Easy Alerts

Herramienta para definición amigable de reglas para eventos complejos orientada a expertos de dominio

Informe de Proyecto de Grado

Gimena Ana Vera Agustoni
gimenavera@gmail.com

Docente Supervisora
Raquel Sosa
raquels@fing.edu.uy

Instituto de Computación
Facultad de Ingeniería
Universidad de la República

Montevideo, Uruguay
Mayo, 2021

Resumen

En la actualidad todo se puede medir y almacenar en datos. Muchas empresas e instituciones a lo largo y ancho del planeta podrían utilizar estos datos para la toma de decisiones. Estos datos pueden provenir de un sinfín de fuentes. Aquí nos encontramos por un lado con personal técnico que podrían dar uso a este tipo de datos; por otro lado, personal experto de dominio que efectivamente podrían realizar un análisis exhaustivo de ellos. Muchas veces el inconveniente radica en que estos grupos de personas son disjuntos.

Existe a su vez un área dentro de las tecnologías de software, centrada en el análisis y la gestión de datos con el objetivo de detectar situaciones de interés lo antes posible. Esta área es el Procesamiento de Eventos Complejos (CEP por sus siglas en inglés). Uno de los inconvenientes con estas tecnologías, es que para su uso se requiere de desarrolladores de software, ya que se usan prácticamente como lenguajes de programación.
El área de CEP se centra en el procesamiento de grandes volúmenes de información de diferentes fuentes en tiempo real, identificando eventos que cumplen con ciertas reglas (o patrones). Esto es vital para la automatización en la gestión de eventos como se propone por ejemplo en domótica o smart cities.

Con el objetivo de facilitar a los usuarios expertos de un dominio la utilización de tecnologías CEP se presenta este proyecto de grado. El foco del proyecto es diseñar una herramienta genérica que se pueda adaptar al uso en diferentes dominios, donde el experto defina las reglas que permiten detectar los eventos deseados mediante el procesamiento de información de diversas fuentes de datos, sin necesidad de conocimientos de programación.

La herramienta considera que las fuentes de datos pueden cambiar y que su procesamiento se puede aplicar a diferentes fenómenos como por ejemplo tráfico, clima, control de condiciones de confort, detección de incendios, etc. por lo que las reglas también deben ser configurables.

En este proyecto se implementa un prototipo de una herramienta con una interfaz amigable llamada Easy Alerts. En ella los expertos en diferentes dominios tienen la posibilidad de configurar los umbrales de los patrones o reglas. Estas reglas son integradas a un motor CEP mediante un adaptador y en caso de cumplirse, se dispara una acción. En EasyAlerts se pueden configurar reglas simples y reglas compuestas. Las reglas compuestas, son las que surgen de la unión de dos reglas simples. El sistema expone un Web Service para proporcionar los patrones generados por el usuario experto y de esta forma ser integrado a un motor CEP.

Palabras Claves

Easy Alerts, Eventos complejos, Detección temprana, sensores, CEP, Drools, Java, Eclipse, Web Services, .Net, SOAP, C#, WCF, SOA

Tabla de Contenido

Tabla de Figuras

1. Introducción

A continuación, se presentan conceptos que ayudarán a comprender el problema planteado, su motivación, objetivos y resultados esperados al finalizar el Proyecto de Grado. En la última sección se brinda al lector una idea de la organización del documento en los capítulos venideros.

1.1. Motivación y Contexto

En el mundo hay diferentes tipos de sucesos que afectan de una manera u otra a su entorno. Por ejemplo, la inundación de un río que genere la evacuación de los ciudadanos que viven cerca o que no permita el paso de vehículos. También puede ser el incendio de un centro comercial, que ocasione pérdidas materiales y eventualmente humanas.

Este tipo de sucesos se generan por uno o más acontecimientos que pueden parecer aislados pero que especialistas en las disciplinas pueden encontrar relación entre ellos. La mayoría de estos sucesos son medibles y se pueden guardar los datos de dichas mediciones. Estos datos pueden ser muy diversos, provenir de diferentes fuentes y en gran cantidad. Existen especialistas de cada dominio que, con estos datos, pueden establecer condiciones en las que estos acontecimientos desencadenan sucesos de interés. Sin embargo, generalmente, este tipo de procesamiento de información necesita ejecutarse con algún software específico que sea capaz de manejar la complejidad de las relaciones y la cantidad de datos obtenidos. Este software específico es del área de procesamiento de eventos complejos. Para hacer buen uso de este tipo de software, se requiere conocimientos de programación, ya que en la práctica es necesario implementar código para definir estas relaciones.

Por lo anterior, surge la necesidad de tener algún tipo de interfaz que sea independiente de las tecnologías de procesamiento de eventos complejos, que permita a especialistas de diferentes dominios utilizar este potencial para procesar los datos sin tener que depender de técnicos informáticos.

1.2. Objetivos del proyecto

Este proyecto tiene como objetivo general proponer una herramienta que permita definir en forma amigable umbrales de patrones o reglas para el procesamiento de eventos complejos.

Este proyecto cuenta con los siguientes objetivos específicos:

- Estudiar la problemática de la detección de eventos complejos, considerando un conjunto dinámico y extensible de fuentes de información.
- Proponer e implementar un prototipo de herramienta para definir reglas en forma amigable por parte de un experto del dominio.
- Utilizar la herramienta con dos casos de estudio para validarla

Con el prototipo se espera obtener:

- Un asistente con patrones para que usuarios expertos puedan establecer umbrales de riesgo y anidación de 2 reglas básicas en la formación de reglas compuestas.
- Un BackOffice donde se administren los usuarios registrados y las reglas ingresadas, considerando que los usuarios pueden pertenecer a diferentes dominios y diferentes organizaciones.

- Una prueba de integración con diferentes sistemas de acciones o alarmas definidos por los usuarios.
- Una prueba de integración con un motor de CEP.

1.3. Resultados alcanzados

Los resultados alcanzados en este proyecto de grado son un prototipo de la herramienta Easy Alert, los casos de estudio con los cuales se valida el prototipo y la documentación que detalla el proceso de análisis, diseño e implementación de dicha herramienta.

El prototipo desarrollado cuenta con una interfaz gráfica donde el usuario experto de un dominio tiene la opción de configurar tanto los patrones como las acciones que se generen en caso de que estos patrones se cumplan. Se definió una capa de abstracción para separar la herramienta de los motores CEP. Para validar los casos de estudio se utiliza uno de los motores de procesamiento de eventos complejos, ofrecidos en el mercado. Los datos que alimentan al motor de procesamiento se simulan de una única fuente externa. La alerta o acción generada es el envío automático de un correo electrónico, de un sms, como también la invocación de un web service/api simulada. Adicionalmente el prototipo cuenta con una interfaz para un usuario administrador que le permite definir nuevos usuarios expertos que usen la herramienta en forma independiente. Pudiendo definir diferentes dominios y diferentes organizaciones a las que pertenecen los usuarios.

El informe elaborado describe el proceso de investigación de herramientas y estudios previos; análisis, diseño e implementación de la herramienta.

1.4. Organización del documento

A continuación, se detalla la estructura del resto del informe. En el Capítulo 2 se encuentra el Marco Conceptual, donde se hace la investigación de las herramientas CEP y de estudios previos. En el Capítulo 3 se explica el Análisis de requerimientos y en el Capítulo 4 se exponen las principales decisiones de Arquitectura y Diseño. En el Capítulo 5 se documentan los puntos más importantes de la Implementación. En el Capítulo 6 se presentan las Pruebas y Casos de estudio. En el Capítulo 7 están las Conclusiones y trabajo a futuro. El Capítulo 8 presenta el Glosario y en el Capítulo 9 las Referencias bibliográficas.

2. Marco Conceptual

Para lograr un entendimiento completo de la problemática, se investigó el paradigma del procesamiento de eventos complejos (CEP por sus siglas en inglés) y algunas de las herramientas que lo implementan. Se decide realizar esta investigación de las herramientas disponibles de CEP dado que es un conjunto de tecnologías de uso específico.

También se buscaron estudios relacionados a CEP, tanto para entender su aplicación en diferentes dominios como para identificar si existe algún trabajo previo sobre interfaces amigables a usuarios que no tengan conocimiento de programación.

2.1. Procesamiento de Eventos Complejos

Un evento es simplemente algo que ocurre: la llegada de un correo electrónico, una señal proveniente de un sensor, un semáforo que se pone en rojo, una transferencia bancaria [1]. Cuando varios de estos eventos se combinan y desencadenan otro evento, este último se conoce como evento complejo.

Los eventos no acostumbran a ocurrir de una forma programada u ordenada, pueden proceder de diversas fuentes y en grandes cantidades. Para este tipo de problemática donde se requiere responder a los eventos según llegan, existen soluciones de software donde el CEP es una ellas.

CEP hace referencia a un conjunto de tecnologías que permiten procesar y relacionar continuamente grandes cantidades de eventos para detectar y responder a las condiciones cambiantes del entorno. Este procesamiento puede ser de utilidad en situaciones que permiten ser modeladas como eventos complejos.

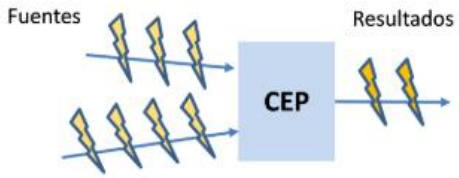


Figura 1: Eventos CEP y resultados. Tomado de [2]

Las tecnologías CEP permiten a partir de eventos de entrada (pueden ser de distintas fuentes) obtener información útil, analizarla (buscar ciertos patrones, por ejemplo) y tomar acciones en función de esa información. Como resultado de ese análisis se pueden identificar amenazas u oportunidades y actuar en consecuencia.

Las acciones tomadas pueden ser variadas, por ejemplo, se puede notificar a cierta persona/organización que deben tomar alguna acción para tratar de reducir un riesgo. La aplicación de CEP puede hacerse en distintos ámbitos. En el ámbito de la administración de redes, por ejemplo, se podría detectar tempranamente la congestión de una red basándose en patrones de comportamiento anormal en tiempo real.

2.2. Herramientas CEP

Las herramientas CEP pueden clasificarse de varias formas, una de las características es sobre qué sintaxis se expresan los patrones. Los patrones se pueden expresar en un lenguaje de procesado de eventos que implementa y extiende SQL-standard, o en una sintaxis propia de la herramienta. En ambos casos se definen patrones que disparan acciones cuando ciertos eventos cumplen con ellos.

Los patrones son una porción de código que intenta modelar una situación. Para esto la mayoría de las tecnologías CEP ponen a disposición del usuario un conjunto de herramientas como por ejemplo operadores temporales, operadores lógicos, ventanas, etc.

En los últimos años han surgido muchas herramientas sobre este tema. Se decidió investigar algunas de las más populares y se detalla a continuación las principales características.

Estos productos son:

- Esper de EsperTech[3]
- Drools de JBoss[4]
- Flink de Apache[5]
- WSO2 Siddhi[6]
- Cloud Platform de Google[7]
- Storm de Apache[8]

2.2.1. Esper

Esper es un framework de CEP de EsperTech[3] disponible para Java y .Net (NEsper). Tiene una versión gratuita y una Enterprise licenciada.
Para tratar con los eventos, Esper ofrece un lenguaje propio: Esper EPL.
Esper procesa grandes volúmenes de mensajes entrantes, ya sea de carácter históricos o en tiempo real.
El compilador y el runtime de Esper pueden correr en cualquier arquitectura y contenedor, ya que no tienen dependencias con servicios externos y no requieren ningún almacenamiento externo.
Esper EPL está organizado en módulos, cada módulo consiste en sentencias (“statements”). Las sentencias son el medio para realizar el análisis evento-tiempo. Estas sentencias son “queries” continuas que analizan eventos y tiempo y detectan situaciones.

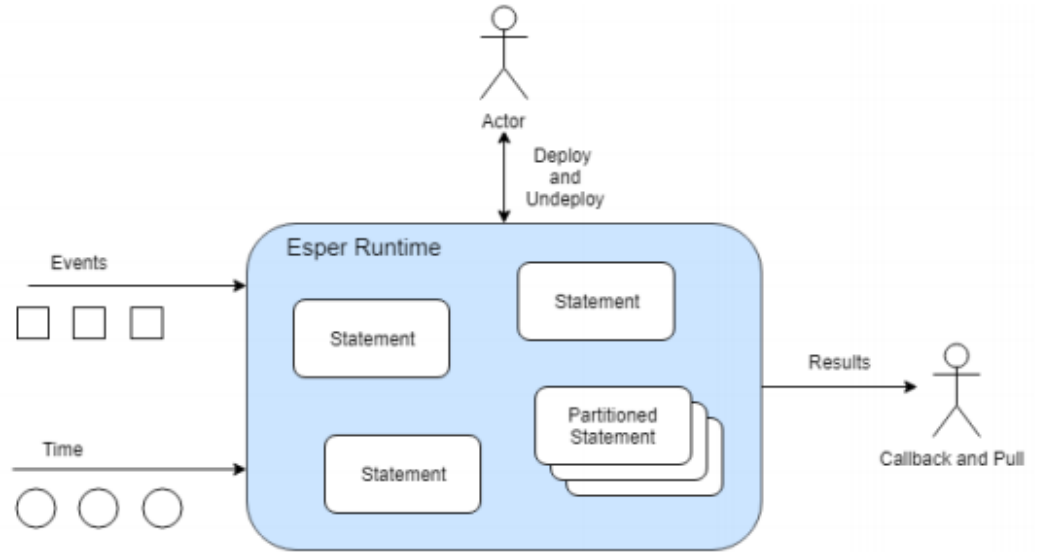


Figura 2: Diagrama de funcionamiento de Esper [20]

2.2.2. Drools

Drools[4] es una plataforma open source de JBoss de modelado de comportamiento unificado escrito en java, que cuenta con varios módulos. Drools proporciona un core de Business Rules Engine (BRE).

Drools es una colección de herramientas que permite separar y trabajar sobre lógica y datos encontrados en cierto dominio.

El desarrollo de patrones en Drools se implementa mediante un lenguaje propio llamado drl. Un típico archivo de patrones en Drools es un archivo con extensión drl. En un archivo drl se pueden tener múltiples patrones y funciones como también declaraciones, importaciones y atributos que pueden ser asignados y utilizados por los patrones.

Drools se dividen en dos partes principales: Authoring y Runtime. El proceso de "Authoring" implica la creación del archivo de patrones (.drl), el cual contiene los patrones a analizar. Este análisis chequea que la sintaxis de los patrones sea la correcta y produce una estructura intermedia que los "describe". Esto es luego pasado al Package Builder que produce paquetes, generando el código y compilando lo necesario para la creación de los paquetes. El runtime involucra la creación de una memoria de trabajo y el manejo de las activaciones. La memoria de trabajo es una parte fundamental del motor Drools: es aquí donde los hechos son insertados donde pueden ser luego modificados o removidos. Los hechos son clases de Java simples los cuales una vez dentro de la memoria de trabajo, puede resultar en uno o más patrones siendo concurrentemente verdaderos y programados para su ejecución. Se comienza con un hecho, se propaga hacia la comparación de patrones y termina con una conclusión.

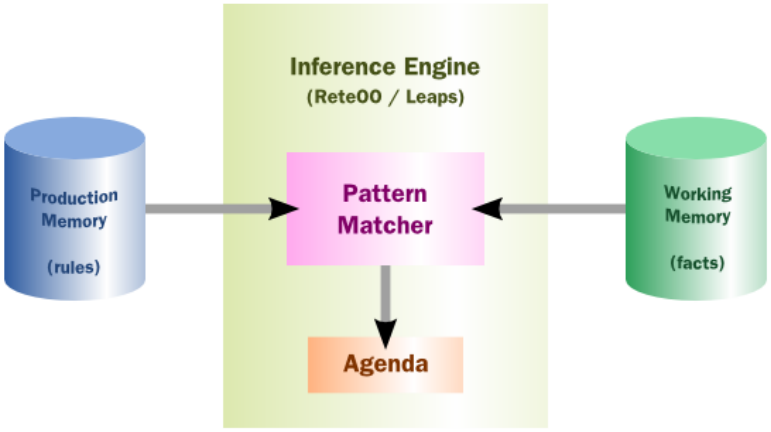


Figura 3: Diagrama de alto nivel Drools[21]

Drools proporciona soporte para los modelos de la especificación Decision Model and Notation (DMN), donde el usuario técnico tiene un editor visual para realizar el modelado. Decision Model and Notation (DMN) es un estándar publicado por Object Management Group. Es un enfoque estándar para describir y modelar decisiones repetibles dentro de las organizaciones.

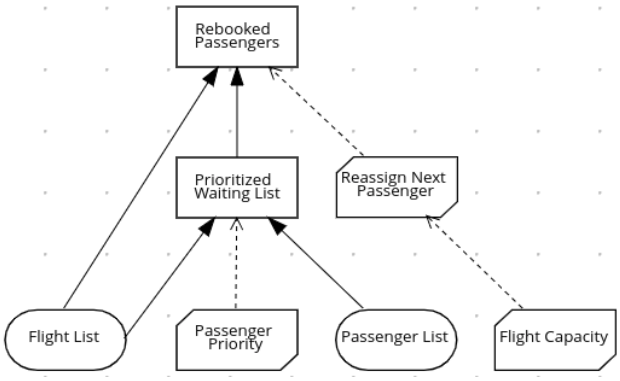


Figura 4: Ejemplo de modelado

2.2.3. Apache Flink

Flink[5] es un motor open source de procesamiento de flujos de datos de Apache, que proporciona capacidades de distribución de datos, comunicación y tolerancia a fallos. Flink proporciona APIs para los lenguajes Java, Scala, Python, R y SQL.

FlinkCEP es la librería de procesamiento de eventos complejos (CEP) implementado sobre Flink que permite detectar patrones de eventos en flujos de eventos, brindado la oportunidad de separar los datos de interés de dicho flujo. FlinkCEP brinda un API, Pattern API, que permite especificar los patrones que se desean detectar en el flujo de datos.

La Pattern API permite definir secuencias de patrones complejos que se deseen extraer del flujo de entrada. Cada secuencia de patrones complejos consiste en múltiples patrones simples. Las secuencias de patrones pueden verse como grafos de patrones, donde las transiciones de un patrón a otro ocurren basados en las condiciones especificadas por el usuario. Una ocurrencia es una secuencia de eventos de entrada que pasa por todos los patrones del grafo complejo, a través de una secuencia válida de transiciones entre patrones.

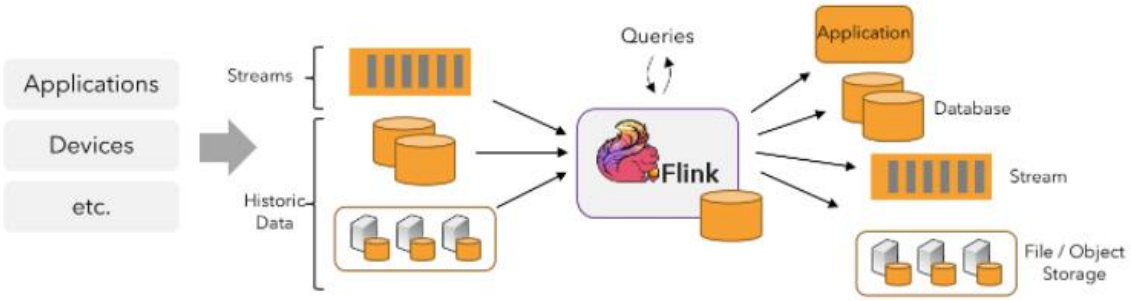


Figura 5: Diagrama de Flink[22]

2.2.4. Siddhi

Siddhi[6] es un sistema open source, nativo en la nube, escalable y de procesamiento de eventos complejos, capaz de construir aplicaciones dirigidas por eventos. Los casos de uso pueden ser análisis de tiempo real, integración de datos y manejo de notificaciones entre otros. En este momento, el proyecto Siddhi está integrado con la plataforma WSO2 y todas las futuras liberaciones serán realizadas como parte de esta. Bajo esta integración, existen dos licenciamientos, uno gratuito para educación o prueba, y otro corporativo con costo.

La lógica del procesamiento de eventos puede ser escrita usando una implementación de SQL conocido como SiddhiQL (Siddhi Streaming SQL) para capturar eventos de diversas fuentes de datos, procesarlos y analizarlos, para finalmente integrándose con múltiples servicios publicar las salidas en variados "sinks" en tiempo real.

