistemas de información geográfica, sensores, internet de las cosas, redes sociales, ESB, CEP, alertas, notificaciones.

Tabla de contenido

1. Introducción

Hoy en día la preocupación por el cuidado del medioambiente ha tomado una gran importancia a nivel global [1], lo que ha dado lugar a diferentes leyes medioambientales. Este proyecto apunta a la construcción de una plataforma tecnológica que permita el monitoreo de parámetros medioambientales para controlar el cumplimiento de dichas leyes.

1.1. Motivación

En la actualidad, se están llevando adelante varias iniciativas para controlar y disminuir la contaminación ambiental en el mundo. Por otro lado, los avances tecnológicos en las áreas de sistemas de información geográfica, plataformas sociales y sensores, permiten la generación y procesamiento de gran cantidad de información. Si bien estos avances presentan una oportunidad para el monitoreo medioambiental, existen algunas dificultades para ser aprovechados de forma integrada. En primer lugar, estas tecnologías utilizan formatos de datos y protocolos de comunicación heterogéneos, que en general son incompatibles. En segundo lugar, esta información no es fácilmente accesible por los usuarios interesados.

Por otro lado, en el contexto empresarial existen herramientas que atacan los problemas de comunicación e integración. Por ejemplo, existen los Enterprise Services Bus (ESB) que facilitan la integración de servicios, así como la tecnología de Complex Event Processing (CEP) que soluciona el procesamiento masivo de datos en tiempo real y permite inferir determinadas situaciones de interés.

La motivación principal de este proyecto es proponer una plataforma basada en ESB y CEP que aborde las problemáticas mencionadas para poder monitorear el cumplimiento de leyes medioambientales y envíe notificaciones en caso de detectar un incumplimiento.

En la Figura 1 se aprecia por un lado la situación actual donde las distintas tecnologías generan información en sus respectivos formatos de datos de manera independiente, mientras que la plataforma a proponer se encargaría de obtener los datos de las distintas tecnologías y presentarlos de manera unificada al usuario.

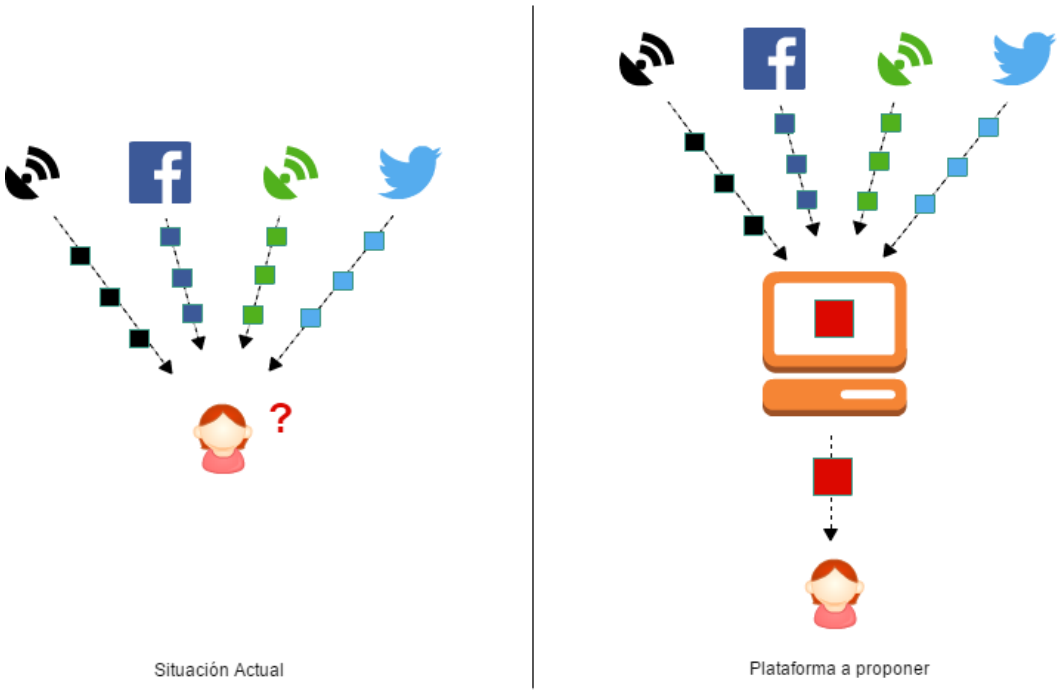


Figura 1 - Realidad actual – Plataforma a proponer.

1.2. Objetivos

El objetivo general de este proyecto es proponer y diseñar mecanismos, basados en ESB y CEP, que permitan monitorear el cumplimiento de leyes medioambientales, generando notificaciones en caso de incumplimiento, mediante el uso de tecnologías de la información geográfica, plataformas sociales y sensores que envían datos en tiempo real.

Para cumplir con este objetivo, se plantean los siguientes objetivos específicos:

1. Analizar el contexto en el que se sitúa el proyecto e identificar los requerimientos que plantea en cuanto a la integración de las distintas fuentes de información, leyes medioambientales y notificaciones a los usuarios.

2. Proponer una solución que permita monitorear el cumplimiento de leyes medioambientales en base a la aplicación de reglas sobre los datos provenientes de las distintas fuentes (p. ej. sensores, redes sociales) y generar notificaciones en caso de incumplimiento.

3. Implementar un prototipo que valide la factibilidad técnica de la solución propuesta utilizando tecnologías de ESB y CEP con capacidades geográficas.

4. Desarrollar un caso de estudio que permita la evaluación funcional de la solución propuesta en el marco de la realidad uruguaya.

1.3. Aportes del proyecto

A continuación, se enumeran los aportes del proyecto:

1. Análisis del contexto en la actualidad, obteniendo los requerimientos necesarios para proponer una solución que integre distintos tipos de fuentes de información para el control de las leyes medioambientales existentes a nivel nacional y global.

2. Propuesta de una solución, basada en tecnologías ESB y CEP, que resuelve los requerimientos relevados. La solución obtiene en tiempo real los datos generados por las distintas fuentes y notifica a los usuarios correspondientes en caso de un incumplimiento de alguna ley medioambiental.

3. Implementación de un prototipo que obtiene datos, de sensores con protocolo de comunicación Sensor Observation Service (SOS) y Twitter utilizando Mule ESB y procesándolos con Esper CEP.

4. Desarrollo de un caso de estudio, simulando un posible escenario de utilización de la plataforma, donde a organizaciones nacionales les interesa controlar parámetros medioambientales del río Santa Lucía. Estas organizaciones poseen fuentes de datos propias y para realizar los controles utilizan sus fuentes y las de otras organizaciones.

5. Evaluación de la factibilidad técnica de incorporar una extensión geográfica a la tecnología de CEP permitiendo la realización de controles en el procesamiento de datos, considerando la ubicación donde se producen.

1.4. Organización del documento

El resto de este documento se organiza de la siguiente manera:

- En el Capítulo 2, se presenta el marco conceptual, con los conceptos principales que se utilizan en este documento. El estudio realizado en este capítulo sienta las bases teóricas para el desarrollo del proyecto.

- En el Capítulo 3, se realiza un análisis de los requerimientos planteados para la solución. Además, se presenta un análisis de soluciones similares existentes.

- En el Capítulo 4, se presenta la solución propuesta y se describen los principales conceptos y decisiones tomadas que llevaron a la solución final.

- En el Capítulo 5, se describe el diseño de la solución y la implementación realizada.

- En el Capítulo 6, se describe un caso de estudio que evalúa funcionalmente la solución propuesta.

- En el Capítulo 7, se presentan conclusiones del trabajo realizado y trabajo a futuro.

El documento posee también dos apéndices. En el primero se especifica el procedimiento necesario para el despliegue del prototipo. En el segundo se describen los pasos a seguir para incorporar nuevos protocolos de comunicación al prototipo.

2. Marco conceptual

En este capítulo se presentan los principales conceptos relevantes para el proyecto: iniciativas y normativas medioambientales, tecnologías web, estándares de sistemas de información geográfica y tecnologías empresariales.

2.1. Iniciativas y normativas medioambientales

En esta sección, se comienza enumerando algunas cumbres internacionales que se realizaron en los últimos años en materia del cuidado medioambiental, para describir luego varias leyes uruguayas con relación al medioambiente.

2.1.1. Principales cumbres internacionales

La preocupación por el cuidado del medioambiente ha crecido en los últimos años y se ha demostrado en las diferentes reuniones y cumbres que se han realizado mundialmente.

La primera vez donde se manifestó una preocupación importante sobre el cuidado del medioambiente fue en la llamada “Declaración de Estocolmo” de 1972. La conferencia fue realizada por las Naciones Unidas (ONU) y tuvo una gran concurrencia de países participantes. En esta cumbre se definieron 26 principios y un plan de acción que incluye diez recomendaciones sobre el cuidado del medioambiente. [2]

En Río de Janeiro en 1992 se realizó una nueva cumbre que fue la de mayor concurrencia y proyectó el tema ambiental hacia el conjunto de los grandes debates mundiales. Esta cumbre es calificada como la más significativa de la historia. Un logro de la cumbre fue la elaboración de la “Agenda 21”. Ésta es un programa de la ONU para promover el desarrollo sostenible y se detallan acciones a tomar a nivel local, nacional y mundial por los gobiernos de los países miembros de la ONU. Es considerado el documento más relevante y ambicioso en el tema ambiental elaborado hasta el presente, debido a su carácter de plan de acción mundial. [2]

En el 2002 se realizó la cumbre de Johannesburgo. A diferencia de las anteriores, se hicieron evaluaciones de los avances logrados en las décadas anteriores. De esta forma se realizó un balance de los compromisos ya cumplidos, y los que quedaron pendientes. En términos más generales fue una afirmación a los compromisos previamente acordados. [3]

2.1.2. Normativa medioambiental uruguaya

En cuanto a las legislaciones nacionales acerca del medioambiente se encuentra el artículo 47 de la Constitución de la República [4] que establece que la protección del medioambiente es de interés general. En éste se indica que las personas deberán abstenerse de cualquier acto que cause depredación, destrucción o contaminación grave al medioambiente.

La ley que reglamenta esta disposición es la 17.283 [5]. Dicha ley se divide en artículos sobre la protección del ambiente, calidad de aire, del agua, del suelo y del paisaje. A continuación, se expone un breve resumen de algunas leyes nacionales, clasificadas por su categoría:

- **Suelo**: El artículo 1 de la ley 15.239 indica que es deber del Estado velar por prevenir y controlar la erosión y degradación de los suelos, las inundaciones y la sedimentación en aguas superficiales destinadas a fines agropecuarios, así como determinar y fijar las dunas. [6]

- **Agua**: La Constitución de la República establece que el agua es un recurso natural esencial para la vida. El acceso al agua potable, constituye un derecho humano fundamental. [7]

  Es importante destacar que el decreto 253/079 clasifica los cursos o cuerpos de agua del país en cuatro clases dependiendo de su uso preponderante. Para cada una de las clases se definen distintos parámetros a controlar, así como los valores mínimos y máximos. Por ejemplo, la clase 1 son aguas destinadas, o que puedan ser destinadas, al abastecimiento de agua potable a poblaciones con tratamiento convencional. En la Figura 2 se definen los rangos aceptables de algunos de los parámetros medibles para esta clase. [8]

DECRETO 253/79

PARAMETRO	ESTANDAR
- pH	Entre 6,5 y 8,5
- OD (Oxígeno disuelto)	Mín. 5 mg/L
- DBO5 – (Demanda Bioquímica de Oxígeno)	Máx 5 mg/L
- AMONIACO LIBRE	Máx 0,02 mg/L en N
- NITRATOS	Máx 10 mg/L en N
- FOSFORO TOTAL	Máx 0,025 mg/L en P
- CIANURO	Máx 0,005 mg/L
- ARSENICO	Máx 0,005 mg/L
- COBRE	Máx 0,2 mg/L
- CROMO TOTAL	Máx 0,05 mg/L
- MERCURIO	Máx 0,0002 mg/L
- NIQUEL	Máx 0,02 mg/L
- PLOMO	Máx 0,03 mg/L
- ZINC	Máx 0,03 mg/L

Figura 2 - Decreto 253/079 - Clase 1 [8].

- **Aire:** El artículo 17 de ley 17.283 establece que queda prohibido liberar o emitir a la atmósfera, directa o indirectamente, sustancias, materiales o energía, por encima de los límites máximos o en contravención de las condiciones que establezca el Ministerio de Vivienda, Ordenamiento Territorial y Medio Ambiente (MVOTMA). [5]

En Uruguay, el MVOTMA posee la coordinación exclusiva de la gestión ambiental integrada del Estado y de las entidades públicas en general. Para esto, utiliza las siguientes herramientas de control para monitorear el cumplimiento de las leyes [9]:

- Auditorías e inspecciones.
- Atención de denuncias.
- Sanciones a infractores.
- Monitoreo en línea de las emisiones al aire, generadas por grandes emprendimientos.
- Recepción de documentos ambientales de los grandes emprendimientos.

En la actualidad, la Dirección Nacional de Medio Ambiente (DINAMA) posee un visualizador geográfico, como se muestra en la Figura 3, en donde se pueden visualizar los datos producidos por las estaciones de aire. También se puede acceder a un historial que posee los valores obtenidos en mediciones de la calidad del agua.

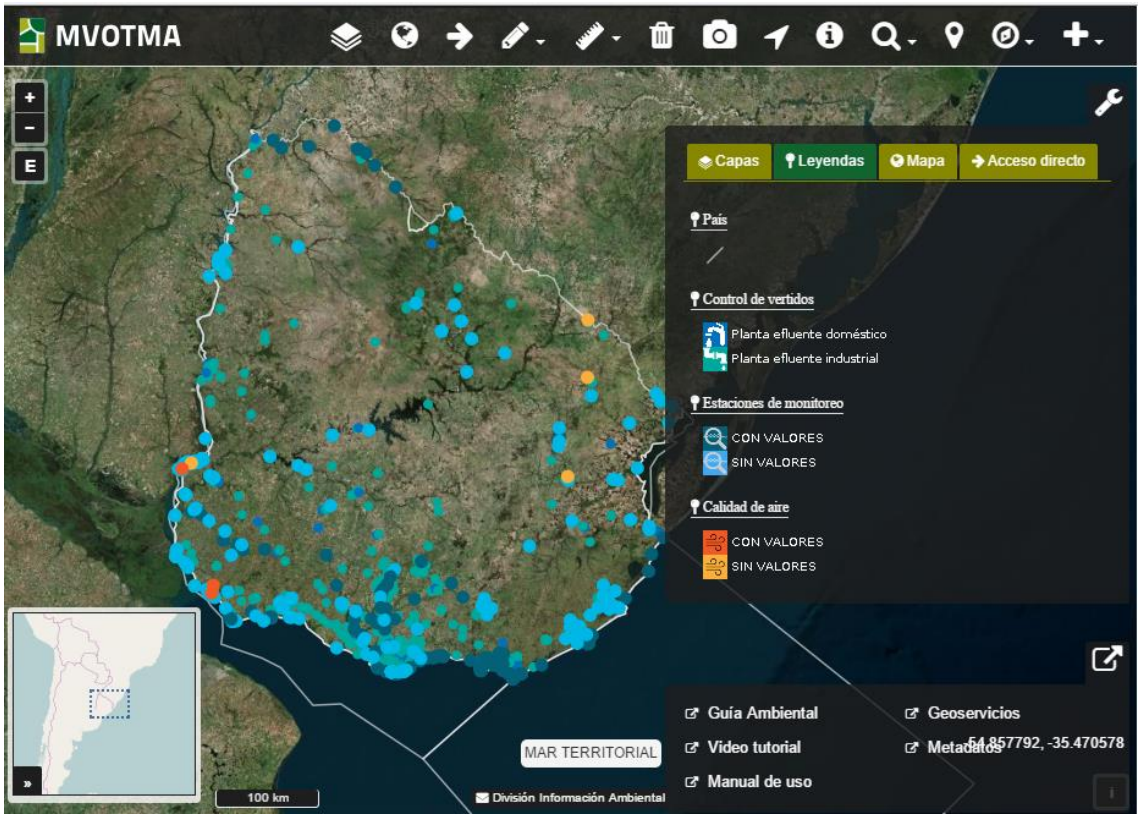


Figura 3 - Visualizador geográfico de la DINAMA[1].

[1] http://www.dinama.gub.uy/visualizador/index.php?vis=siq

2.2. Tecnologías relevantes al contexto

En esta sección, se brinda una descripción de distintas tecnologías relevantes al contexto del proyecto.

2.2.1. Internet de las cosas

Internet de las Cosas (Internet of Things, IoT) para dispositivos inteligentes se puede definir como la interconexión de sensores y dispositivos que poseen la habilidad de compartir información entre plataformas a través de una red unificada. De esta forma, se desarrolla un ambiente para la creación de aplicaciones que puedan consumir la información provista. [10]

En IoT los elementos de la vida cotidiana (denominados cosas, objetos o máquinas), se complementan con la tecnología de comunicación e informática y se unen al marco de las comunicaciones. En este marco las tecnologías inalámbricas y cableadas proporcionan capacidades de comunicación e interacción cumpliendo una variedad de servicios basados en las interacciones persona-persona, máquina-máquina, persona-máquina, máquina-persona. Estas máquinas conectadas a los objetos serán los nuevos usuarios de internet y generarán tráfico de datos. [11]

El uso de IoT puede traer beneficios en distintas áreas [10]:

- Salud: Monitoreo de pacientes, modelado y contención de propagación de enfermedades, información predictiva para alertar a los profesionales y anticipar decisiones políticas en casos de pandemia.
- Servicios de emergencia: Supervisión remota del personal, sensores integrados en los edificios para ayudar a los rescatistas ante emergencias o situaciones de catástrofes.
- Monitoreo de multitudes: Monitoreo del flujo de multitudes para el manejo de posibles emergencias, uso eficiente de espacios públicos y comerciales.
- Transporte: Información del tráfico en tiempo real y optimización de rutas, monitoreo de accidentes para coordinación en las tareas de respuesta de las emergencias.
- Agua: Control de la calidad del agua, fugas y usos.
- Infraestructura: Control de temperatura, humedad, consumo de energía, calefacción y ventilación.
- Medioambiente: Monitoreo del ruido, de la industria, control de vías fluviales y contaminación del aire.

Algunas de las plataformas más utilizadas en IoT son: Xively[2], Carriots[3] y ThingSpeak[4]. Estas plataformas permiten conectar sensores para que les envíen las mediciones obtenidas.

Un sensor [12] es un dispositivo diseñado para detectar diversas acciones, ya sea físicas o químicas (temperatura, presión, etc.), y transformarlas en otra magnitud, normalmente eléctrica, que se pueda cuantificar y manipular para luego transmitirla adecuadamente. El sensor siempre está en contacto con la variable de instrumentación con lo cual aprovecha sus propiedades para adaptar la señal que mide para que otro dispositivo la interprete. Los sensores también podrían estar conectados a una computadora para tener acceso a bases de datos en la cual se guarden los valores obtenidos.

2.2.2. Redes sociales de Internet

Una red social es una estructura social que está compuesta por personas, organizaciones o entidades que se conectan entre sí por uno o varios criterios como amistad, parentesco o profesional, entre otros. Las redes sociales de Internet se han convertido en un fenómeno social que revoluciona la manera de comunicación y la interacción que hasta el momento tenían los ciudadanos. [13]

Las redes sociales de Internet son una de las principales fuentes de información de las personas jóvenes. Dentro de las ventajas que brindan las redes sociales, se encuentra la información en tiempo real acerca de lo que sucede a nuestro alrededor. Cualquier persona puede expresar, manifestar o transmitir sucesos que ocurran. [13]

Por ejemplo, el 25 de enero de 2016 se registró un terremoto de magnitud 6.3 en la escala abierta de Richter en el mar de Alborán. El temblor hizo que varios usuarios de Twitter escribieran tweets con la palabra terremoto. Se realizaron más de 90.000 tweets con dicha palabra en un corto lapso. [14]

Para poder comunicar estas redes sociales con otros programas se necesita que la red social cuente con una interfaz de programación de aplicaciones (Application Programming Interfaces, API[5]).

Una API simplifica en gran medida el trabajo de un programador, ya que no tiene que desarrollar programas desde cero. Estas permiten al informático usar funciones predefinidas para interactuar con otro programa o sistema. [15]

Las redes sociales de Internet más importantes, como Facebook o Twitter, poseen interfaces de aplicaciones en donde un usuario puede consumir alguna de las funciones o

[2] https://www.xively.com/
[3] https://www.carriots.com/
[4] https://thingspeak.com/
[5] Una API es una especificación formal sobre cómo un módulo de un programa se comunica con otro. http://www.abc.es/tecnologia/consultorio/20150216/abci--201502132105.html

procedimientos, que contienen, para obtener información. En particular Twitter posee operaciones que permiten consumir tweets con una palabra en particular en tiempo real.

2.2.3. Sistema de información geográfica

Los Sistemas de Información Geográfica (Geographic Information System, GIS) constituyen una tecnología que forma parte del ámbito más extenso de los Sistemas de Información, que permiten analizar, evaluar y proponer alternativas de soluciones para la toma de decisiones con respecto de la información geográfica. [16]

Se estima que gran parte de la información que trata una empresa, puede ser georreferenciada, es decir que dicha información puede ser localizada en un lugar físico determinado por referencias geográficas (coordenadas, barrios, etc.). Con esto, la geoinformática suma el componente geográfico a la gestión de la administración de las empresas que manejan información susceptible de ser georreferenciada, logrando una mejor y más eficiente gestión de los recursos. [16]

Básicamente un GIS está compuesto por varios elementos fundamentales: hardware, software, datos, cartografía, metodología y personas. [16]

Existen dos grandes maneras de estructurar la información real en un sistema informático: utilizando un modelo vectorial y utilizando un modelo raster. En el modelo raster, la característica básica es que, al crear una trama de celdas o pixeles, cada una de ellas tiene una única propiedad geográfica. En este modelo es más importante la propiedad geográfica que los límites exactos. Por otro lado, en el modelo vectorial cada entidad geográfica se representa a partir de tres elementos básicos, llamados primitivas: puntos, líneas y polígonos. En este modelo los límites de los objetos se representan de forma explícita. La Figura 4 muestra un ejemplo de las diferencias del modelo raster con el vectorial. [16]

Uno de los formatos más populares utilizados para guardar datos vectoriales es el Shapefile. Este formato de representación vectorial fue desarrollado por Enviromental Systems Research Institute (ESRI), permite guardar tanto geometrías como sus atributos asociados. [17]

Sin embargo, el formato Shapefile posee algunas desventajas: se encuentra compuesto de por lo menos tres archivos, no soporta el formato raster y no soporta nombres de atributos de más de diez caracteres. Por dichas desventajas, se suele utilizar una base de datos relacional para el almacenamiento de las geometrías y sus atributos. En una base de datos relacional se utiliza SQL para el acceso a los datos y su posterior análisis. Además, permite almacenar formatos raster y utiliza índices espaciales que permiten acceder más rápidamente a las geometrías. [18]

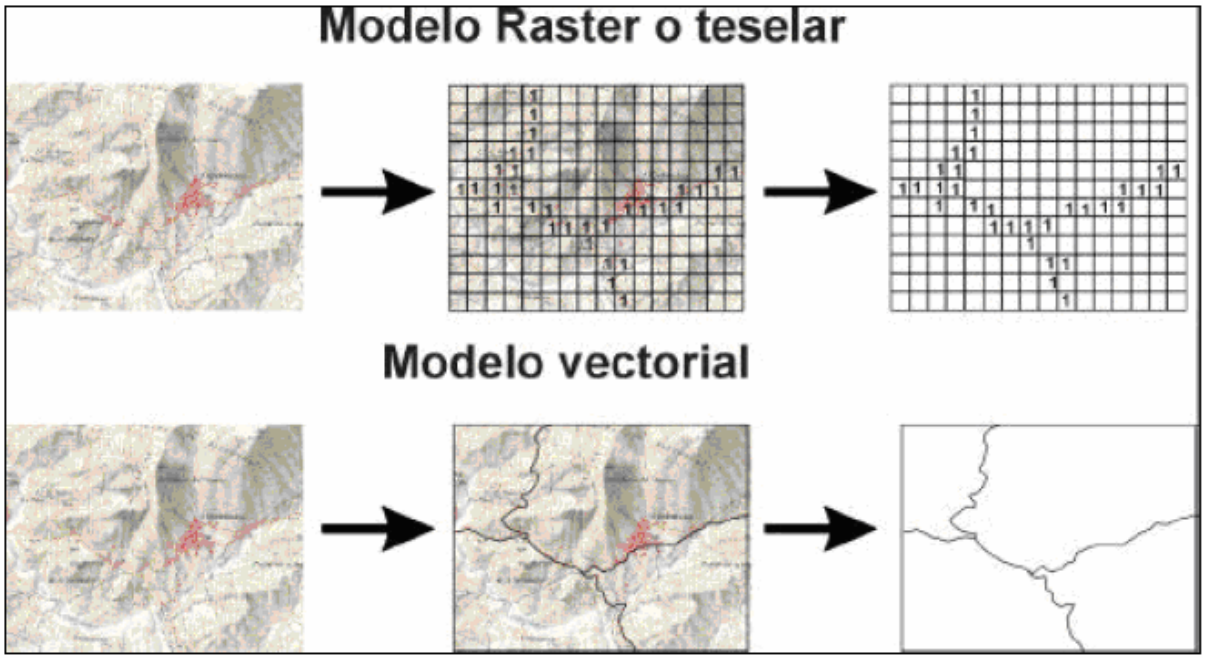


Figura 4 - Diferencia modelo raster y vectorial [16].

2.3. Estándares de sensores

Uno de los componentes de mayor relevancia dentro de IoT son los sensores. Actualmente existen diversos estándares que especifican formatos y protocolos de comunicación entre los mismos y con otros sistemas. Muchos de estos formatos y protocolos están definidos en estándares que son propuestos por Open Geospatial Consortium (OGC) [19]. OGC es una organización sin fines de lucro comprometida en la elaboración de estándares abiertos para la comunidad geográfica mundial. Estos estándares se elaboran a través de un consenso y están disponibles libremente para cualquier persona que desee mejorar el intercambio de datos geográficos.

En la siguiente sección se listan algunos de los estándares existentes.

2.3.1. Sensor Web Enablement

Sensor Web Enablement (SWE) [20] es un framework que agrupa un conjunto de estándares de OGC que permite que distintos tipos de sensores, transductores y repositorios de datos de sensores puedan ser descubiertos, accedidos y usados a través de la web. Esto tiene aplicaciones en diferentes campos como la meteorología, monitoreo del medioambiente, prevención de riesgos, seguridad en plantas industriales, etc.

El uso de estándares permite integrar diferentes tecnologías de sensores, computadoras y redes, de manera independiente de tecnologías cerradas de una empresa en particular.

En las siguientes secciones se detallan algunos de los estándares que componen el SWE.

2.3.1.1. Observations and Measurement

El estándar Observation and Measurement (O&M) (versión 2.0) [21], [22] se encuentra dentro del ISO 19.156 y define esquemas XML para las observaciones y medidas tomadas por un sensor, como también para las funciones relacionadas con la toma de dichas observaciones. O&M es necesario al utilizar Sensor Observation Service (SOS) para la implementación de arquitecturas SWE y para el soporte general de los sistemas OGC.

O&M define una serie de modelos de información usando diagramas de clases Unified Modeling Language (UML), coberturas, características, observaciones, valores, fenómenos observables y medibles (tal como se ve en la Figura 5) y la implementación de los modelos en un esquema XML (como se aprecia en la Figura 6) compatible con Geography Markup Language (GML).

O&M solo permite medidas cuyos resultados sean expresados como cantidades, categorías, valores temporales y geométricos, así como conjuntos y series de los mismos.

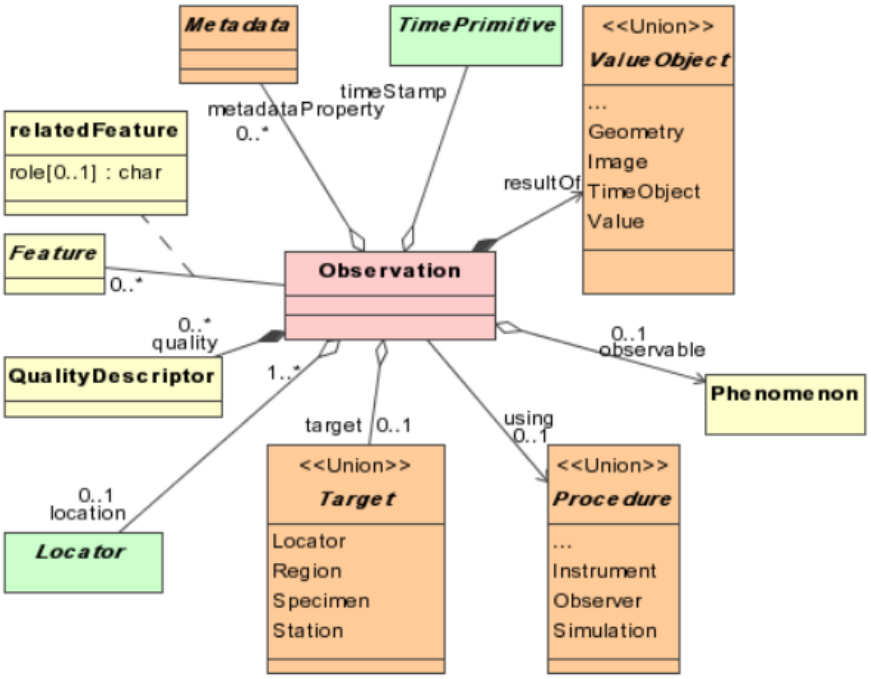


Figura 5 - Modelo de observación [23].

En la Figura 6, se muestra un ejemplo del XML compatible con GML[6]. En dicho XML se observa que se mide la radiación provista por el satélite Landsat.

[6] Geography Markup Language (GML) es un estándar de OGC que especifica una gramática XML para expresar atributos geográficos. http://www.opengeospatial.org/standards/gml

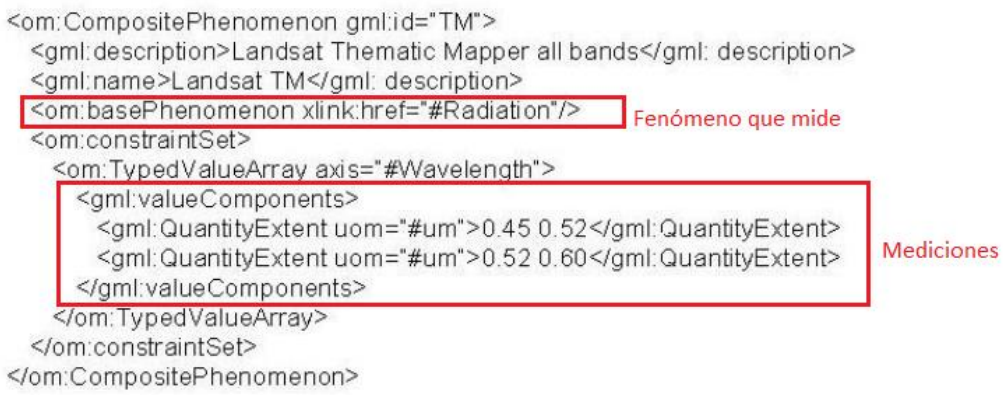


Figura 6 - Esquema XML para descripción de fenómenos [23].

2.3.1.2. Sensor Model Language

El estándar Sensor Model Language (SensorML) (versión 2.0) [21], [24] define un esquema XML para describir sistemas de sensores y procesos asociados a las observaciones del sensor. Provee la información necesaria para descubrir el sensor, la ubicación de las observaciones tomadas y el procesamiento de bajo nivel de dichas observaciones.

El objetivo es dar información del sensor para encontrar datos, y ayudar al proceso y análisis de las mediciones del sensor, la geolocalización de los datos y obtener las propiedades fundamentales en cuanto a sensores.

Se puede obtener la siguiente información [21]:

- Características de la observación: propiedades físicas medidas (p. ej. humedad, temperatura), características de calidad (p. ej. precisión, exactitud) entre otras.
- Características geométricas: forma, tamaño.
- Descripción y documentación: Información sobre el sensor, historia e información de referencia.

La descripción del sensor se divide en nueve componentes informativos, alguno de los cuales poseen subcomponentes tal como se aprecia en la Figura 7.

Las propiedades necesarias para la descripción del sensor son [21]:

- IdentifiedAs: retorna la identificación del sensor (identificador, nombre y tipo del sensor).
- documentConstrainedBy: tiempo de observación para el cual el sensor es válido junto con el nivel de seguridad y restricciones legales de su uso.
- measures: describe los parámetros del sensor, las características de las medidas y las características de respuestas relativas a las medidas.

- ObservationLocatedUsing: muestra el estado del sensor, indicando su localización, orientación, velocidad, aceleración, transformaciones de coordenadas y operaciones matemáticas útiles. La posición y orientación queda especificada dentro de un sistema de coordenadas georreferenciadas o en relación a coordenadas locales, usando la norma ISO-19.111 para especificar sistemas y transformaciones de coordenadas.

El resto de las propiedades son opcionales, conteniendo únicamente las propiedades que son relevantes al tipo u objetivo del sensor.

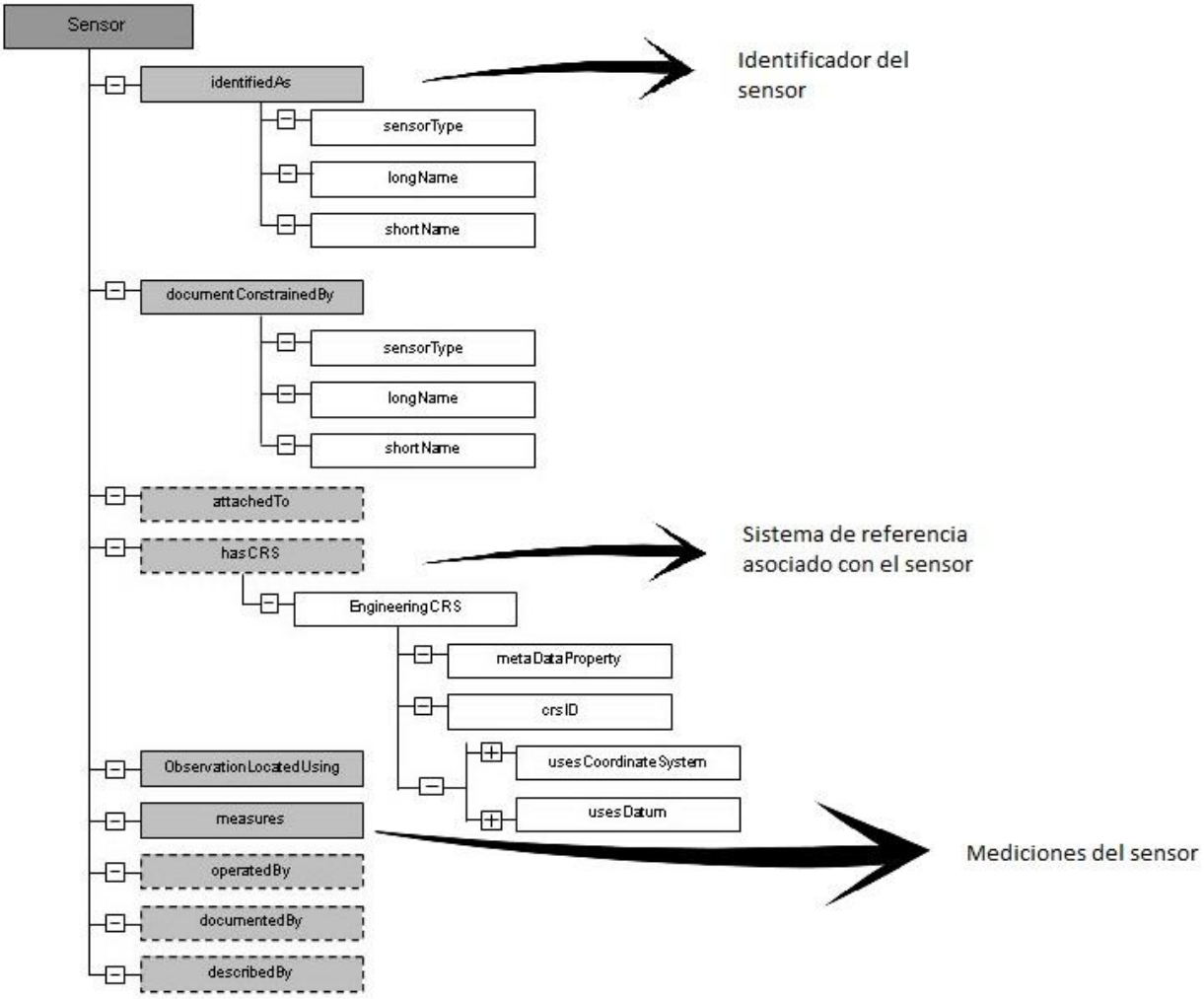


Figura 7 - Diagrama de componentes de SensorML [21].

2.3.1.3. Sensor Observation Service

Sensor Observation Service (SOS) (versión 2.0) [25] es una especificación de Web Service para la solicitud, filtrado y recuperación de observaciones e información de un sistema de sensores en tiempo real.

El resultado de las consultas se obtiene en el formato O&M para modelar observaciones de sensores, y la especificación SensorML para modelar sensores y sistemas de sensores.

La utilización de este estándar facilita el acceso a los datos y además facilita la integración en GIS y en infraestructuras de datos geográficos.

A continuación, se listan algunas de las operaciones que posee (la versión 2.0) [26]:

- GetCapabilities: Son los metadatos del servicio SOS. Retorna un XML con información de la interfaz, datos de los sensores, períodos en los cuales el sensor tiene datos disponibles.
- GetObservation: Obtiene los datos de observación y medida del sensor mediante una consulta espacio-temporal. Los datos se obtienen con el formato O&M.
- DescribeSensor: Devuelve los metadatos del sensor en formato SensorML. Contiene la información de los sensores, la identificación y clasificación, posición y fenómenos observados y también puede incluir más información, como por ejemplo, datos de la calibración.
- InsertSensor: Publica un nuevo sensor.
- InsertObservation: Inserta datos de los sensores.

2.4. Otros estándares GIS

Una forma de interactuar con los mapas geográficos es utilizar algunos de los estándares definidos por OGC, los más conocidos son Web Map Service y Web Feature Service.

2.4.1. Web Map Service

El estándar Web Map Service (WMS) (versión 1.3) [27] definido en el ISO 19.128, es un servicio cartográfico que produce mapas de datos geográficos en forma dinámica a partir de datos georreferenciados. No se obtienen los datos originales, sino que se obtiene una representación gráfica de ellos. Por lo general se obtienen en formatos PNG, GIF o JPEG, y algunas veces en SVG o WebCGM.

El estándar especifica las siguientes operaciones:

- GetCapabilities: Obtiene los metadatos del servicio, que contienen una descripción de los datos publicados y de los parámetros aceptados por el servicio.
- GetMap: Obtiene una imagen de un mapa con parámetros geográficos y dimensionales definidos. El cliente que utiliza esta función obtiene un conjunto de pixeles que forman una imagen de un mapa que corresponde a una zona geográfica dada o a un conjunto de elementos gráficos dentro de dicha zona.

Además, permite que el cliente pueda elegir qué capas desea utilizar para formar la imagen a obtener, así como también especificar el sistema de referencia a usar, el formato de la imagen a recibir y el área geográfica a cubrir.

- GetFeatureInfo: Es utilizada por el cliente para obtener información particular de alguna de las entidades presentes en el mapa.

2.4.2. Web Feature Service

El estándar Web Feature Service (WFS) (versión 2.0.2) [28] definido en el ISO 19.142 específica distintos tipos de operaciones para la consulta y edición de entidades geográficas vectoriales. En los párrafos siguientes se listan los distintos tipos de operaciones que especifica WFS.

Las operaciones de descubrimiento permiten identificar qué capas tiene un servicio [29]:

- GetCapabilities: Retorna un archivo XML con información general del servicio y específica del servicio WFS, como por ejemplo el autor, los sistemas de referencia soportados, objetos geográficos que contiene, formatos de salida de la imagen y las operaciones soportadas por cada tipo de entidad.

- DescribeFeatureType: Retorna la descripción de los tipos de objetos geográficos que el servicio puede ofrecer en un archivo XML. También se especifica para cada tipo de objeto geográfico la manera de codificar los objetos para enviarlos como datos de entrada en operaciones de inserción, actualización o sustitución, y la manera de modificarlos cuando son enviados como datos de salida (en las respuestas de las operaciones GetPropertyValue, GetFeature o GetFeatureWithLock).

Las operaciones de consulta permiten obtener entidades geográficas y los valores de sus atributos filtrando de acuerdo a las restricciones previamente definidas por el cliente [29]:

- GetFeature: Retorna una selección de objetos geográficos en formato GML. También es posible realizar un filtro basado en sus propiedades para obtener los objetos geográficos deseados y realizar tanto consultas geográficas como no geográficas.

- GetPropertyValue: Retorna el valor (o parte de él) de una propiedad de un objeto geográfico. Esta operación posee varios elementos que contienen las descripciones de las consultas. Se diferencia con GetFeature en que en esta operación no se obtiene el GML, sino que solo se obtienen los valores de las propiedades solicitadas.

También posee operaciones de bloqueo que permiten el acceso exclusivo a las entidades geográficas con el propósito de modificarlas o eliminarlas. Estas operaciones son LockFeature y GetFeatureWithLock.

Las operaciones transaccionales permiten crear, modificar, reemplazar y eliminar entidades almacenadas en la base de datos [29]:

- Transaction: Una petición transaccional se compone de operaciones que modifican instancias de objetos geográficos que son accesibles vía web. Luego de que se termina la transacción, el servicio realiza las modificaciones realizadas en la base de datos que se encuentra conectada, generando un documento XML con el resultado de la operación.

  Las operaciones de modificación que pueden llevarse a cabo son: insert, update, replace y delete.

Por último, las operaciones de consultas almacenadas permiten crear, eliminar, listar y describir expresiones de consulta con parámetros que se almacenan en el servidor, para que se puedan invocar repetidas veces con diferentes valores de parámetros. Éstas son CreateStoredQuery, DropStoredQuery, ListStoredQueries y DescribeStoredQueries.

2.4.3. Simple Features Access for SQL

El estándar Simple Features Access (SFA) (versión 1.2.1) (ISO 19.125) [30] específica un esquema SQL que permite almacenar, recuperar, consultar y actualizar las colecciones de atributos mediante una interfaz SQL.

En una implementación SQL, los atributos de un tipo son almacenados en una tabla con sus respectivas geometrías. Cada atributo es representado como una fila en dicha tabla. Los atributos no geográficos son mapeados en columnas con su respectivo tipo, mientras que los atributos geográficos son mapeados en columnas de tipo geometría. De esta forma, SFA define una serie de operaciones SQL que permite operar sobre dichos tipos.

Todas las geometrías que se definen según este esquema, son geometrías en un espacio bidimensional, y cada objeto geométrico está asociado a un sistema de referencia que lo define. Existe un objeto llamado Geometry del cual heredan los restantes objetos formando una jerarquía bien definida tal como se aprecia en la Figura 8.

Los métodos del objeto Geometry son:

- Métodos básicos que proveen información sobre el objeto (dimensión, tipo de geometría, sistema de referencia, etc.).
- Métodos para comprobar relaciones geográficas entre objetos geométricos (cruza a, contiene a, se intersecta con, etc.).
- Métodos que realizan algún tipo de análisis (unión de geometrías, distancia entre geometrías, área de influencia de una geometría, etc.).

Además, cada objeto derivado de la raíz Geometry tiene sus propios métodos específicos, dentro de alguno de los grupos anteriores.

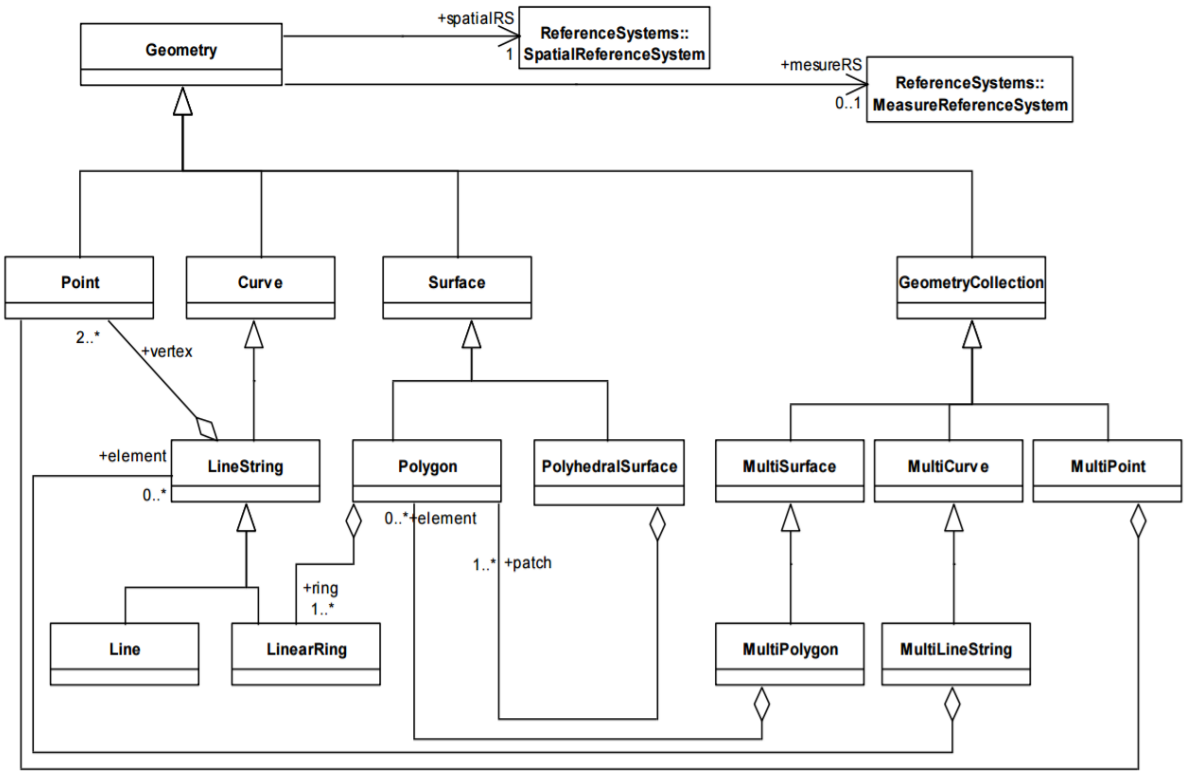


Figura 8 - Jerarquía de tipos de Geometría de SQL [30].

2.5. Tecnologías empresariales

En esta sección se describe qué es un ESB. Además, se muestran patrones de diseño utilizados para el ESB. El capítulo finaliza describiendo el procesamiento de eventos complejos.

2.5.1. Enterprise Service Bus

Un Enterprise Service Bus es una plataforma de integración basada en estándares que combina mensajería, Web Services, transformación de datos y ruteo inteligente para conectar y coordinar de forma confiable la interacción de un gran número de aplicaciones diversas a través de empresas extendidas (empresas más socios de negocios) con integridad transaccional. [31]

Anteriormente, las aplicaciones se conectaban punto a punto con otras aplicaciones utilizando sus interfaces. Esto generaba que hubiera problemas de acoplamiento y que no fuera sencillo la comunicación con una nueva aplicación. Con el uso de ESB, las aplicaciones dejan de

comunicarse directamente y pasan a hacerlo a través de mensajes que circulan por el ESB logrando disminuir el acoplamiento entre ellas.

A continuación, se listan algunas de las principales características que brindan los ESB:

- **Conectividad con varios protocolos:** Un ESB suele incluir varios adaptadores que le permiten conectarse con distintos protocolos de comunicación. Esto además facilita la conexión entre dos sistemas que utilicen protocolos distintos.
- **Transformación de mensajes:** El ESB es capaz de transformar un mensaje, ya sea agregando, modificando o quitando datos, posibilitando la comunicación entre dos aplicaciones que utilicen distintos formatos o modelo de datos.
- **Ruteo de mensajes:** Cuando llega un mensaje al ESB, éste determina estática o dinámicamente el o los destinos del mensaje.
- **Flujos de mediación:** Especifican una secuencia de operaciones que debe ejecutarse al recibir un nuevo mensaje.

En la Figura 9 se presenta un ejemplo básico de una situación en la que sería útil utilizar un ESB. Se pueden ver las relaciones de comunicación entre una aplicación Java, una aplicación .Net y un servidor de sensores.

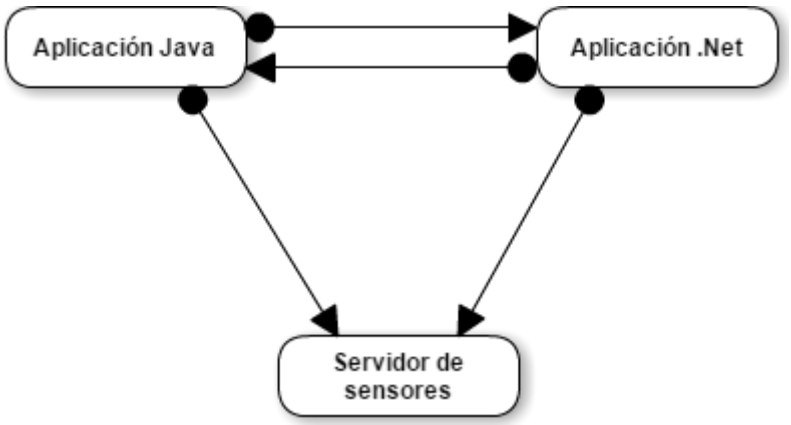


Figura 9 - Diferentes aplicaciones comunicadas entre sí.

En este ejemplo, una posible problemática es la comunicación directa entre las aplicaciones, ya que pueden llegar a resultar complejas. Entre más aplicaciones y más relaciones existan entre ellas, mayor será la complejidad y dependencia, logrando un sistema rígido, poco flexible ante cambios y difícil de mantener.

Otro posible problema podría ser que no todas las aplicaciones soporten la comunicación con las demás, con lo cual se necesitaría la construcción de un adaptador para comunicarlas.

Si en lugar de tener el sistema anterior, se tiene un elemento que actúe de mediador entre todas las aplicaciones, se logra que éste sea el único punto de interacción entre las aplicaciones abstrayendo la complejidad de la comunicación entre ellas.

En la Figura 10 se presenta el sistema si se incluye un ESB como mediador entre las distintas aplicaciones.

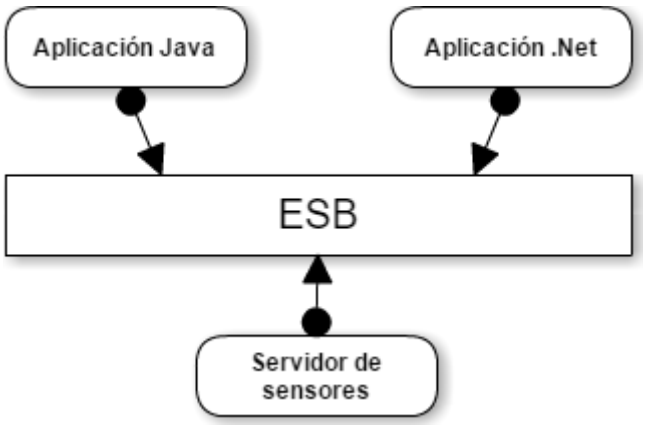


Figura 10 - Comunicación de las aplicaciones a través de un ESB.

Primeramente, se observa que se simplifican las interacciones ya que ahora se encapsulan en el ESB volviéndose las mismas transparentes para las aplicaciones. Con esta solución se logra centralizar las invocaciones, disminuir el acoplamiento, lograr mayor flexibilidad, reusabilidad y, en consecuencia, facilitar el mantenimiento del sistema.

El ESB además de actuar de intermediario, provee componentes de integración de uso común y estandarizados, prontos para utilizarse en poco tiempo lo cual logra reducir significativamente los tiempos de implementación.

2.5.2. Enterprise Integration Patterns

Las necesidades y problemas que surgen en la integración de sistemas y aplicaciones son comunes a la mayoría de las organizaciones y son mayores conforme crece el tamaño o complejidad tecnológica. De esta forma se originaron los Enterprise Integration Patterns donde se intenta resolver esta problemática.

Los Enterprise Integration Patterns (EIPs) [32] identifican problemas comunes que se tienen cuando se quiere integrar varios sistemas y presentan una solución de manera unificada para resolverlos.

En la Figura 11 se muestran los distintos patrones de mensajería separados por categorías.

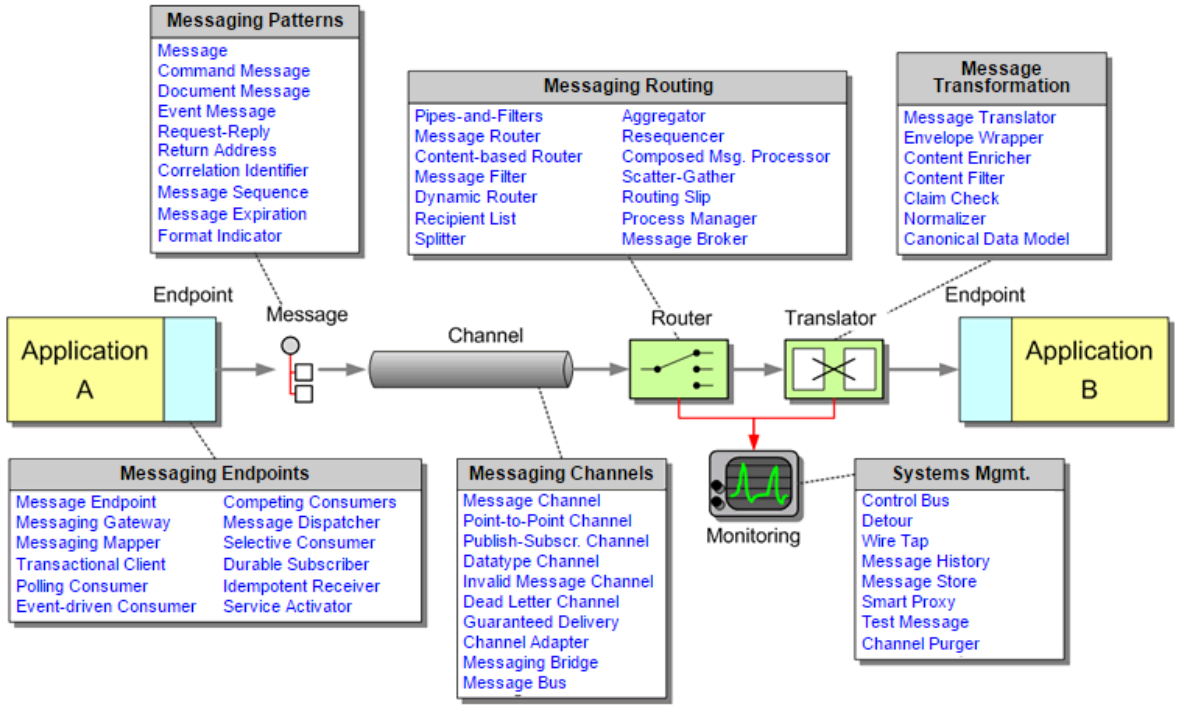


Figura 11 - Patrones de mensajería [32].

Se describen en la siguiente lista los patrones de mensajería de la Figura 11.

- **Routing Patterns:** describen cómo se enrutan los mensajes desde el emisor hasta el receptor correcto. Consumen un mensaje de un canal y vuelven a publicarlo, por lo general, sin modificaciones.
- **Transformation Patterns:** cambian el contenido de un mensaje, por ejemplo, para adaptarse a diferentes formatos de datos usados por el emisor y receptor. Puede ser necesario agregar datos al mensaje, extraer datos u ordenarlos.
- **Endpoint Patterns:** describen cómo los clientes del sistema de mensajería producen o consumen los mensajes.
- **System Management Patterns:** describen las herramientas de administración del sistema.

Los ESBs existentes en el mercado suelen tener implementados estos patrones para su utilización.

Una solución ESB consiste en la utilización de uno o más de los patrones mencionados anteriormente, que permite la resolución de un problema dado. En la Figura 12, se presenta un ejemplo de una solución ESB que resuelve el problema de enviar un mensaje a múltiples destinos.

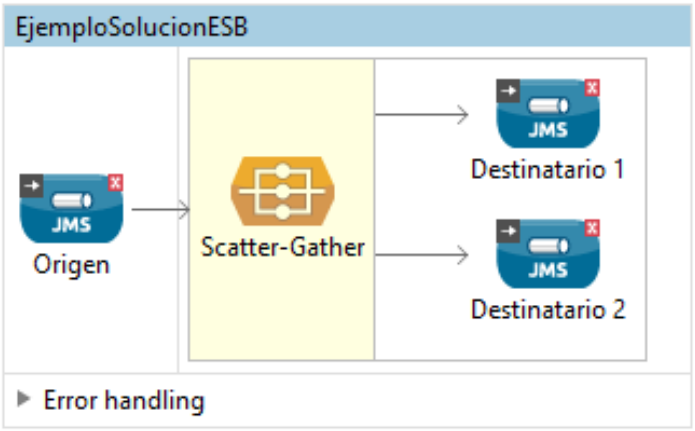


Figura 12 – Solución ESB envío de mensajes.