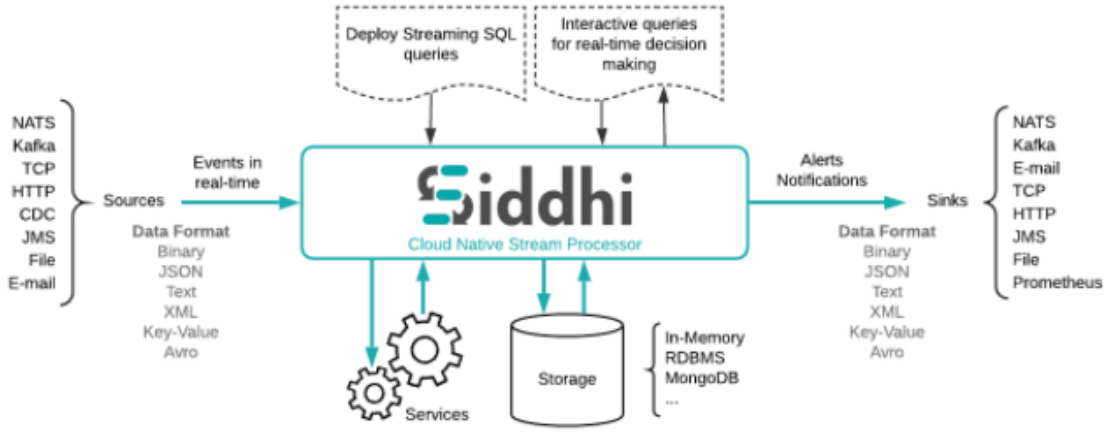


Figura 6: Diagrama de alto nivel de arquitectura de Siddhi[23]

2.2.5. Google Cloud Platform

Google Cloud Platform[7] Brinda una arquitectura que puede ser usada para procesamiento de eventos complejos. Esto se debe a que cuenta con ciertos productos que unidos forman un sistema capaz de obtener datos de diversas fuentes, analizarlos y procesarlos en tiempo real y de ser necesario ejecutar acciones.

El componente principal para poder procesar eventos complejos, Dataproc, brinda un licenciamiento por recursos usados.

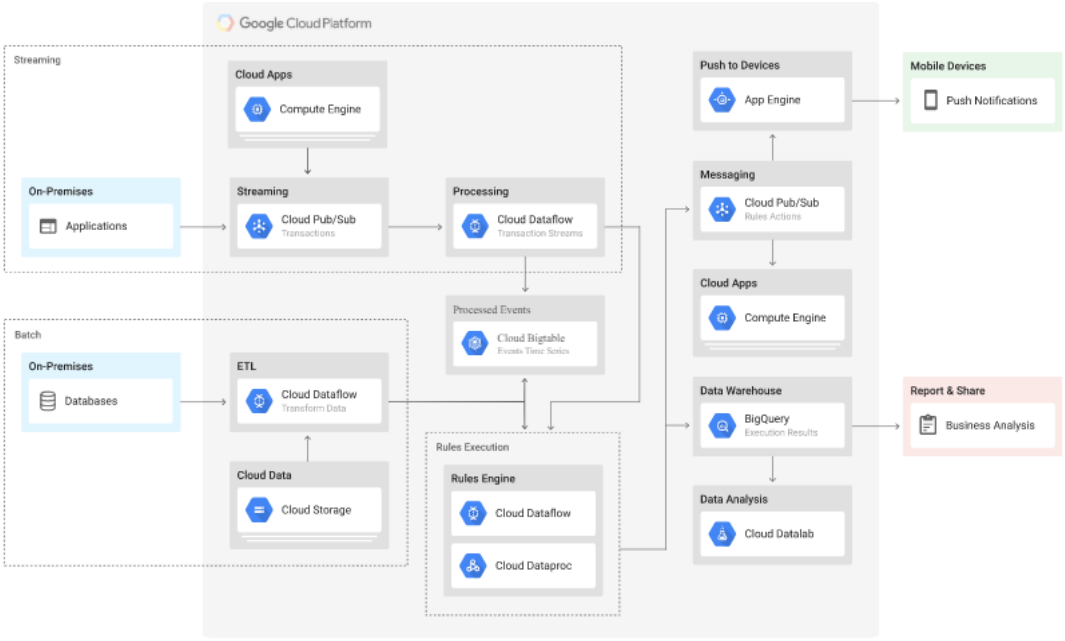


Figura 7: Diagrama de arquitectura Google Cloud Platform[7]

2.2.6. Apache Storm

Apache Storm[8] es un sistema de tiempo real distribuido, gratuito y open source. La abstracción del core de Storm es sobre flujos. Un flujo es una secuencia ilimitada de tuplas, el cual puede ser transformado de una manera confiable a un nuevo flujo mediante las primitivas que Storm proporciona. Las primitivas básicas de Storm que proveen la transformación de flujos se denominan “spouts” y “bolts”.

Un spout es una fuente de flujos mientras un bolt consume cualquier número de flujos de entrada, los procesa y posiblemente emite nuevos flujos. Esta compleja transformación requiere múltiples pasos y por lo tanto múltiples bolts. Los Bolts pueden correr funciones, filtrar tuplas, hacer agregación de streaming, hablar con bases de datos entre otros.

Redes de spouts y bolts se empaquetan en una topología que es el mayor nivel de abstracción que se somete al cluster de Storm para la ejecución. Una topología es un grafo de transformaciones de flujos donde cada nodo es un spout o un bolt. Las aristas del grafo indican que bolt está subscrito a qué flujo. Cuando un spout o bolt emite una tupla a un flujo, envía la tupla a cada bolt que subscribe al flujo.

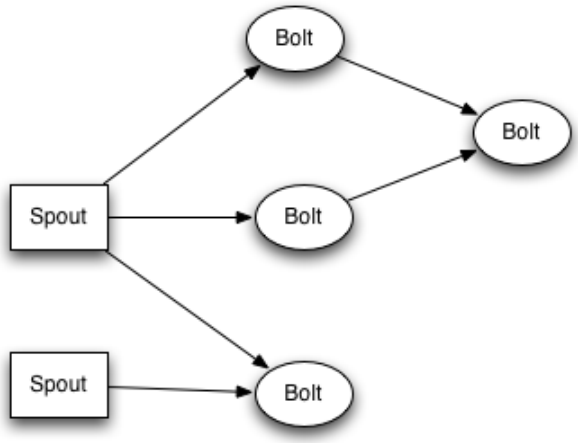


Figura 8: Topología de Apache Storm[27]

2.2.7. Comparativa de herramientas CEP

Luego de realizar la investigación a alto nivel de algunas de las herramientas del mercado, se realiza una tabla comparativa con sus principales características:

	Proveedor	Licenciamiento	Primitivas	Lenguaje	Disponible
Esper	EsperTech	Gratuito con versión enterprise licenciada	Statements	Esper EPL	Java y .Net
Drools	JBoss	Gratuito	Hechos y reglas	Sintaxis propia drl	Java
FlinkCep	Apache	Gratuito	Flujos y patrones	Sintaxis propia	Java, Scala, Python, R
Siddhi	WoS2	Gratuito para estudio y prueba Privativo para organizaciones	Eventos y sinks	Siddhi SQL	Java
Storm	Apache	Gratuito	Spouts y Bolts	Sintaxis propia	Java
Goggles Cloud Platform	Google	Por cantidad de uso	Diversos	Sintaxis propia	Propio

2.3. Estudios previos

Se identificó que el área de procesamiento de eventos complejos puede ser aplicada a muchos dominios. Hay una cantidad de trabajos relacionados con estas tecnologías en muy diversos escenarios y con diferentes objetivos. Dentro de los estudios realizadados se destacaron cinco estudios muy interesantes sobre procesamiento de eventos complejos aplicados a diferentes dominios:

- A model-driven approach for facilitating user-friendly design of complex event patterns
- Preliminary Development and Evaluation of Lightning Jump Algorithms for the Real-Time Detection of Severe Weather
- CASA and LEAD: Adaptive Cyberinfrastructure for Real-Time Multiscale Weather Forecasting
- Scalable CEP for Smart Cities
- COMPLEX EVENT PROCESSING AND DATA MINING FOR SMART CITIES

2.3.1. A model-driven approach for facilitating user-friendly design of complex event patterns

Este estudio fue realizado por Juan Boubeta-Puig, Inmaculada Medina -Bulo, Guadalupe Ortíz en febrero de 2014 por la Universidad de Cádiz[9].

Esta publicación presenta una herramienta desarrollada para dar abstracción a los usuarios expertos de un dominio específico cualquiera, mediante una interfaz para escribir código EPL (Event Processing Language) que pueda ser insertado en cualquier motor CEP, sin necesidad de que el experto de dominio conozca el lenguaje específico del motor CEP. Se le brinda al usuario experto, una lista de operadores y operandos para que logre de esta forma desarrollar sus reglas independientemente del dominio y del motor CEP a utilizar. En la siguiente figura se muestran las fases del enfoque de los autores para definir patrones de eventos de una forma amigable al usuario.

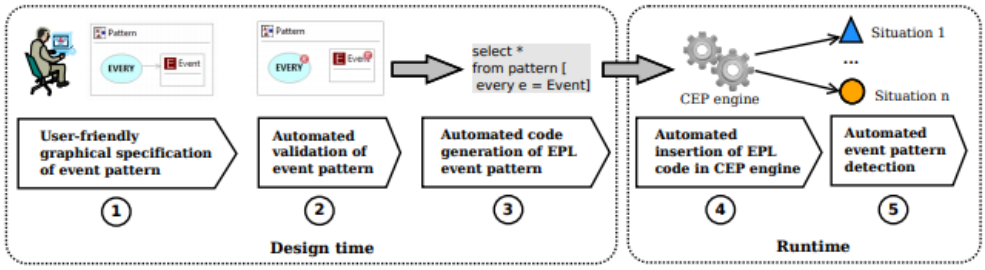


Figura 9: Fases del enfoque (Boubetta et al 2014)

El caso de uso presentado muestra cómo es posible detectar un brote epidémico de influenza a partir de datos de pacientes con cierta sintomatología y en cierta región.

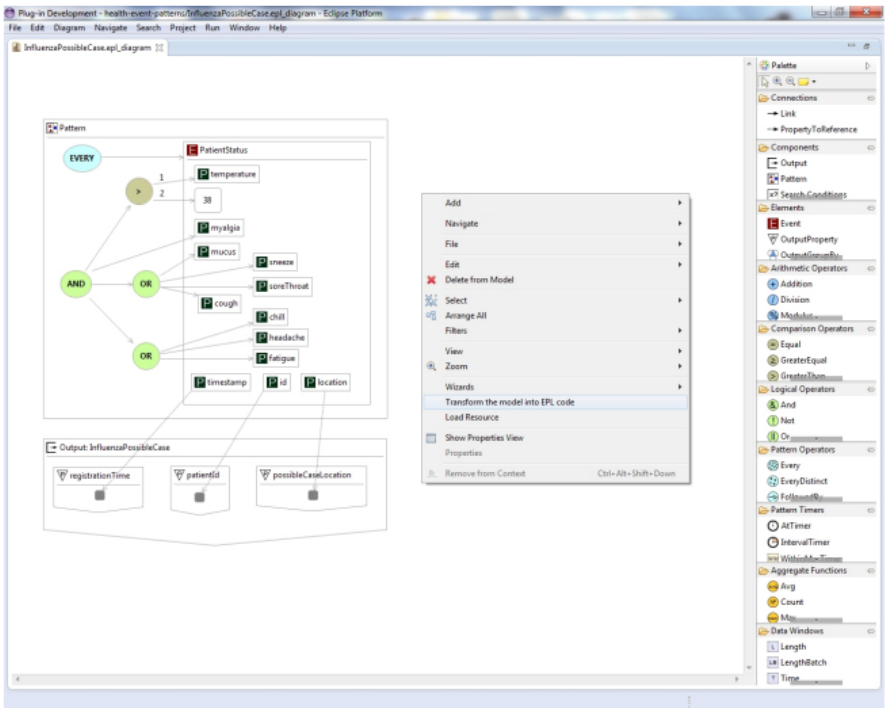


Figura 10: Interfaz con caso de estudio (Boubetta et al 2014)

2.3.2. Semantic Complex Event Processing for Social Media Monitoring - A Survey

El siguiente estudio tiene como autores a Robin Keskisarkka y Eva Blomqvist y fue presentado en la conferencia internacional de semántica web del 2014 [10].

Esta investigación utiliza varias herramientas de procesamiento complejo de eventos, con otras relacionadas a la semántica. Con esto se identificó que se puede llegar a detectar terremotos, evaluar la propagación de un brote de influenza, ayudando al alivio de desastres, por nombrar algunas cosas.
Se realizó el experimento con varias herramientas como ETALIS, C-SPARQL, CQELS e INSTANS para luego analizar estas herramientas por expresividad, performance y razonamiento.

2.3.3. Adaptive Cyberinfrastructure for Real-Time Multiscale Weather Forecasting

Esta investigación se trata de dos proyectos, CASA y LEAD, fusionados. Fue llevada a cabo por varios investigadores de diferentes lugares: Beth Plale and Dennis Gannon, Indiana University Bloomington; Jerry Brotzge and Kelvin Droegemeier, University of Oklahoma, Norman; Jim Kurose and David McLaughlin, University of Massachusetts Amherst; Robert Wilhelmson, University of Illinois at Urbana-Champaign; Sara Graves, University of Alabama in Huntsville; Mohan Ramamurthy, University Corporation for Atmospheric Research; Richard D. Clark and Sepi Yalda, Millersville University; Daniel A. Reed, University of North Carolina at Chapel Hill; Everette Joseph, Howard University; and V. Chandrasekar, Colorado State University. Esta publicación se realizó en noviembre del 2006[11].

El primer trabajo crea una red de sensores, colaborativa y distribuida. Mientras que el segundo ofrece un workflow dinámico y manejo de datos, en un framework con web service diseñado para soportar sistemas sobre demanda, en tiempo real y dinámicamente adaptables. La interacción de estos sistemas se muestra en la Figura 10.

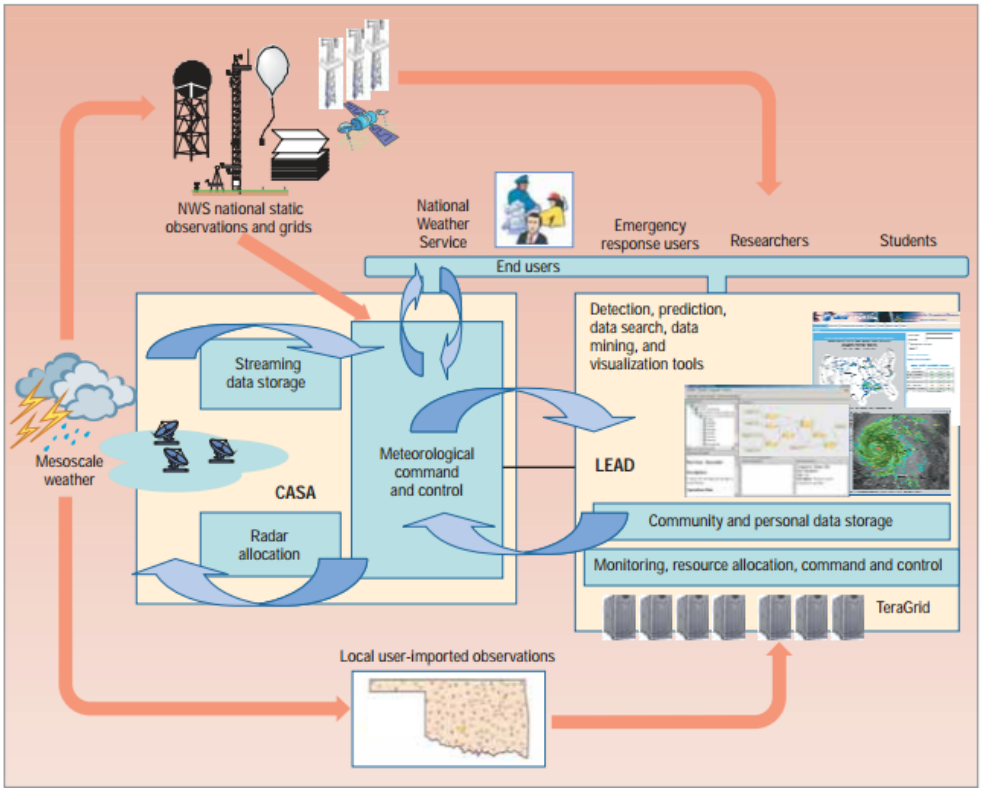


Figura 11: Interacción de los sistemas CASA y LEAD (Bethe Plale et al 2006)

2.3.4. Scalable CEP for Smart Cities

Esta investigación fue realizada por Fernando Freire Scatone de la Universidad de San Pablo[12].

Este trabajo se centra en explicar qué es CEP y qué son las ciudades inteligentes, para luego hacer una reflexión sobre cómo toda la información que es recolectada puede volverse inútil si no contamos con un procesamiento adecuado de la misma.

La propuesta que realizan es investigar todas las herramientas que ejecutan un procesamiento complejo de eventos, y evaluar el rendimiento de cada una de ellas, para luego poder seleccionar la de menor uso de recursos.

También pretende monitorear el uso de CPU, de memoria y de ancho de banda, para poder identificar si se pudiera agregar o quitar recursos a medida que el procesamiento de eventos complejos lo amerite, descubrir los requerimientos de las ciudades inteligentes que afectan la escalabilidad de los sistemas CEP y tratar de resolver esa problemática.

Otro punto para tener en cuenta es usar la disposición natural de la ciudad para mejorar el procesamiento, basado en la localización geográfica y la información relevante a las personas de ese lugar.

2.3.5. Complex event processing and data mining for smart cities

Presentado en octubre del 2012, en Eslovenia en la conferencia: Conference on Data Mining and Data Warehouses (SiKDD 2012) por Alexandra Moraru[13].

Este estudio habla de cómo se ha trabajado CEP independiente del campo de data mining. Lo que presenta este documento son los primeros pasos hacia la definición de un marco que permita una integración perfecta del CEP y data mining. Se presentan los escenarios de ciudades inteligentes como un buen campo de trabajo para la experimentación. Se discute un caso de uso concreto, Londres y se presentan resultados preliminares para los datos reales que se han recopilado.

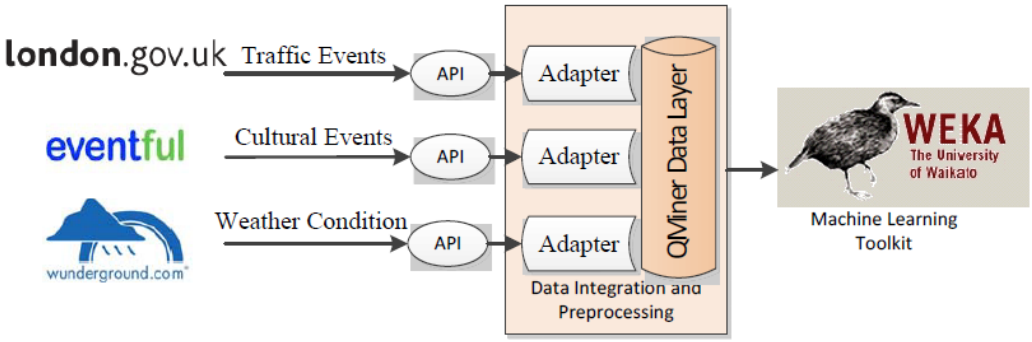


Figura 12: CEP y data mining para ciudades inteligentes (Moraru et al 2012)

2.3.6. Comparativa de estudios previos

	Dominio de aplicación	Herramienta utilizada	Tipo de proyecto
A model-driven approach for facilitating user-friendly design of complex event patterns	Genérico	Framework propio	Creación de una interfaz amigable donde el usuario pueda generar sus reglas sin depender de un dominio o de un motor CEP en específico.
Semantic Complex Event Processing for Social Media Monitoring	Redes Sociales	ETALIS, C-SPARQL, CQELS e INSTANS	Se investiga cómo a través de las redes sociales se puede inferir ciertos comportamientos con un buen proceso de la data.
CASA and LEAD	Clima	Framework realizado en proyecto LEAD	Se unieron dos proyectos anteriores. LEAD y CASA, uno de ellos se centra en sensores y el otro en manejo de la data. La unión de los dos es la que nos ofrece … sobre demanda y en tiempo real.
Scalable CEP for Smart Cities	Ciudades inteligentes	Varios motores de procesamiento de CEP	Identificar cual motor consume menos recursos
Complex event processing and data mining for smart cities	Ciudades inteligentes	WEKA toolkit	Se estudia como diferentes eventos (culturales, climáticos, etc) pueden influir en el tráfico

3. Análisis

Se analiza construir una herramienta con una interfaz gráfica amigable para los usuarios expertos de dominio, en donde se configuran patrones y acciones sin necesidad de contar con usuarios técnicos.