2.5.3. Procesamiento de eventos complejos

Esta sección inicia con una breve descripción de CEP. Luego se describen los patrones habitualmente utilizados, un metamodelo para CEP y una posible extensión geográfica para dicho modelo. La sección finaliza describiendo el lenguaje utilizado por algunos motores CEP y una arquitectura comúnmente utilizada para el procesamiento de eventos complejos. Previamente es necesario definir la palabra evento para poder entender las siguientes subsecciones.

Un evento [33] es definido como algo que ocurre en un sistema. Por ejemplo, la llegada de un correo electrónico, la creación de un nuevo tweet o una medición de un sensor. Los eventos pueden estar relacionados con otros eventos ya sea por el tiempo, por causalidad o por agregación. También dentro de los eventos sucede que algunos son más importantes que otros.

2.5.3.1. Descripción de CEP

El procesamiento de eventos complejos (Complex Event Processing, CEP) [34] es una tecnología de computación en la cual conjuntos de eventos, muchas veces miles, son convertidos a eventos comprensibles por máquinas. Los eventos complejos suelen ser utilizados en el ambiente empresarial para obtener la información de lo que está ocurriendo en tiempo real. CEP relaciona atributos en común de los eventos obtenidos de una o más fuentes, con la finalidad de ejecutar una acción.

Por ejemplo, CEP puede ser utilizado en los mercados financieros ya que son muy cambiantes en poco tiempo. Algunas decisiones de compra y venta pueden ser realizadas en menos de un milisegundo luego de obtener un nuevo dato cambiario como evento. En estos sistemas se encuentra todo automatizado ya que no es posible que lo realice una persona debido al tiempo que le llevaría. [34]

El objetivo principal del procesamiento de eventos complejos es la identificación de determinados eventos importantes para el cliente con el fin de tomar acciones de forma oportuna. Para realizar esta identificación, se definen los patrones de eventos complejos.

2.5.3.2. Patrones de eventos

A continuación, se muestran algunos de los patrones de eventos más utilizados en aplicaciones con procesamiento de eventos complejos [35]:

- Filtrado de eventos: Sirve para encontrar determinados datos en un flujo de eventos, filtrando aquellos que no son relevantes.
- Detección de nuevos eventos: Permite identificar nuevos eventos a medida que se generan.
- Enriquecimiento del evento: Permite agregar información a los eventos entrantes, de acuerdo a sus propiedades.
- Agrupación de eventos: Sirve para agrupar eventos de acuerdo a alguna de las propiedades indicadas por el usuario.
- Correlación de eventos: Permite correlacionar eventos de uno o más flujos de acuerdo a datos en común.
- Detección de evento faltante: Alerta si un evento esperado no llega luego de un determinado tiempo.

2.5.3.3. Metamodelo CEP

En [36] los autores definen un metamodelo para CEP que define los elementos del modelo y las relaciones de los mismos, el cual se presenta en la Figura 13. En este metamodelo se observan las siguientes clases:

- CEPDomain: Es la clase principal, todo modelo tendrá una única instancia CEPDomain. Representa un dominio CEP compuesto por uno o más eventos.
- Events: Describe un evento para un CEPDomain. Los eventos tienen un tipo y pueden llegar a tener una imagen con su representación gráfica. Además, los eventos pueden tener una o más propiedades.
- EventProperty: Representa la propiedad o atributo de un evento. Estas propiedades tienen un nombre y son de uno de los siguientes tipos: Unknown, Boolean, Integer, Long, Double, Float o String.

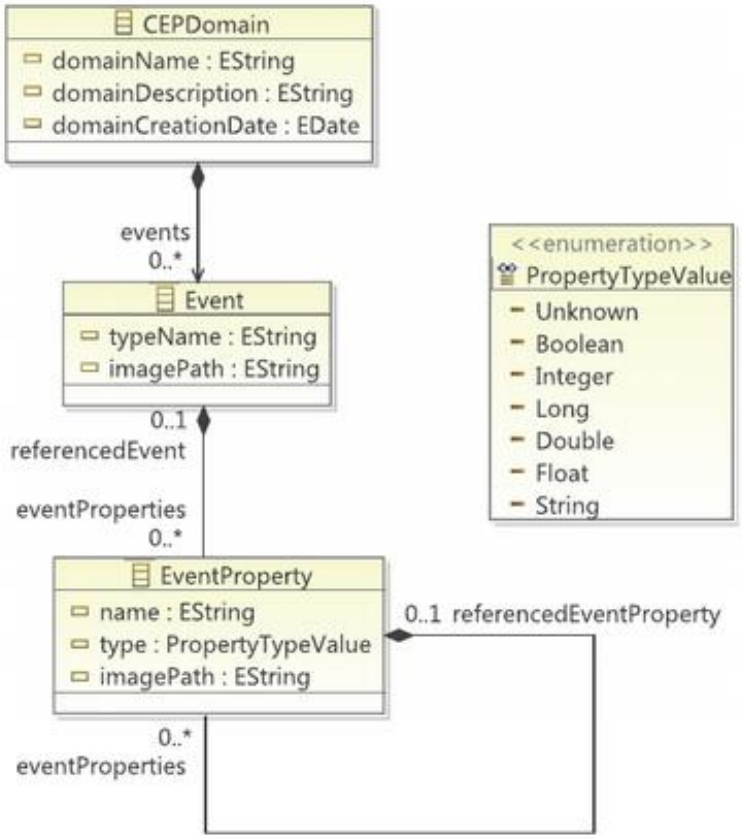


Figura 13 - Metamodelo de dominio CEP [36].

2.5.3.4. Extensión geográfica

Una de las características de las tecnologías CEP es que no suelen ofrecer soporte para eventos con geometrías como las vistas en la sección 2.4.3. Tampoco ofrecen operadores geográficos para relacionar eventos según su posición geográfica. Para resolver dicha problemática, en [37] los autores definen una extensión geográfica para el metamodelo de CEP. En la extensión se pueden definir propiedades de tipo Geometría en el evento y también agregar funciones geométricas para relacionar las geometrías de los eventos. En la Figura 14 se muestra el modelo de dominio considerando los eventos geográficos.

De esta manera, es posible definir un área de interés en el mapa, con el fin de únicamente tener en cuenta los eventos que estén incluidos dentro de esa área. Esto es posible realizarlo con la función geométrica Intersects.

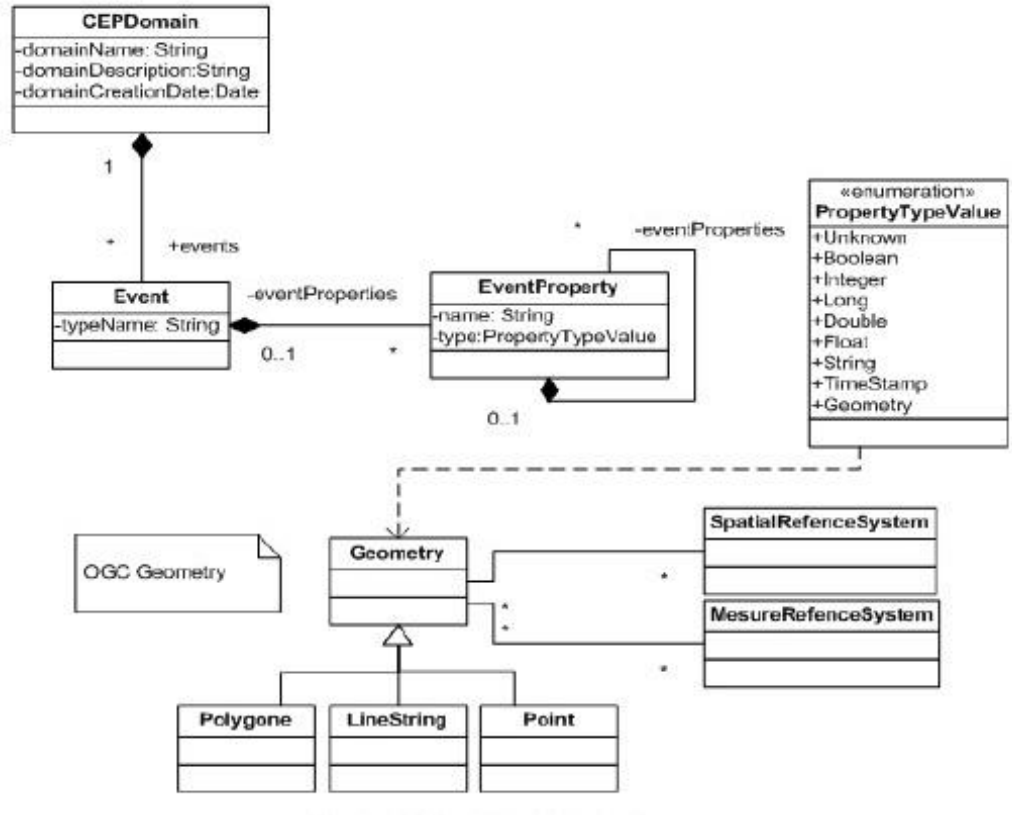


Figura 14 - Modelo de dominio de los eventos geográficos [37].

2.5.3.5. Event Processing Language

Event Processing Language (EPL) [38] es un lenguaje SQL extendido y posee las cláusulas SELECT, FROM, WHERE, GROUP BY, HAVING BY y ORDER BY. Como los eventos se componen de datos, los conceptos de correlación de SQL como la unión, filtrado y la agregación se pueden aprovechar de manera eficaz. Los flujos de eventos reemplazan a las tablas como fuentes de datos y los eventos reemplazan a las filas como datos básicos.

EPL pueden clasificarse en tres categorías [39]:

- Orientados a flujos: están basados en SQL, en álgebra relacional y extendidos para soportar ventanas temporales de datos. Un ejemplo de EPL orientado a flujo es Esper[7].

- Orientados a reglas: este tipo de EPL se clasifica en tres subtipos:

[7] http://www.espertech.com/esper/index.php

  - Reglas de producción: son reglas de tipo: “condición entonces acción”, esto significa que cuando ocurre la “condición” se ejecuta la “acción”. Un ejemplo de este tipo de EPL es Drools Fusion[8].

  - Reglas activas o de evento-condición-acción: están basadas en que cuando ocurre un evento, se evalúan las condiciones y en caso de corresponder se ejecuta la acción correspondiente. Un ejemplo de este tipo de EPL es IBM Operational Decision Managment[9].

  - Reglas de programación lógica: están basados en afirmaciones lógicas y en base de datos deductivas. Un ejemplo de este tipo es ELE[10].

- Imperativos: en este caso, mediante programación imperativa se definen el conjunto de operadores a aplicar sobre los eventos donde cada operador especifica una transformación sobre los eventos recibidos. Un ejemplo de este tipo es Apama[11].

EPL es muy similar a SQL, pero en este lenguaje se utilizan vistas en lugar de tablas. Las vistas representan las distintas operaciones que se necesitan para estructurar y obtener los datos que vienen de los flujos de eventos.

Las consultas EPL se crean y almacenan en el motor CEP utilizado y envían los resultados a los suscriptores del mismo. A continuación, se puede ver un ejemplo sencillo de utilización del lenguaje EPL:

“select avg(price) from StockTick.win:time(30 sec) where symbol='IBM'”

La cláusula SELECT especifica las propiedades del evento que se desea obtener. La cláusula FROM especifica las definiciones de los flujos de eventos y nombres a utilizar. La cláusula WHERE especifica las condiciones de búsqueda que indica el evento o la combinación de eventos a buscar. En el ejemplo anterior se devuelve el precio promedio de las acciones de IBM en los últimos 30 segundos.

Las consultas EPL pueden ser tan simples como la del ejemplo o pueden llegar a ser mucho más complejas.

2.5.3.6. Event Driven Architecture

Event Driven Architecture (EDA) [40] se basa en recibir eventos y de acuerdo a los eventos recibidos tomar determinadas acciones. EDA es una arquitectura de bajo acoplamiento y alta cohesión ya que los creadores de los eventos no poseen conocimiento del posterior procesamiento que tendrá el evento.

[8] http://drools.jboss.org/drools-fusion.html
[9] http://www-03.ibm.com/software/products/en/odm
[10] https://code.google.com/archive/p/etalis/
[11] http://www2.softwareag.com/corporate/products/apama_webmethods/analytics/default.aspx

El flujo del evento es representado de acuerdo a cuatro capas lógicas.

1. Generadores: Los generadores son quienes crean los eventos, como por ejemplo pueden ser una aplicación, un servicio, un mail o un sensor.

2. Canal: El canal del evento es por donde viaja el mismo. Es el nexo entre los generadores de eventos y el motor de procesamiento.

3. Procesamiento: Una vez recibido el evento este es evaluado por el motor de procesamiento. Para evaluarlo existen reglas previamente definidas por el usuario que permiten definir qué acciones tomar.

4. Consecuencia: Es la consecuencia que ocurre luego de evaluar los eventos obtenidos. Estas acciones pueden ser invocar un servicio, publicar un nuevo evento o notificar a usuarios.

EDA es generalmente utilizada en CEP debido a que recibe eventos, los interpreta y de acuerdo a reglas definidas por el usuario toma determinadas acciones.

3. Análisis de la problemática planteada

En este capítulo se presentan los requerimientos que se identificaron y el análisis realizado para resolver la problemática planteada en el Capítulo 1. Además, se presentan soluciones existentes que intentan resolver problemáticas relacionadas a la planteada.

3.1. Análisis de requerimientos

De acuerdo a la problemática planteada en este proyecto se detallan, a continuación, los requerimientos relevados que deben ser resueltos por la solución a desarrollar.

3.1.1. Control de acceso basado en roles

El sistema debe poder ser utilizado por usuarios pertenecientes a distintos roles, con acceso a diferentes funcionalidades según su rol. Los roles que puede tener un usuario dentro de una organización son: Administrador de organización, administrador de leyes y administrador de alertas. De acuerdo al análisis realizado se identificaron cinco roles predefinidos:

- **Superusuario**: Este rol tiene acceso total a la plataforma y a todas sus funcionalidades.
- **Administrador de organización:** Este rol es el responsable de administrar una organización. Debe tener acceso y control total de la organización, incluyendo la administración de la misma.
- **Administrador de leyes:** Este rol es responsable de la gestión de leyes en su organización.
- **Administrador de alertas:** Este rol es el responsable de la gestión de alertas en su organización.
- **Cliente:** Este rol solo puede utilizar las funciones básicas de la plataforma.

Cabe destacar que un usuario puede tener distintos roles en la plataforma, uno por cada organización a la que pertenezca.

3.1.2. Gestión de usuarios

RGU1: El sistema debe permitir dar de alta, modificar y eliminar usuarios. Por defecto el rol de un nuevo usuario es el de cliente y queda asociado a la organización llamada “Organización pública”.

RGU2: El sistema debe permitir al usuario seleccionar con cuál de las organizaciones a las que pertenece desea trabajar.

3.1.3. Gestión de organizaciones

Dentro de las organizaciones se distinguen dos tipos: las públicas y las privadas. Las organizaciones públicas permiten que cualquier usuario se suscriba a ella, mientras que en las privadas es necesario ser invitado por el administrador de organización de dicha organización. Los usuarios suscriptos o invitados a una organización recibirán notificaciones en caso del incumplimiento de una ley medioambiental de la organización.

RGO1: El sistema debe permitir dar de alta organizaciones, quedando éstas en estado pendiente de aprobación.

RGO2: El sistema debe permitir al rol superusuario aprobar o rechazar organizaciones. Se le debe notificar la decisión al usuario que creó la organización. En caso de aprobación el usuario debe quedar con el rol de administrador de organización.

RGO3: El sistema debe permitir al rol administrador de organización modificar y eliminar la organización.

RGO4: El sistema debe permitir al rol administrador de organización invitar a otros usuarios a formar parte de la organización que pertenece, seleccionando el rol que tendrá dentro de la misma. Estos usuarios deben ser notificados por el sistema.

RGO5: El sistema debe permitir a los usuarios suscribirse y desuscribirse a organizaciones públicas.

3.1.4. Gestión de tipos de fuentes

Los tipos de fuente son protocolos (p. ej. SOS y Twitter API) que el sistema soporta y poseen una solución ESB que consume los datos de la fuente en tiempo real.

RGTF1: El sistema debe permitir el alta de tipos de fuentes. El sistema debe recibir los parámetros ingresados por el usuario y una solución ESB para consumir la fuente en tiempo real.

3.1.5. Gestión de fuentes

RGF1: El sistema debe permitir dar de alta, modificar y eliminar fuentes. Al dar de alta la fuente, ésta queda asociada a la organización en la cual el usuario se encuentra trabajando. Se debe indicar si la fuente es de uso público, privado o compartido.

Las fuentes de uso público deben poder ser consultadas y utilizadas por cualquier usuario del sistema. Las fuentes privadas solo deben poder ser consultadas y utilizadas por determinados usuarios. Si la fuente se encuentra asociada a la organización pública, sólo puede ser consultada y utilizada por el usuario que la creó. En cambio, si la fuente se encuentra asociada a otra organización, puede ser consultada y utilizada por los miembros pertenecientes a dicha organización. Las fuentes compartidas pueden ser consultadas por los miembros de la

organización asociada a la fuente y los miembros de las organizaciones con las cuales se encuentre compartida.

RGF2: El sistema debe poder compartir fuentes entre organizaciones.

RGF3: El sistema debe procesar los datos de las fuentes en tiempo real.

3.1.6. Visualización de datos

RVD1: El sistema debe permitir visualizar la última medición de una fuente.

RVD2: El sistema debe proporcionar una vista gráfica con los valores de las mediciones realizadas en un determinado período de tiempo.

3.1.7. Gestión de leyes

RGL1: El sistema debe permitir al administrador de organización o administrador de leyes dar de alta, modificar y eliminar leyes.

3.1.8. Gestión de alertas

RGA1: El sistema debe permitir dar de alta, modificar y eliminar alertas. Al dar de alta la alerta queda asociada a la organización en la cual el usuario se encuentra trabajando. Se debe indicar si la alerta es de uso público o privado.

RGA2: El sistema debe permitir a los usuarios suscribirse y desuscribirse a alertas públicas.

RGA3: El sistema debe suscribir automáticamente a los miembros y usuarios suscriptos de una organización, al momento que se da de alta una alerta asociada a dicha organización.

RGA4: El sistema debe comenzar inmediatamente a monitorear el incumplimiento de la ley luego que la alerta se da de alta.

RGA5: El sistema debe permitir la creación de alertas seleccionando las fuentes que se desea utilizar.

RGA6: El sistema debe permitir la creación de alertas seleccionando una zona geográfica. Se deben tener en cuenta todas las fuentes que se encuentren en la zona definida.

RGA7: El sistema debe permitir la creación de alertas preventivas y reactivas. Las preventivas son utilizadas para prevenir, por ejemplo, el incumplimiento de una ley, mientras que la reactiva notifica cuando se está incumpliendo una ley.

3.1.9. Envío de notificaciones

REN1: El sistema debe notificar a los usuarios suscriptos a una alerta, cuando se incumple la ley asociada.

REN2: El sistema debe permitir que el usuario pueda elegir las vías de comunicación, por las cuales se recibe la notificación.

3.2. Modelo conceptual

En base a estos requerimientos, que debe satisfacer el sistema, se obtiene el modelo conceptual que se aprecia en la Figura 15.

Se observa que los usuarios tienen su correspondiente rol de acuerdo a las organizaciones a las que pertenecen. Los roles pueden ser superusuario, administrador de organización, administrador de leyes, administrador de alertas y cliente.

Los administradores de organización y los administradores de leyes son quienes ingresan leyes a la plataforma. En la ley se ingresan los parámetros que la ley controla, por ejemplo, Ph, Cromo Total, etc. Las leyes se pueden estar relacionadas entre ellas mismas y también pueden estar relacionadas con alertas. Las leyes pueden tener categorías que indican donde se aplica la ley como, por ejemplo, agua, suelo o bosque.

Las alertas son creadas por usuarios y pueden ser de tipo preventivas o reactivas. Las alertas procesan los datos generados las fuentes y, en caso de detectar un incumplimiento, se generan las notificaciones correspondientes.

Las fuentes son publicadas por los usuarios, y poseen un tipo de protocolo existente en la plataforma.

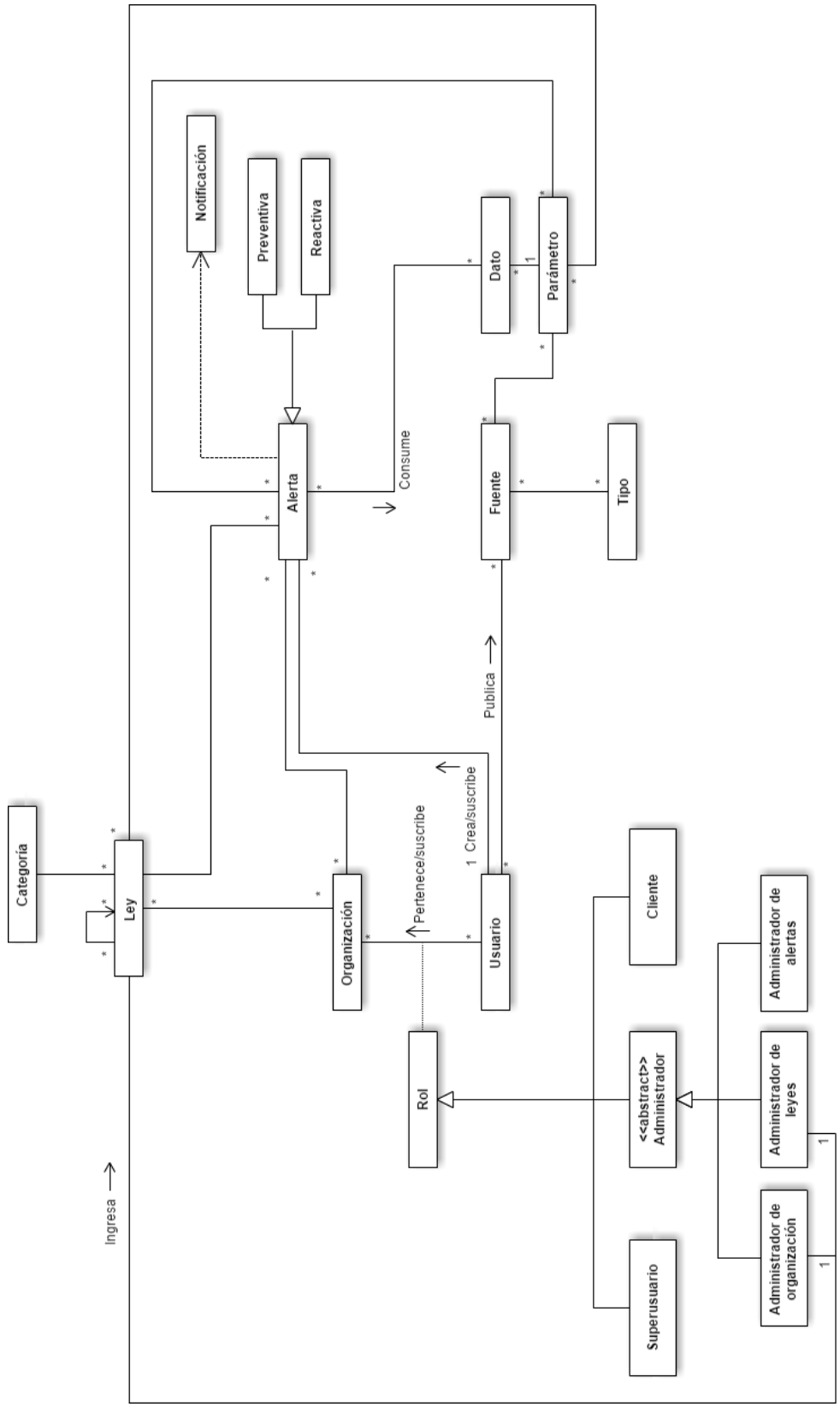


Figura 15 - Modelo conceptual.

3.3. Análisis de soluciones similares

En la actualidad existen muchas plataformas IoT que permiten a los usuarios conectar sus dispositivos. Se describe a continuación las características más importantes de las plataformas IoT más populares. [41]

3.3.1. Xively

Xively[12] anteriormente llamada Pachube, fue creada en 2007. Es una plataforma que pone a disposición de cualquier usuario la posibilidad de conectar sus sensores a una nube, con el fin de permitirle construir aplicaciones basadas en los datos provenientes de los mismos. Alcanzó su máxima popularidad tras el desastre nuclear que ocurrió en Fukushima, debido a la gran cantidad de consultas realizadas sobre los datos generados por un sensor de radiación. De esta manera se podía contrastar la información generada por el sensor con los datos que ofrecía el gobierno japonés.

Esta plataforma IoT proporciona la posibilidad de obtener datos de los dispositivos mediante los protocolos HTTP/S, Sockets/Websocket y MQTT.

3.3.2. Carriots

Carriots[13] es una plataforma española en la nube, que ofrece una plataforma como servicio (PaaS) orientada a proyectos IoT y máquina a máquina (M2M). Carriots no es una plataforma de código abierto. Ofrece un servicio para conectar dispositivos a una nube privada. Desde la web de Carriots aseguran que cualquier dispositivo es compatible con su plataforma siempre y cuando tenga acceso a internet. Carriots ofrece una API propia basada en REST para poder comunicarse con la plataforma y gestionar los datos de una forma sencilla. REST se caracteriza principalmente por la existencia de recursos que pueden ser accedidos utilizando un identificador global.

3.3.3. ThingSpeak

ThingSpeak[14] es una plataforma abierta de aplicaciones diseñada para permitir conectar personas con objetos. Se caracteriza por ser una plataforma de código abierto con una API para almacenar y recuperar datos de los objetos usando el protocolo HTTP sobre Internet o vía Local Area Network (LAN). También se encuentra integrada con MATLAB permitiendo realizar análisis de los datos obtenidos. Dentro de ThingSpeak existen aplicaciones ya desarrolladas que son un complemento para los proyectos.

[12] https://www.xively.com/
[13] https://www.carriots.com/
[14] https://thingspeak.com/

3.3.4. Conclusiones

En la Tabla 1 se muestran algunas funcionalidades existentes de las soluciones mencionadas anteriormente.

Funcionalidad	**Xively**	**Carriots**	**ThingSpeak**
Insertar sensor	Sí	Sí	Sí
Control de redes sociales	No	No	Sí
Gestión de organizaciones	No	No	No
Alertas de acuerdo a fuentes	Si	Sí	Sí
Alertas por zona	No	No	No
Notificaciones	Sí	Sí	Sí
Visualización de datos	Sí	Sí	Sí
Consumir en tiempo real	Sí	Sí	Sí
Compartir fuentes	Sí	Sí	Sí
Fuentes públicas y privadas	Sí	Sí	Sí

Tabla 1 - Comparación plataformas IoT.

Se observa en la Tabla 1 que solamente la plataforma ThingSpeak realiza un control del contenido de las publicaciones en las redes sociales. Si bien esta plataforma cumple la mayoría de las funcionalidades requeridas, no se adecua a las necesidades del proyecto. Esto se debe porque, al igual que Xively y Carriots, no cuentan con las funcionalidades requeridas, como por ejemplo gestión de leyes, gestión de organizaciones y alertas por zona.

4. Descripción de la solución

En este capítulo se presentan las principales características de la solución propuesta, denominada Sistema de Monitoreo Ambiental (SIMONA). Se comienza con una descripción general de la solución y luego se describen los principales componentes. El capítulo finaliza describiendo la interacción entre los componentes.

4.1. Descripción general

Se propone una plataforma que utiliza tecnologías ESB y CEP, para satisfacer los requerimientos relevados en el Capítulo 3. Esta plataforma permite que los usuarios puedan ingresar sus fuentes de datos y mediante el uso del ESB, la plataforma consume en tiempo real los datos generados por ellas.

Por otro lado, los usuarios tienen la posibilidad de crear distintas alertas sobre parámetros medioambientales, con el fin de poder realizar un control y seguimiento de los mismos. En caso de que algún parámetro tome valores fuera del rango establecido, los usuarios son notificados. Debido a que el volumen de datos, generados por las fuentes, puede llegar a ser muy grande, se utiliza un motor CEP para realizar el procesamiento de los datos.

Además, los usuarios tienen la posibilidad de ingresar tipos de fuente (p. ej. SOS). Éstos se suman a los existentes en la plataforma y quedan disponibles para que cualquier usuario, que posea una fuente de ese tipo, la pueda dar de alta (p. ej. URL de un servidor SOS particular).

En la Figura 16, se muestra un diagrama de alto nivel de los distintos componentes que conforman la solución propuesta. Se observa que los usuarios acceden a la plataforma a través de un navegador web. Desde allí acceden a todas las funcionalidades que posee el sistema, considerando el rol con que ingresa. En el ESB, se encuentran las soluciones que consumen en tiempo real los datos generados por los sensores y por las redes sociales mediante el componente “Consumir Fuente”. Estos datos son convertidos a eventos por el componente “Evento Dato Fuente” y son enviados al motor CEP para su procesamiento y al mismo tiempo, el componente “Guardar Dato Fuente” registra los datos generados por la fuente en la base de datos del sistema. Si en el procesamiento de datos se encuentran valores por fuera del rango establecido en la alerta, el componente “Notificación” notifica a los usuarios correspondientes.

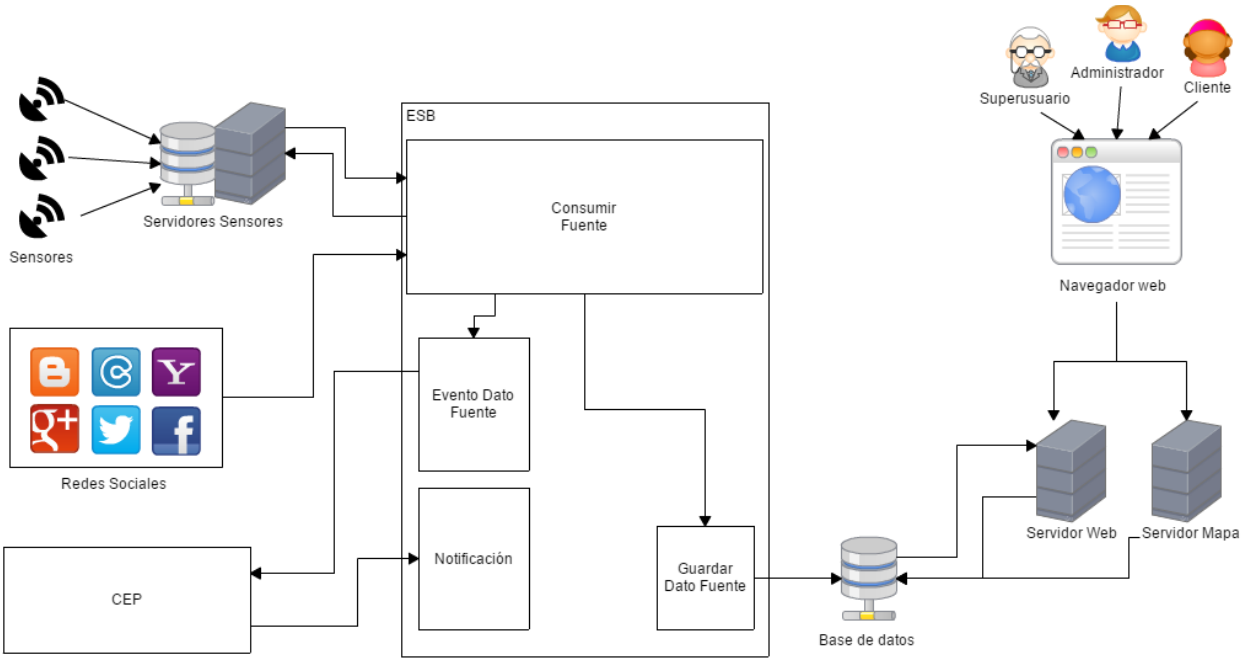


Figura 16 – Solución general.

4.2. Principales funcionalidades

Es esta sección se describen los aspectos más importantes de la solución propuesta.

4.2.1. Manejo de roles

Para solucionar el requerimiento del control de acceso basado en roles, se almacena el rol que tiene el usuario en cada una de las organizaciones a la que pertenece. Al momento que el usuario ingresa a la plataforma, el sistema le permite seleccionar una de las organizaciones a la cual pertenece. El sistema obtiene el correspondiente rol dentro de la organización y le permite utilizar únicamente las funcionalidades correspondientes a su rol.

4.2.2. Extensibilidad de los tipos de fuente

En esta sección se describe la solución propuesta para la creación de un nuevo tipo de fuente.

Para lograr que el sistema fuese extensible, éste permite agregar tipos de fuentes. La solución propuesta consiste en la posibilidad que el usuario proporcione una solución ESB que resuelva el consumo en tiempo real de la fuente y posteriormente sea utilizado para el consumo de las fuentes de este tipo.

Debido a que cada fuente necesita un conjunto de parámetros distintos para poder obtener sus datos, la solución contempla que se pueda indicar cuáles son dichos parámetros para que posteriormente, cuando un usuario dé de alta una nueva fuente de este tipo complete

esos campos. Es decir, los tipos de fuente son parametrizables para dar soporte a varias fuentes concretas de ese tipo.

La solución ESB que el usuario proporciona, debe resolver el consumo en tiempo real de la fuente. Estos datos obtenidos deben ser procesados en SIMONA, lo cual implica guardar el valor de la medición en la base de datos y enviarlo al motor CEP para su procesamiento. La forma de resolver esta situación consiste en que la solución ESB creada por el usuario debe generar un archivo JSON[15] con un formato establecido, tal como se aprecia en la Figura 17. En el JSON se puede observar el identificador de la fuente, los parámetros que ésta mide y las mediciones obtenidas, entre otros. Las mediciones están compuestas por la fecha cuando se realizó la medición, el parámetro que mide y su valor. Luego este JSON debe ser enviado a una cola de mensajes preestablecida por el sistema.

```
{
    "idFuente": "28",
    "sistemaReferencia": "4326",
    "x": "7.651968812254194",
    "y": "51.935101100104916",
    "z": "12",
    "elementoFuente": "1",
    "idTipoParametro": "Temperatura:2;Ph:1;",
    "valores": [{
        "identificadorParametro": "Temperatura",
        "fecha": "2016-11-07T23:42:01.280Z",
        "valor": "-53.988796"
    },
    {
        "identificadorParametro": "Temperatura",
        "fecha": "2016-11-07T23:42:09.609Z",
        "valor": "-53.236885"
    }]
}
```

Figura 17 - Ejemplo del JSON con los datos de la fuente.

La solución ESB es almacenada por el sistema y será utilizada, cada vez que se dé de alta una fuente de este nuevo tipo.

[15] “JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos.” - http://www.json.org/json-es.html

Por otro lado, el sistema procesa los archivos JSON que arriban a la cola de mensaje, encargándose de la inserción en la base de datos y el envío al motor CEP para su procesamiento.

En la Figura 18, se aprecia el diagrama de flujo que ocurre al obtener los datos de una fuente de un nuevo tipo.

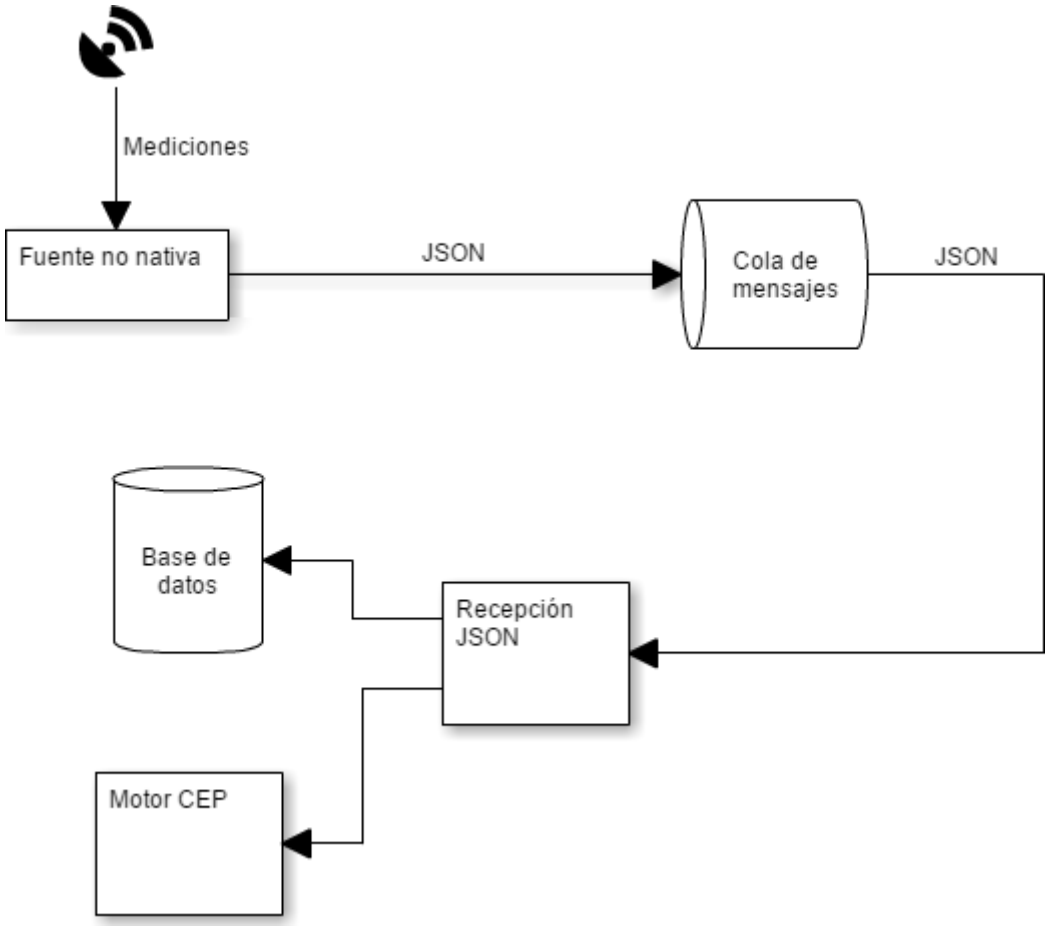


Figura 18 – Diagrama de flujo de datos.

De esta manera se consigue la posibilidad de crear tipos de fuentes, logrando que el sistema sea extensible.

4.2.3. Consumir fuente en tiempo real

Para solucionar el requerimiento del consumo de datos en tiempo real de las fuentes, el sistema tiene almacenada una solución ESB, por cada tipo de fuente, encargada de consumir los datos en tiempo real.

Al dar de alta una nueva fuente, el sistema realiza una copia de la solución ESB almacenada, correspondiente al tipo de la fuente, y graba un conjunto de propiedades. Estas propiedades son necesarias para poder identificar a la fuente en el sistema, así como también contiene los

parámetros específicos del tipo de fuente para poder consumir los datos. En este caso, por cada fuente registrada en el sistema, se estará ejecutando una aplicación de la solución ESB encargada de consumir los datos.

En la Figura 19, se muestra un ejemplo de un posible archivo de propiedades.

```
sensorSOS.id = 28
sensorSOS.identificador = "Sensor1"
sensorSOS.url = localhost
sensorSOS.puerto = 8080
sensorSOS.path = 52n-sos-webapp/service
sensorSOS.frecuencia = 12
sensorSOS.topic = esperTopic
sensorSOS.activeMQ = tcp://localhost:61616
sensorSOS.sistemaReferencia = 4326
sensorSOS.x = 7.651968812254194
sensorSOS.y = 51.935101100104916
sensorSOS.z = 12
sensorSOS.elementoFuente = 1
sensorSOS.idTipoParametro = Temperatura:2;Ph:1;
```

Figura 19 – Archivo de propiedades de una fuente.

4.2.4. Generación de alertas

Para solucionar los requerimientos de gestión de alertas, se proponen distinguir las alertas en dos tipos: las alertas creadas a partir de fuentes de tipo red social y las alertas creadas a partir de fuentes de tipo sensor. Esta distinción se realiza debido a que la información generada en las redes sociales suele no ser tan confiable y exacta como la del sensor. Por lo cual, se separan las alertas de acuerdo a su tipo.

Las alertas que utilizan fuentes de redes sociales, están asociadas a las fuentes de dicho tipo (palabra clave) en una determinada zona geográfica. Además, la alerta tendrá el número de ocurrencias de la palabra que deben ocurrir en un período de tiempo para que se realice una notificación. La solución se realiza de esta manera para contemplar que pueden existir mensajes que contengan la palabra clave, pero no estar hablando de lo que realmente se busca.

Por otro lado, las alertas que utilizan sensores pueden ser de dos tipos: uno es seleccionando las fuentes manualmente y el otro es a partir de una zona geográfica. Esta distinción se realiza debido a la confiabilidad de las fuentes. En la alerta por zona geográfica se tienen en cuenta todas las fuentes que se encuentren en la zona definida. De esta forma, puede suceder que en la zona definida existan fuentes que no se quieran utilizar debido a distintos motivos (p. ej.

confiabilidad). Mientras que las alertas por fuentes, permiten seleccionar las fuentes que se deseen utilizar.

Al igual que en el módulo de gestión de fuentes, existe una solución ESB almacenada en el sistema. En este caso, la solución ESB recibe los eventos generados con los datos de las fuentes y los procesa en el motor CEP. Luego, en caso de ser necesario, es enviado a otro componente del sistema encargado de las notificaciones a los usuarios. Aquí también se estará ejecutando una aplicación de la solución ESB por cada alerta que se crea en el sistema, en la cual las propiedades son modificadas de acuerdo a los datos de la alerta creada.

Los motores CEP utilizan distintos lenguajes para la definición de las reglas a controlar. Estos lenguajes requieren de ciertos conocimientos especiales que no todos los usuarios del sistema pueden tener. Por este motivo la solución contempla esta situación. El sistema provee una interfaz amigable (rangos de valores para los distintos parámetros) para que el usuario que dé de alta alertas. Luego, el sistema se encarga de realizar la traducción de la información ingresada por el usuario al lenguaje CEP.

4.2.5. Envío de notificaciones

Para solucionar los requerimientos de las notificaciones a los usuarios, el sistema posee un componente de notificaciones. Este componente se encarga de notificar a los usuarios, cuando se incumple una ley asociada a una alerta. Este componente será una solución ESB.

Al momento que se incumple una ley asociada a una alerta, se envía un mensaje al componente de notificaciones. Este componente recibe dicho mensaje y envía las notificaciones a los usuarios que correspondan (son los usuarios suscritos a la alerta). Cada usuario recibirá una notificación por cada una de las vías que este tiene asignadas.

4.3. Interacción entre componentes

En esta sección se muestra la interacción entre los componentes del sistema previamente mencionados. En la Figura 20 se presenta el flujo entre los componentes mostrando un pequeño ejemplo de cómo funciona el sistema.

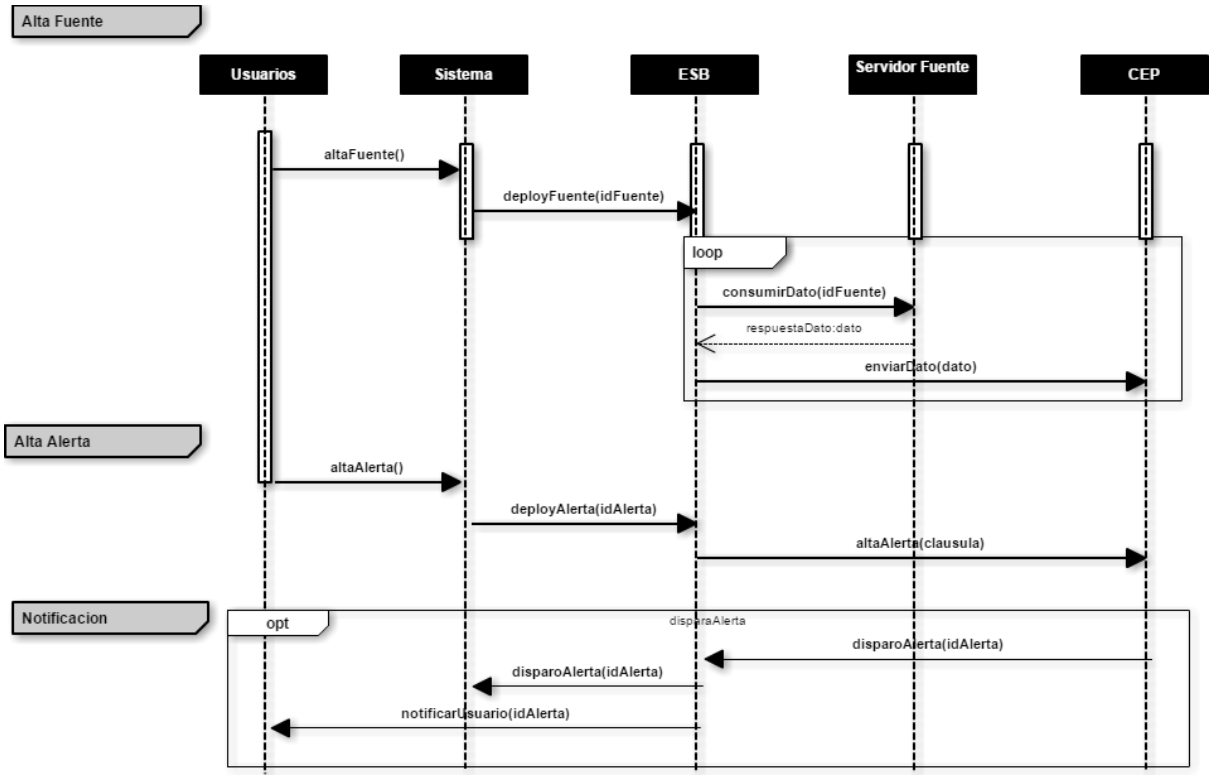


Figura 20 - Diagrama de secuencia.

En una primera instancia, un usuario da de alta una fuente en el sistema. En ese momento el sistema genera una nueva copia de la solución ESB del tipo de fuente seleccionado y ésta es desplegada en el ESB para consumir los datos. Cada cierto tiempo, la aplicación consulta con el servidor de la fuente para obtener los nuevos datos generados por la misma y los envía al motor CEP para su procesamiento.

Por otro lado, otro usuario da de alta en el sistema una alerta que utiliza diferentes fuentes, entre ellas la creada anteriormente. Al dar de alta la alerta, queda registrada en el motor CEP la regla generada con las condiciones de la alerta.

Cuando el motor CEP procesa un dato de una fuente y concluye que está incumpliendo alguna de las reglas, le avisa al ESB de la alerta en cuestión y éste se encarga de notificar a los usuarios correspondientes sobre la alerta que se está incumpliendo.

5. Diseño e implementación

En este capítulo se describe la arquitectura de la solución propuesta, las decisiones de diseño y la selección del producto ESB a utilizar. El capítulo finaliza describiendo las tecnologías utilizadas y detalles de la implementación realizada.

5.1. Arquitectura

En esta sección se presenta la arquitectura del sistema que consta de dos grandes componentes: una aplicación web que brinda una interfaz de administración y otro componente que se encarga de la recepción de los datos generados por las fuentes, control de las alertas y la notificación a los usuarios. En esta sección también se describen los patrones utilizados y sus respectivos diseños. En la Figura 21, se aprecia cómo queda definida la arquitectura del sistema.

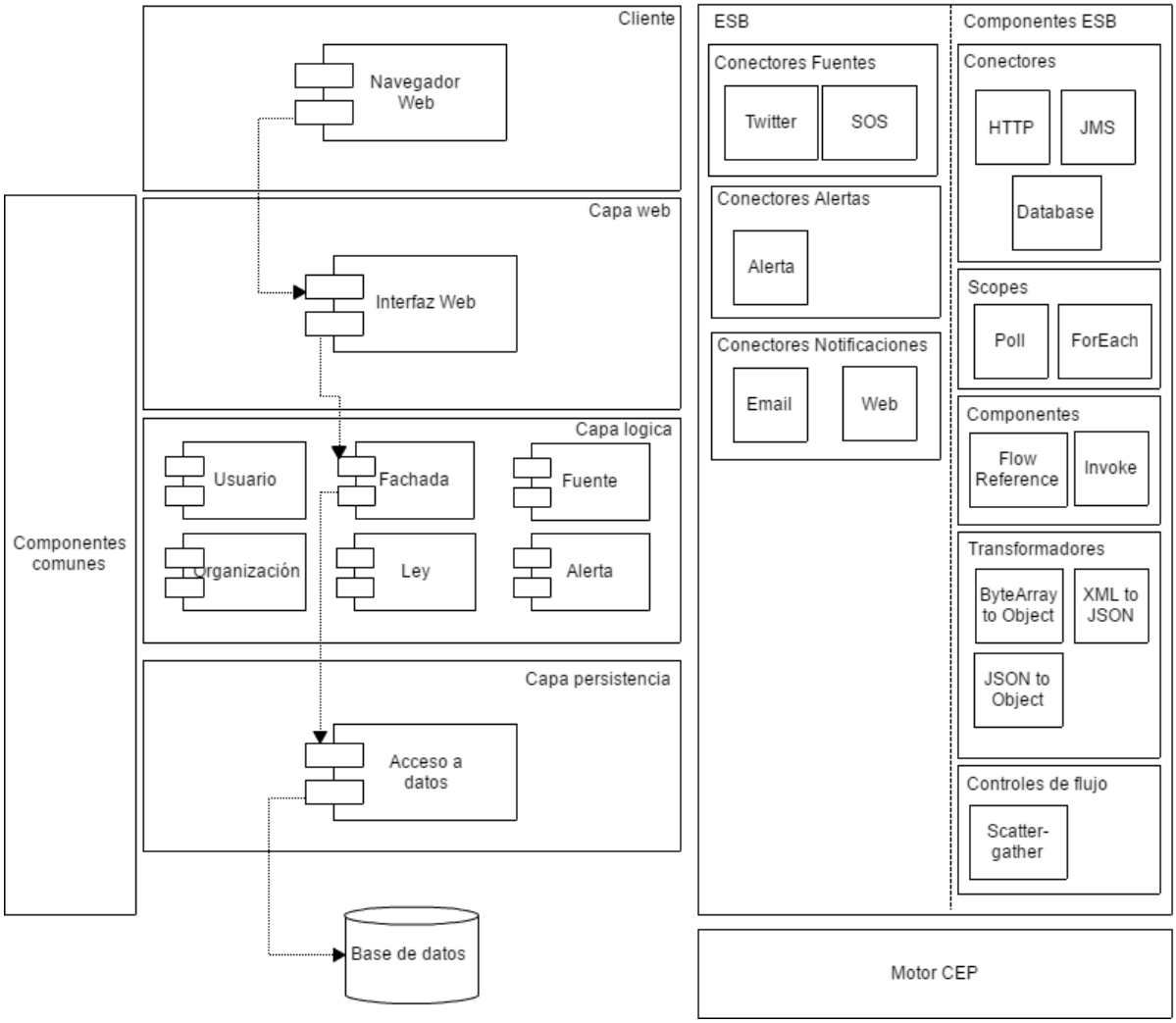


Figura 21 - Arquitectura del sistema.

Para la construcción de la solución se utilizaron principalmente dos patrones de arquitectura [42]:

- Arquitectura en capas: Se utiliza para separar la lógica del sistema con la presentación del mismo. Se divide el sistema en la capa web, capa lógica y capa de persistencia.
- Arquitectura orientada a eventos: Se utiliza en el sistema para manejar la recepción de datos de fuentes y su posterior flujo.

5.1.1. Arquitectura del sistema de administración

La arquitectura del sistema de administración es una arquitectura en capas. Se divide al sistema en las siguientes capas:

- Capa web: En esta capa se encuentran los componentes web que se utilizan en la interfaz de usuario (p. ej. JSP, Servlet, Javascript).
- Capa lógica: Esta capa posee los componentes de negocio. El componente principal es la fachada, debido a que todas las peticiones que se realizan en la web interactúan con este componente. Luego se encarga de derivar las peticiones al componente correspondiente. De forma de simplificar la Figura 21, estas últimas peticiones no son incluidas en el diagrama
- Capa persistencia: Esta capa se utiliza para acceder y almacenar datos en la base de datos.

5.1.2. Arquitectura general de SIMONA

SIMONA está basada en una arquitectura orientada a eventos (Event Driven Architecture, EDA). Esta arquitectura define como evento, al dato obtenido de una fuente. La estructura del evento está compuesta por el dato de la medición, el parámetro que mide, la fecha y su ubicación geográfica. En la Figura 22, se presenta un ejemplo de las capas lógicas del flujo del evento.

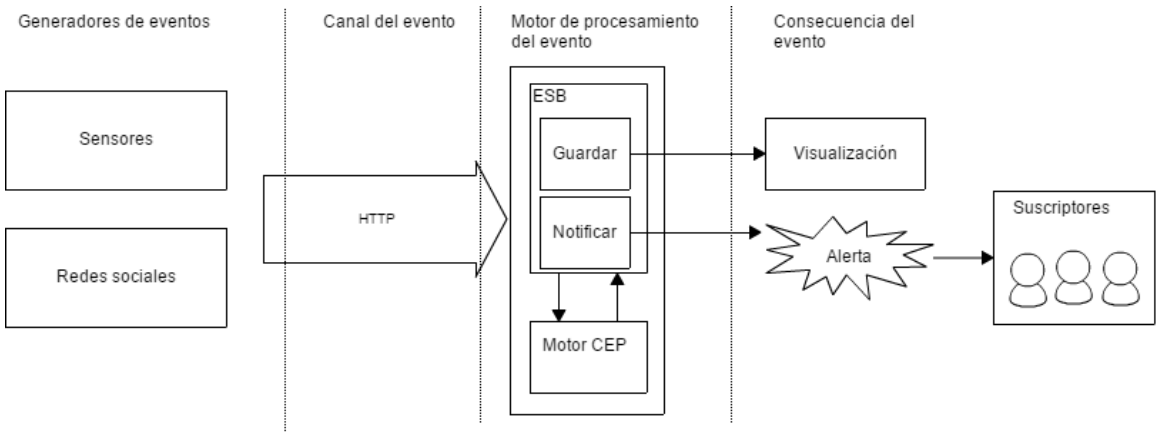


Figura 22 - Flujo del evento.

Las capas lógicas del flujo del evento son:

- **Generador de evento**: Los generadores de eventos son las fuentes, que pueden ser tanto sensores como redes sociales. El evento es generado cada vez que se obtiene una medición realizada por la fuente.
- **Canal de evento:** El canal por el cual se transmite el evento es HTTP.
- **Motor de procesamiento del evento:** El evento es recibido por el ESB y es enviado al motor CEP para su procesamiento. Luego de procesado, se almacenan las mediciones obtenidas y en caso de que la medición incumpla alguna regla de la alerta, se procede a la notificación de los usuarios correspondientes.
- **Consecuencia del evento:** La consecuencia del evento es la visualización de la medición en la web y la posible notificación a los usuarios.

5.2. Decisiones de diseño

En el desarrollo de la solución nos enfrentamos con algunos problemas. En esta sección se muestran las decisiones tomadas con sus respectivas justificaciones.

- Una de las interrogantes planteadas durante el desarrollo de la solución fue la manera de obtener los datos de las fuentes. En una primera etapa la idea era recorrer todas las fuentes ingresadas en el sistema cada un cierto periodo de tiempo, e ir realizando los correspondientes pedidos de datos. El problema de esta solución es la escalabilidad, es decir, si se tienen muchas fuentes llevaría demasiado tiempo recorrer todas las fuentes realizando los pedidos correspondientes. Además, no todas las fuentes generan nuevas mediciones con la misma frecuencia, por lo tanto, se estarían realizando muchos pedidos donde no se recibirán nuevos datos.

  La solución a este problema fue realizar el despliegue de una solución ESB por cada fuente. De esta manera las fuentes son completamente independientes entre ellas. Además, se agregó la frecuencia con la que una fuente genera nuevas mediciones. Por lo tanto, cada fuente realizará la obtención de nuevos datos, de acuerdo a la frecuencia definida.

- Otra de las decisiones tomadas fue en cuanto a la utilización del motor CEP. En un principio se tenía pensado utilizar el motor CEP como un módulo independiente al ESB. Al analizar el ESB elegido, se encontró que el mismo tenía un conector para utilizar el motor CEP. Se realizaron pruebas para corroborar el correcto funcionamiento del mismo y al ser satisfactorias, se decidió utilizar el conector provisto por el ESB.

- En cuanto a las reglas y patrones a declarar en el motor CEP (en el lenguaje del motor) para el procesamiento de los eventos, se plantearon dos maneras diferentes de

hacerlo. Una opción era brindarle al usuario la posibilidad que complete las reglas que debía cumplir la alerta en el lenguaje del motor, mientras que la otra opción era que el usuario defina la alerta eligiendo las operaciones aritméticas y valores a controlar en una interfaz. Luego el sistema realiza la traducción al lenguaje del motor. La primera opción brinda una amplia gama de posibilidades en cuanto a las reglas y patrones que se pueden realizar en una alerta, pero tiene la desventaja que los usuarios deben conocer el lenguaje del motor. La segunda opción genera reglas y patrones en el lenguaje del motor simples y predeterminados, pero brinda la posibilidad que cualquier usuario pueda dar de alta alertas.

Teniendo en cuenta las dos posibilidades, se decidió optar por la segunda opción ya que los usuarios de la aplicación pueden no tener conocimientos del lenguaje del motor.