En este capítulo se profundiza en los requerimientos relevados, los casos de uso y diagramas de secuencia del sistema más importantes, así como también el modelo de entidad relación.

A continuación en la Figura 13, se presenta un diagrama de alto nivel con las interacciones entre los actores y el sistema:

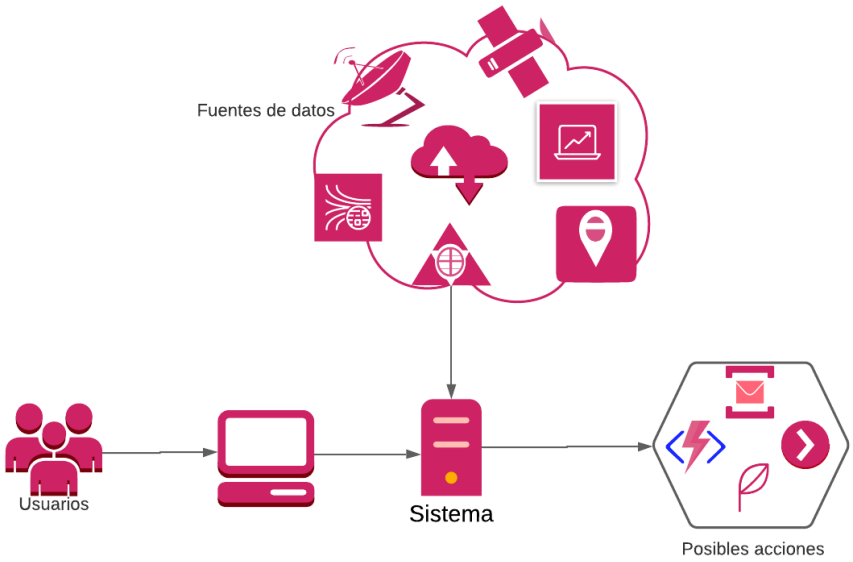


Figura 13: Diagrama de interacción entre sistemas

Los usuarios expertos de las diferentes organizaciones acceden al sistema a través de una interfaz web. El sistema les da la capacidad de crear patrones para detectar situaciones de interés. Estos patrones configurados en el sistema luego podrán ser utilizados en una herramienta de procesamiento de eventos complejos y de esta forma disparar alertas si se cumplen las reglas.

3.1. Análisis de requerimientos

En base a las necesidades planteadas como motivación de este proyecto, se define como principal objetivo, la creación de una herramienta intuitiva y simplificada, que permita a un usuario sin conocimientos de programación crear reglas o patrones que respondan a necesidades de su campo de trabajo.

Se considera importante la idea de priorizar el buen diseño del sistema ya que esto permite al usuario experto de dominio tener la posibilidad de crear patrones de manera sencilla. En esta interfaz se requiere que el usuario experto pueda configurar reglas que detecten situaciones de interés sin la necesidad de un técnico informático. Como acción a realizar cuando la situación de interés es detectada, se configuran alertas con el objetivo de tomar medidas tempranas.

Los usuarios expertos pueden tener diferentes necesidades y conocimientos, pertenecer a variadas organizaciones y requerir de diversas fuentes y tipos de datos. Para contemplar este requerimiento se decide

que la herramienta sea un sistema multi organización. Para administrar esta herramienta se define el tipo de usuario administrador. A diferencia de los usuarios expertos, los cuales pertenecen a cada organización, los administradores son transversales a las mismas, con la función de dar soporte de primer nivel a las organizaciones utilizando esta herramienta.

3.1.1. Tipos de usuario

Los tipos de usuarios requeridos son:

Usuario Administrador

Este usuario tiene un rol de administrador transversal a las organizaciones, lo que le permite visualizar las mismas y desde ahí los usuarios de cada una de ellas. De esta manera puede crear un usuario de tipo experto nuevo para la organización seleccionada y también editar los existentes. Este tipo de usuarios puede visualizar todas las reglas ingresadas por las diferentes organizaciones, los dominios con sus eventos y todas las alertas, pero no puede editar ninguna de estas entidades.

Usuario Experto

El usuario experto cuenta con el conocimiento en un dominio para definir bajo qué umbrales configurar una regla. Sin conocimientos de programación, puede generar patrones que cuando genere coincidencias en un motor CEP, disparen acciones. Este usuario puede visualizar tanto las alertas como las reglas configuradas por él o por otros usuarios expertos que pertenezcan a su organización.

3.1.2. Tipos de reglas

Se cuenta con dos tipos de reglas: reglas simples y reglas compuestas

Regla Simple

Esta regla monitorea un único evento perteneciente a un dominio definiendo los umbrales para los cuales se requiere ejecutar la alerta. La regla simple se crea por defecto como inactiva.

Regla Compuesta

Esta regla se define seleccionando dos reglas simples y agrupándolas. Como precondición, es necesario que las reglas ya estén ingresadas como simples. Lo que se inserta para estas reglas aparte de la selección de las dos reglas simples, es el nombre de la nueva regla y la alerta que será disparada si la regla se cumple. Esta regla también se crea por defecto como inactiva.

3.1.3. Requerimientos funcionales

Usuario Administrador

RA-1 Login de usuario

El sistema proporciona un login de usuarios, donde los mismos acceden al sitio ingresando nombre de usuario y contraseña.

RA-2 Listado de organizaciones
El sistema permite al usuario administrador visualizar las organizaciones creadas en el sistema.

RA-3 Listado de Usuarios
El sistema permite listar los usuarios existentes en cada organización para editar sus datos o agregar usuarios nuevos.

RA-4 Alta Usuario Experto
El sistema permite el alta de nuevos usuarios de tipo experto asociados a una organización.

RA-5 Editar datos de Usuario
El sistema permite la edición de los datos de usuarios que pueden ingresar a la herramienta.

RA-6 Listado de reglas
El sistema permite al usuario administrador visualizar todas las reglas que se van generando.

RA-7 Listado de dominios
El sistema deberá permitir al usuario administrador visualizar los diferentes dominios ingresados al sistema.

RA-8 Listado de eventos de dominio
El sistema permite al usuario administrador visualizar los eventos de un dominio.

RA-9 Listado de alertas
El sistema permite al usuario administrador visualizar todas las alertas ingresadas en el sistema.

Usuario Experto

RE-1 Login de usuario
El sistema permite un login de usuarios, donde los mismos acceden al sitio ingresando nombre de usuario y contraseña.

RE-2 Listado de tipos de alertas
El sistema permite visualizar a los usuarios expertos el listado de tipos de alertas configuradas para su organización.

RE-3 Alta tipo de alertas
El sistema permite al usuario experto el alta de un nuevo tipo de alerta para su organización.

RE-4 Listado de reglas simples
El sistema permite al usuario experto visualizar las reglas simples creadas por él o por usuarios de su organización.

RE-5 Creación de reglas simples
El sistema permite que el usuario experto configure nuevas reglas. Para esto se elige un evento de su dominio, una cantidad de tiempo y un valor. Para la regla también se configura una alerta con su respectivo mensaje.

RE-4 Listado de reglas compuestas
El sistema permite al usuario experto visualizar las reglas compuestas creadas por él o por usuarios de su organización. Al seleccionar una regla compuesta, se muestra las dos reglas simples que la componen.

RE-6 Creación de reglas compuestas
El sistema permite a partir de dos reglas simples armar reglas compuestas. Donde también el usuario ingresa el nombre de la nueva regla y la alerta a disparar en caso de cumplirse.

Alertas

Los tipos de alertas son variados, dependiendo de las necesidades de cada organización. Se podrá realizar envío de correo electrónico, envío de SMS, envío de notificaciones Push, conexiones vía Web Services, publicaciones en redes sociales, entre otras.

Dashboard

En el dashboard se permite tanto a los usuarios administradores como a los usuarios expertos de una organización, visualizar diferentes métricas.

Usuarios administradores

- cantidad de organizaciones en el sistema
- cantidad de reglas creadas totales (diferenciadas entre simples y compuestas)
- cantidad de reglas activas
- cantidad de alertas totales ejecutadas
- cantidad de reglas por organización
- cantidad de usuarios por organización

Usuarios expertos

- cantidad de reglas creadas por su organización
- cantidad de alertas ejecutadas de la organización
- cantidad de reglas creadas por usuario

Interfaz

El diseño de la interfaz web es amigable para el usuario experto, considerando que es suficiente contar con conocimientos básicos en informática.

Interfaz para usuario administrador:

- Login de usuario
- Listado de funcionalidades
- Dashboard
- Visualización de dominios
- Visualización de eventos de domino
- Visualización de organizaciones
- Visualización de usuarios de cada organización
- Creación de usuario

- Edición de usuario
- Visualización de reglas totales
- Visualización de alertas

Interfaz para usuario experto:

- Login de usuario
- Listado de funcionalidades
- Dashboard
- Visualización de reglas de la organización
- Creación de reglas simples
- Creación de reglas compuestas
- Visualización de alertas de la organización
- Creación de alertas

3.1.4. Requerimientos no funcionales

Los requerimientos no funcionales para esta herramienta son:

- Plataforma Web
- Sistema multitenant
- Los datos que se ingresan desde la interfaz web son almacenados en un servidor de base de datos.
- La interfaz de usuario es simples y amigable
- La aplicación web funciona correctamente en Firefox, Chrome e IE
- Uso de simbología y unidades adecuadas
- Responsive

3.2. Casos de uso

En esta sección se describen los diagramas de casos de uso con los requerimientos más relevantes del sistema; en ellos se muestra hacia qué componentes están dirigidas las necesidades del usuario para cada caso de uso.

Casos de uso Usuario Experto

A continuación, se presenta el diagrama de casos de uso con las funcionalidades más destacadas que el usuario experto puede llevar a cabo en la herramienta.

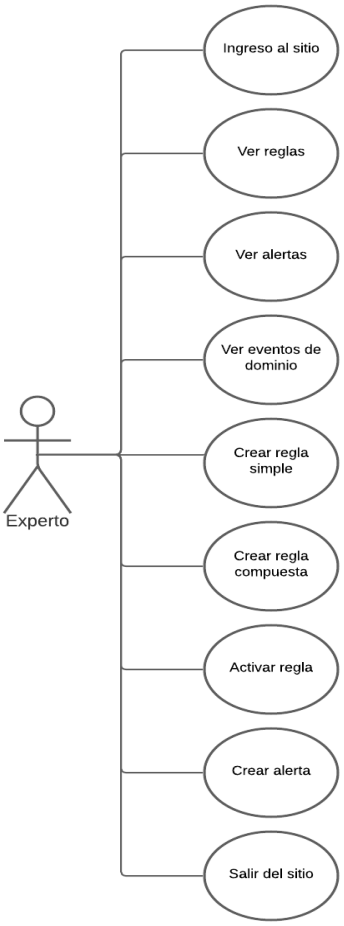


Figura 14: Casos de uso del usuario experto

CU-1 Ingreso al sitio
Para ingresar al sitio el usuario cuenta con un nombre de usuario y contraseña. Estos datos son proporcionados por el usuario administrador cuando se da de alta al usuario experto en el sistema.

CU-2 Ver reglas
El usuario experto selecciona del menú de la interfaz web la opción Reglas, donde se le muestran las reglas ingresadas por él como por los usuarios de su misma organización.

CU-3 Ver alertas
El usuario experto selecciona desde el menú de la aplicación web la opción de Alertas, donde se le listan las configuradas por él y por los usuarios de su organización.

CU-4 Ver eventos de dominio
El usuario experto ve listados los eventos de su dominio en la pantalla de creación de reglas para seleccionar el evento clave para la regla a crear.

CU-5 Crear regla simple
El usuario experto crea una regla simple desde el menú Reglas de la interfaz web. Luego que las reglas son listadas, el usuario experto selecciona un evento de dominio, el valor en la unidad que corresponda, el operador y la cantidad de tiempo. Estos valores se conocen como umbral de la regla.

Luego se selecciona una alerta y se configura el destinatario y el mensaje que tendrá esta cuando llegue a destino.

CU-6 Crear regla compuesta
El usuario experto selecciona de la lista de reglas simples ingresadas, dos de ellas para generar una nueva regla compuesta. Esta regla compuesta mantiene los umbrales de las reglas simples, pero tiene su propio nombre, así como su propia alerta. Para que la regla compuesta se cumpla, deben cumplirse las dos reglas simples simultáneamente.

CU-7 Activar regla
Una vez las reglas están creadas, tanto simples como compuestas, el usuario experto puede activarlas o inactivarlas.

CU-8 Crear alertas
Se crea una nueva alerta si el usuario experto lo necesita para configurar su regla. Para esto se debe especificar el tipo, la descripción y la configuración.

CU-9 Salir del sitio
El usuario experto sale del sitio al dirigirse al botón de salir.

Casos de uso Usuario Administrador

Se muestra a continuación el diagrama de casos de uso para usuario administrador de la herramienta.

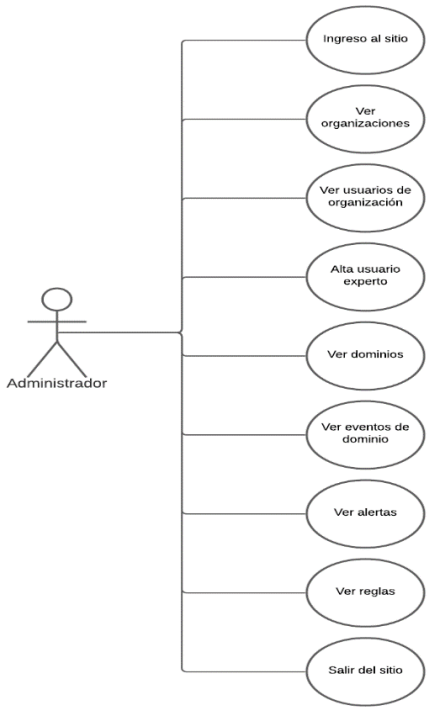


Figura 15: Diagrama de casos de uso del usuario administrador

CU-A1 Ingreso al sitio
Para ingresar al sitio el usuario cuenta con un nombre de usuario y contraseña.

CU-A2 Ver organizaciones
El administrador puede ver todas las organizaciones registradas en el sistema.

CU-A3 Ver usuarios de organización
El administrador puede ver todos los usuarios de cada organización registrada en el sistema.

CU-A4 Alta usuario experto
El usuario administrador puede crear usuarios del sistema para que utilicen los expertos al entrar a la aplicación.

CU-A5 Ver dominios
El administrador puede ver todos los dominios creados en el sistema seleccionando la opción desde la interfaz web.

CU-A6 Ver eventos de dominio
El usuario administrador selecciona desde el menú de la interfaz web ver los dominios y a partir de ahí ve los eventos de cada uno.

CU-A7 Ver alertas
El administrador puede ver las alertas creadas por todos los usuarios del sistema, no se filtrará por organización.

CU-A8 Ver reglas
El administrador puede ver las reglas creadas por todos los usuarios del sistema, no se filtra por organización. Ve las reglas simples y las compuestas. Para las compuestas también puede ver las simples que la componen.

CU-A9 Salir del sitio
El usuario administrador sale del sitio al dirigirse al botón de salir.

3.3. Diagrama de secuencia del sistema

A continuación, se muestran algunos diagramas de secuencia del sistema, los cuales modelan las interacciones que suceden durante un caso de uso particular. En especial se detallan los escenarios de creación de regla simple y compuesta.

Crear regla simple

En la siguiente figura se muestra el diagrama de secuencia del sistema cuando el usuario experto desea crear una regla simple.

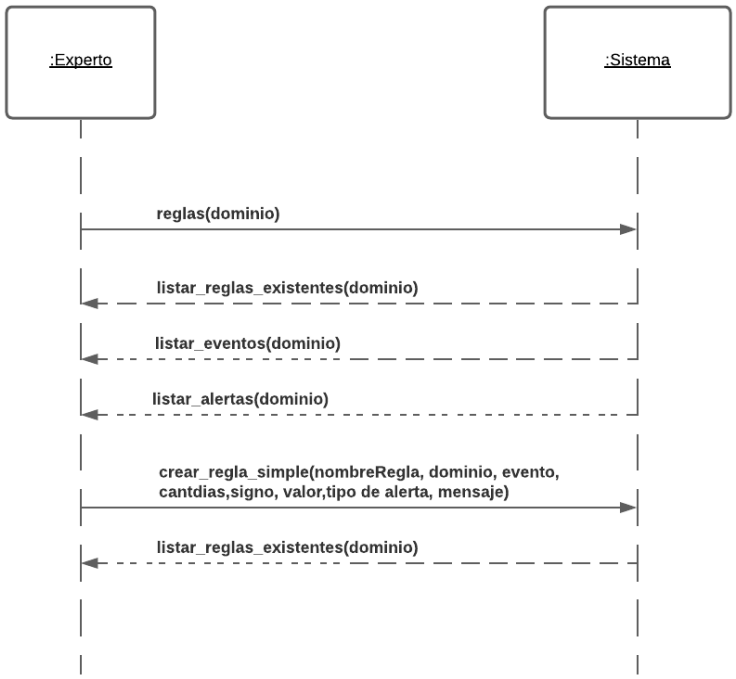


Figura 16: Diagrama de secuencia del sistema para crear regla simple

En el diagrama de la Figura 16 se presenta la interacción entre el usuario experto y el sistema durante la creación de una regla simple. El usuario experto accede a la pantalla de reglas del sistema mediante la opción de menú Reglas. El sistema lista todas las reglas existentes que pertenecen a la organización del usuario experto. También los eventos del dominio y las alertas asociados a esa organización. Luego el usuario experto configura los datos necesarios para crear una nueva regla, que son el nombre, el evento de dominio, la cantidad de días, el valor y el operador, tipo de alerta y mensaje de esta. Cuando le llega al sistema los datos de la regla a crear, este devuelve nuevamente la lista de reglas existentes incluyendo la nueva.

Crear regla compuesta

A continuación, se detalla el diagrama de secuencia del sistema cuando el usuario desea crear una regla compuesta.

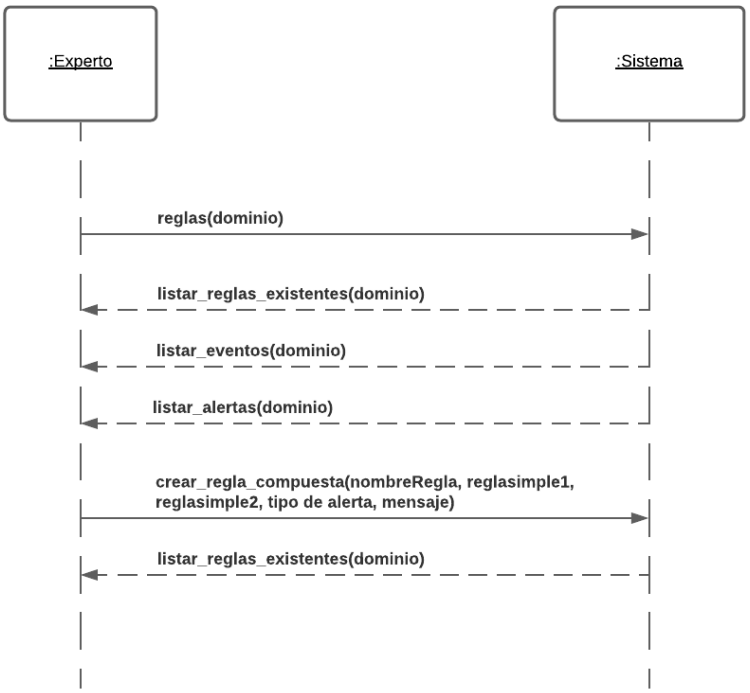


Figura 17: Diagrama de secuencia del sistema para crear regla compuesta

En el diagrama de la Figura 17, se presentan los pasos entre el usuario experto y el sistema, para crear una regla compuesta. Primero el usuario experto accede a la opción Reglas del menú. El sistema le presenta todas las reglas simples y compuestas creadas hasta el momento, los eventos de dominio y las alertas asociados a su organización. El usuario experto, para crear una regla compuesta debe ingresar un nombre, dos reglas simples del listado de reglas simples existentes y el tipo y mensaje de la alerta. Una vez completados estos datos, el sistema devuelve el listado de reglas existente de su organización, con la nueva regla compuesta incluida.