- En un comienzo del proyecto, se quería tener una solución ESB por cada fuente registrada en el sistema. En el caso de las fuentes de redes sociales, existe una restricción en cuanto a la cantidad de conexiones simultáneas que se pueden realizar con un mismo usuario. Como no es viable solicitar por cada fuente de tipo red social un usuario y contraseña para consumir los datos, el sistema posee un único usuario para realizar el consumo de los datos. Debido a la restricción mencionada, en este caso solo se tiene una única solución ejecutando en el ESB por cada red social, la cual se encarga de escuchar todas las palabras claves registradas en el sistema. Una desventaja de realizarlo de esta manera, es que al momento de dar de alta una nueva fuente de tipo red social, es necesario parar la solución ESB, modificarla y volverla a ejecutar.

  Otra posible solución, para no detener la ejecución de las fuentes de tipo red social, sería leer las palabras a buscar desde un archivo o base de datos.

5.3. Selección de producto ESB

Para la selección del ESB a utilizar se realizó una breve relevamiento y comparación entre diferentes opciones del mercado, analizando las ventajas y desventajas de las mismas, para seleccionar la mejor opción de acuerdo a las necesidades del proyecto.

Se analizaron las siguientes soluciones ESB: Fuse ESB[16], Talend ESB[17], WSO2 ESB[18], Microsoft Biztalk Server[19], JBoss ESB[20] y Mule ESB[21].

[16] https://developers.redhat.com/products/fuse/overview/
[17] https://www.talend.com/resource/enterprise-service-bus/
[18] http://wso2.com/products/enterprise-service-bus/
[19] https://www.microsoft.com/en-us/cloud-platform/biztalk
[20] http://jbossesb.jboss.org
[21] https://www.mulesoft.com/platform/soa/mule-esb-open-source-esb

Existen diversas maneras de evaluar los ESB considerando las características de los mismos. En la Figura 23 se presenta un posible criterio de evaluación para los ESB. [43]

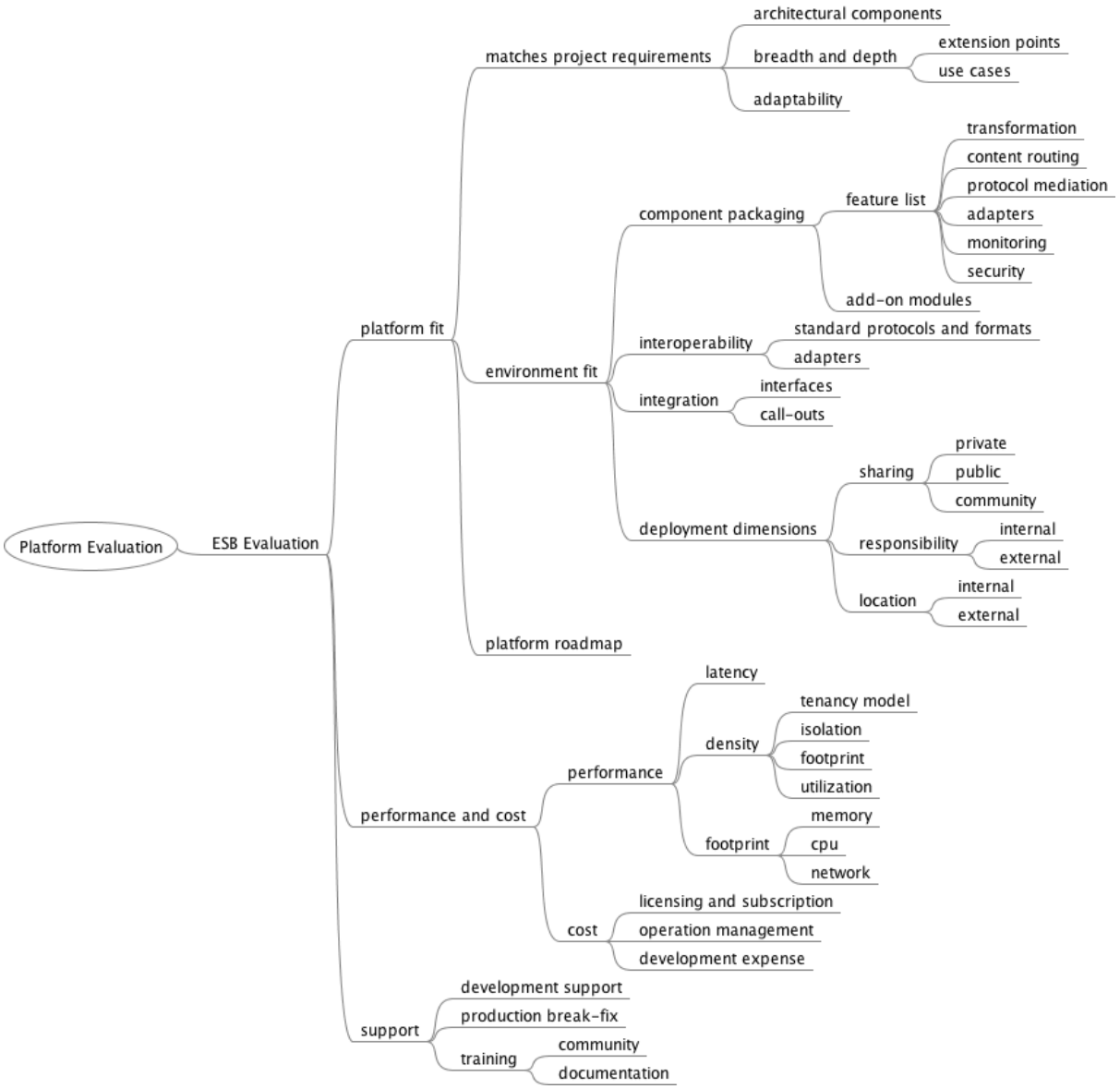


Figura 23 – Criterio de evaluación de ESB [43].

En la elección del ESB a utilizar, se priorizaron los siguientes criterios debido a las necesidades del proyecto:

- Adaptación de la plataforma (Plataform fit):
    - **Integración CEP:** El ESB a elegir debe permitir integrarse con un motor CEP, debido a la solución utiliza procesamiento de eventos complejos.
    - **Conectores:** El ESB debe poseer una gran variedad de conectores hacia otras tecnologías y/o aplicaciones (p. ej. redes sociales)

    - **Usabilidad:** El ESB a elegir, debe ser sencillo e intuitivo de utilizar debido al tiempo del proyecto. Además, debe permitir agregar nuevas aplicaciones, sin detener a las otras.

- Performance y costos (Performance and cost):
    - **Performance:** El ESB debe poder trabajar con grandes volúmenes de datos, sin sufrir una degradación considerable.
    - **Licencia:** El ESB a utilizar debe ser código abierto debido al contexto del proyecto.
- Soporte (Support)**:**
    - **Comunidad:** Se busca que el software a utilizar cuente con una extensa comunidad activa y buena documentación para facilitar el desarrollo y la resolución de problemas.

A continuación, se describen brevemente algunas soluciones ESB:

Fuse ESB: Es una plataforma para integrar aplicaciones, datos, servicios de manera local o en la nube. Utiliza tecnologías de código abierto para brindar los servicios de enrutamiento y transformación de protocolos. También posee JBoss A-MQ que es una plataforma de mensajería fiable de gran rendimiento. Fuse utiliza Apache Camel para proporcionar la conectividad con aplicaciones externas, y además incluye más de 150 conectores (p. ej. Twitter, Facebook).

Talend ESB: Es parte de la suite de Talend y puede ser utilizado independientemente de los otros componentes de la plataforma. Talend ESB es de código abierto y gratuito. Además, se adhiere a protocolos abiertos, para garantizar la mayor interoperabilidad posible. La suite está construida en base a Eclipse, por dicho motivo es intuitiva de usar para quienes utilizan este IDE. Posee una arquitectura modular y distribuida. También ofrece ruteo y mediación de mensajes basado en reglas y en los patrones de integración mencionados en la Sección 2.5.2. Se puede construir rápido y eficazmente Web Services SOAP sobre HTTP y XML sobre JMS, entre otros. Posee más de 500 conectores incluidos por defecto, sin embargo, no posee conectores para las redes sociales más populares (p. ej. Facebook, Twitter). [44], [45]

WSO2 ESB: Es parte de la plataforma de integración WSO2. WSO2 ESB es de código abierto. Posee una arquitectura distribuida y modular, permitiendo que los componentes estén desacoplados y en distinta ubicación. Se puede utilizar la nube de WSO2 para alojar los servicios. Tiene más de 150 conectores incluidos por defecto, entre ellos se encuentran los conectores de las redes sociales Twitter, Facebook e Instagram. [46]

Microsoft Biztalk Server: El servidor provee Biztalk ESB Toolkit. Biztalk ESB es una colección de herramientas y librerías para generar una arquitectura de mensajería bajamente acoplada. Biztalk funciona como middleware donde provee herramientas para la mediación entre los servicios y sus consumidores. Provee ruteo de mensajería, validación y

transformación de mensajes y permite la integración con Web Services. Posee conectores con redes sociales. [47], [48]

JBoss ESB: Utiliza una arquitectura basada en los principios SOA, es bajamente acoplada y utiliza mensajería asincrónica. Algunas de las funcionalidades que provee Jboss ESB son: integración con JMS, SQL y Web Services, transformaciones de mensajes y ruteo basado en contenidos. Además, posee un editor gráfico que permite mejorar la usabilidad. [49], [50]

Mule ESB: Está basado en Java, es de fácil instalación y configuración, ligero y escalable, permitiendo conectarse con nuevas aplicaciones a lo largo del tiempo. Esta plataforma ofrece dos versiones: Community y Enterprise. La versión Community es de código abierto. Posee una gran variedad de conectores, entre ellos conectores con motores CEP y con redes sociales. Posee una interfaz gráfica que facilita la usabilidad. [51]

Teniendo en cuenta las principales necesidades relevadas, se concluye que la solución que mayor se adapta es Mule ESB. Esto se debe a que es de código abierto, posee conectores para las redes sociales más populares y para motores CEP.

5.4. Tecnologías utilizadas

En esta sección se detallan las tecnologías utilizadas, así como un breve análisis de las mismas. En la Figura 24, se presenta un diagrama con las tecnologías utilizadas en la implementación de la solución.

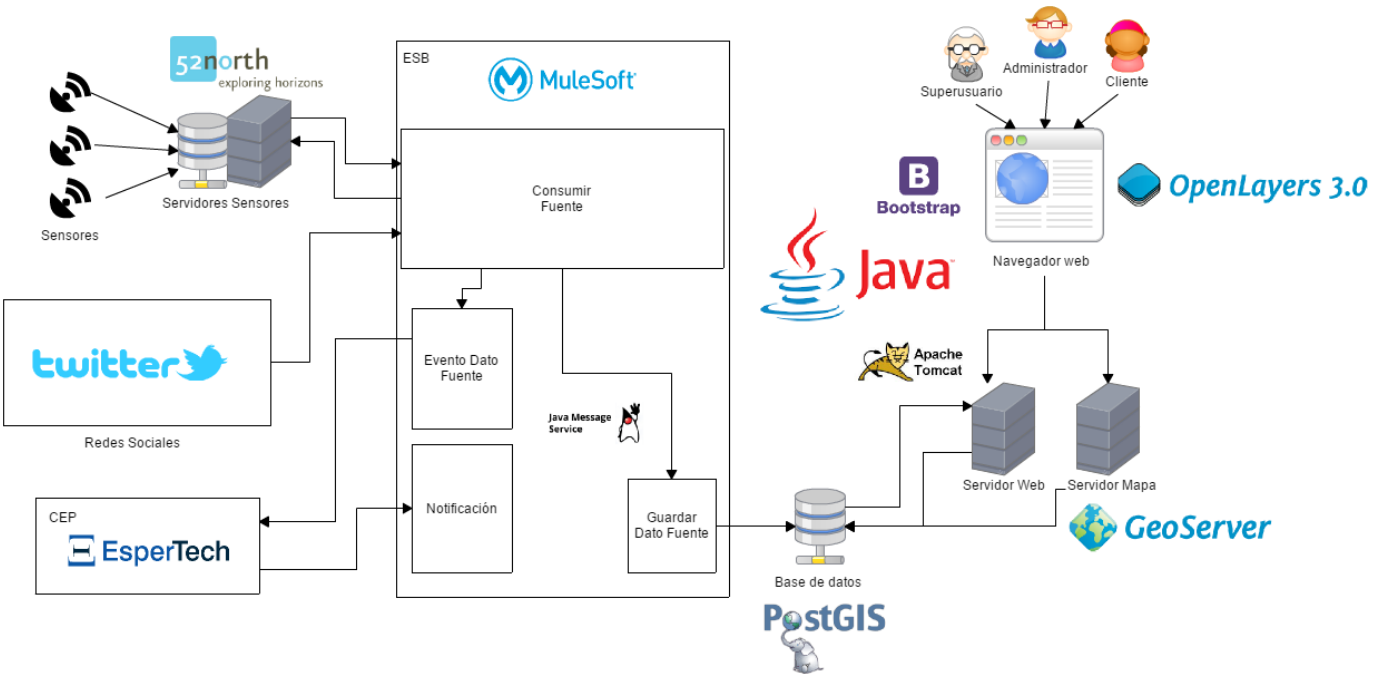


Figura 24 – Diagrama de tecnologías utilizadas.

5.4.1. Mule ESB

Mule ESB tiene muchas funcionalidades, entra las que se destacan [52]:

- Creación y alojamiento de servicios: publican y reciben servicios reutilizables.
- Mediación de servicios: encapsulamiento de servicios, separando la lógica de negocio de la lógica de mensajería.
- Enrutamiento de mensajes: ruteo, filtrado, agregado y reordenamiento de mensajes basados en el contenido y reglas.
- Transformación de datos: intercambio datos a través de distintos protocolos.

Además, Mule ESB maneja todas las interacciones entre las aplicaciones y componentes de manera transparente, independientemente de si están en la misma máquina o a través de Internet. [52]

En la Figura 25, se presenta un ejemplo de una posible utilización de Mule ESB.

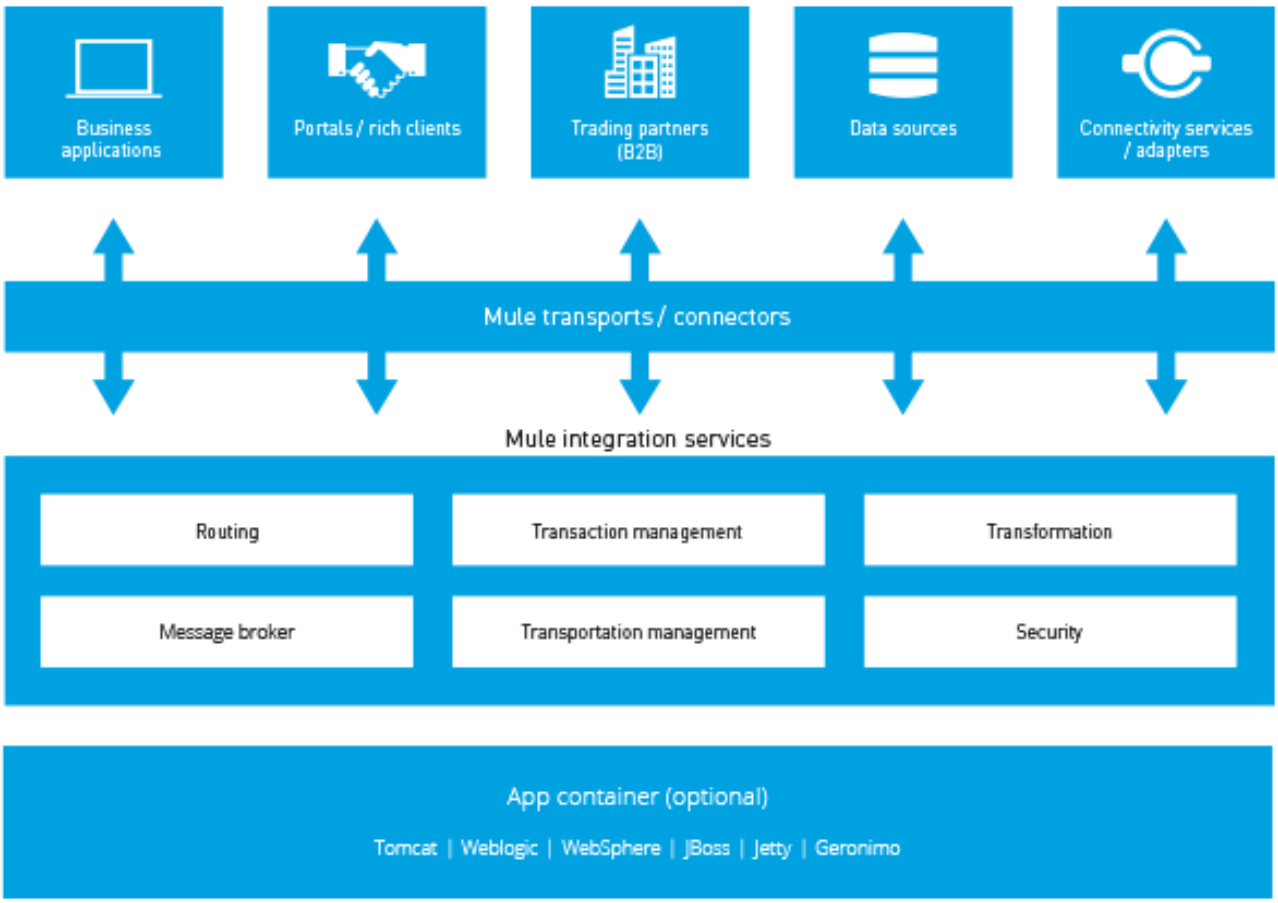


Figura 25 - Ejemplo de Mule ESB [52].

Mule posee varias ventajas con respecto a otros ESB [52]:

- Los componentes pueden ser de cualquier tipo.
- Fácil reutilización de componentes sin necesidad de realizar ningún cambio ya que no se necesita utilizar código específico para funcionar.
- Los mensajes pueden estar en cualquier formato, desde SOAP hasta archivos binarios.
- Se puede implementar variedades de topologías, no solo ESB, debido a que es ligero y embebido. Proporciona aplicaciones seguras y escalables, adaptables al cambio.

5.4.2. Twitter APIs

Twitter posee varias APIs [53], las cuales poseen una buena documentación, así como una gran variedad de ejemplos.

Una funcionalidad que distingue a Twitter es su Streaming API [54]. Esta herramienta brinda a los desarrolladores una baja latencia para acceder a los tweets publicados. La API permite que una vez configurado el streaming, se reciban nuevos tweets que cumplan con alguno de los filtros previamente definidos, a través de una conexión HTTP persistente. Para la implementación de esta herramienta, el desarrollador debe obtener una clave. Esta clave solo puede ser utilizada por una aplicación.

Los tweets obtenidos desde Streaming API están en formato JSON. Uno de los campos que puede tener el tweet es la ubicación geográfica. En el JSON se muestran tres atributos en referencia a la ubicación:

- Coordinates: Representa las coordenadas del punto en donde se realizó el tweet a través de un GeoJSON[22].
- Geo: Deprecado
- Place [55]: Indica la zona geográfica en donde se realizó el tweet.

Para que los tweets contengan la ubicación en que fue realizado, el usuario deberá tener configurado el envío de su ubicación. [56]

A su vez Twitter posee una REST API [53] donde se puede invocar operaciones geográficas como, por ejemplo:

- GET geo/search: Dada una latitud y longitud, IP o nombre devuelve el id de un lugar conocido.

[22] Es un formato para representar geografías con sus atributos. http://geojson.org/

- GET geo/reverse_geocode: Dada una latitud y longitud devuelve veinte lugares cercanos.
- GET geo/id/:place_id: Dado un id de lugar, devuelve el nombre y coordenadas de ese lugar.

También se evaluó utilizar la API de Facebook [57], pero no fue posible debido a que no permite realizar streaming. Hasta hace unos años, Facebook incluía en su API, operaciones para realizar streaming en las publicaciones que realizaban sus usuarios. En la actualidad, debido a los cambios en las políticas de privacidad hacia los usuarios, Facebook decidió quitar esas operaciones de su API.

5.4.3. Esper CEP

Esper [58] es una herramienta de código abierto para el procesamiento de eventos complejos. Permite detectar determinados eventos y ejecutar las acciones correspondientes cuando dicho evento ocurre. Además, Esper permite manejar grandes volúmenes de datos.

Esper utiliza el lenguaje Event Processing Language (EPL) para poder aplicar filtros a los eventos. Además, este lenguaje es extensible permitiendo implementar nuevas funciones. Dicho lenguaje presenta muchas similitudes con SQL.

5.4.4. Java Message Service

Java Message Service (JMS) [59] es una API de Java que permite a las aplicaciones crear, enviar, recibir y leer mensajes. Esta API permite el envío de mensajes asíncronos y confiables, es decir no llegan paquetes repetidos y si se envía el mensaje se asegura que llegue.

5.4.5. Geoserver

Geoserver [60] es un servidor de mapas basado en Java que permite a los usuarios compartir y editar datos geográficos, utilizando los estándares de OGC.

Algunas de las características del Geoserver es la compatibilidad con los estándares WMS y WFS y su conectividad con una base de datos PostGIS.

5.4.6. OpenLayers

OpenLayers [61] es una librería en JavaScript que permite visualizar un mapa en una página web. Desde noviembre de 2007 OpenLayers forma parte de los proyectos de OGC.

Esta librería permite cargar capas al mapa mediante los estándares WMS, WFS y también permite crear, modificar y eliminar geometrías mediante WFS Transaction, mencionado en la Sección 2.4.2. entre otros.

5.4.7. 52° North's Sensor Web

52° North's Sensor Web es una comunidad de 52º North [62], que se enfoca en el desarrollo de servicios para lograr obtener en tiempo real los datos observados por sensores, utilizando el conjunto de estándares, de OGC, SWE mencionado en la Sección 2.3.1.

52° North's Sensor Web posee también una aplicación web en la cual es posible ingresar y obtener mediciones de sensores utilizando el estándar SOS.

5.5. Implementación

En esta sección se presentan detalles de la implementación de los módulos y las funcionalidades más relevantes al proyecto.

5.5.1. Tipos de fuentes predefinidos

El sistema tiene dos tipos de fuente predefinidos: Sensores con protocolo SOS y Twitter, ambos mencionados en el Capítulo 2.

5.5.1.1. Sensores con protocolo SOS

El sistema posee predefinido el tipo de fuente correspondiente al protocolo de sensores SOS. Esto implica que se encuentra implementada una solución ESB encargada de consumir los datos de un sensor con protocolo SOS en tiempo real. La solución se comunica con el servidor SOS del sensor y solicita las mediciones que este generó en un determinado período de tiempo. Luego de obtener las mediciones las registra en la base de datos y las envía al motor CEP para su procesamiento.

5.5.1.2. Twitter

El sistema también posee predefinido el tipo de fuente Twitter, que corresponde a escuchar un conjunto de palabras claves que se mencionan en los tweets. La solución ESB encargada de consumir los datos en tiempo real, escucha las palabras claves y realiza un filtro de los tweets obtenidos (se descartan aquellos que no poseen ubicación geográfica). Finalmente, al igual que en SOS, se almacena un registro en la base de datos y se envía al motor CEP para su procesamiento.

5.5.2. Estructura de evento

Un evento es generado a partir del dato recibido de la medición de una fuente. Está representado mediante una clase Java. Se definen dos clases de evento de acuerdo al tipo de fuente, una para sensores y otra para redes sociales.

La estructura del evento de los sensores (denominado “EventoSensor”) posee los siguientes atributos:

- id: Este atributo sirve para identificar la fuente de donde proviene el dato.
- valor: Es el resultado de la medición realizada.
- nombreParametro: Indica el nombre del parámetro medido.
- elementoFuente: Es el elemento que mide la fuente. Pueden ser agua, tierra, aire, etc.
- idTipoParametro: Es el identificador que utiliza el sistema para reconocer el tipo de parámetro.
- fechaMedicion: Es la fecha en que se realizó la medición.
- geom: Indica las coordenadas donde se encuentra ubicada la fuente.

Mientras que la estructura del evento generado por redes sociales (denominado “EventoRedSocial”) posee los siguientes atributos:

- idFuente: Este atributo sirve para identificar la fuente de donde proviene el dato.
- mensaje: Es el mensaje realizado por el usuario dentro de la red social.
- geom: Indica las coordenadas desde donde se generó el mensaje.
- fechaMedicion: Es la fecha en que se realizó el mensaje.
- usuario: Es el nombre del usuario en la red social que realizó el mensaje.

Por cada dato medido por una fuente, se generará un evento independiente en el sistema. Las fuentes de tipo sensor generan eventos “EventoSensor” y las de tipo red social generan eventos “EventoRedSocial”.

5.5.3. Creación de alertas en motor CEP

Para dar de alta alertas en el sistema, el usuario completa la información necesaria de la misma, en particular, para cada parámetro, se ingresa el rango de valores de los parámetros de la alerta. A su vez, en el caso de alertas por distancia o por redes sociales, se selecciona la zona geográfica en donde tendrá validez la alerta.

Con los datos ingresados, se crea una regla en el motor CEP, que será utilizada cuando se procesen los datos obtenidos de las fuentes.

5.5.3.1. Extensión geográfica CEP

Para lograr identificar si una fuente cumple con los requisitos geográficos especificados en la alerta, se siguieron los lineamientos del artículo mencionado en la Sección 2.5.3.4. que permitió extender el metamodelo CEP para soportar eventos geográficos. Con dicha

extensión geográfica fue posible definir en los eventos, el atributo geográfico “geom”, correspondiente a la ubicación de la fuente.

Por otro lado, se definió una función geográfica “interseccionMultiPoligono” que recibe como parámetro dos geografías, una con la ubicación donde se generó el evento y otra con el multipoligono donde se definió la alerta. Esta función indica si la ubicación en la que se generó el evento intersecta la geometría de la alerta.

Para la implementación de esta función se utilizó la API JTS Topology Suite[23] que provee funciones espaciales para el procesamiento de geometrías. Además, cumple el estándar SFA para SQL publicado por OGC.

5.5.3.2. Traducción de reglas a EPL

Debido a que el motor Esper CEP utiliza EPL como lenguaje para trabajar los eventos y reglas, es necesario realizar una traducción de la alerta, definida por el usuario, a EPL. Esta traducción tiene en cuenta los valores de cada parámetro y en caso de que la alerta tenga asociada una geografía, se añadirá la función geográfica implementada.

A continuación, se brinda un ejemplo de una traducción realizada por el sistema:

*select * from EventoSensor where (interseccionMultiPoligono(geom, 'MULTIPOLYGON(((-6360477.99766609 -4004688.78591695,-6403282.73350579 -4173461.74437062,-6134224.39394197 -4211374.51040007,-6360477.99766609 -4004688.78591695)))')) and ((3=idTipoParametro and 1=elementoFuente and valor > 88))*

Esta consulta se ejecuta cada vez que se recibe un evento de tipo “EventoSensor”, en donde se comparan los campos del evento con los de la alerta. En este ejemplo se aprecia la utilización de la función “*interseccionMultiPoligono”* la cual verifica que la ubicación en la que se generó el evento (“geom”), se intersecte con la geografía de la alerta. Además, se controla que el parámetro del evento, sea alguno de los que controla la alerta y su valor se encuentre dentro del rango definido.

En el caso de las alertas generadas a partir de fuentes de tipo red social, la traducción tiene algunas diferencias ya que no existen parámetros y rangos de valores. A continuación, se muestra un ejemplo de la traducción realizada por el sistema:

*select * from EventoRedSocial (mensaje regexp '.*(?i)azufre.*').win:time(44 sec) where (interseccionMultiPoligono(geom, 'MULTIPOLYGON(((-6344579.09578277 -4037709.58213615,-6397167.77124297 -4164900.79720268,-6121994.46941634 -4171015.7594655,-6145231.32601503 -4093967.23495404,-6344579.09578277 -4037709.58213615)))')) having count(*) >= 22*

En este caso se aprecia que el evento que se recibe es “EventoRedSocial”, en donde se debe cumplir que el mensaje contenga la palabra que se controla en la alerta y que la ubicación

[23] https://live.osgeo.org/es/overview/jts_overview.html

intersecta a la geografía de la alerta. Además posee dos controles más, “*win:time(44 sec)” y* “*having count(*) >= 22”*. Estos se utilizan para indicar la cantidad de mensajes que deben ocurrir en determinado período de tiempo para que se cumpla la regla de la alerta.

5.5.4. Generación y despliegue automático de aplicaciones

En esta sección se detallan los despliegues de las nuevas aplicaciones ESB con su respectivo diagrama donde se indica el funcionamiento.

5.5.4.1. Nuevo sensor SOS

El sistema posee una solución ESB, la cual se encarga de consultar los datos generados por un sensor con protocolo SOS. La misma posee un archivo de propiedades, en el cual se almacenan los datos necesarios para identificar al sensor en el sistema y poder obtener sus mediciones. Esta solución no se encuentra desplegada en el ESB.

Al momento en el que un usuario da de alta un sensor SOS, se realiza una copia de la solución ESB previamente mencionada. Luego, utilizando los datos ingresados por el usuario, se graban las propiedades necesarias para el correcto funcionamiento de la solución. Finalmente se despliega la copia de la solución ESB y a partir de este momento, ésta comienza a consumir los datos de la fuente.

En la Figura 26 se muestra la solución ESB encargada de consumir los datos de un sensor SOS.

El flujo se ejecuta cada un cierto período de tiempo (1), especificado por el usuario, en el cual se realiza una petición HTTP al servidor SOS del sensor (2). En la petición se solicitan las mediciones generadas desde la última petición de datos hasta la fecha y hora actual. La respuesta del servidor se transforma a formato JSON (3), en caso de que no se obtengan nuevas mediciones, no se realiza ninguna acción. En caso contrario, las mediciones obtenidas son procesadas de manera individual. Por cada medición (4), se identifican los siguientes campos: el identificador de la fuente (5), el parámetro medido (6), la fecha en la que se realizó la medición (7) y el valor medido (8). Estos campos, por un lado, son almacenados en la base de datos (9), mientras que por el otro lado son utilizados para generar un evento CEP (10). Este evento es enviado a una cola de mensajes (11), para luego ser procesado en Esper.

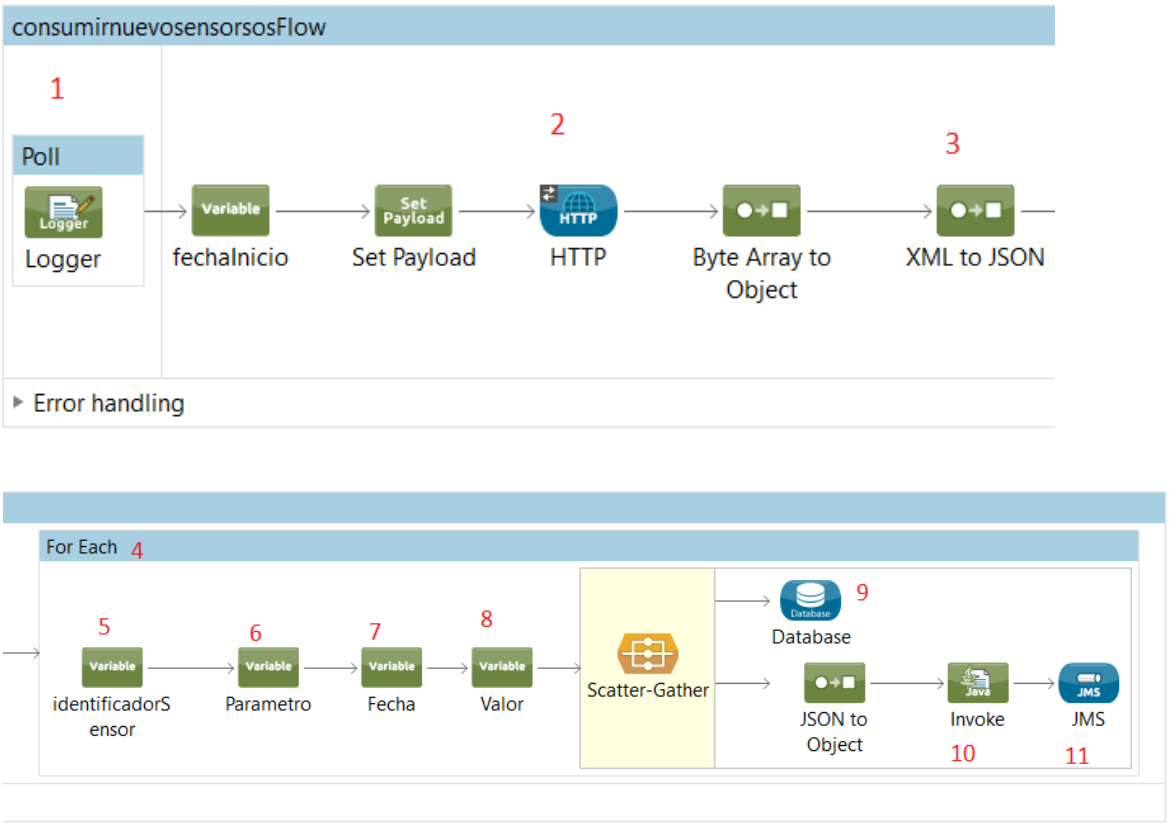


Figura 26 – Solución ESB para consumir sensores SOS.

5.5.4.2. Twitter

El sistema posee una solución para ESB, la cual se encarga de obtener los tweets que realizan los usuarios de Twitter, que contengan una o varias palabras ingresadas en el alta de fuentes. La solución posee un archivo de propiedades, en el cual se almacenan los datos necesarios para obtener los tweets en tiempo real.

A diferencia de las fuentes de tipo sensor SOS, donde se ejecuta una copia de la solución ESB por cada fuente creada, en Twitter solo existe una única solución ejecutándose en el ESB.

Al momento en el que un usuario da de alta una fuente de tipo Twitter con una palabra clave, ésta es agregada al archivo de propiedades de la solución. En este caso, es necesario detener la ejecución de la solución en el ESB y volverla a desplegar, ya que Mule ESB no responde a los cambios en las propiedades sin detener la solución.

En la Figura 27 se muestra la solución encargada de consumir los datos de Twitter.

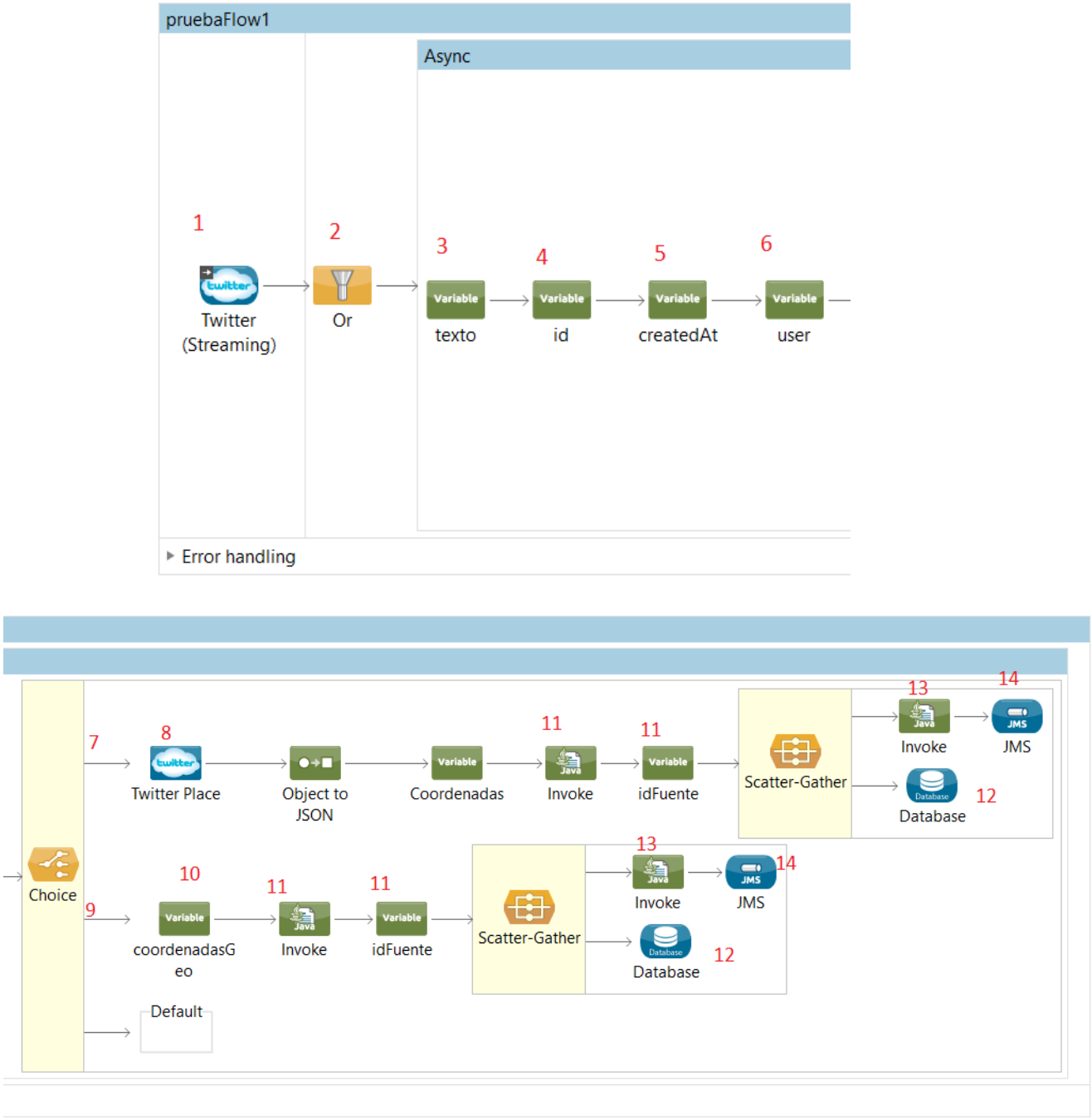


Figura 27 – Solución ESB para consumir Tweets.

El flujo se encuentra continuamente recibiendo los tweets que se generan en Twitter (1), obteniendo únicamente aquellos que contengan alguna de las palabras dadas de alta en las fuentes. Cuando se obtiene algún tweet, se eliminan los que no tengan la ubicación en la que fueron realizados (2). Con los tweets restantes, se obtienen los datos más relevantes, como lo son el texto del tweet (3), el identificador en twitter (4), la fecha en la que fue realizado (5) y el usuario que lo realizó (6). Posteriormente, existen dos posibilidades: que la ubicación sea

el punto exacto en donde fue realizado el tweet, o que contenga el identificador de la zona geográfica en donde se realizó. En el caso que contenga el identificador de la zona geográfica (7), se procede a obtener las coordenadas de la zona realizando una petición a la API de Twitter (8) con el identificador de la zona obtenido. En el caso que se tenga el punto exacto en que fue realizado el tweet (9), se obtiene directamente la ubicación (10). Luego, ambos flujos obtienen el identificador de la fuente (interno del sistema) (11), para finalmente, almacenar los datos en la base de datos (12) y generar un evento complejo (13). Este evento es enviado a una cola de mensajes (14), para luego ser procesado por Esper.

5.5.4.3. Nueva fuente de tipos no predefinidos

Como se mencionó en la Sección 4.2.2, el usuario debe proporcionar una solución ESB, que no se encuentra desplegada, para consumir en tiempo real las mediciones de la fuente de un nuevo tipo. Esta solución envía estas mediciones, mediante un archivo JSON, a una cola de mensajes preestablecida.

Al igual que en sensores SOS, la solución posee un archivo de propiedades, en el cual se almacenan los datos necesarios para identificar a la fuente en el sistema y poder obtener sus mediciones.

Al momento en el que un usuario da de alta una fuente, se realiza una copia de la solución ESB previamente mencionada. Luego, utilizando los datos ingresados por el usuario, se graban las propiedades necesarias para el correcto funcionamiento. Finalmente, se despliega la copia de la solución ESB y ésta permanece consumiendo los datos de la fuente.

Por otro lado, el sistema posee una solución ESB que se encarga de recibir todas las mediciones enviadas a la cola de mensajes por parte de las soluciones mencionadas anteriormente. El flujo correspondiente a dicha solución se presenta en la Figura 28.

El flujo comienza con la recepción del archivo JSON (1), luego se obtienen los valores más significativos de la fuente: identificador de la fuente (2), sistema de referencia (3), las coordenadas (4, 5 y 6), el elemento (7) y parámetros que mide la fuente (8). Por cada medición (9), se obtienen los datos de ella: identificador del parámetro (10), la fecha en que se realizó (11) y el valor de la medición (12). Luego hay dos posibles flujos (13): uno para las fuentes de tipo sensor (14) y otro para las fuentes de tipo red social (15). Ambos flujos, almacenan las mediciones en la base de datos (16) y generan el evento CEP (17). Este evento es enviado a una cola de mensajes (18), para luego ser procesado en Esper.

Figura 28 - Solución ESB para recibir mediciones de fuente de tipos no predefinidos

5.5.4.4. Nueva alerta

El sistema posee la implementación de dos soluciones ESB, no desplegadas, una para las alertas generadas por sensores y otra para las redes sociales. Las mismas poseen un archivo de propiedades respectivamente, en el cual se almacenan los datos necesarios para identificar a la alerta en el sistema y la cláusula a controlar por el motor CEP.

En la Figura 29 se muestra la solución ESB utilizada para las alertas de sensores y en la Figura 30 se presenta la solución ESB para las alertas de redes sociales.

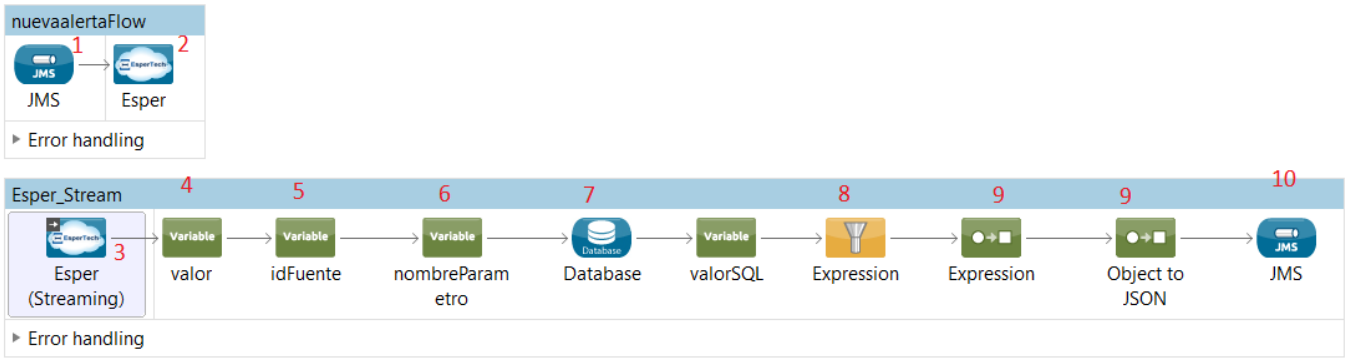


Figura 29 - Solución ESB para alertas de sensores.

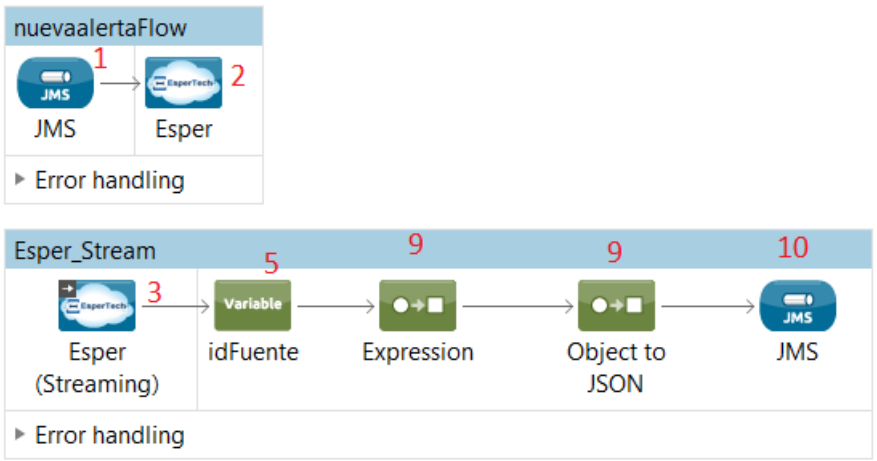


Figura 30 - Solución ESB para alertas de redes sociales.

Ambas soluciones poseen dos flujos. El primero se encarga de recibir, mediante la cola de mensajes (1), los eventos generados a partir de los datos de las fuentes y se envían al motor CEP para su procesamiento (2). Este flujo es igual en ambas soluciones.

El segundo flujo es el encargado de procesar el evento generado. Este flujo recibe los eventos generados y filtra los eventos que no cumplen con las condiciones de la regla EPL. El flujo continúa de manera distinta dependiendo del tipo de alerta:

- En la solución de alerta por sensores, en caso de que cumpla con las reglas definidas en el motor CEP (3), se obtienen los datos más significativos del evento: valor (4), identificador de la fuente (5) y parámetro (6). Luego se realiza un nuevo control (7), para controlar que la fuente que generó el evento pueda ser utilizada por la alerta en cuestión. Este control es necesario debido a que las fuentes pueden ser privadas, con lo cual, los datos generados por ellas, no pueden ser utilizadas por todas las alertas. Dicho control básicamente consiste en obtener el usuario u organización que creó la alerta y verificar que tenga permisos para utilizar la fuente asociada al evento. Si la

fuente cumple con el control realizado (8), continua con el flujo, en caso contrario se descarta.

- En la solución de alerta por redes sociales, en caso de que cumpla con las reglas definidas en el motor CEP (3), se obtiene el identificador de la fuente (5) asociada al evento.

El flujo de ambas soluciones finaliza con la creación de un archivo JSON (9) con los datos de la alerta y los datos del evento. Estos datos son enviados, mediante una cola de mensajes JMS (10), a la solución ESB encargada de las notificaciones.

5.5.4.5. Notificaciones

El sistema posee dos soluciones ESB en ejecución, que se encargan de realizar las notificaciones a los usuarios. En las Figuras 31 y 32 se presentan las soluciones ESB para el módulo de notificaciones, generadas a partir de fuentes de tipo sensor y de redes sociales respectivamente.

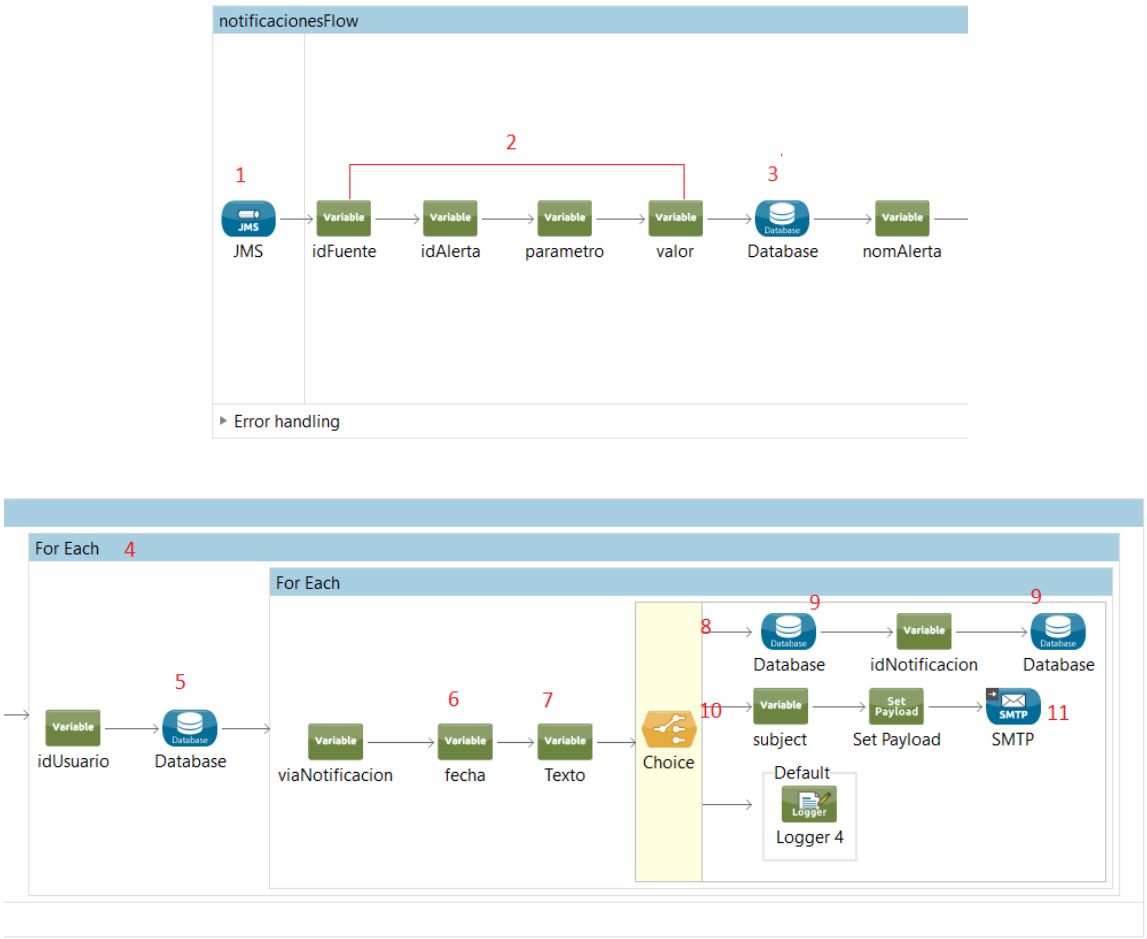


Figura 31 – Solución ESB para las notificaciones de fuentes de tipo sensor.

Ambas soluciones funcionan de manera similar, a diferencia del mensaje de notificación generado. Esto es debido a que la información que se posee sobre la fuente y el dato que generó la notificación, es distinta dependiendo del tipo de fuente.

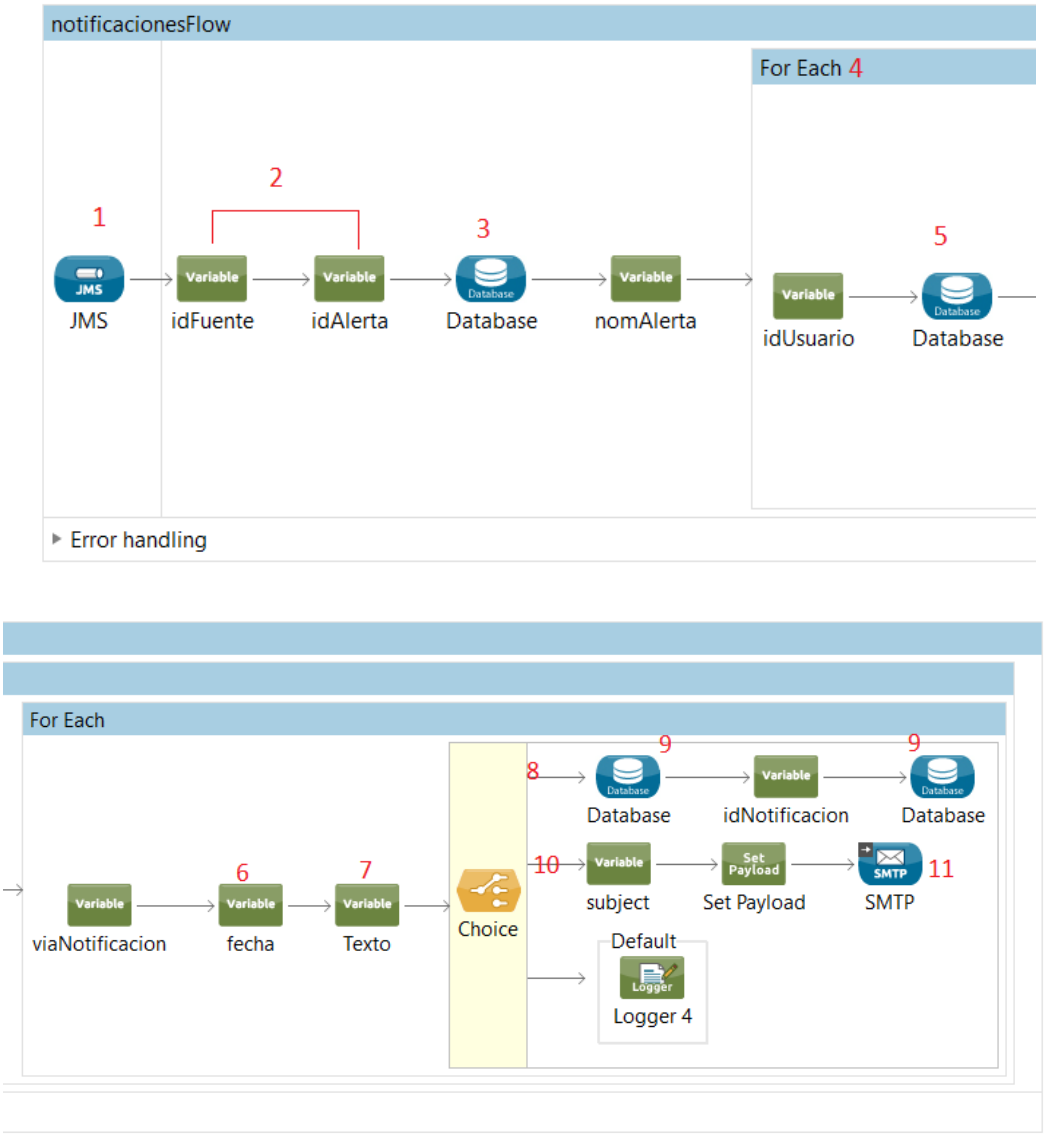


Figura 32 - Solución ESB para las notificaciones de fuentes de redes sociales

En ambos casos, las soluciones reciben los mensajes de la cola JMS (1) (cada aplicación recibe los mensajes de una cola diferente). Luego se obtienen los datos más relevantes (2). El flujo continúa obteniendo desde la base de datos, los usuarios que deben ser notificados (3) (los usuarios que se encuentran suscritos a la alerta). Posteriormente, para cada usuario que deba ser notificado (4), se obtiene desde la base de datos, las vías de notificación que tiene configuradas (5). Además, se obtiene la fecha y hora actual (6) y se genera el texto (7) que se mostrará en la notificación. Si la vía de notificación es web (8), se almacena en la base de datos los datos de la notificación (9) para que la web se encargue de realizar la notificación

correspondiente. Mientras que si la vía de notificación es email (10), se procede a realizar el envío del mail (11).

5.5.5. Limitaciones del prototipo

En esta sección se listan algunas de las limitaciones del prototipo.

- **Alertas de distintos tipos de fuentes**

Una de las limitaciones del prototipo es que no permite la creación de alertas, utilizando simultáneamente fuentes de tipo sensor y fuentes de tipo red social.

- **Unidades**

Otra limitación del prototipo es el manejo y conversión de unidades, debido a que actualmente en el prototipo se trabaja con una única unidad por parámetro. Este módulo se encargaría de realizar las conversiones necesarias para que los datos generados por las fuentes, sean en la misma unidad que lo registrado en las alertas.

- **Seguridad**

En cuanto a la seguridad del prototipo es posible mejorarla, por ejemplo, utilizando https en la web. Además, se podría agregar mayor seguridad a las transacciones geográficas.

6. Caso de estudio

En este capítulo se muestra un caso de estudio con el objetivo de validar funcionalmente el prototipo construido. Se comienza por la descripción de la realidad de instituciones nacionales, para proceder con la ejecución del mismo.

6.1. Descripción de la realidad

El pasado año fue de interés nacional intentar controlar los niveles de contaminación del río Santa Lucía debido a que su estado fue crítico, afectando notoriamente la calidad de agua. [63]

Por este motivo el Ministerio de Industria, Energía y Minería (MIEM) instaló sensores en el río con el fin de controlar los parámetros contaminantes. Por otro lado, el MVOTMA es el encargado de controlar el cumplimiento de las leyes medioambientales vigentes. Además, es de principal interés para Obras Sanitarias del Estado (OSE) conocer el estado del agua y por este motivo, también posee sensores en el río. Cada una de estas organizaciones realiza sus controles de manera independiente.

A su vez, existen ciudadanos que utilizando la red social Twitter donde comunican problemáticas encontradas en el río.

El objetivo de este caso de estudio es lograr que las organizaciones mencionadas anteriormente colaboren entre ellas, para beneficiarse de los distintos datos que cada una obtiene de manera individual. Además, las organizaciones tendrán en cuenta la información generada por los distintos usuarios de las redes sociales.

En la Figura 33 se presenta la descripción de la realidad del caso de estudio.

Figura 33 - Descripción de la realidad en el caso de estudio.

6.2. Simulación de sensores SOS

En un principio, para la ejecución del caso de estudio se deseaba utilizar datos en tiempo real generados por sensores. Se realizaron consultas a distintas organizaciones que poseen sensores medioambientales para poder hacer uso de los mismos. Sin embargo, esto no fue posible debido a distintos motivos.

Se decidió entonces, crear un módulo por fuera de los objetivos del proyecto, para que este se encargue de simular el funcionamiento de sensores con el protocolo SOS.

El módulo consiste en un sistema web, en el cual se pueden dar de alta sensores con sus respectivos parámetros. Luego, el sistema da de alta el sensor en el servidor SOS (utilizando el servidor provisto por 52° North) y a partir de ese momento, se simularán los datos de los parámetros del sensor, siendo enviados al servidor SOS.

Este módulo también permite ingresar la frecuencia con la cual desea que se simulen los datos, así como el rango de valores que se desea que tomen los datos de cada parámetro del sensor.

En la Figura 34 se muestra un sensor dado de alta en el sistema de simulación. Se observa que el sensor de nombre RSL-Sensor001, se encuentra activo con una frecuencia de 30

segundos, es decir que cada 30 segundos se generaran valores aleatorios para los parámetros P y Ph dentro de los rangos establecidos.