3.4. Modelo de dominio

Primero se presenta el modelo conceptual, donde se capturan las clases más importantes para la herramienta planteada.

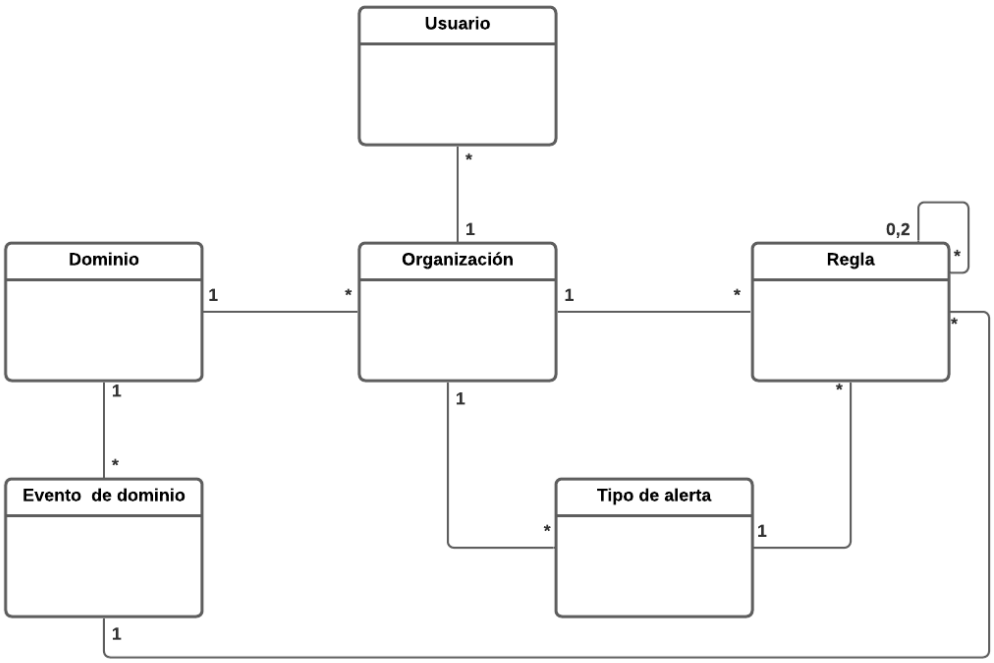


Figura 18: Modelo conceptual

En la Figura 18 se presenta el modelo conceptual. El usuario pertenece a una única organización y cada organización puede tener n usuarios. La organización está asociada a un único dominio y los dominios tienen n eventos de dominio. Cada regla tiene un único evento de dominio, un único tipo de alerta y una única organización. Si es una regla compuesta, tiene a su vez dos reglas simples asociadas. El tipo de alerta asociado a una regla pertenece al conjunto de tipos de alertas asociados a la organización de la regla.

A continuación, en el modelo de dominio se representan de forma visual los tipos de objetos más importantes en el contexto del sistema. Se utiliza la notación Unified Modeling Language (UML), el modelo de dominio se representa con un conjunto de entidades en los que no se define ninguna operación.

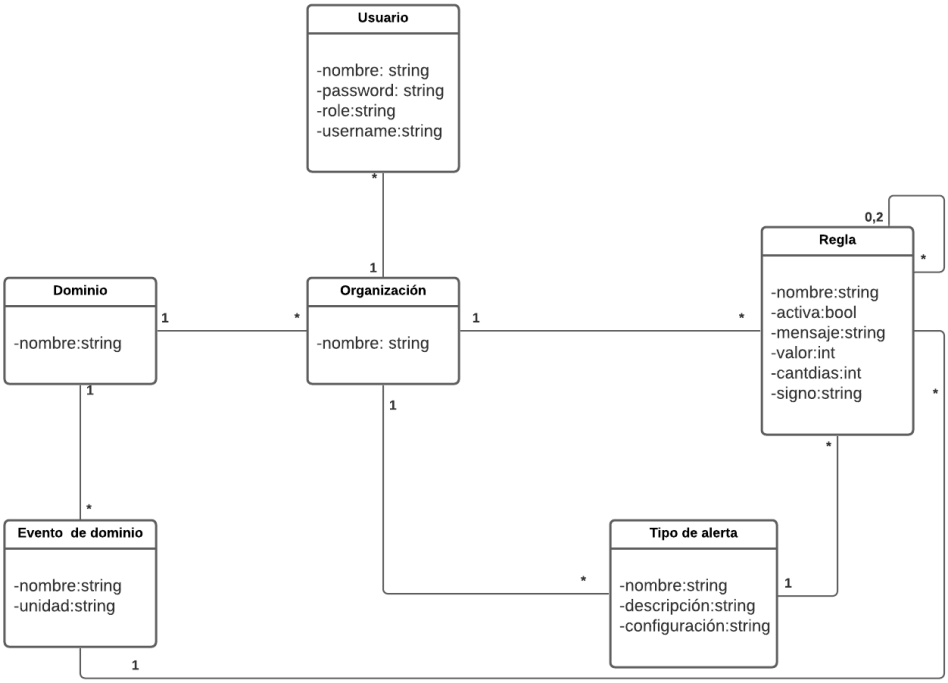


Figura 19: Modelo de dominio

C1 – Usuario

La clase Usuario representa tanto los usuarios expertos como los administradores. A los usuarios expertos, los da de alta un usuario administrador el cual ingresa los datos de este como también la organización a la que pertenecen. Los usuarios administradores son creados por base de datos.

C2 – Organización

Esta clase modela la organización a la que pertenece cada usuario. Contando con diferentes organizaciones estaremos dando a la herramienta más escalabilidad ya que se puede proporcionar la misma para muchas organizaciones, pero siendo el mismo sistema.

C3 - Dominio

La clase dominio modela los diferentes dominios que el sistema contempla.

C4 - Evento de dominio

La clase evento de dominio modela los diferentes fenómenos que el sistema contempla.

Los atributos más importantes de esta clase son el nombre del evento y la unidad con la que se mide sus ocurrencias.

C5 - Regla

Las reglas son las configuradas por el usuario experto. Las mismas están asociadas a un evento de dominio, a una organización y a un tipo de alerta. Para crear una nueva regla, el usuario configura dependiendo el evento que haya seleccionado, el valor que le agrega a las ocurrencias y la cantidad de días. Por otro lado, configura la alerta asociada y el mensaje. Las reglas se crean inactivas por defecto, pero se pueden activar posteriormente. Si el usuario experto quiere crear una regla más compleja, debe seleccionar dos reglas simples ya ingresadas y configurarles solamente la alerta y el nombre. Con esto está creando una regla

compuesta.

C6 – Tipo de alerta

Para cada regla se puede crear una nueva alerta o utilizar las ingresadas de la lista que se presenta. Las alertas pertenecen a una organización. Luego, se ingresa la configuración deseada (dirección de email, nro de teléfono, endpoint, etc).

4. Diseño y Arquitectura

Esta sección proporciona una apreciación global y comprensible de la arquitectura del sistema usando diferentes puntos de vista para mostrar distintos aspectos de este. Intenta capturar y llegar a las decisiones de arquitectura críticas para diseñar la herramienta. A medida que transcurre el capítulo, se detallan en profundidad las diferentes etapas.

Como primera decisión de diseño, se realizará una arquitectura cliente-servidor en 3 capas. Se optó por este diseño, ya que es una arquitectura típica para el tipo de herramienta a construir. Con respecto al diseño de la base de datos, se utiliza una arquitectura Multi-Tenant, ya que uno de los requerimientos es que el sistema sea multi organización. En este enfoque todos los usuarios comparten el mismo conjunto de tablas y un identificador de cada tenant. Se elige este enfoque de esquema compartido ya que tiene los más bajos costos de hardware y de copia de seguridad, dado que le permite servir al mayor número de clientes por base de datos del servidor.

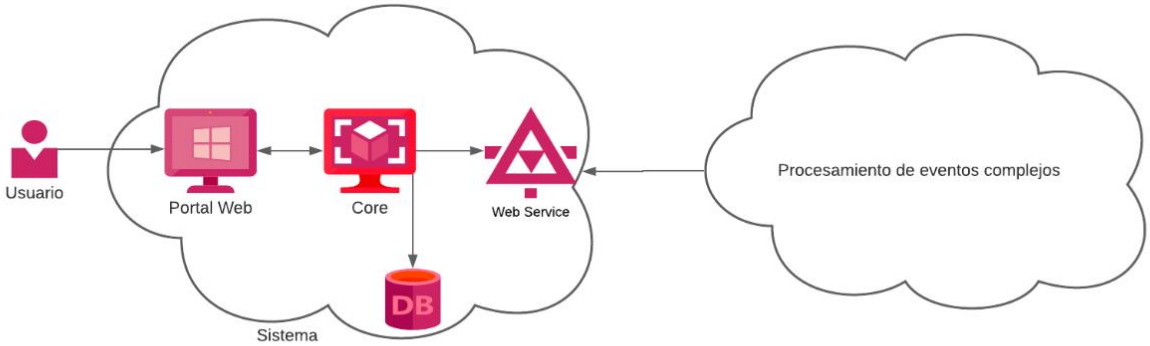


Figura 20: Arquitectura general de sistema

El diagrama de la Figura 20 presenta la arquitectura del sistema. El usuario experto accede al sistema mediante el portal web, el cual se comunica con el componente de negocio y persistencia. Se expone un web service que retorna las reglas configuradas en el sistema. El componente de procesamiento de eventos complejos es externo al sistema y puede ser integrado a cualquiera de los productos CEP del mercado.

4.1. Casos de uso relevantes a la arquitectura

Para definir la arquitectura, se toma como insumo el modelo de casos de uso presentado en el capítulo 3, para identificar requerimientos funcionales claves y requerimientos no funcionales asociados. Los requerimientos funcionales claves, son aquellos que representan las funcionalidades más importantes que deben ser proporcionadas por la herramienta. Para este diseño, se tomaron los siguientes casos de uso:

- Crear usuario experto
- Crear regla simple

Crear usuario experto

Se crea un usuario que será el experto capaz de configurar las reglas con sus respectivas alertas y eventos de dominio. Para crear este usuario, aparte de sus datos, se requerirá agregar la organización a la que pertenece. La organización, al ser creada, tiene asociado un dominio con sus respectivos eventos de dominio. De esta manera, las reglas, las alertas y los usuarios que este nuevo experto verá, son los de su organización.

Crear regla simple

Configura una nueva regla simple a partir de los eventos de dominio de la organización a la que pertenece el usuario experto y declarando una cantidad de tiempo y un umbral de valor para el disparo de la alerta. También se configura el tipo de alerta que se va a disparar con su respectivo mensaje.

4.2. Diagrama arquitectura en capas

Como se indicó en la introducción del capítulo, se selecciona la arquitectura básica de tres capas ya que este sistema deberá conectarse a la interfaz gráfica de usuario y a un mecanismo de almacenamiento persistente de datos, aislando de esta forma la capa de lógica de la aplicación. Las tres capas de esta arquitectura son la capa de presentación, la capa de negocios y la capa de acceso a datos. A continuación, se presenta un diagrama y un detalle de las capas y los paquetes contenidos en cada una de ellas.

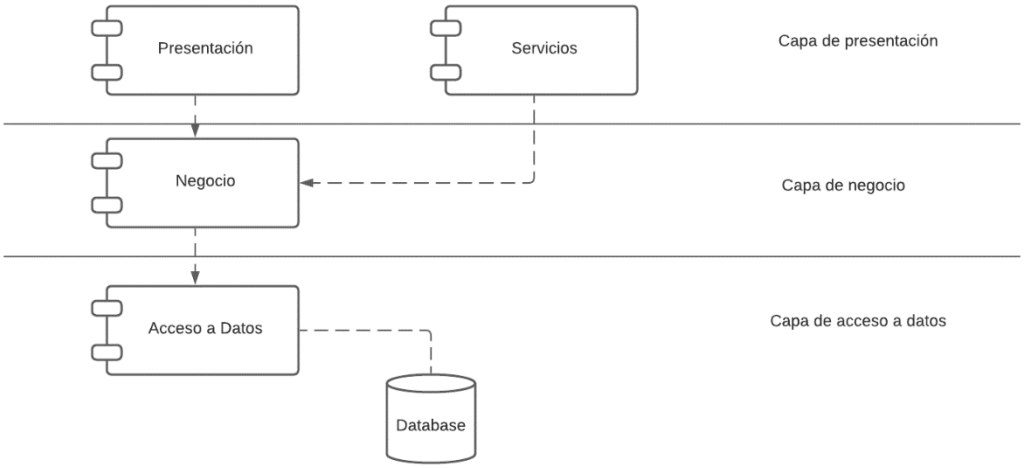


Figura 21: Diagrama de arquitectura en capas

Capa de presentación

En esta capa se encuentra la interfaz gráfica web, que es donde los usuarios expertos y administradores gestionan las reglas y alertas, así como los usuarios. Es la encargada de la interacción del usuario final con el sistema. También en esta capa se encuentra la exposición de servicios.

Capa de negocio

En esta capa se encuentran tanto las entidades, como la lógica de negocio. Este subsistema se encarga principalmente de proveer todos los datos necesarios para la capa de presentación y para los servicios web. Tiene toda la lógica asociada a la aplicación.

Capa de acceso a datos

La capa de datos se encarga de almacenar la información que se ingresan desde la interfaz web. Los componentes DAOs (Data Access Objects), se alojan en la capa de datos, pero son compartidos por la capa de negocios cumpliendo la tarea de transportar información.

4.3. Descomposición en subsistemas

Un sistema se estructura en varios subsistemas principales, los cuales son unidades de software independientes que interactúan entre sí. En esta sección detallaremos la descomposición en subsistemas. Un subsistema es un sistema en sí mismo, se componen de módulos y tienen interfaces definidas que se usan para comunicarse con otros subsistemas. Un módulo es parte de un subsistema cuya función es brindar servicios a otros módulos y a su vez este recibe los servicios proporcionados por otros módulos.

El siguiente diagrama de subsistemas define diferentes niveles de módulos, los módulos de un mismo nivel tienen responsabilidad de abstracción similar. Los módulos de un nivel están para atender los pedidos de los subsistemas con los que interactúa.

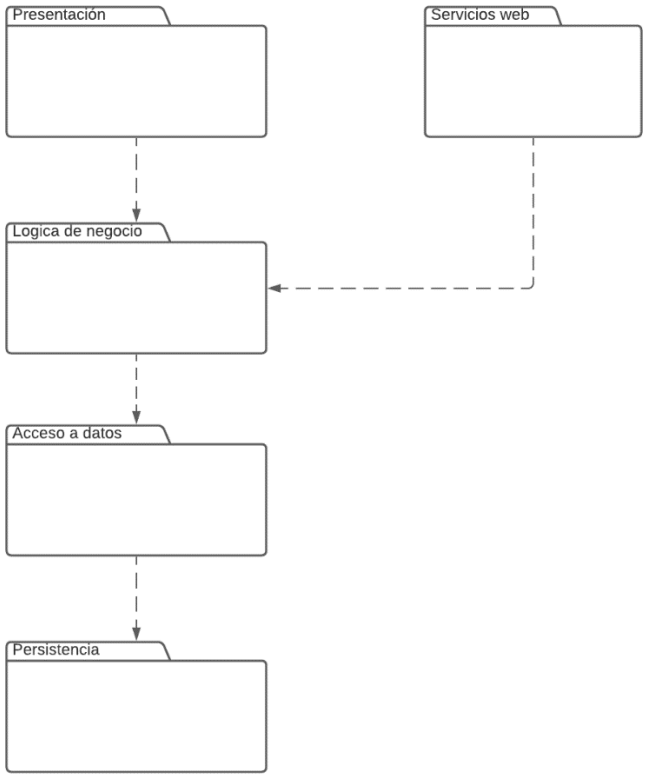


Figura 22: Diagrama de subsistemas

En la Figura 22 se presenta la forma en que los diferentes subsistemas interactúan entre sí. En el primer nivel tenemos el módulo de presentación y de servicios web, ambos módulos obtienen información del nivel inmediato inferior donde se aloja la lógico de negocios. La lógica de negocios se comunica con el nivel inmediato inferior que es donde está el acceso a datos y finalmente el último nivel es la persistencia.

4.4. Diagrama de despliegue

Se presenta a continuación el diagrama de despliegue general del sistema con la siguiente descripción de cada nodo.

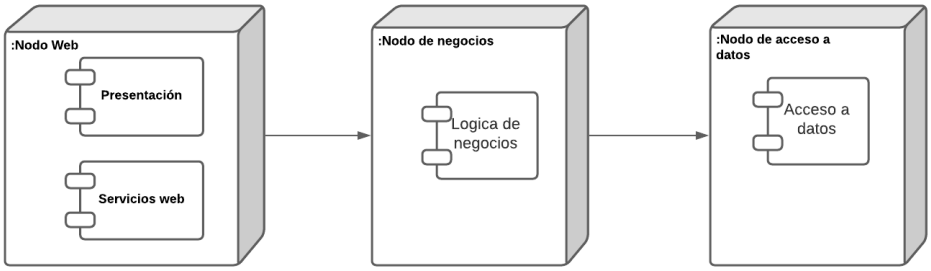


Figura 23: Diagrama de despliegue general del sistema

Nodos
A continuación, se explica el contenido de cada nodo y la tecnología que se utiliza en cada uno.

Nodo de acceso a datos
Este servidor es el encargado de alojar los datos del sistema y prestar los servicios que actúan o consultan sobre los mismos. Se accede al mismo por medio del servidor de aplicaciones. Se utilizará un motor de base de datos.

Nodo de negocios
Este nodo es el principal intermediario entre la interfaz de usuario y los datos del sistema. El mismo contiene la mayor parte de la lógica de negocio de este. En este servidor se aloja toda la lógica de negocio referente a usuarios, dominios, reglas y alertas.

Nodo Web
Este nodo aloja la publicación web, que es el punto de entrada de los usuarios a través del navegador web. Este a su vez se comunica con el servidor de aplicaciones a través de http. En este nodo también se exponen los servicios web.

4.5. Dominio

En esta sección se describen las decisiones tomadas para hacer el diseño de la base de datos. Uno de los puntos a destacar, es que al ser esta una aplicación multi tenant, se creó una tabla para persistir las organizaciones y asociarlos a los usuarios correspondientes del mismo.

4.5.1. Entidades identificadas

Las entidades que conforman este sistema son: las reglas, los dominios, los eventos de dominio, los usuarios, las organizaciones y las alertas. A continuación, se detallan los atributos principales identificados en esta etapa del diseño. A partir de este punto a las organizaciones le llamaremos tenant.

- Las reglas están identificadas por un id y cuentan con un identificador de tenant. Tienen un nombre, un operador, un valor que es el umbral configurado por el usuario experto desde el portal web y la

cantidad de tiempo que se va a contabilizar para cumplir la regla. Por otro lado, interesa saber el evento de dominio al que hace referencia la regla y el tipo de alerta que se debe disparar. También necesitamos saber si la regla es simple, o es compuesta y si el valor seteado es acumulado o es diario.

- El dominio se identifica por un id, y como atributos tienen el id del tenant para el cual es creado y un nombre.
- Del evento, se conoce el id, que lo identifica, el nombre y la unidad en la que se mide, así como también el dominio al que pertenecen.
- Los usuarios están identificados por un id y por el id del tenant al cual pertenecen, tienen también un nombre y un rol, que puede ser administrador o experto. Adicionalmente, se almacena el nombre de usuario y la contraseña para acceder al portal.
- El tenant modela la organización, y tiene un id que lo identifica y un nombre.
- Los tipos de alertas están identificados por un id y un id de regla, también tienen un nombre, una descripción, un tenant y la configuración para dispararla.

4.5.2. Modelo de Entidad - Relación

El diagrama siguiente modela la relación entre las entidades y los atributos, destacando la clave primaria de cada una de ellas.