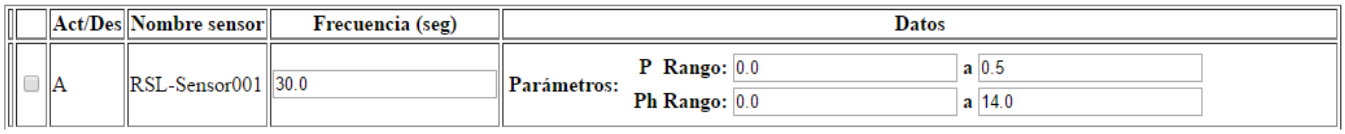


Figura 34 – Sensor simulado.

Luego, en SIMONA se darán de alta los sensores de tipo SOS utilizando la información del sensor que se simuló.

6.3. Ejecución del caso de estudio

Para el caso de estudio se asume que ya fueron dados de alta en el sistema el conjunto de usuarios y organizaciones utilizadas (MVOTMA, MIEM y OSE).

Con el fin de lograr el objetivo propuesto, las distintas organizaciones realizan una serie de procedimientos listados a continuación.

6.3.1. Alta nuevo tipo de fuente

Comenzamos el caso de estudio dando de alta un nuevo tipo de fuente. Esto es debido a que el MIEM posee sensores que utilizan un protocolo que actualmente no es soportado por el sistema al que denominaremos “Protocolo X”. Por este motivo, el MIEM implementó una solución ESB para Mule, la cual consume en tiempo real los datos de los sensores que utilizan el Protocolo X. La implementación de la misma, fue realizada cumpliendo los requisitos que impone el sistema, los cuales se detallan en el Apéndice 2.

En la Figura 35 se muestra el alta del Protocolo X en SIMONA. Aquí se aprecia que el MIEM proporciona la solución que implementó para consumir los datos en tiempo real y además indica los campos que deben ser completados por los usuarios para poder consumir los datos del sensor. En este caso, se necesita que los usuarios indiquen el identificador del sensor, así como también la información (URL, puerto, path) de donde se deben consumir los datos.

A partir de este momento, el sistema incluye el Protocolo X dentro de los protocolos de fuentes existentes para el consumo en tiempo real.

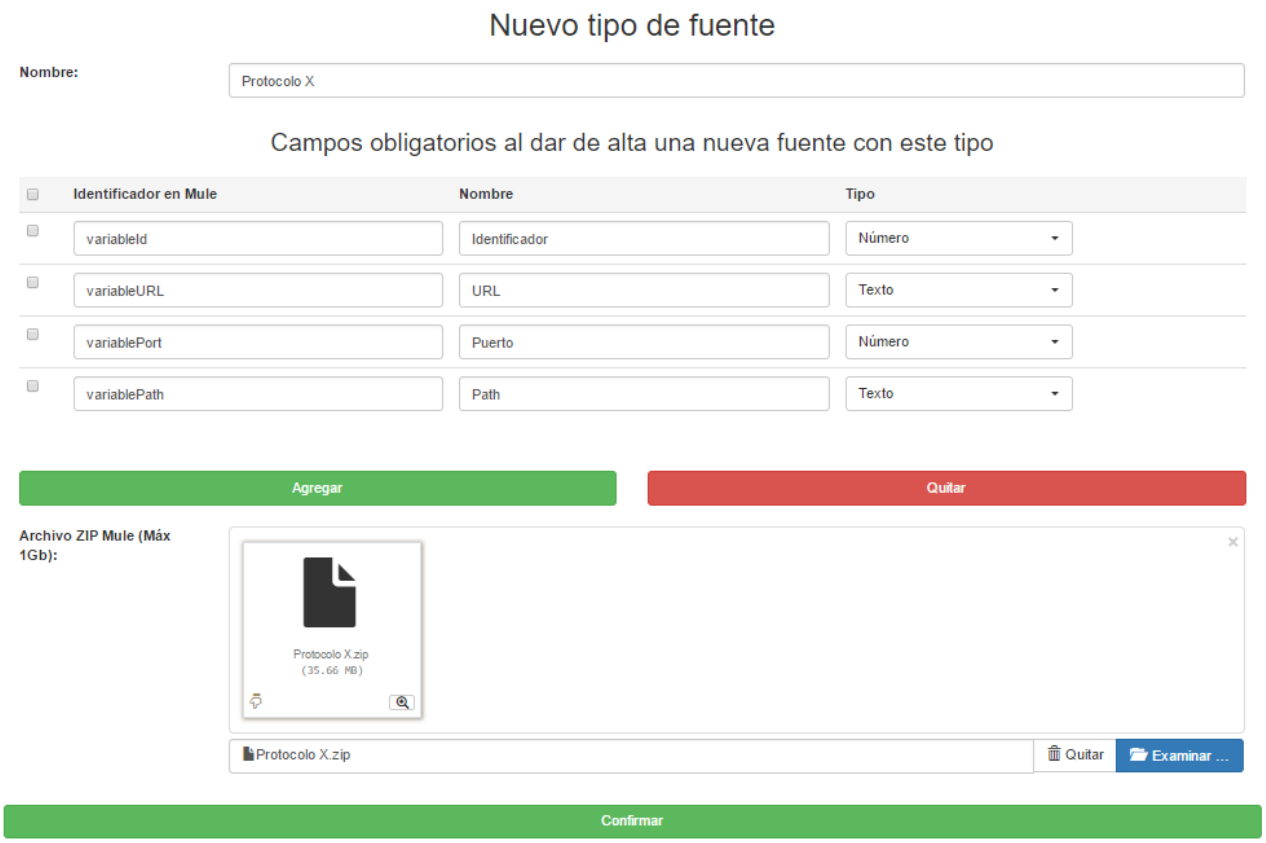


Figura 35 – Alta nuevo tipo de fuente.

6.3.2. Alta fuente

En segunda instancia, las organizaciones proceden a dar de alta sus sensores en la plataforma.

En la Tabla 2 se detallan las fuentes que son dadas de altas por cada organización, con sus respectivos protocolos de comunicación y el conjunto de parámetros que miden.

Todas estas fuentes son compartidas entre las tres organizaciones (MVOTMA, MIEM y OSE) de forma que cualquiera de ellas puede utilizar los datos que estas generan para sus respectivas alertas.

Organización	Nombre fuente	Protocolo	Parámetros
MIEM	MIEM - Sensor SL 1	X	pH
			Cr
			Zn
	MIEM - Sensor SL 2	X	pH
			CN−
			Pb
	MIEM - Sensor SL 3	X	Cr
			pH
	MIEM - Sensor SL 4	X	Cr
OSE	OSE - Sensor SL 1	SOS	pH
			Cr
	OSE - Sensor SL 2	SOS	Ph
			P
	OSE - Sensor SL 3	SOS	Cr
			Hg
	OSE - Sensor SL 4	SOS	Ph
			Cr
MVOTMA	TWEET - Contaminación	Twitter	Contaminación

Tabla 2 - Fuentes creadas.

En la Figura 36 se presenta el ejemplo de una de las fuentes del MIEM dada de alta en el sistema. Entre los datos que el sistema le solicita al usuario, se encuentran los campos que se indicaron al dar de alta el nuevo tipo de fuente “Protocolo X” (Identificador, URL, Puerto y Path).

Al dar de alta la fuente, se realiza una copia de la solución ESB y se realiza el deploy de la misma en Mule, y ésta queda consumiendo los datos de la fuente en tiempo real. En la Figura 37 se presenta el archivo de propiedades de la solución, en el cual se aprecian las variables que utiliza el proyecto junto con los valores que ingresó el usuario.

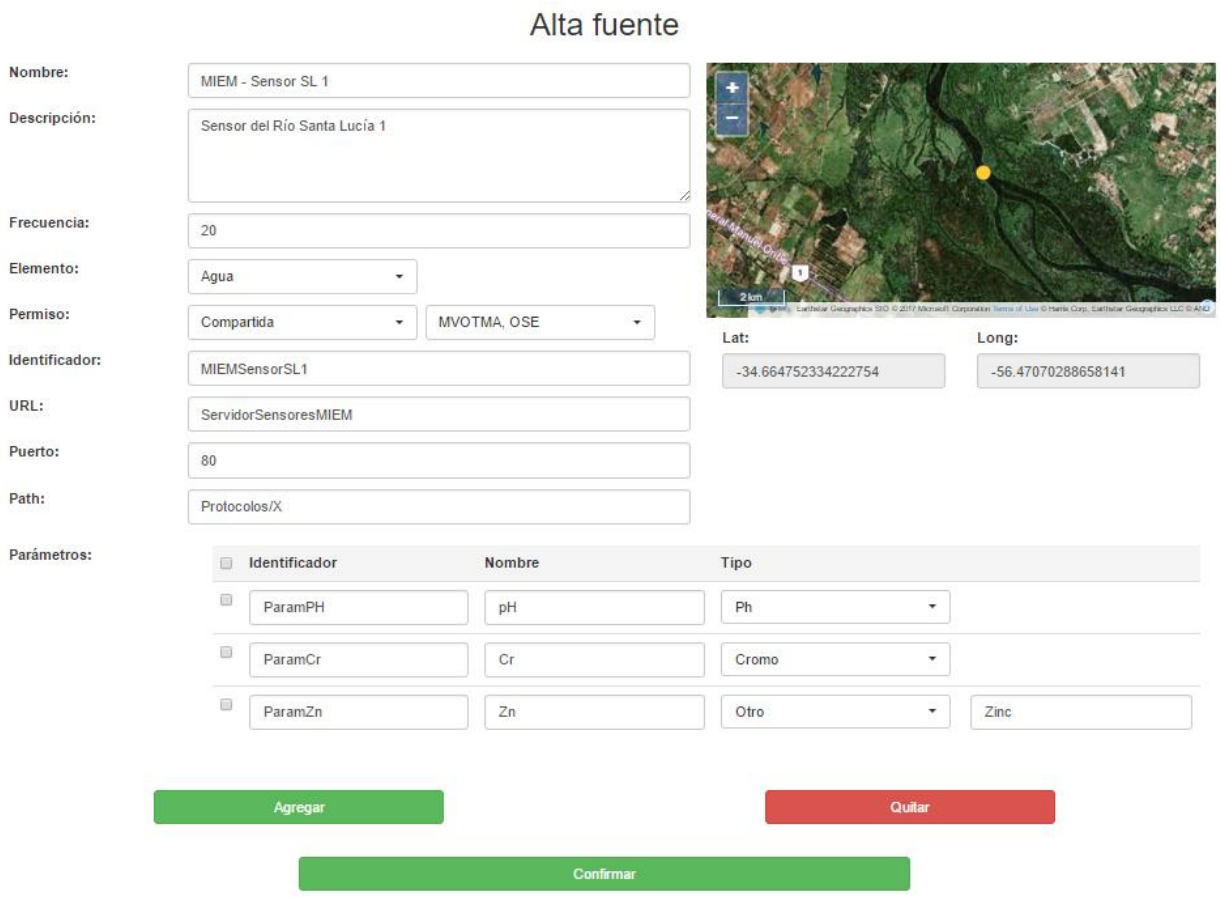


Figura 36 – Nueva fuente con protocolo X.

```
fuente.id  = 57
fuente.frecuencia= 20
fuente.cola= datosCola
fuente.activeMQ= tcp://localhost:61616
fuente.sistemaReferencia= 4326
posicion.x= -6344273.347669629
posicion.y= -4144721.4217353966
posicion.z= 0
fuente.elementoFuente= 1
fuente.idTipoParametro= ParamPH:1;ParamCr:14;ParamZn:15;
variableId= MIEMSensorSL1
variableURL= ServidorSensoresMIEM
variablePort= 80
variablePath= Protocolos/X
```

Figura 37 – Archivo de propiedades de la nueva fuente con protocolo X.

Por otro lado, es de interés para MVOTMA utilizar la red social Twitter como fuente de información. En particular le interesa conocer todos los tweets donde se mencione la palabra “contaminación”. Para esto se da de alta una fuente de tipo Twitter. En las Figuras 38 y 39 se presenta el alta de la fuente de la red social Twitter y el archivo de propiedades de la solución ESB respectivamente.

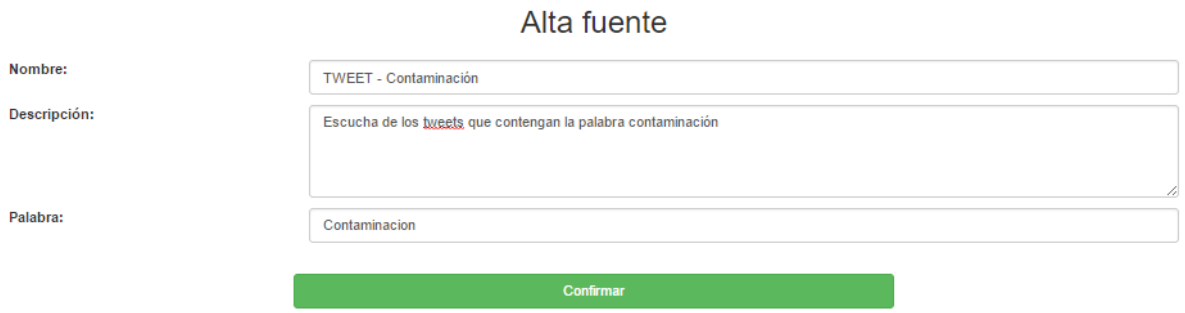


Figura 38 – Nueva fuente twitter.

```
twitter.accessKey=3328120347-0P3U6xKkGCkzK4g5Qv
twitter.accessSecret=vvZ4dCOHtXOsttuOtxF7u4u
twitter.consumerKey=TAj8VjtWxcTMiw2
twitter.consumerSecret=TXH5IE3qOqFOFNNFsMcVkIxn2C5J
twitter.topic=twitterTopic
twitter.activeMQ=tcp://localhost:61616
twitter.idFuentes=incendio:39;agua turbia:47;Contaminacion:48;azufre:49;
```

Figura 39 – Archivo de propiedades de la fuente Twitter.

En el caso de OSE, posee sensores con el protocolo de comunicación SOS. Para darlos de alta en el sistema, en un primer paso se ingresa la URL de donde se consumirán los datos del sensor y el identificador del mismo. Con estos datos el sistema obtiene la ubicación y los parámetros que mide y se los muestra precargados al usuario para que solamente complete los otros datos requeridos, esto se aprecia en la Figura 40.

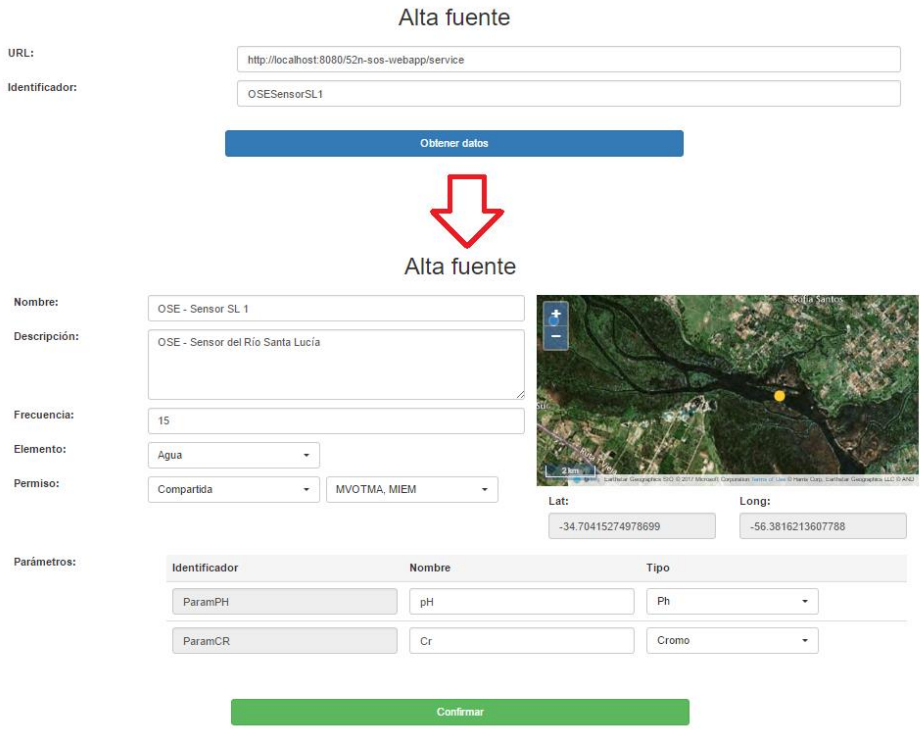


Figura 40 – Nueva sensor con protocolo SOS.

En la Figura 41 se muestra el archivo de propiedades de la solución Mule, generado para el sensor con protocolo SOS.

```
sensorSOS.id = 58
sensorSOS.identificador = "OSESensorSL1"
sensorSOS.url= localhost
sensorSOS.puerto= 8080
sensorSOS.path= 52n-sos-webapp/service
sensorSOS.frecuencia= 15
sensorSOS.topic= esperTopic
sensorSOS.activeMQ= tcp://localhost:61616
sensorSOS.sistemaReferencia= 3857
sensorSOS.x= -6276373.379981053
sensorSOS.y= -4123749.0120996516
sensorSOS.z= 0
sensorSOS.elementoFuente= 1
sensorSOS.idTipoParametro= ParamPH:1;ParamCR:14;
```

Figura 41 – Archivo de propiedades del sensor con protocolo SOS.

De esta forma, las organizaciones ya incorporaron sus fuentes de información al sistema para que éstas sean posteriormente utilizadas para la creación de nuevas alertas.

6.3.3. Alta ley

En tercer lugar, el MVOTMA procede a dar de alta un nuevo decreto 253/079.

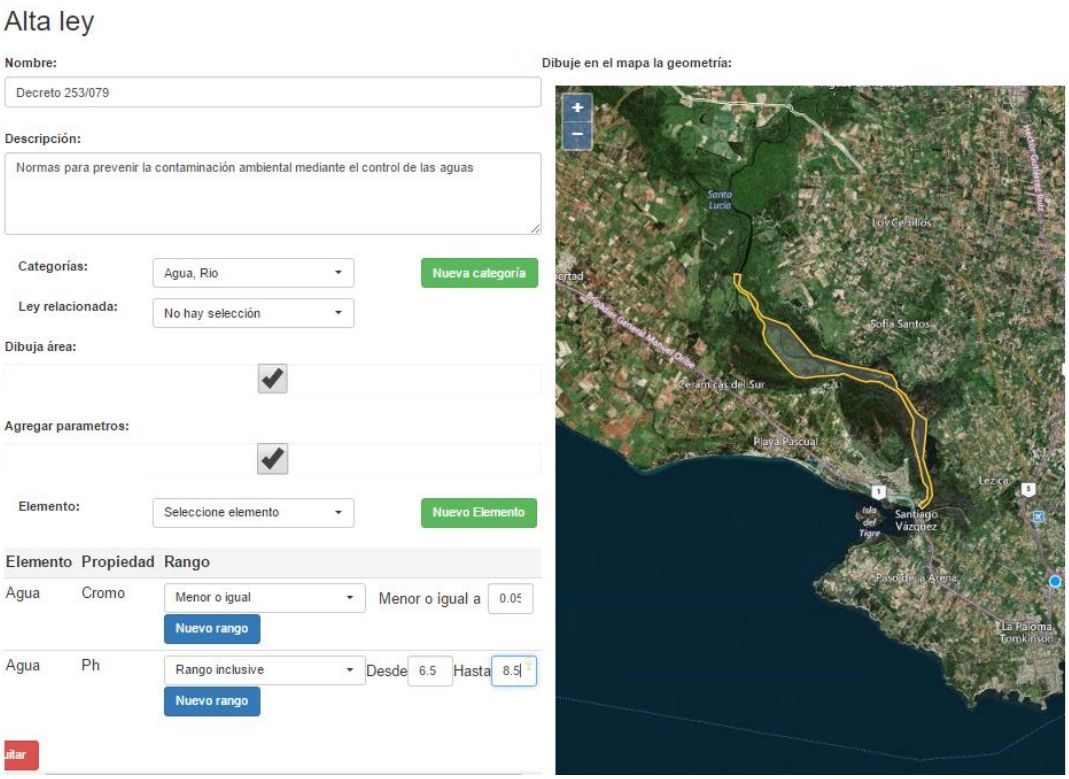


Figura 42 – Alta decreto 253/079.

En la Figura 42 se muestra cómo quedaría dado de alta el decreto en SIMONA. Se encuentra seleccionada el área geográfica en donde tendrá validez el decreto (parte de la cuenca del Río Santa Lucía), así como también algunos de los parámetros que especifica el decreto junto con los valores que deben cumplir. En este ejemplo, el pH debe encontrarse entre 6.5 y 8.5 y el Cromo total debe ser menor o igual a 0.05 mg/L.

6.3.4. Alta alerta

Para finalizar con el caso de estudio, el MVOTMA procede a crear algunas alertas para controlar el estado medioambiental del río Santa Lucía. Por un lado, crea una alerta para llevar el control de los parámetros del decreto creado en la sección anterior, mientras que por otro lado crea otra alerta para la red social Twitter buscando la palabra “Contaminación” en las zonas cercanas a la cuenca del río.

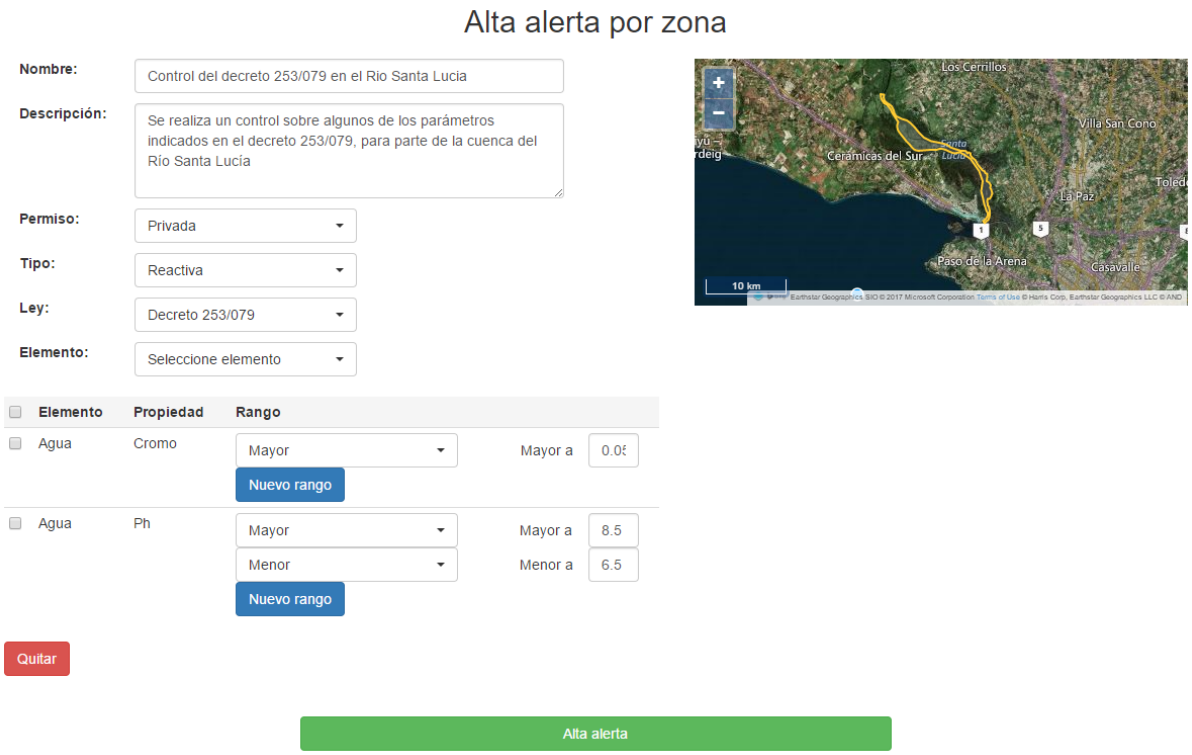


Figura 43 – Alta alerta por zona.

En la Figura 43 se aprecia el alta de una alerta por zona, en ella se indica la zona geográfica en donde la misma tendrá validez. Está asociada al decreto 253/079 creado en la sección anterior, y para los parámetros seleccionados se ingresaron el rango de valores en los cuales se debe notificar a los usuarios. El decreto indicaba que el cromo total debía tener un valor máximo de 0.05 mg/L y el pH estar entre 6.5 y 8.5, con lo cual, en la alerta, se especifica que se debe notificar a los usuarios cuando el cromo supere el valor 0.05 mg/L o el pH sea menor a 6.5 o mayor a 8.5.

Al dar de alta la alerta, se genera una copia de la solución ESB, y se le realiza el deploy en Mule. La solución se encargará de verificar que la fuente se encuentre dentro de la zona geográfica definida. También, compara los valores de las mediciones generadas por las fuentes, con los valores declarados en la alerta, utilizando el motor CEP. Para esto se genera la cláusula EPL asociada a los datos ingresados en la alerta y se graba en el archivo de propiedades de la solución. En la Figura 44 se aprecia parte del archivo de propiedades generado. La imagen fue modificada, quitando parte de los puntos que conforman el polígono dibujado en la alerta.

Cabe destacar, que en esta alerta se utilizarán todas las fuentes de tipo sensor que se encuentren dentro de la zona geográfica definida que sean públicas, propias de la organización, o compartidas con la misma.

```
alerta.id = 67
alerta.eventoTopic = esperTopic
alerta.eventoActiveMQ = tcp://localhost:61616
alerta.clausula = select * from EventoSensor where ( interseccionMultiPoligono(geom,
'MULTIPOLYGON(((-6288440.8758215755 -4115092.5186463553,-6288431.321193039
-4115713.5695011728,-6288278.447136469 -4115971.544471635,-6288096.909194292
-4116649.923097666,-6286959.908398548 -4117366.52023784,-6286682.824171016
-4117911.1340643708,-6286281.529772519 -4118608.6219474734,-6285937.563145236
....
-4117911.134064371,-6286463.067714697 -4117509.839665874,-6286759.261199302
-4116936.561953735,-6287160.555597799 -4116583.040697917,-6287580.959253367
-4116363.2842415967,-6287877.152737972 -4116086.2000140627,-6288030.026794543
-4115627.5778443515,-6287905.816623579 -4115130.7371604983,-6288440.8758215755
-4115092.5186463553)))')) and ( (1=idTipoParametro and 1=elementoFuente and valor > 9 or valor < 7)
or (14=idTipoParametro and 1=elementoFuente and valor > 0) )
alerta.notificacionTopic = notificacionTopic
alerta.notificacionActiveMQ = tcp://localhost:61616
```

Figura 44 – Archivo de propiedades de la alerta por zona.

Además, como se mencionó al comienzo de la sección, el MVOTMA crea también una alerta para la red social Twitter. En la Figura 45 se aprecia el alta de la misma.

Figura 45 – Alta alerta Twittter.

En la imagen se muestra que el usuario indicó, que la alerta se generará si en un lapso de 120 segundos, ocurren 10 tweets dentro de la zona geográfica dibujada, que contengan la palabra “Contaminación”. En la Figura 46 se presenta el archivo de propiedades generado para la solución del ESB.

```
alerta.id = 68
alerta.eventoTopic = twitterTopic
alerta.eventoActiveMQ = tcp://localhost:61616
alerta.clausula = select * from EventoTwitter (tweet regexp '.*(?i)Contaminacion.*').win:time(120
sec) where ( interseccionMultiPoligono(geom, 'MULTIPOLYGON(((-6295391.86808126
-4110401.19603535,-6280715.9586505 -4106961.52976252,-6264664.18271062
-4110477.63306364,-6260154.39804179 -4122936.86867412,-6260689.45723979
-4128822.51985208,-6262676.8199752 -4141281.75546256,-6274524.55935941
-4144033.48848083,-6281709.64001821 -4140976.00734942,-6290270.58718615
-4129816.20121979,-6298067.16407124 -4120185.13565585,-6295391.86808126 -4110401.19603535)))'))
having count(*) >= 10
alerta.notificacionTopic = notificacionTwitterTopic
alerta.notificacionActiveMQ = tcp://localhost:61616
```

Figura 46 – Archivo de propiedades de la alerta generada de Twitter.

6.3.5. Recepción de notificación

Luego de tener las fuentes ingresadas en SIMONA y las alertas creadas, resta esperar a que alguna de las fuentes realice una medición que cumpla con las reglas definidas en las alerta. En caso que esto ocurra, el sistema procede a notificar a los usuarios suscriptos a dicha alerta. Esta notificación puede ser vía mail y/o notificaciones en la web.

En caso de que el usuario tenga configurado recibir las notificaciones vía web, el sistema le mostrará una notificación al momento que se genera la alerta. Además, todas las alertas son almacenadas, y en caso de que el usuario no se encuentre en el sistema al momento que se generó la alerta, recibirá las notificaciones la próxima vez que ingrese al mismo. En la Figura 47 se muestra un ejemplo de las notificaciones que recibe el usuario si se encuentra en el sistema al momento que se generó la alerta.

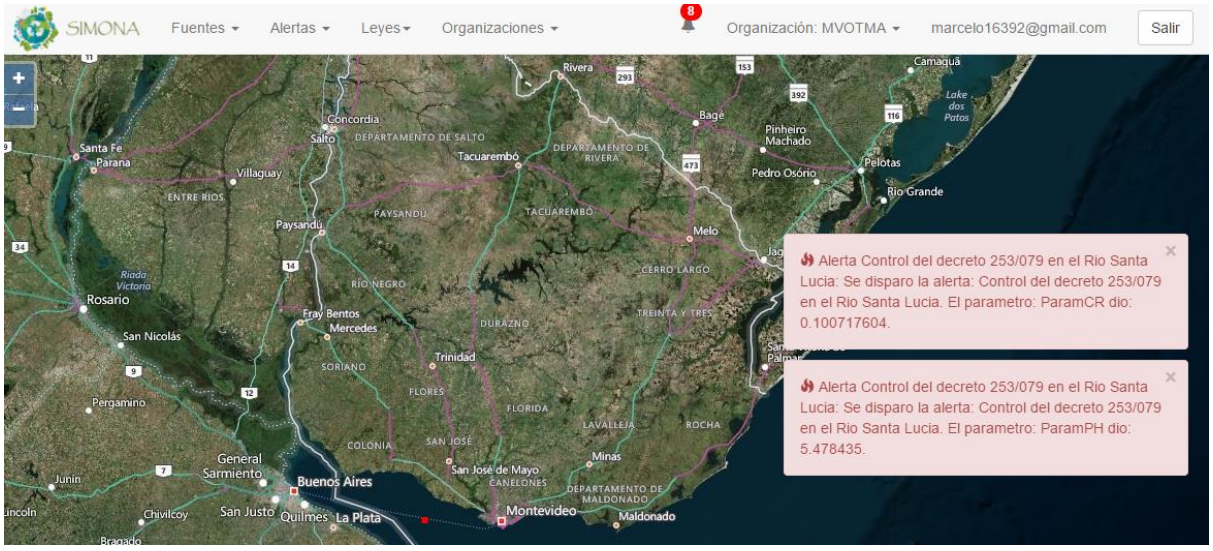


Figura 47 – Notificación instantánea del incumplimiento.

En las Figuras 48 y 49 se listan las notificaciones que aún no fueron leídas y el detalle de la notificación respectivamente.

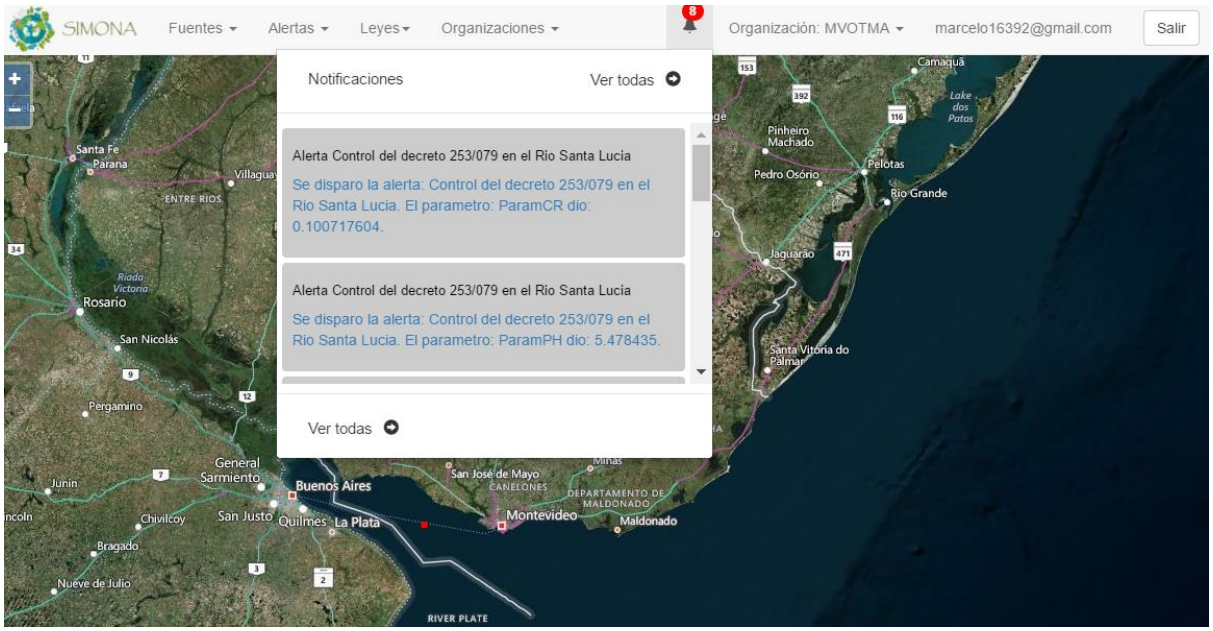


Figura 48 – Notificaciones sin leer.

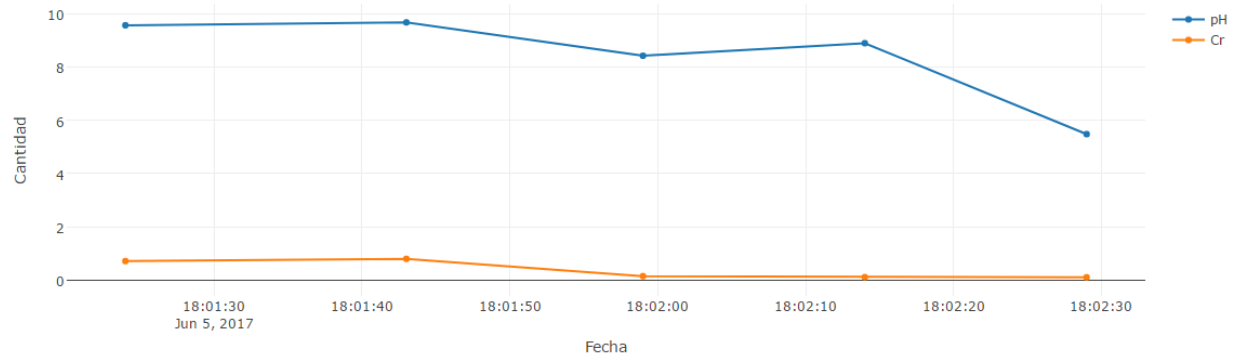
Figura 49 – Detalle de la notificación.

En caso de que el usuario tenga configurado las notificaciones vía email, el usuario recibe un mail tal como se aprecia en la Figura 50.

Figura 50 – Mail de la notificación recibida.

6.3.6. Visualizar datos

Los usuarios también pueden visualizar los datos que han medido las fuentes desde que fueron ingresadas al sistema.



Fecha	Parametro	Valor
05/06/2017 18:02:29	pH	5.478435
05/06/2017 18:02:29	Cr	0.100717604
05/06/2017 18:02:14	Cr	0.10734236
05/06/2017 18:02:14	pH	8.894091
05/06/2017 18:01:59	Cr	0.14425391
05/06/2017 18:01:59	pH	8.422188
05/06/2017 18:01:43	Cr	0.7926376
05/06/2017 18:01:43	pH	9.675685
05/06/2017 18:01:24	Cr	0.71018714
05/06/2017 18:01:24	pH	9.566624

Figura 51 – Visualización de los datos.

En la Figura 51, se aprecia una consulta realizada sobre el sensor “OSE - Sensor SL 1”. Aquí se visualizan los datos de dos maneras diferentes, por un lado, hay una gráfica en donde se muestra cómo variaron los valores de los parámetros a medida que avanzó el tiempo y por otro lado se muestra una tabla, en donde se muestran en detalle los valores obtenidos.

7. Conclusiones y trabajo a futuro

En este capítulo inicialmente se presentan las conclusiones del proyecto, y luego se presentan las posibles mejoras que se pueden realizar a la solución desarrollada.

7.1. Conclusiones

El objetivo general del proyecto consistió en proponer y diseñar mecanismos que permitan monitorear el cumplimiento de leyes medioambientales en tiempo real, utilizando distintas fuentes de información y tecnologías de información geográfica.

Para cumplir con el objetivo se realizó un análisis de las leyes medioambientales nacionales e internacionales. Además, se analizaron las distintas fuentes de información que ayudan a monitorear parámetros medioambientales. En el análisis realizado se obtuvo una serie de requerimientos funcionales que permitieron definir los conceptos más relevantes de la solución: Fuente, Alerta, Ley, Notificación, Tipo de Fuente, Organización.

Luego se diseñó una solución basada en ESB y CEP que aporta al monitoreo en tiempo real de leyes medioambientales. Esta solución analiza los datos generados por las distintas fuentes de información y notifica a los usuarios cuando se incumple alguna ley medioambiental.

Se implementó un prototipo que permitió validar la factibilidad técnica de la solución propuesta. En cuanto a las tecnologías utilizadas, Esper CEP demostró su capacidad para el rápido procesamiento de datos. Además de ofrecer la posibilidad de extender la herramienta con funciones geográficas. En cuanto a Mule ESB, se observó una gran madurez de la misma y la facilidad de uso, debido a su intuitiva interfaz gráfica.

Finalmente se realizó un caso de estudio para validar funcionalmente la solución propuesta. El caso de estudio consistió en mostrar cómo se podría utilizar eficazmente el sistema planteando una hipótesis en donde ciertas organizaciones nacionales necesitan controlar parámetros medioambientales del río Santa Lucia, compartiendo sus fuentes de información.

7.2. Trabajo a futuro

A continuación, se identifican posibles mejoras a realizar a la solución.

- **Denuncias medioambientales**

Una posible mejora a la solución es contar con un módulo de denuncias que permita a los ciudadanos realizar denuncias medioambientales y realizar el seguimiento de las mismas. Estas denuncias serian comunicadas al organismo correspondiente.

- **Generación de alertas más complejas**

Actualmente, las reglas EPL generadas automáticamente por el sistema utilizan un conjunto reducido de las funcionalidades del lenguaje. Por ejemplo, cuando se define una alerta con varios parámetros, alcanza con que un único parámetro tenga un valor incorrecto para notificar a los usuarios. Podría ser útil generar alertas que notifiquen al usuario, si varios de los parámetros generan valores incorrectos en un tiempo determinado.

Para solucionar este problema se podría implementar la lógica necesaria, para realizar traducciones más complejas desde la interfaz del sistema al lenguaje EPL. Esto implica que los usuarios sin conocimiento del lenguaje EPL, a través de la interfaz, puedan generar nuevas consultas.

Por otro lado, para los usuarios con conocimiento del lenguaje EPL se podría otorgarle mayor libertad, permitiendo que ellos definan las reglas EPL a utilizar en las alertas.

- **Procesamiento del lenguaje natural en redes sociales**

En el prototipo desarrollado con respecto a los datos generados por las redes sociales, se tiene en cuenta los datos que contiene exactamente el texto a buscar. Sin embargo, no contempla datos que utilicen sinónimos o conjugaciones verbales diferentes al texto buscado, que representan lo mismo.

Se puede crear un nuevo módulo que permita mejorar las búsquedas de datos en redes sociales, utilizando procesamiento del lenguaje natural.

Referencias

[1] «Medio Ambiente y Cambio Climático», *Naciones unidas de Perú*, [en línea]. Disponible: http://onu.org.pe/temas/medio-ambiente-y-cambio-climatico. [Último acceso: Junio 2017].

[2] Silvia Jankilevich, «Las cumbres mundiales sobre el ambiente Estocolmo, Río y Johannesburgo 30 años de Historia Ambiental», *Universidad de Belgrano*, [en línea]. Disponible: http://repositorio.ub.edu.ar/bitstream/handle/123456789/690/106_jankilevich.pdf?sequence=1&isAllowed=y. [Último acceso: Junio 2017].

[3] «Informe de la Cumbre Mundial sobre el Desarrollo Sostenible», *Naciones Unidas*, [en línea]. Disponible: http://www.cepal.org/rio20/noticias/paginas/6/43766/WSSD_Informe.ESP.pdf. [Último acceso: Junio 2017].

[4] «Constitución de la República», *Parlamento de Uruguay*, [en línea]. Disponible: https://parlamento.gub.uy/documentosyleyes/constitucion. [Último acceso: Junio 2017].

[5] «Ley Nº 17.283», *Poder Legislativo Uruguay*, [en línea]. Disponible: https://legislativo.parlamento.gub.uy/temporales/leytemp8310829.htm. [Último acceso: Junio 2017].

[6] «Ley N° 15.239», *Poder Legislativo Uruguay*, [en línea]. Disponible: https://legislativo.parlamento.gub.uy/temporales/leytemp7267814.htm. [Último acceso: Junio 2017].

[7] «Ley General de Medio Ambiente», *Importa que lo sepas*, [en línea]. Disponible: http://www.impo.com.uy/medioambiente/. [Último acceso: Junio 2017].

[8] «Decreto 253/079», *Ministerio de Vivienda Ordenamiento Territorial y Medio Ambiente*, [en línea]. Disponible: http://www.mvotma.gub.uy/ciudadania/item/10003601-decreto-253-079.html. [Último acceso: Junio 2017].

[9] «Control ambiental de emprendimientos y actividades», *Ministerio de Vivienda Ordenamiento Territorial y Medio Ambiente*, [en línea]. Disponible: http://www.mvotma.gub.uy/control-ambiental-de-emprendimientos-y-actividades.html. [Último acceso: Junio 2017].

[10] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, Marimuthu Palaniswami, «Internet of Things (IoT): A vision, architectural elements, and future directions», *Future Generation Computer Systems*, July 2012, [en línea]. Disponible: https://arxiv.org/ftp/arxiv/papers/1207/1207.0203.pdf, [Último acceso: Junio 2017].

[11] Hakima Chaouchi, «The Internet of Things: Connecting Objects», Febrero 2013, [Último acceso: Junio 2017].

[12] «Sensor», *Real Academia Española*, [en línea]. Disponible: http://dle.rae.es/srv/search?m=30&w=sensor. [Último acceso: Junio 2017].

[13] «Definición de Red social», *Definición ABC*, [en línea]. Disponible: https://www.definicionabc.com/social/red-social.php. [Último acceso: Junio 2017].

[14] «Así se ha contado en Twitter el terremoto de Andalucía y Melilla», *El Pais*, [en línea]. Disponible: http://verne.elpais.com/verne/2016/01/25/articulo/1453706650_429434.html. [Último acceso: Junio 2017].

[15] «¿Qué es una API y para qué sirve?», *ABC*, [en línea]. Disponible: http://www.abc.es/tecnologia/consultorio/20150216/abci--201502132105.html. [Último acceso: Junio 2017].

[16] Juan E. Gutiérrez Palacios, «Los Sistemas de Información Geográfica». [Último acceso: Junio 2017].

[17] «ESRI Shapefile Technical Description», *ESRI*, [en línea]. Disponible: https://www.esri.com/library/whitepapers/pdfs/shapefile.pdf. [Último acceso: Junio 2017].

[18] Colin Childs, «The top nine reasons to use a file geodatabase», *ESRI*, [en línea]. Disponible: https://www.esri.com/news/arcuser/0309/files/9reasons.pdf. [Último acceso: Junio 2017].

[19] «Open Geospatial Consortium», *OGC*, [en línea]. Disponible: http://www.opengeospatial.org/. [Último acceso: Junio 2017].

[20] «Sensor Web Enablement (SWE)», *OGC*, [en línea]. Disponible: http://www.opengeospatial.org/ogc/markets-technologies/swe. [Último acceso: Junio 2017].

[21] M. Iniesto, P. Carballo, «Sensor Web Enablement: Todos los sensores conectados a la web», [en línea]. Disponible: http://www.cartesia.org/geodoc/topcart2004/conferencias/60.pdf. [Último acceso: Junio 2017].

[22] «Observations and Measurements», *OGC*, [en línea]. Disponible: http://www.opengeospatial.org/standards/om. [Último acceso: Junio 2017].

[23] Simon Cox, «Observations and Measurements», *CSIRO Exploration & Mining*, 2003, [en línea]. Disponible: https://www.seegrid.csiro.au/wiki/pub/Xmml/PublicationsPresentations/OM_SWE_dulles.pdf. [Último acceso: Junio 2017].

[24] «Sensor Model Language (SensorML)», *OGC*, [en línea]. Disponible: http://www.opengeospatial.org/standards/sensorml. [Último acceso: Junio 2017].

[25] «Sensor Observation Service», *OGC*, [en línea]. Disponible: http://www.opengeospatial.org/standards/sos. [Último acceso: Junio 2017].

[26] «Sensor Observation Service», *52 north*, [en línea]. Disponible: http://52north.org/communities/sensorweb/sos/index.html. [Último acceso: Junio 2017].

[27] «Web Map Service (WMS)», *OGC*, [en línea]. Disponible: http://www.opengeospatial.org/standards/wms. [Último acceso: Junio 2017].

[28] «Web Feature Service (WFS)», *OGC*, [en línea]. Disponible: http://www.opengeospatial.org/standards/wfs. [Último acceso: Junio 2017].

[29] «Servicio de descarga (WFS) Versión 2.0», *Infraestructura de Datos Espaciales de España*, [en línea]. Disponible: http://www.idee.es/resources/documentos/RD_wfs_v2_0.pdf. [Último acceso: Junio 2017].

[30] «Simple Feature Access - Part 2: SQL Option», *OGC*, [en línea]. Disponible: http://www.opengeospatial.org/standards/sfs. [Último acceso: Junio 2017].

[31] D. Chappell, «Enterprise Service Bus: Theory in Practice», 2004. [Último acceso: Junio 2017].

[32] Gregor Hohpe, Bobby Woolf, «Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions», 2004. [Último acceso: Junio 2017].

[33] David B. Robins, «Complex Event Processing», [en línea]. Disponible: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.457.4226&rep=rep1&type=pdf. [Último acceso: Junio 2017].

[34] W. Roy Schulte, David Luckham, «Real-time intelligence and how it uses complex-event processing (CEP)», [en línea]. Disponible: http://www.complexevents.com/2013/09/17/understanding-real-time-intelligence/. [Último acceso: Junio 2017].

[35] Alexandre Alves, Shailendra Mishra, «Design Patterns for Complex Event Processing (CEP)», *Oracle Corporation*, [en línea]. Disponible: http://www.oracle.com/technetwork/server-storage/ts-4783-159494.pdf. [Último acceso: Junio 2017].

[36] Juan Boubeta-Puig, Guadalupe Ortiz, Inmaculada Medina-Bulo, «ModeL4CEP: Graphical domain-specific modeling languages for CEP domains and event patterns», 2015, [Último acceso: Junio 2017].

[37] Federico Herrera, Laura González, Daniel Calegari, «Complex Event Processing with Geospatial Support for Monitoring and Controlling Compliance with Environmental Regulations», 2016, [Último acceso: Junio 2017].

[38] «Chapter 5. EPL Reference: Clauses», *EsperTech*, [en línea]. Disponible: http://www.espertech.com/esper/release-5.2.0/esper-reference/html/epl_clauses.html. [Último acceso: Junio 2017].

[39] Juan Boubeta Puig, «Model-Driven Development of Domain-Specific Interfaces for Complex Event Processing in Service-Oriented Architectures», [Último acceso: Junio 2017].

[40] Brenda M. Michelson, «Event-Driven Architecture Overview», *Elemental Links*, [en línea]. Disponible: http://elementallinks.com/el-reports/EventDrivenArchitectureOverview_ElementalLinks_Feb2011.pdf. [Último acceso: Junio 2017].

[41] Carlos García Muelas, «Integración de Redes Telemáticas», *Universitat Oberta de Catalunya*, [en línea]. Disponible: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/40187/6/cgmuelasTFC0115memoria.pdf. [Último acceso: Junio 2017].

[42] «Arquitectura de Software», *Facultad de Ingeniería Udelar*, [en línea]. Disponible: https://www.fing.edu.uy/tecnoinf/mvd/cursos/ingsoft/material/teorico/is05-ArquitecturaDeSoftware.pdf. [Último acceso: Junio 2017].

[43] Kushlani de Silva, «WSO2 Advantage Webinar : ESB Evaluation Framework », *WSO2*, [en línea]. Disponible: http://wso2.com/library/webinars/2012/08/wso2-advantage-webinar-esb-evaluation-framework/. [Último acceso: Junio 2017].

[44] «Talend is the Best Source for Open Source ESB», Talend, [en línea]. Disponible: https://www.talend.com/resource/open-source-esb/. [Último acceso: Junio 2017].

[45] «Enterprise Service Bus», Talend, [en línea]. Disponible: http://www.talend.com/resource/enterprise-service-bus/. [Último acceso: Junio 2017].

[46] «WSO2 Enterprise Service Bus», *WSO2*, [en línea]. Disponible: http://wso2.com/products/enterprise-service-bus/. [Último acceso: Junio 2017].

[47] «What Is an Enterprise Service Bus?», *Microsoft*, [en línea]. Disponible: https://msdn.microsoft.com/es-es/biztalk/biztalk-esbtoolkit.aspx. [Último acceso: Junio 2017].

[48] «Overview of the BizTalk ESB Toolkit», *Microsoft*, [en línea]. Disponible: https://msdn.microsoft.com/en-us/library/ff699634(v=bts.70).aspx. [Último acceso: Junio 2017].

[49] «JBoss ESB», *Redhat*, [en línea]. Disponible: http://jbossesb.jboss.org/. [Último acceso: Junio 2017].

[50] «JBoss ESB - Features», *Redhat,* [en línea]. Disponible: http://jbossesb.jboss.org/features.html. [Último acceso: Junio 2017].

[51] «ESB Solutions», *MuleSoft*, [en línea]. Disponible: https://www.mulesoft.com/platform/soa/mule-esb-open-source-esb. [Último acceso: Junio 2017].

[52] «What is Mule ESB?», *MuleSoft,* [en línea]. Disponible: https://www.mulesoft.com/resources/esb/what-mule-esb. [Último acceso: Junio 2017].

[53] «REST APIs», *Twitter*, [en línea]. Disponible: https://dev.twitter.com/rest/public. [Último acceso: Junio 2017].

[54] «Streaming APIs», *Twitter*, [en línea]. Disponible: https://dev.twitter.com/streaming/overview. [Último acceso: Junio 2017].

[55] «Places», *Twitter*, [en línea]. Disponible: https://dev.twitter.com/overview/api/places. [Último acceso: Junio 2017].

[56] «Geo Guidelines», *Twitter*, [en línea]. Disponible: https://dev.twitter.com/overview/terms/geo-developer-guidelines. [Último acceso: Junio 2017].

[57] «Información general sobre la API Graph», *Facebook*, [en línea]. Disponible: https://developers.facebook.com/docs/graph-api/overview. [Último acceso: Junio 2017].

[58] «Esper: Event Processing for Java», *EsperTech*, [en línea]. Disponible: http://www.espertech.com/products/esper.php. [Último acceso: Junio 2017].

[59] «The Java EE 6 Tutorial», *Oracle*, [en línea]. Disponible: http://docs.oracle.com/javaee/6/tutorial/doc/bncdr.html. [Último acceso: Junio 2017].

[60] «What is Geoserver?», *Geoserver*, [en línea]. Disponible: http://geoserver.org/about/. [Último acceso: Junio 2017].

[61] «A high-performance, feature-packed library for all your mapping needs», *OpenLayers*, [en línea]. Disponible: http://openlayers.org/. [Último acceso: Junio 2017].

[62] «Sensor Web Community», *52 north*, [en línea]. Disponible: http://52north.org/communities/sensorweb/. [Último acceso: Junio 2017].

[63] «Santa Lucía en estado "crítico"», *El País*, [en línea]. Disponible: http://www.elpais.com.uy/informacion/rio-santa-lucia-critico-contaminacion.html. [Último acceso: Junio 2017].

Apéndices

Apéndice 1: Procedimiento para el despliegue del prototipo

En este apéndice, se describen los pasos a seguir para poder ejecutar correctamente el prototipo implementado dentro de la máquina virtual entregada.

1. Levantar ActiveMQ
    a) Abrir una consola en la carpeta "C:\apache-activemq-5.8.0\bin"
    b) Ejecutar el comando "activemq“
2. Levantar Tomcat
    a) Ejecutar startup.bat que se encuentra en la carpeta "C:\Apache tomcat 9.0\bin"
    b) Esperar a que quede levantado para los siguientes pasos
3. Levantar Mule
    a) Ejecutar en una consola el comando "mule"
    b) Esperar a que quede levantado para los siguientes pasos
4. Levantar la "Logica"
    a) Abrir una consola en la carpeta "C:\Users\Marcelo\Desktop\Datos"
    b) Ejecutar el comando "java -jar Logica.jar"
    c) En los marcadores del Chrome está el enlace a la página.
5. Levantar "ProcesarSimularSensores" (realizar este paso si se quiere generar datos con la simulación de sensores)
    a) Abrir una consola en la carpeta "C:\Users\Marcelo\Desktop\Datos"
    b) Ejecutar el comando "java -jar ProcesarSimularSensores.jar"
    c) En los marcadores del Chrome está el enlace a la página.

Apéndice 2: Creación nuevo tipo de fuentes

En esta sección se describe el procedimiento para incorporar nuevos tipos de fuentes a la plataforma.

Para la creación de un nuevo tipo de fuente el usuario deberá implementar una solución ESB para Mule que se encargue del consumo en tiempo real de los datos de una fuente. Las soluciones ESB implementadas en Mule cuentan con un archivo de propiedades, llamado “mule-app.properties”. La solución implementada deberá tener este archivo vacío, debido a que la plataforma grabará un conjunto de propiedades en él.

Para la creación de un nuevo tipo de fuente el usuario ingresa a la opción “Nuevo tipo de fuente”. En la Figura 52, se presenta la pantalla para el ingreso de un nuevo tipo de fuente.

Figura 52 – Nuevo tipo de fuente.

El usuario deberá ingresar:

- Nombre: Nombre del nuevo tipo de fuente.
- Archivo ZIP Mule: Solución ESB implementada.
- Campos obligatorios: Variables utilizadas en la solución ESB que permiten el consumo de los datos de una fuente concreta. Estas variables están compuestas por tres campos:
    - Identificador en Mule: Es el nombre de la variable que se utiliza en la solución Mule implementada.
    - Nombre: Es el nombre a mostrar en la pantalla de alta fuente del nuevo tipo.

- Tipo: Es el tipo de dato de la variable (p. ej. Texto, número).

Estos campos son solicitados al momento de dar de alta una fuente de este nuevo tipo, y también son grabados en el archivo de propiedades con el valor que el usuario ingresa (IdentificadorEnMule=valor).

Además de las propiedades descritas anteriormente, el archivo de propiedades contendrá propiedades de la fuente en la plataforma. Las propiedades son:

- fuente.id: Identificador de la fuente en la plataforma.
- fuente.frecuencia: Frecuencia estimada, en segundos, en que la fuente genera nuevos datos.
- fuente.cola: Identificador de la cola JMS donde se enviarán datos generados.
- fuente.activeMQ: Dirección de la cola JMS donde se enviarán datos generados.
- fuente.idTipoParametro: Identificador en la plataforma de los parámetros que componen la fuente.
- fuente.elementoFuente: Identificador en la plataforma del elemento de la fuente.
- fuente.sistemaReferencia: Sistema de referencia de la ubicación de la fuente.
- posición.x: Coordenada x de la ubicación de la fuente.
- posición.y: Coordenada y de la ubicación de la fuente.
- posición.z: Coordenada z de la ubicación de la fuente.

En particular la propiedad correspondiente a la frecuencia puede ser de utilidad para la implementación de la solución ESB.

Con respecto a la implementación de la solución ESB, debe finalizar enviando un archivo JSON a una cola de mensajes JMS. Los datos necesarios para el envío a la cola JMS se encuentran en el archivo de propiedades. El archivo JSON deberá seguir el formato utilizando en el ejemplo de la Figura 53.

```
{
    "idFuente": "28",
    "sistemaReferencia": "4326",
    "x": "7.651968812254194",
    "y": "51.935101100104916",
    "z": "12",
    "elementoFuente": "1",
    "idTipoParametro": "Temperatura:2;Ph:1;",
    "valores": [{
        "identificadorParametro": "Temperatura",
        "fecha": "2016-11-07T23:42:01.280Z",
        "valor": "-53.988796"
    },
    {
        "identificadorParametro": "Temperatura",
        "fecha": "2016-11-07T23:42:09.609Z",
        "valor": "-53.236885"
    }]
}
```


Figura 53 - Ejemplo del JSON con los datos de la fuente.

Los campos idFuente, sistemaReferencia, x ,y ,z, elementoFuente, idTipoParametro se encuentran en el archivo de propiedades.