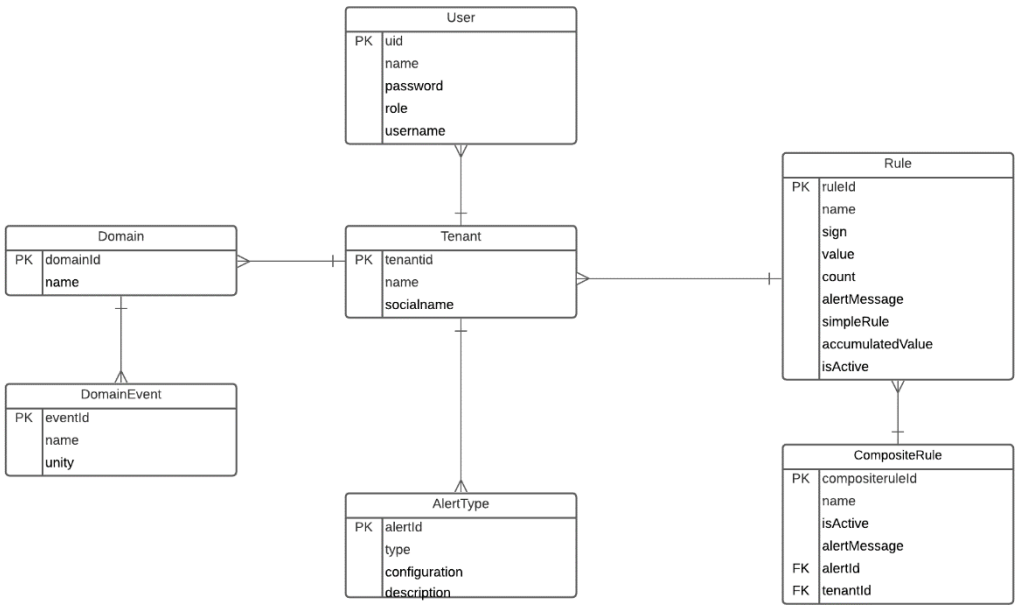


Figura 24:Diagrama Entidades

Observaciones

A continuación, se explican las restricciones del diagrama.

Unicidad

- Una regla se identifica por su nombre, a pesar de tener un id de sistema que será único también.

- Los demás conceptos modelados que sean únicos se identificarán por su id correspondiente

Dependencias circulares
El tipo de alarma de una regla debe pertenecer al mismo tenant que la regla.
El evento de dominio de una regla debe pertenecer al mismo dominio que el tenant de la regla.

Entidades

Nombre	User
Descripción	Todas las personas que acceden al portal web.

Nombre	Tenant
Descripción	Cada organización que utiliza la herramienta cuenta con un tenant para que se puedan identificar sus usuarios.

Nombre	DomainEvent
Descripción	Entidad que representa los principales eventos que componen la regla y pertenecen a un dominio.

Nombre	Domain
Descripción	Se representa con esta entidad, los dominios a los cuales van a pertenecer los eventos que componen la regla como las organizaciones.

Nombre	Rule
Descripción	Representa las reglas simples creadas por el usuario experto desde el portal web.

Nombre	CompositeRule
Descripción	Representa las reglas compuestas creadas por el usuario experto desde el portal web. Se conforma por dos reglas simples.

Nombre	AlertType
Descripción	Detalla los diferentes tipos de alarmas que la plataforma soporta

5. Implementación

En el marco de este proyecto se desarrolla un prototipo reducido como prueba de concepto. En esta sección de la documentación, se describe el proceso de implementación del prototipo de la herramienta propuesta en este proyecto que llamaremos Easy Alerts.

Partiendo de la arquitectura definida en el capítulo anterior, se seleccionan las tecnologías a utilizar, así como los supuestos y restricciones tomadas para este prototipo en particular. En el capítulo 2 de este informe se presentó la investigación realizada de algunas herramientas de CEP. La herramienta elegida para implementar este prototipo es Drools. La selección de las restantes herramientas para implementar este prototipo se basó en la utilización de tecnologías ya conocidas. Al finalizar el capítulo se describen las dificultades encontradas durante esta implementación.

5.1. Decisiones tomadas

A continuación, se realiza un resumen de las decisiones tomadas para la creación del prototipo con foco en la interfaz amigable para el usuario experto.

- Reglas simples
  - Los operadores proporcionados al usuario experto son “>”, ”>=”, ”<”, ”<=”, ”=”
- Reglas compuestas
  - Se compone únicamente de 2 reglas simples
- Alertas
  - Se enviarán alertas a través de emails, SMS y mensajes simulando la invocación a APIs
- Fuentes de datos
  - Se simula el consumo de eventos de dominio a través de archivos de texto
- Formato de datos
  - El formato de datos es lista de listas, con los valores nombre de evento, media, fecha.
- Periodicidad
  - Se toma para este prototipo como unidad de tiempo el día para la creación de reglas y la obtención de los datos.
- Creación de entidades
  - Tanto las organizaciones como los dominios y los eventos de dominio se crean por base de datos.

5.2. Easy Alerts

Para la implementación de la interfaz gráfica del portal “Easy Alerts”, se decidió desarrollar un proyecto Web utilizando la tecnología .Net Framework 4.5. Se realizó un proyecto del tipo ASP.Net Web Application para la capa de presentación, mientras que la capa de servicios se realizó con una implementación del tipo WCF Service Application bajo el protocolo SOAP. El lenguaje utilizado es C# con ASP para la presentación.

La estructura del proyecto y el contenido de cada carpeta es el siguiente:

- Contenido: en este directorio se encuentran las páginas web.
- Dominio: aquí están definidas las clases de las entidades, en este caso alertas, usuarios, eventos de dominio, dominios, organizaciones y reglas. También en el mismo directorio, se encuentra la clase Fachada que se utilizó para centralizar todos los llamados al nivel de persistencia.
- Persistencia: es donde se alojan los métodos que hacen las consultas a la base de datos.

En esta solución de .Net, es donde también se expone el web service que obtiene las reglas ingresadas por los usuarios expertos en el portal web, desde la base de datos y es consumido por el adaptador realizado para ser interpretado por el motor de reglas de Drools.

Portal Web

Se desarrolló el portal web para que el usuario experto tenga una interfaz amigable en donde configurar las reglas a procesar por el motor. Este portal es responsive para poder ser accedido desde el navegador de cualquier dispositivo. Para el diseño se utilizó Boostrap, un framework ampliamente utilizado con el cual los usuarios están familiarizados. Se prestó especial atención a la utilización de colores, la ubicación de los botones y menús como también a la disposición de los elementos de interés dentro del sitio.
A continuación, se presenta una descripción con sus respectivas imágenes de las diferentes pantallas y usos del portal.

Login

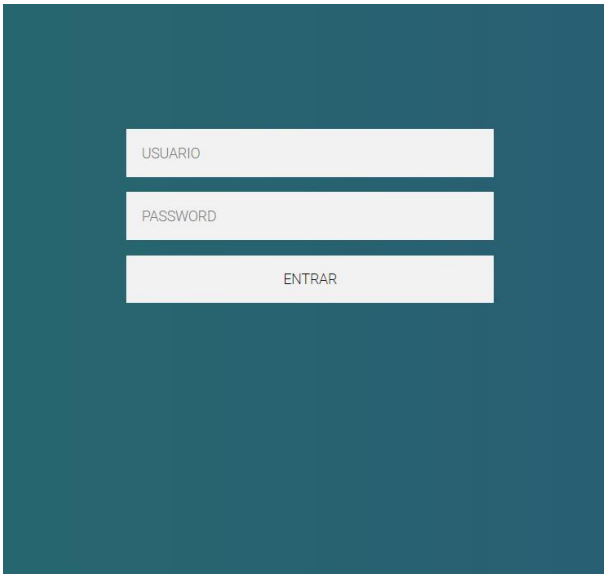


Figura 25: Login de la aplicación

El usuario tanto administrador como experto accede al portal a través de un login que le solicita su nombre de usuario y contraseña.

Pantalla principal

La plataforma Easy Alerts tiene una interfaz orientada a facilitar al usuario el acceso a las funcionalidades. Para esto, las siguientes secciones están disponibles:

- Barra superior
- Menú

- Dashboard

Barra superior

- Logo: Se presenta el logo de Easy Alerts
- Menú: con esta opción se colapsa o despliega el menú
- Perfil de usuario: Se presenta el nombre del usuario logueado así como su tipo (Administrador, Experto)
- Cerrar sesión: presionando el ícono azul, el usuario se desloguea del sistema.

Menú

- Dashboard
- En el caso del usuario Administrador verá las siguientes opciones
    - Dominios
    - Reglas
    - Organizaciones
    - Alertas

- En el caso del usuario Experto se presentan las siguientes
    - Reglas
    - Alertas

Dashboard

- Administrador
    - Total de reglas ingresadas en todo el sistema
    - Total de usuarios del sistema
    - Total de organizaciones del sistema
    - Distribución de reglas por organización
    - Actividad de reglas por organización

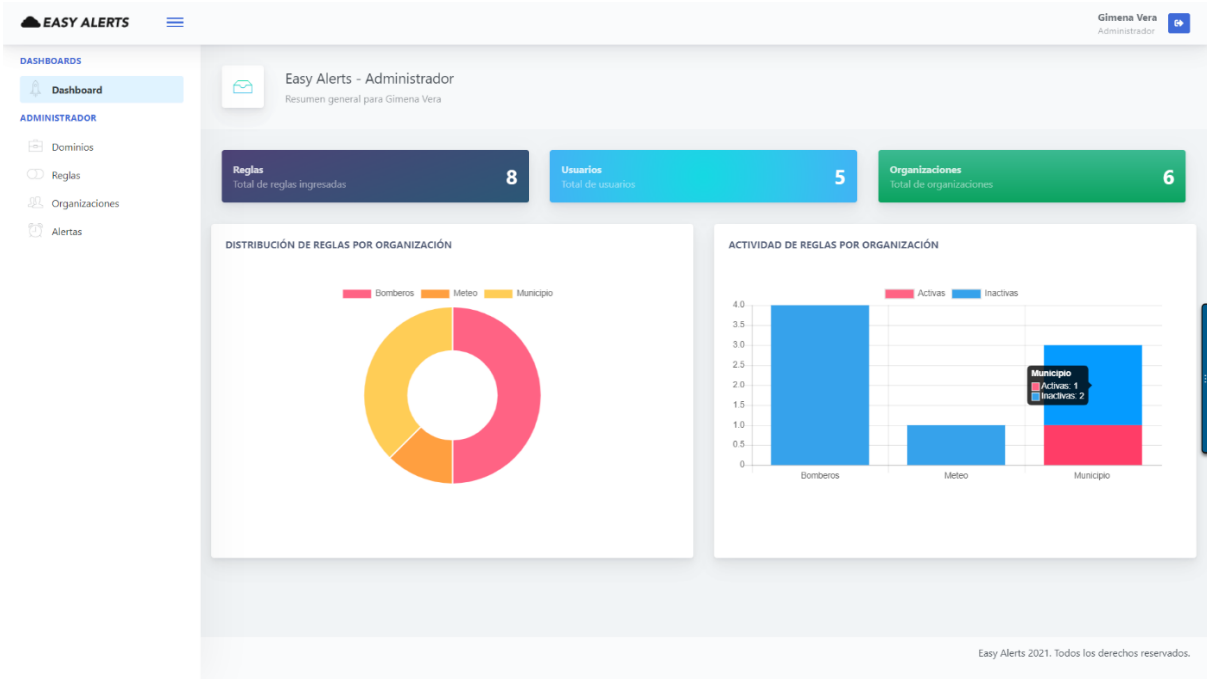


Figura 26: Dashboard administrador

- Experto
  - Total de reglas ingresadas por su organización
  - Total de reglas activas de su organización
  - Total de usuarios de su organización
  - Distribución de los distintos tipos de alertas utilizados en las reglas creadas por su organización
  - Actividad de reglas por tipo de alerta de su organización

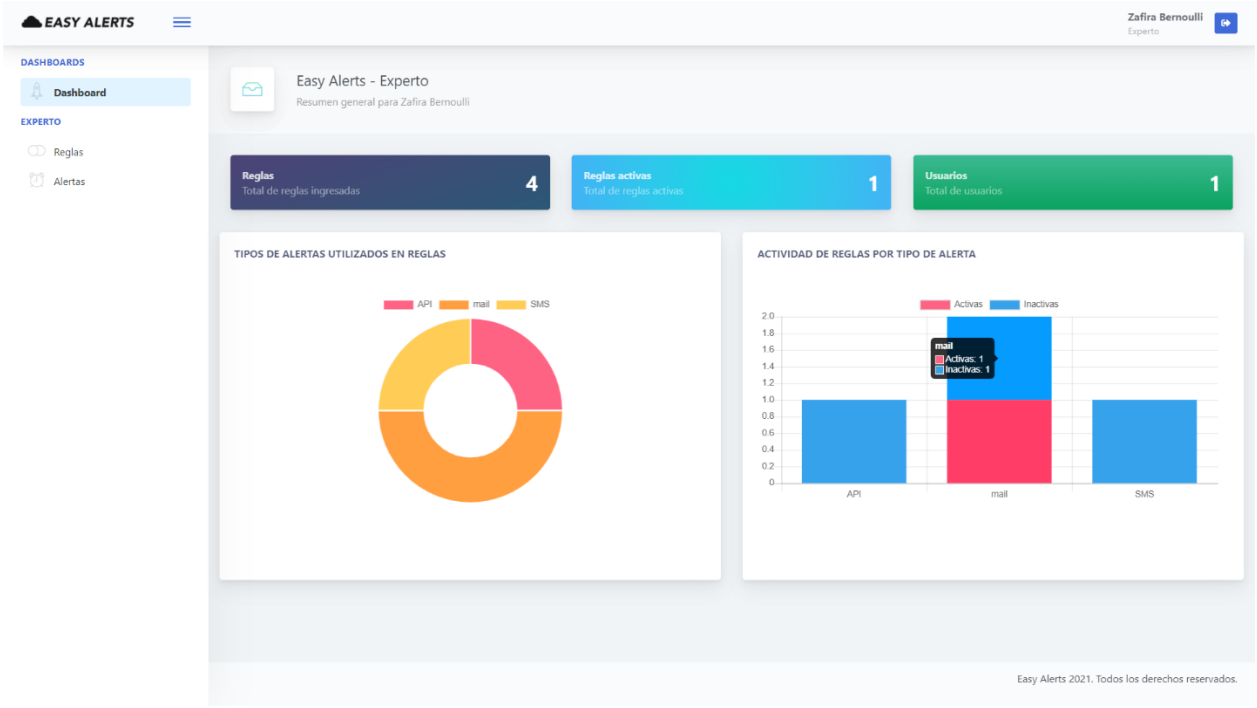


Figura 27: Dashboard experto

El sistema es responsive por lo que puede ser accedido desde cualquier navegador tanto de pc, Tablet o Smartphone. En la siguiente imagen, se muestra la adaptabilidad del sistema a diferentes resoluciones de pantalla de dispositivos móviles.

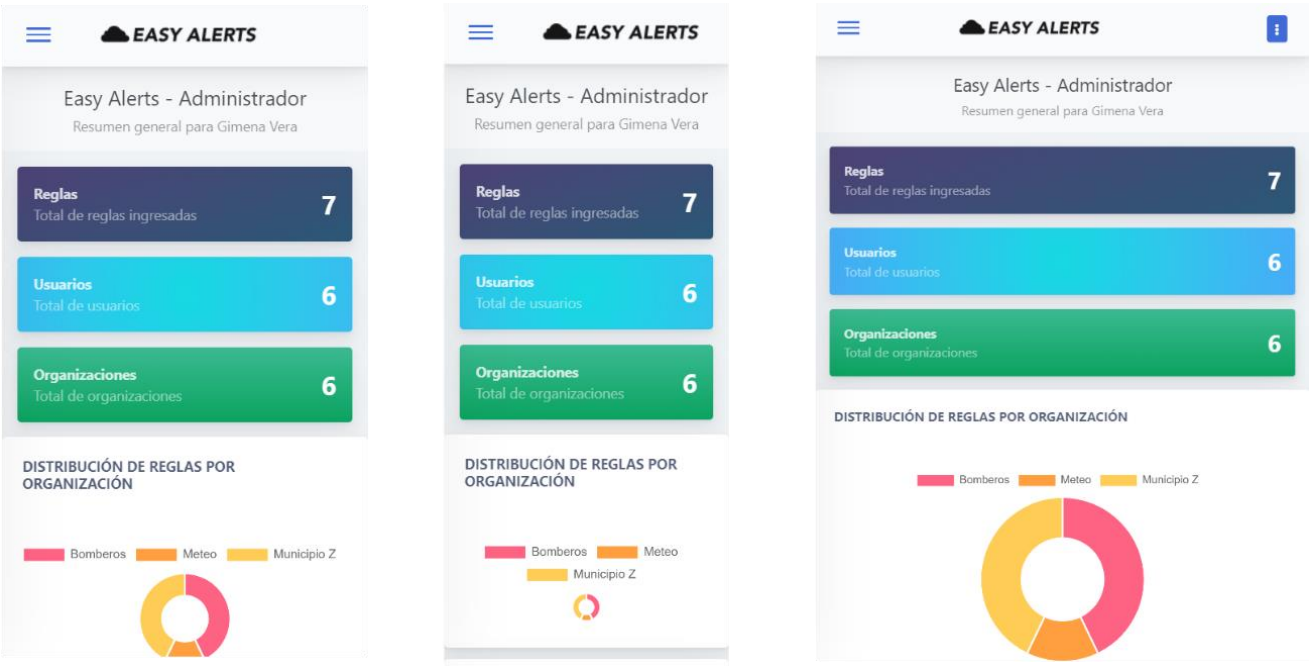


Figura 28: Vistas del dashboard en diferentes dispositivos

<u>Funcionalidades para usuario Administrador</u>

<u>Dominios</u>

Se le presenta al administrador el listado de los dominios existentes en el sistema y una opción para ver los eventos de cada uno de ellos.

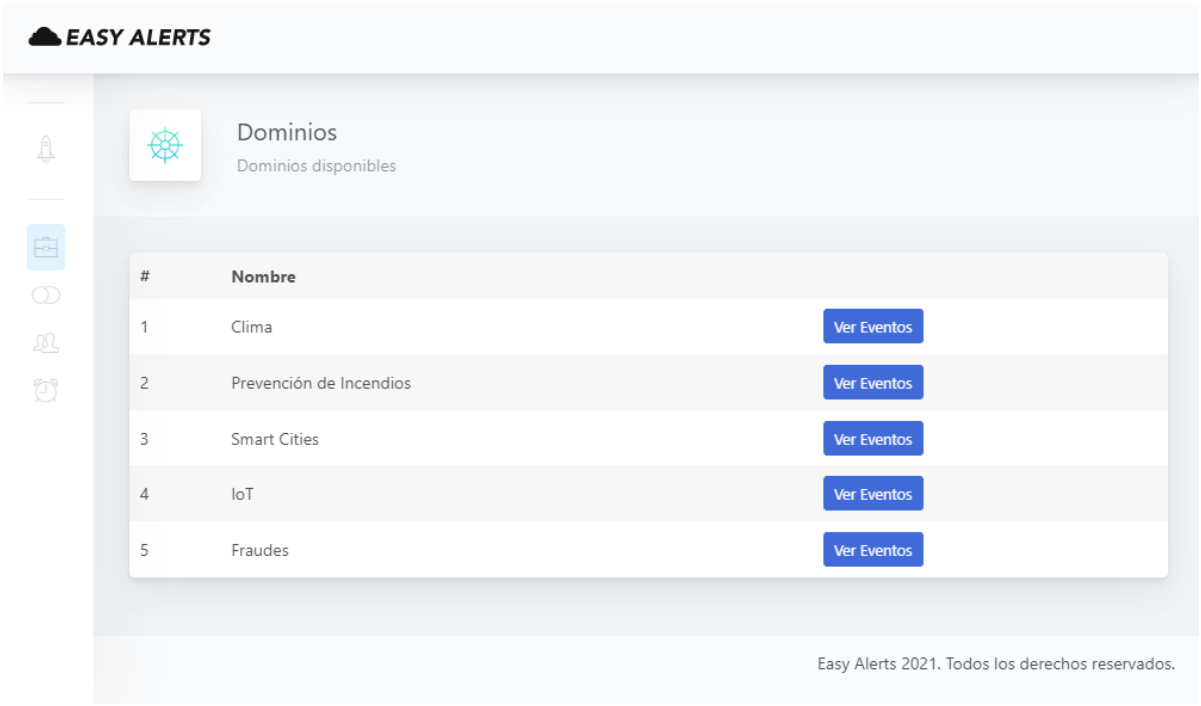


Figura 29:Listado de dominios

Reglas

El usuario administrador puede ver todas las reglas ingresadas en el sistema por usuarios de cualquier organización. Se listan tanto las reglas creadas simples como compuestas. El administrador no puede cambiar la activación de estas ni crear nuevas reglas de ningún tipo.

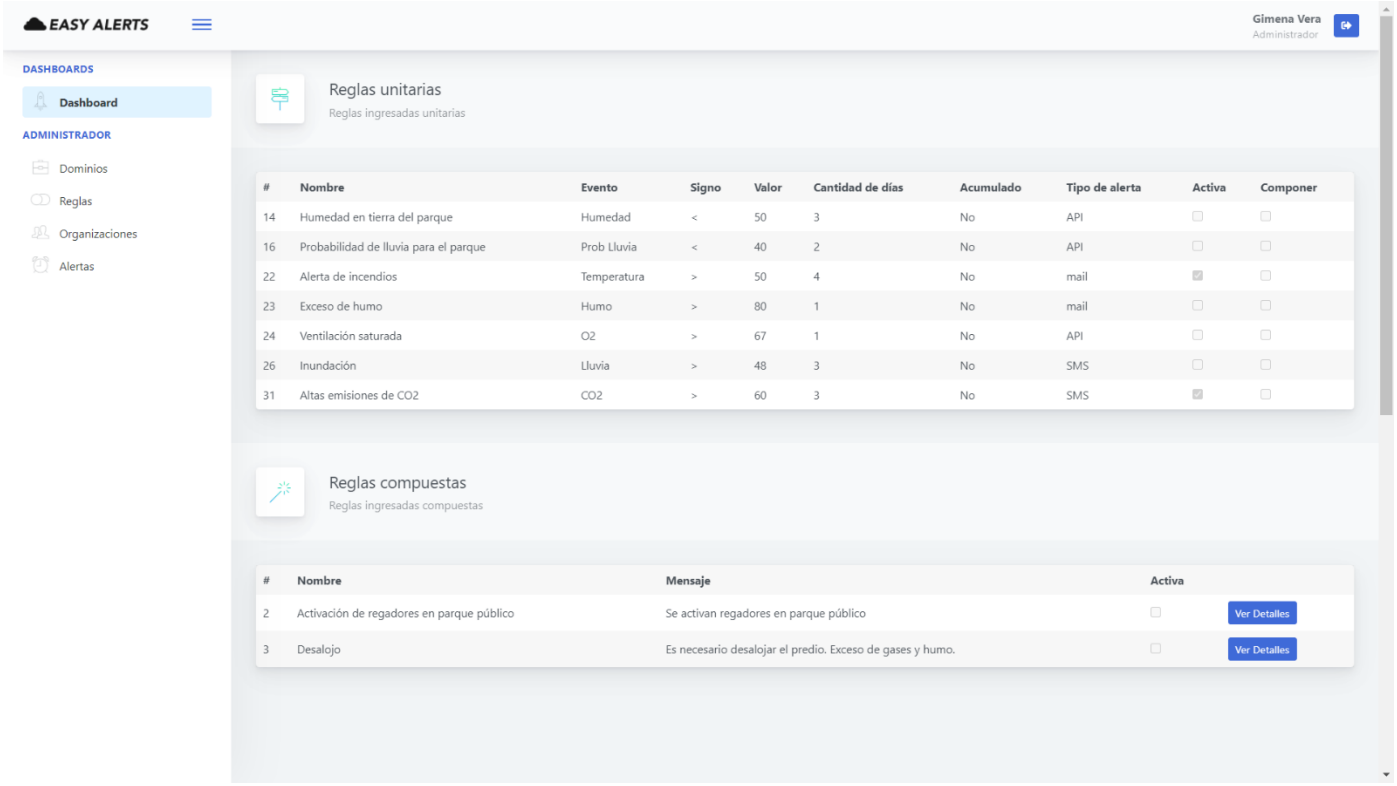


Figura 30: Listado de reglas

En el listado de reglas compuestas se encuentra una opción para ver cuales reglas simples la componen.

Alertas

En el listado de tipos de alarmas disponibles se puede ver la descripción de cada una, así como la configuración y a la organización a la cual pertenecen.

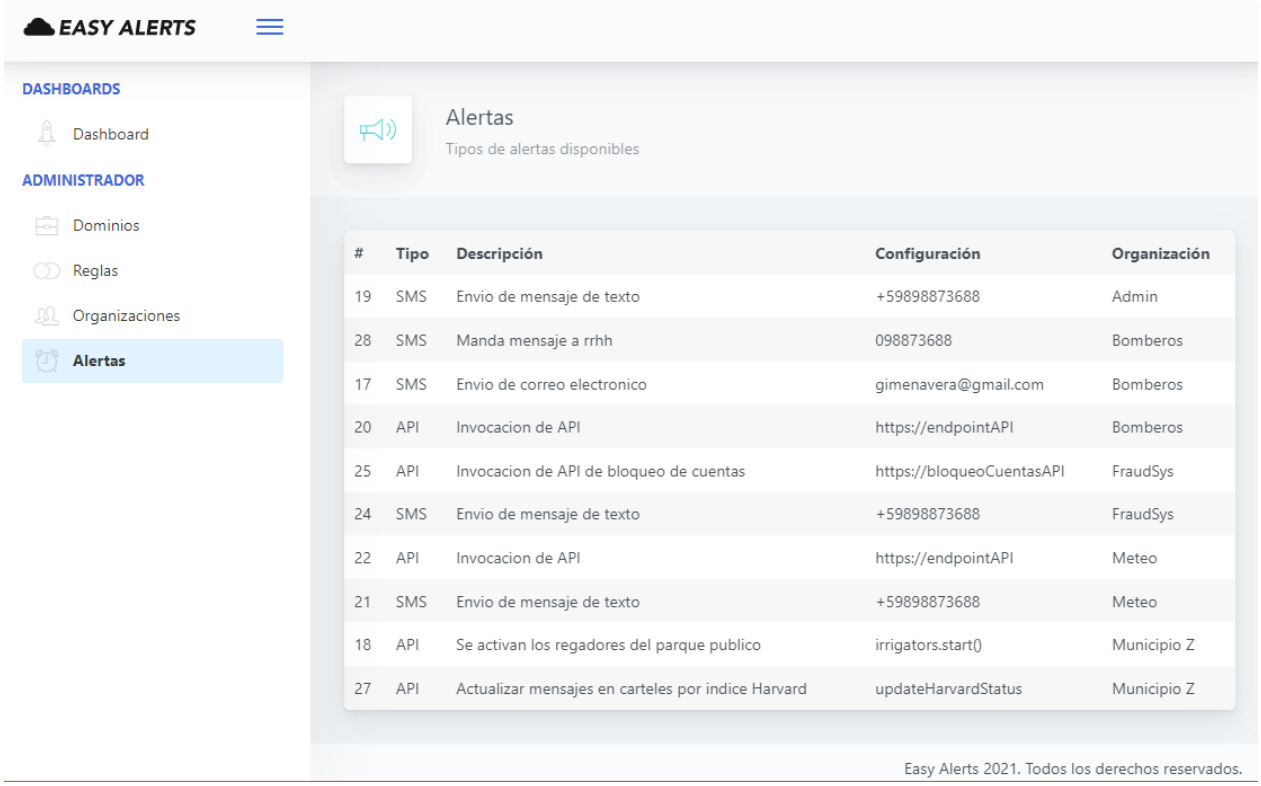


Figura 31: Listado de alertas

Organizaciones

Bajo esta opción se listan las diferentes organizaciones operando con Easy Alerts. Cada una de ellas presenta en el listado un acceso para administrar los usuarios de cada una.

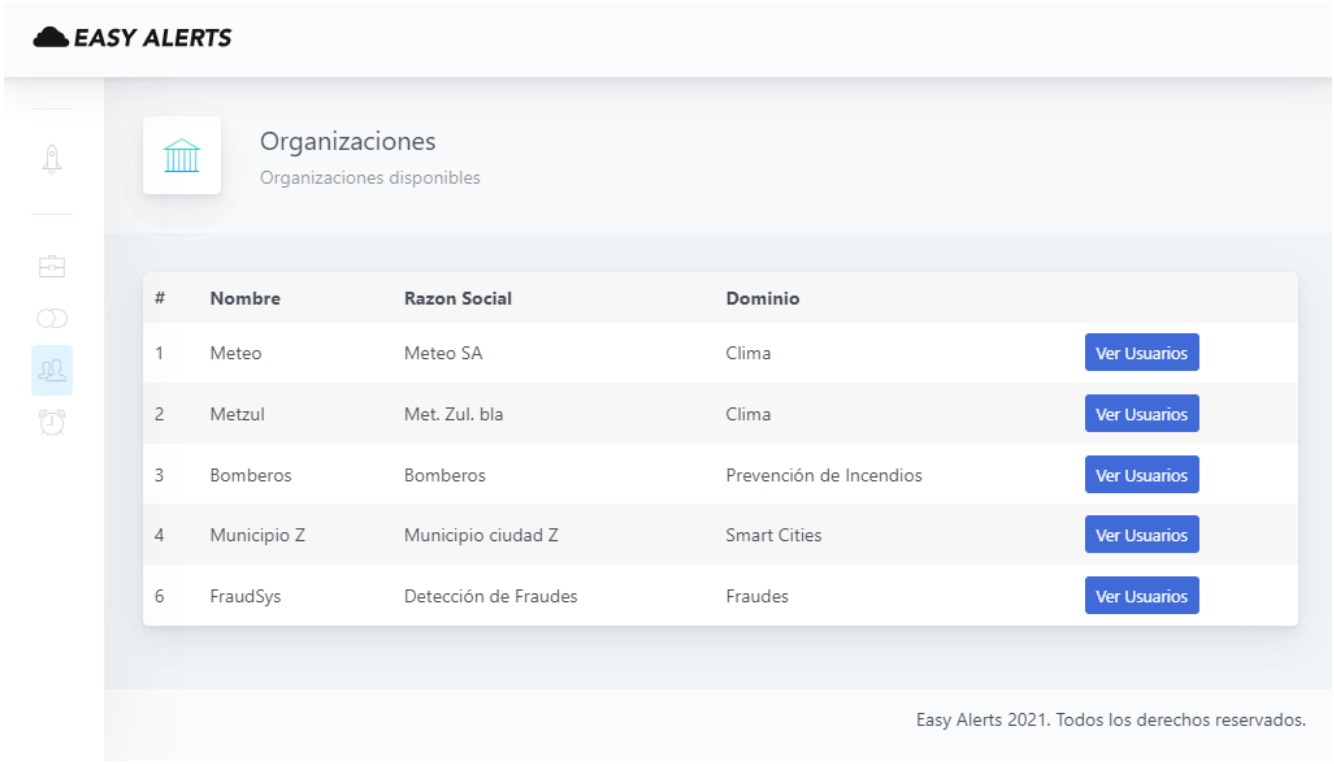
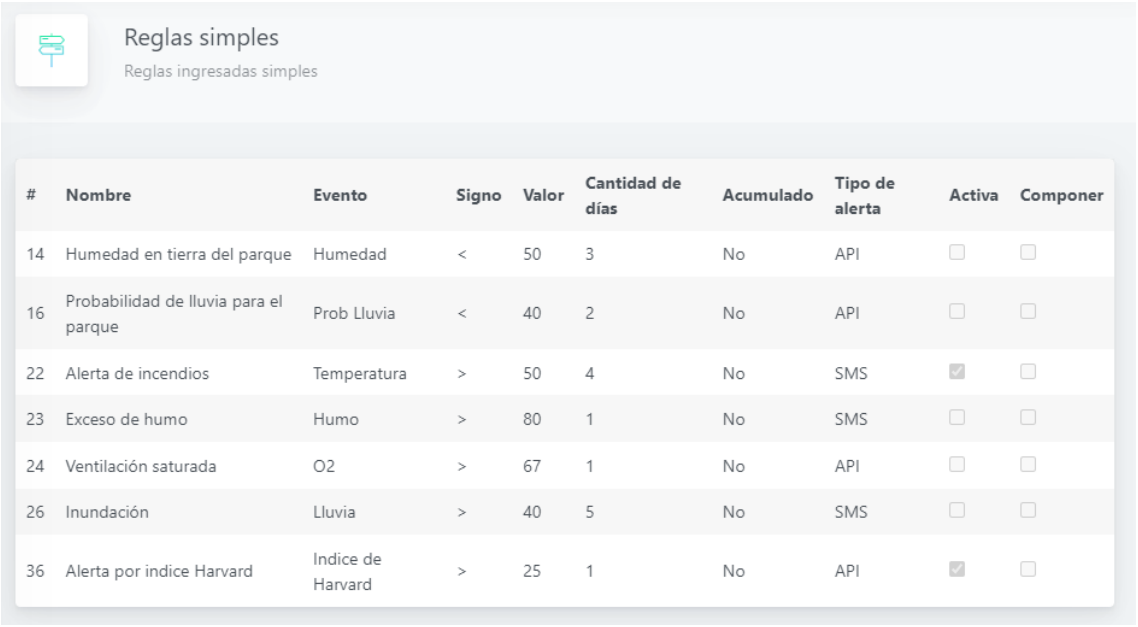


Figura 32: Listado de organizaciones del sistema

Funcionalidades para usuario Experto

Reglas

El usuario experto al entrar a la opción Reglas puede ver todas las reglas creadas por usuarios que pertenecen a la misma organización que él, tanto simples como compuestas. Esta vista es igual que la del usuario administrador. La mayor diferencia que encontramos aquí es que el usuario experto puede cambiar el estado de activación de las reglas ya creadas y también crear nuevas.



#	Nombre	Evento	Signo	Valor	Cantidad de días	Acumulado	Tipo de alerta	Activa	Componer
14	Humedad en tierra del parque	Humedad	<	50	3	No	API	☐	☐
16	Probabilidad de lluvia para el parque	Prob Lluvia	<	40	2	No	API	☐	☐
22	Alerta de incendios	Temperatura	>	50	4	No	SMS	☑	☐
23	Exceso de humo	Humo	>	80	1	No	SMS	☐	☐
24	Ventilación saturada	O2	>	67	1	No	API	☐	☐
26	Inundación	Lluvia	>	40	5	No	SMS	☐	☐
36	Alerta por indice Harvard	Indice de Harvard	>	25	1	No	API	☑	☐

Figura 33: Listado de reglas para experto

Creación de regla simple

Para crear una regla simple, se configuran los umbrales a partir de cuales debería dispararse la acción. También se selecciona el tipo de alerta a disparar y se le ingresa un mensaje.

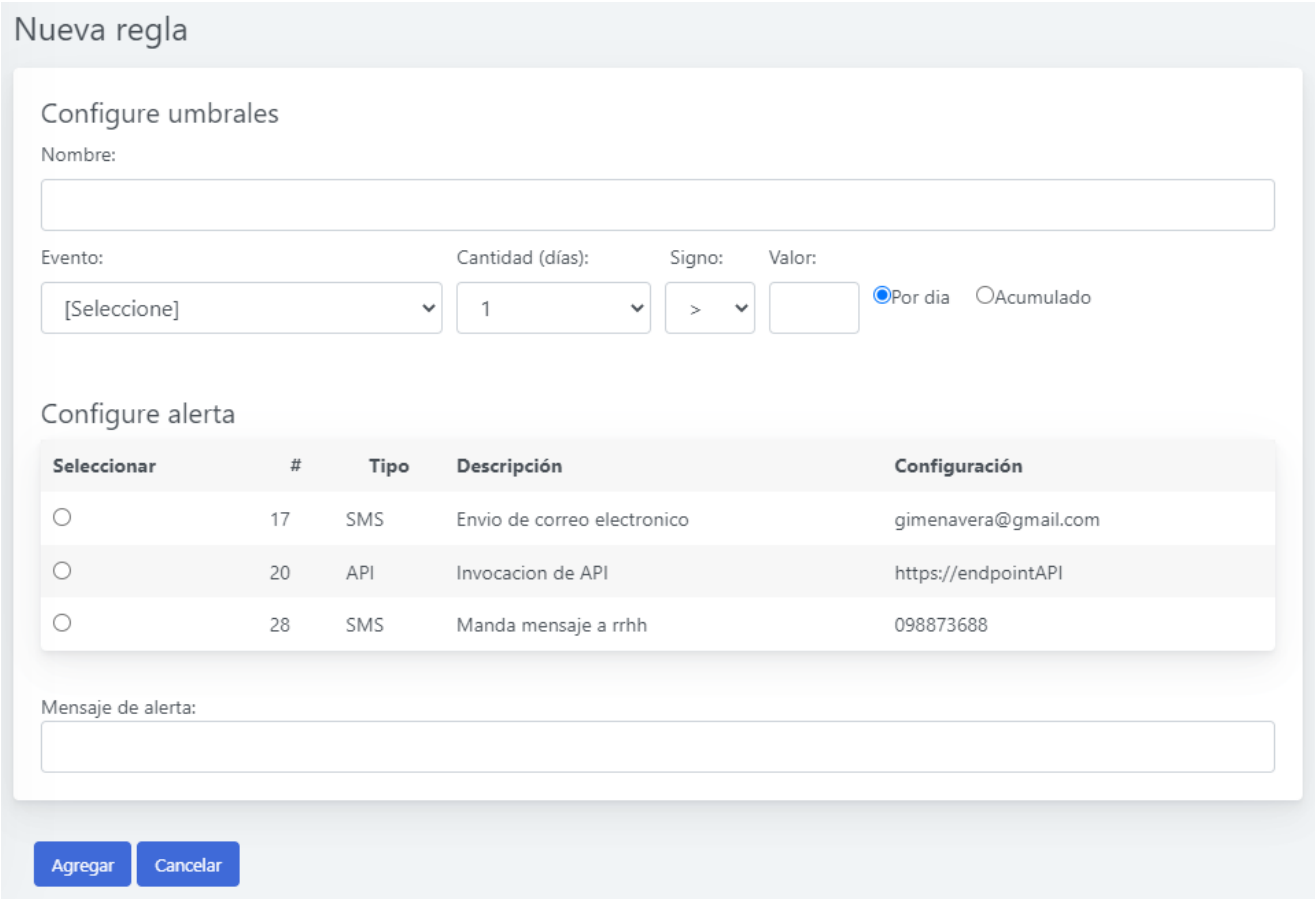


Figura 34: Creación de nueva regla

Creación de regla compuesta

Para crear una regla compuesta, es suficiente por marcar dos reglas simples y configurarle un nombre, un tipo de alerta y el mensaje a enviarse en caso de dispararse.

Alertas

El usuario experto puede visualizar los diferentes tipos de alertas creados por todos los usuarios que pertenecen a su organización, editarlos o crear nuevos.

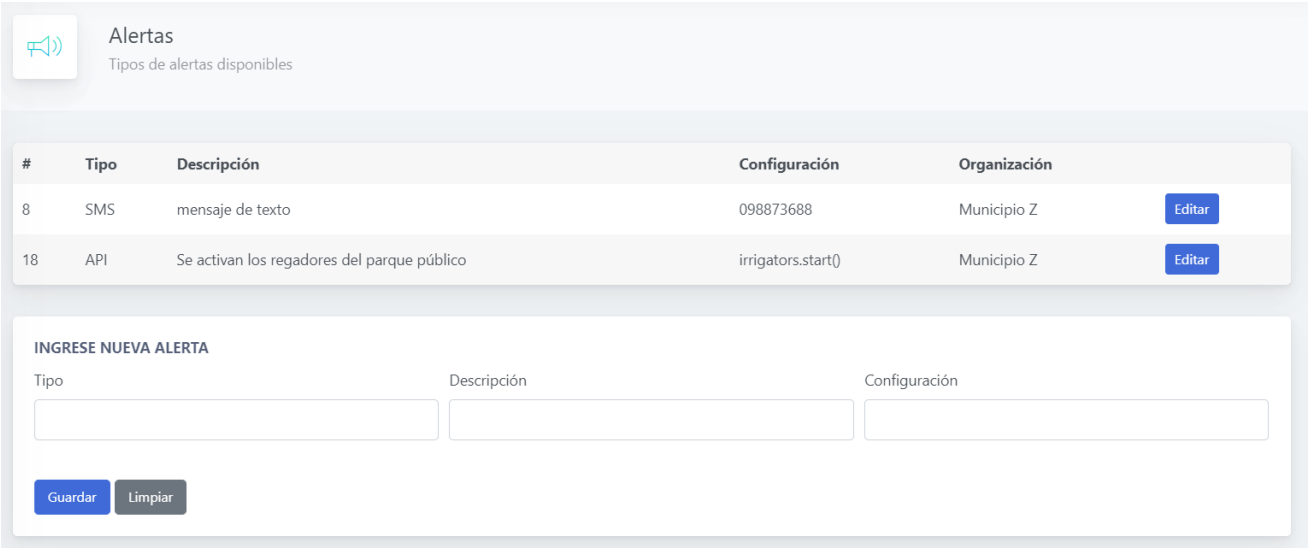


Figura 35: Alertas de una organización

Seguridad

Este prototipo cuenta con manejo de usuarios y perfiles con el objetivo de controlar quien genera las reglas y de esta forma evitar notificar una alarma malintencionada, que pueda resultar en falsos positivos, generando acciones de los diferentes grupos interesados que podrían implicarles posibles gastos o publicidad negativa para la organización entre otros efectos no deseados.

Web services

Se expone un web service SOAP (Simple Object Access Protocol) realizado sobre WCF, que devuelve la lista de reglas ingresadas en el sistema por los usuarios expertos. Al exponer un web service se hace más escalable el sistema dado que puede ser consumido por diferentes plataformas y da la posibilidad de que se expanda con más métodos.

5.3. Desarrollo del prototipo

A continuación, se detallan los principales módulos del prototipo desarrollado.

5.3.1. Base de datos

Se comenzó creando la base de datos en MySQL. La interfaz que se utilizó para manejar la base de datos es SQLYog. Se crearon las tablas especificadas en el diseño para modelar las entidades del prototipo.

5.3.2. Integración con Motor CEP

Para poder probar el objetivo más general de este proyecto el cual es que desde la interfaz de Easy Alerts se pueden configurar reglas de forma simple y que estas puedan ser interpretadas y procesadas por un motor CEP, se desarrolló un adaptador que integra el web Service de Easy Alerts con, en este caso, el motor de Drools.

A continuación, se explica sobre el adaptador, la integración del motor, la fuente de datos utilizada y las alertas configuradas para ser disparadas en caso de cumplirse los patrones.

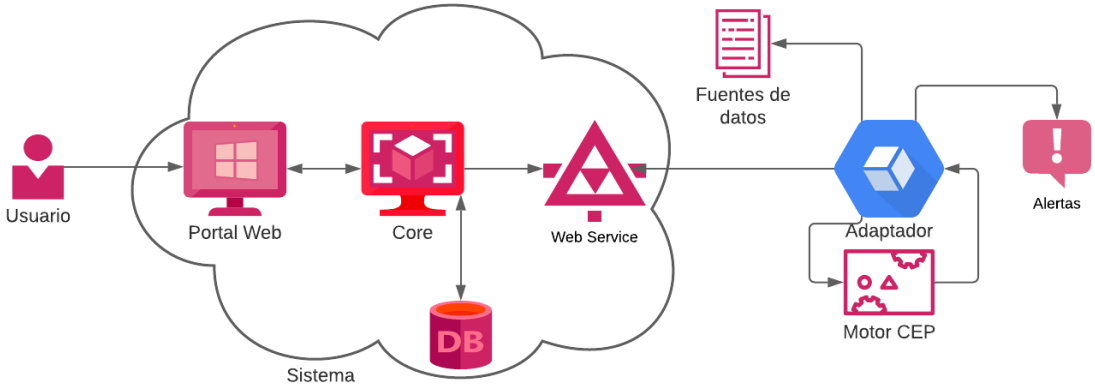


Figura 36: Arquitectura general del prototipo

Adaptador

Para que esta herramienta pueda ser integrada a un sistema CEP, se crea un adaptador en java que consume el servicio web expuesto por Easy Alerts. Drools tiene una memoria de trabajo donde se insertan los diferentes tipos de datos en forma de objetos [14,15]. Por este motivo el adaptador carga objetos de tipo regla (EventRule) dentro de la memoria de trabajo del motor CEP. Por otro lado, también este adaptador es el que obtiene los eventos de dominio de la fuente de datos utilizada por el prototipo. Se crea un objeto de tipo evento (Event) para cada uno de ellos y se agrega también dentro de la memoria de trabajo. Una vez cargados todos los objetos en la memoria de trabajo, se dispara la validación de reglas invocando al motor CEP. Si hay un match, se dispara la alerta.

Motor de CEP – Drools

Drools es el framework elegido para el manejo del motor de reglas. El manejador de este motor es el que integra mediante el adaptador, el uso del web service expuesto por Easy Alerts. Una vez que procesa los datos de reglas y eventos con los patrones del motor, tiene la capacidad de decidir si cumple con las reglas

establecidas y de ser así, dispara las alertas configuradas.

Para poder utilizar el motor de reglas Drools, se deben generar reglas de negocio, las cuales son una sentencia del tipo *cuando/entonces*: cuando se cumple una condición, entonces se desencadena una acción. Para este prototipo, se utiliza un conjunto acotado de las funcionalidades que Drools proporciona.

Utilizamos un solo archivo de reglas del tipo *drl*, donde se crea la regla genérica para reglas del tipo simple y también se crean reglas genéricas para el tipo compuesto.

En la sección de código siguiente se muestra un ejemplo para comprender mejor la sintaxis:

```
rule "simpleRule"
salience 2
no-loop true
when
    r: EventRule(simpleRule == true)
    c: ClimaticEvent(Name == r.getEventName())

then
    int currentValue = c.getMedia();
    int days = r.getCantDays();
    if (r.isAccumulateRule())
    {
        r.setAccumulateValue(r.getAccumulateValue()+ currentValue);
        update(r);
        currentValue = r.getAccumulateValue();
    }

    //incrementa la cantidad de días de ocurrencias
    int cantCurrentDays = r.getCount()+1;
    r.setCount(cantCurrentDays);
    update(r);

    if (currentValue > r.getValue() && (r.getCount() >= days)){
        //cumple la condición, se dispara la alarma
        r.fireAlert(r);
        //se inicializan el contador de días de ocurrencias
        r.setCount(0);
        r.setAccumulateValue(0);
        update(r);
    }

end
```

Figura 37: Código que interpreta la regla simple

En el “when” una vez que se encuentra desde la memoria de trabajo, un objeto de tipo EventRule, se le asigna la variable r. Luego, cuando se encuentra un evento, donde el nombre del evento (por ejemplo, Viento) sea igual al nombre del evento de la regla, entra en el “then”.

En el “then” se validan las condiciones y se modifican los objetos. En este caso, se le suma el valor de la media del evento (dado que se considera que se obtiene el dato una vez por día) al acumulado. La cantidad de días se incrementa y se hace la pregunta de si el acumulado es mayor al valor y la cantidad de días es igual a las configuradas por el experto. Si es así, se inicializan las variables dinámicas de la regla y se disparan las alertas que fueron seteadas desde el portal web. Esto es un ejemplo simple, de los que se utilizaron para validar el prototipo.

Para que este proceso se lleve a cabo en un sistema real, se deberá ejecutar como un servicio del sistema operativo, y debería estar siempre “Running”. Para este prototipo se crea un jar ejecutable el cual una vez comienza a correr se mantiene de esta forma invocando cada cierto periodo de tiempo al web service y también consumiendo de las fuentes de datos. El “main” del proceso, inserta en la memoria de trabajo estos nuevos objetos, y dispara las reglas a ser procesadas.

Fuente de datos

Para el prototipo, se consideran dos conjuntos de datos principales, los de tipo "hechos" que son los eventos de dominio y los de tipo "reglas" que son las reglas que el usuario experto ingresa en el portal web. A su vez, el conjunto de datos de tipo "hechos" pueden tener diferentes características. En el caso de este prototipo, el web service expuesto en la capa de negocios es el encargado de alimentar al motor con las reglas ingresadas desde el portal web por el usuario experto. Para los datos de tipo "hechos", se simulan de otra fuente de datos, obteniéndose de archivos de texto. Estos archivos se generan uno por día, pretendiendo modelar la llegada diaria de los datos como fue definido para la creación de esta herramienta.

Formatos de datos

Para este prototipo se define un formato de datos especifico:

1. Debe ser una lista de listas
2. Cada lista está conformada por {nombre evento, medida media del día, fecha con formato DD/MM/YYYY}
3. El separador es un pipe "|"

```
Humedad|40|16/02/2021
Prob Lluvia|12|16/02/2021
Lluvia|3|16/02/2021
```

Figura 38: Ejemplo de fuente de datos

5.4. Dificultades encontradas

Una de las problemáticas fue al momento de formular los parámetros de la regla de la siguiente forma:

- x= cantidad de días
- y=operador
- z= valor

con el objetivo de que la sentencia que decide si se incrementa el acumulado o se dispara la regla sea de la siguiente forma: X Y Z

Siendo la intención que la regla sea lo más genérica posible, se diseñó inicialmente de la forma explicada anteriormente. Esto debió ser modificado, generando una regla por cada posible operador, donde el único cambio es el signo de la comparación:

```
r.getSign().equals("<")
```

También surgieron diferentes dificultades a la hora de realizar el adaptador y que este se comunique con el motor CEP. En especial por desconocimiento de las directivas propias del lenguaje del motor, con lo referente al manejo de los loops y de la jerarquía que se le da a la validación de cada regla. Luego de investigar y estudiar la documentación se pudieron superar los obstáculos y generar una correcta comunicación entre el adaptador y el sistema Easy Alerts.

6. Pruebas y casos de estudio

En esta sección se presentan dos casos de estudio que se utilizaron para validar la solución planteada y mostrar el funcionamiento general del sistema.

El objetivo de los casos de estudios es validar el correcto funcionamiento del prototipo enfocándonos en las principales funcionalidades de este. El primer caso permite validar el ciclo completo desde que se configura una regla simple en el sistema hasta que se procesa por el motor y se dispara la acción. El segundo caso presenta el manejo de las reglas compuestas que son las que hacen al sistema potencialmente más efectivo.

6.1. Caso regla simple

Problema planteado para regla simple: Inundación

Se presenta como usuario (ficticio) de Easy Alerts una institución gubernamental de Uruguay: Meteo. Esta empresa nos indica su necesidad de tener alertas por inundación para las empresas de transporte, cuando los arroyos en el interior del país no dan paso por las rutas o caminos. Nos cuentan que, si el conductor del camión u ómnibus identifica que algún camino no da paso recién cuando llega a verlo, a veces no es posible realizar el desvío, que sí sería posible de saberlo antes de comenzar el trayecto. Cuando hay posibilidad, por experiencias anteriores, de que algún camino esté inundado se le enviará a quien corresponda una alerta con esta información.

Para esto, dimos de alta en el sistema la empresa *Meteo* con su dominio = *Clima* (Nro 1 en la Figura 39) y sus eventos de dominio; mediante la opción “Ver Eventos” (Nro 2 en la Figura 39) se visualiza la lista de los eventos previamente configurados para ese dominio (Figura 40). Por otro lado, *Meteo* proporciona los endpoints para consumir los web services de los pluviómetros de la cuenca afectada.

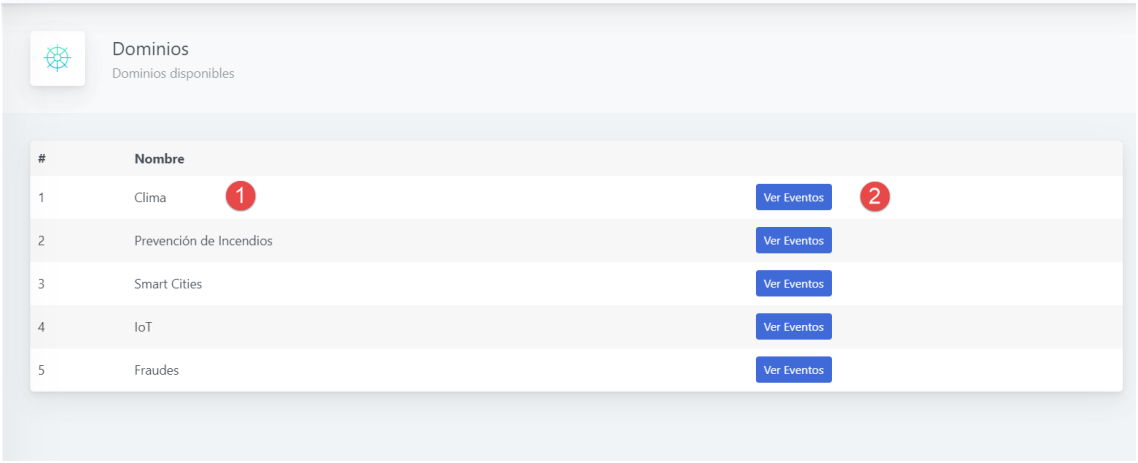


Figura 39: Dominio caso de estudio 1

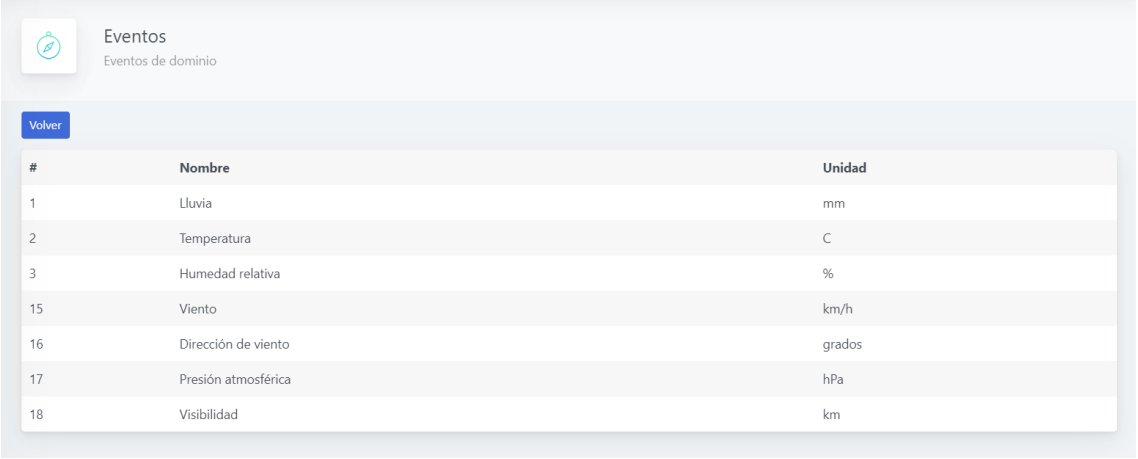


Figura 40: Eventos de dominio caso de estudio 1

A continuación, se presenta la forma en la que queda Easy Alerts una vez creada la organización al registrarse al sistema. Los usuarios expertos que se comiencen a crear mediante la interfaz web se presentarán de la siguiente manera:

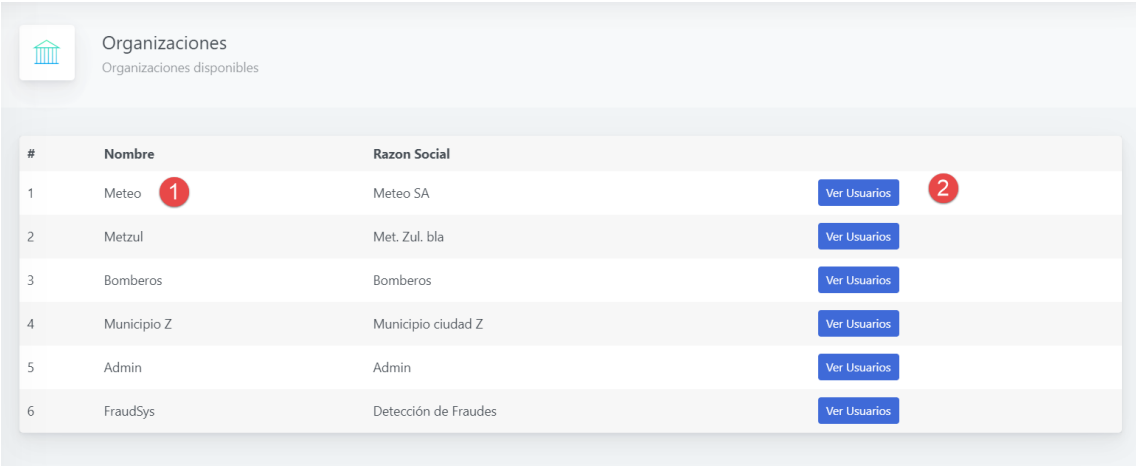


Figura 41: Listado de organizaciones caso de estudio 1

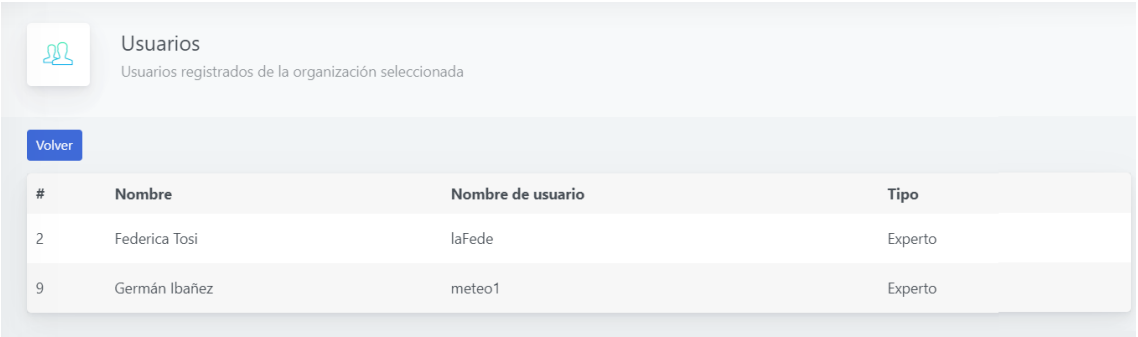


Figura 42: Usuarios expertos de caso de estudio 1

Bajo la opción de Organizaciones se listan las organizaciones creadas en el sistema. En la lista de la Figura 41 vemos en el Nro1 la organización Meteo con su razón social. Luego bajo la opción “Ver Usuarios” (Nro 2 en la Figura 41) el sistema direcciona a otro listado de los usuarios expertos creados para esa organización (Figura 42).

El usuario experto *Meteo1* accede a la plataforma con su nombre de usuario y password y lo primero que

puede visualizar es el dashboard con los indicadores de su organización:

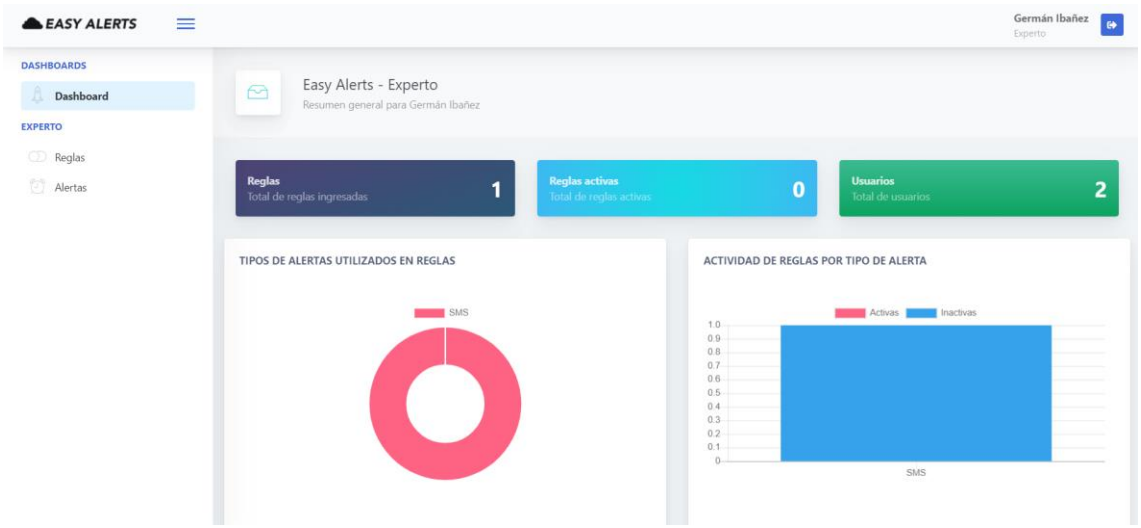


Figura 43: Vista del portal con usuario experto meteo1

Se consume la información que proporcionan los pluviómetros en las zonas cercanas. El arroyo que solicitan monitorear es el Arroyo Miguelete (dpto. Colonia) ruta 21 km 221[17].

Figura 44: Rotonda entre las rutas 21 y 22, Arroyo Miguelete

El experto configura la regla teniendo en cuenta experiencias pasadas donde se identificó que, para la inundación en esta zona en particular, alcanza con medirse más de 40mm de lluvia por día durante 5 días consecutivos. Para esto se elige el envío de correos y se configuró la regla de la siguiente forma:

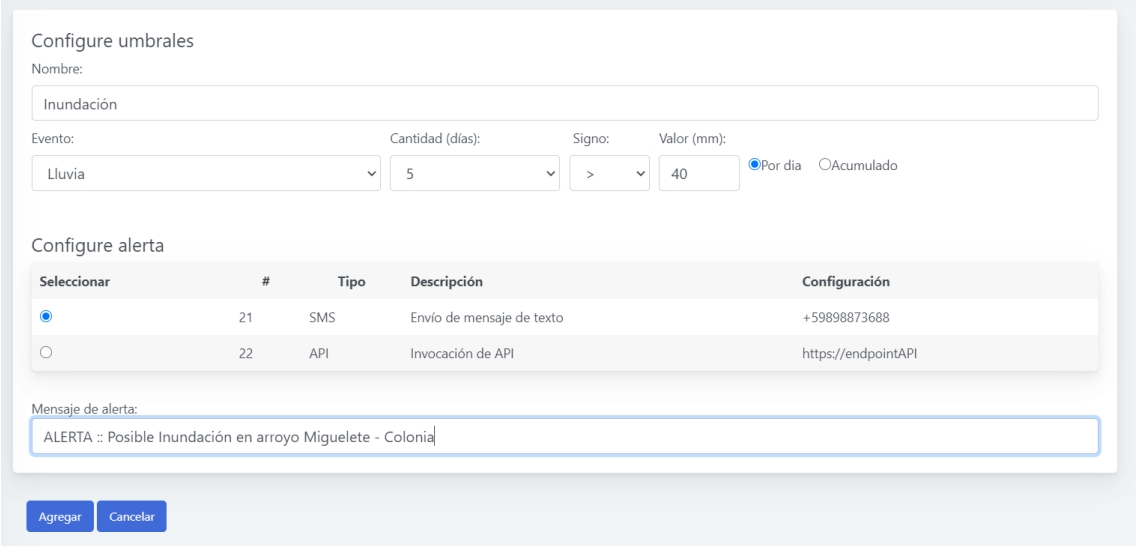


Figura 45: Configuración de regla simple inundación caso estudio 1

Figura 46: Regla simple creada: Inundación. Caso de estudio 1

En la simulación de los datos de eventos, se consume el siguiente conjunto para que la regla se dispare:

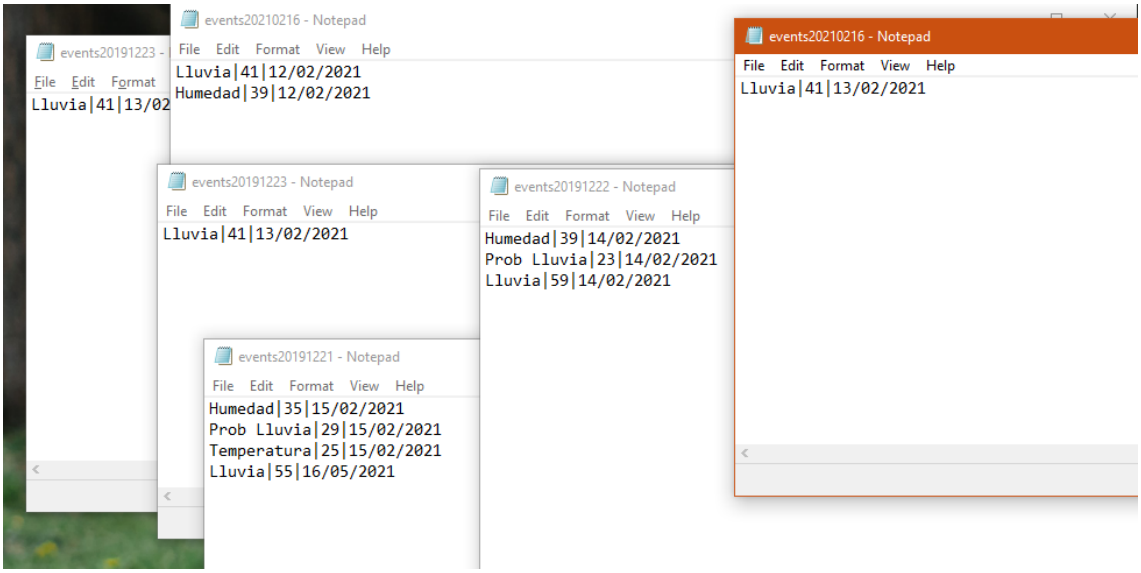


Figura 47: Datos simulados para inundación caso de estudio 1

6.2. Caso regla compuesta

Problema planteado para regla compuesta: Riego en parque público

El municipio Z lleva adelante un proyecto piloto de smart cities que de ser exitoso se extenderá a otros municipios de la ciudad. Para uno de los temas, se solicita a Easy Alerts un mecanismo automático de riego

en el parque de la zona. El municipio proporciona las conexiones para obtener los datos desde los sensores de humedad del suelo del parque, como también una interfaz para activar los regadores en caso de ser necesario. Por medio de la configuración previa se ingresa el nuevo dominio = *smart cities* y sus eventos. El usuario experto del municipio Z *smart1* configura en Easy Alerts, una regla compuesta para llevar adelante la tarea. Esta regla tiene como objetivo optimizar recursos, siguiendo los lineamientos de las smart cities, por lo que se desea activar los regadores únicamente de ser necesario. Esta necesidad se identifica considerando tanto la probabilidad de lluvia como la humedad del suelo. Para combinar estos dos fenómenos se utiliza una regla compuesta.

En la Figura 48 se muestra como primero se crea una regla simple basándose en experiencias previas, donde se considera necesario regar el parque si el porcentaje de humedad en el suelo es menor a 50% cada día durante 3 días consecutivos.

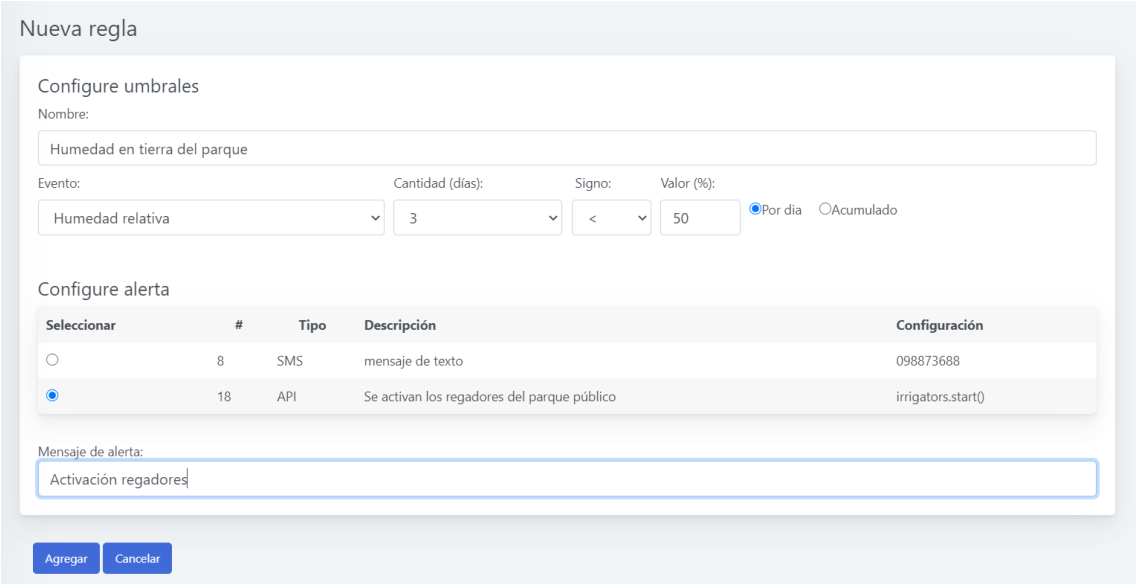


Figura 48: Configuración regla simple 1 para caso de estudio 2

Luego, como se ve en la Figura 49, se crea otra regla simple donde se configura el umbral de probabilidad de lluvia, cuya lectura es proporcionada por el Instituto de Meteorología. Esta regla se configura con una probabilidad de lluvia menor a 40% durante dos días para que se cumpla.

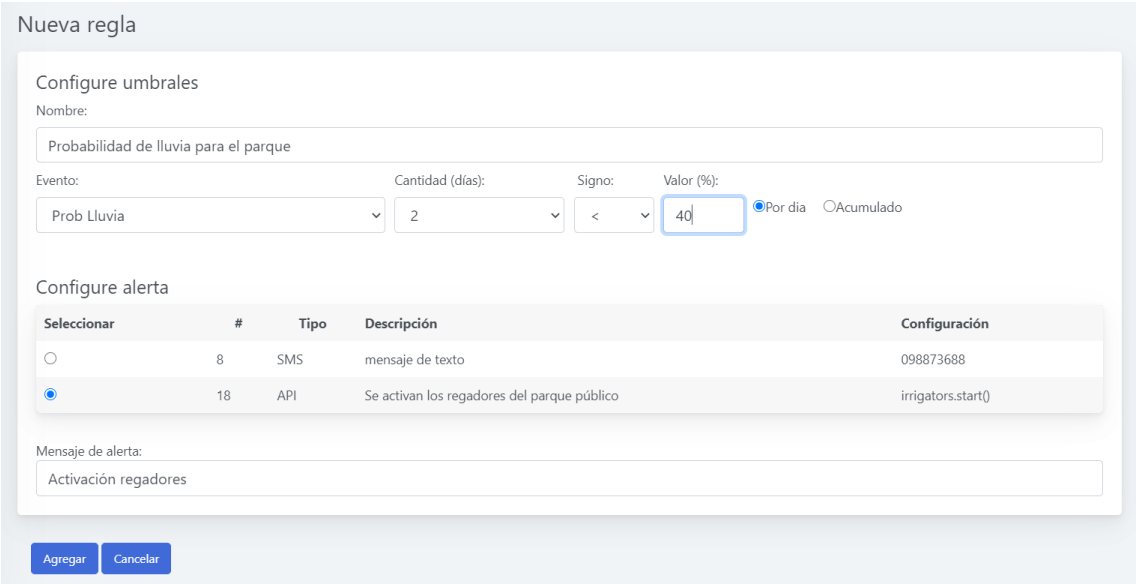


Figura 49: Configuración regla simple 2 para caso de estudio 2

Componiendo ambas reglas como se muestra en la Figura 50, se crea la nueva regla compuesta deseada. Enviando como alerta la acción de prender los regadores del parque público del municipio mediante el API proporcionado por la organización.

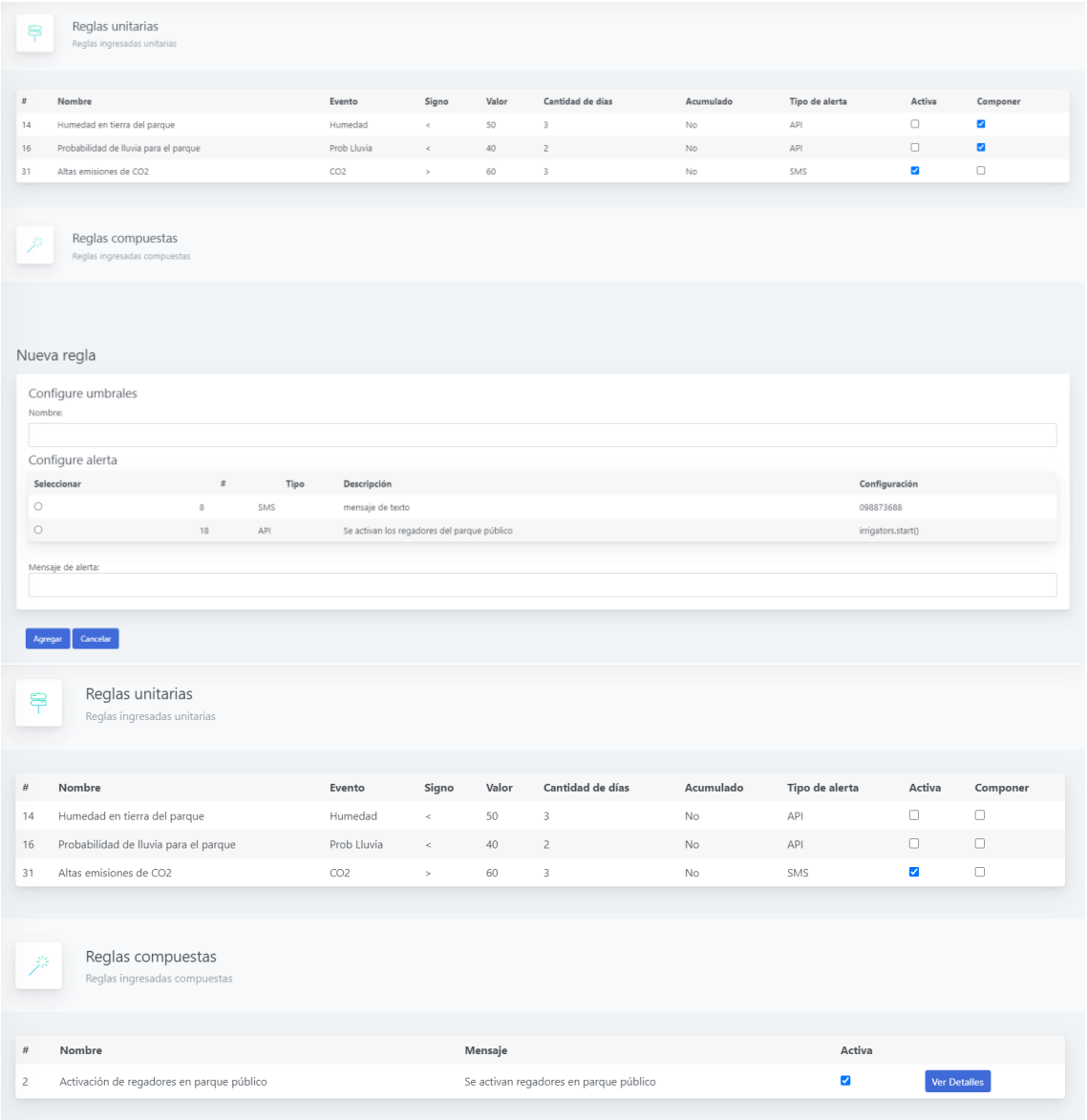


Figura 50: Regla compuesta: smart cities caso de estudio 2

Como se muestra en las figuras, la interfaz gráfica es la misma que cuando deseamos ingresar una regla simple. La diferencia es que, para agregar una regla compuesta, es necesario tener previamente ingresadas como simples las reglas que la componen. Luego de seleccionar dos reglas simples del listado, se le indica un nombre y un tipo de alerta para crear la regla compuesta.

6.3. Resultados de las pruebas

Una vez finalizado el desarrollo de la interfaz se comenzaron las pruebas de la herramienta para confirmar que el sistema tenía la capacidad deseada. En particular la configuración de reglas que puedan ser interpretadas por un adaptador conectado a un motor de procesamiento de eventos complejos.

El sistema se desarrolló por componentes como se mostraron en la arquitectura, y en la integración de cada nuevo componente se realizaron pruebas de integración y regresión para asegurarse de que ningún componente previo se vea afectado.

Durante la creación de los casos de estudio se detectaron errores en la interfaz que fueron solucionados, así como también se probaron los flujos alternativos. Se realizó este tipo de pruebas abarcando todas las pantallas y opciones.

El objetivo principal de los casos de estudio fue validar el uso del prototipo con escenarios que se asemejan a la realidad, pero también se utilizaron para probar la plataforma. Como conclusión, se logró validar las principales funcionalidades en un ciclo completo de uso del prototipo con los resultados esperados.

7. Conclusiones

7.1. Gestión del proyecto

Este proyecto de grado se dividió en diferentes etapas:

- *Análisis*
- *Diseño*
- *Implementación*
- *Pruebas*
- *Documentación*

Análisis: Tomando la problemática planteada que motivó este proyecto como punto de partida, se comenzó con la investigación de herramientas y trabajos existentes para comprender mejor el contexto del problema. Se continuó con el análisis de los requerimientos que tendría el sistema y sus principales casos de uso.

Diseño: Esta etapa consistió en diseñar la arquitectura tanto del sistema general, como también del prototipo a crear. Desde la base de datos como también de los datos que serían consumidos cumpliendo con los requerimientos funcionales y no funcionales definidos en la etapa de análisis. Se diseñó la interacción de los componentes adaptando la estrategia de abstracción y modularidad. En este punto se identificó que, para validar la herramienta con los casos de prueba, había que crear un adaptador con alguno de los motores CEP. De esta manera se puede probar efectivamente que la forma en la que el usuario experto crea la regla en la interfaz amigable sirve para generar patrones en un motor CEP.

Implementación: Se implementó el prototipo con el alcance definido y el adaptador de forma de validar la viabilidad de la herramienta diseñada con los casos de prueba. Para la implementación se comenzó configurando el ambiente de desarrollo dividido en tres módulos principales: base de datos, portal web y servicios en .Net, adaptador en Java.

Pruebas: Al finalizar la implementación del prototipo y del adaptador se realizaron pruebas y dos casos de estudio con la finalidad de validar que la herramienta cumpla con los objetivos del proyecto de grado. Para las pruebas se simularon casos de aplicación real, brindando dos escenarios de diferentes dominios.

Documentación: Durante todo el transcurso de este proyecto, se fue generando la documentación que refleja las definiciones realizadas en cada etapa.

La planificación inicial de este proyecto preveía una duración de 15 meses, con las siguientes fases:

Figura 491: Plan del proyecto

La duración real del proyecto fue 25 meses debido en parte a la situación familiar durante las etapas iniciales de la pandemia. Entre abril y octubre de 2020 no hubo actividad. Luego en las etapas finales el proceso de corrección de la documentación también se extendió más de lo planificado.

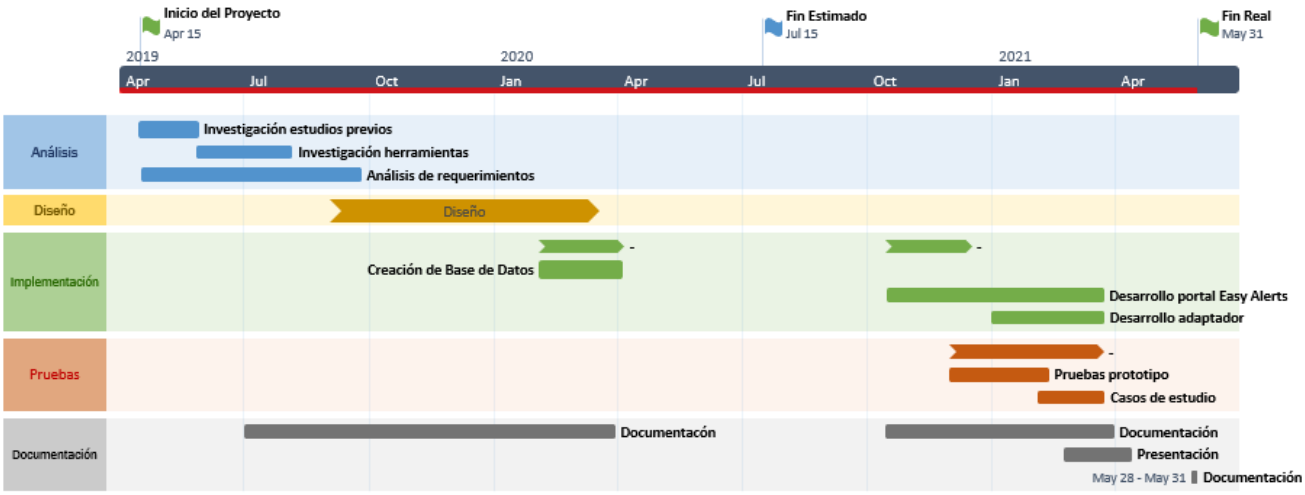


Figura 502: Cronograma real del proyecto

7.2. Conclusiones

Luego de culminar las diferentes etapas de este proyecto de grado y habiendo logrado implementar un prototipo validado con los respectivos casos de prueba, se concluye que se cumplieron los objetivos planteados al inicio de este trabajo.

En primer lugar, el prototipo desarrollado se considera intuitivo y amigable para usuarios no técnicos y las configuraciones allí ingresadas son lo suficientemente robustas para generar reglas en un motor CEP. Con las investigaciones realizadas de los estudios previos, se identificaron escenarios de uso de la herramienta, lo cual también se muestra con los casos de estudio que se puede utilizar en varios dominios.

A lo largo de las fases del proyecto se fueron tomando ciertas decisiones para mantener el carácter genérico de la herramienta. El usuario experto de dominio, sin tener ninguna formación en desarrollo de software puede ser capaz, mediante un conjunto reducido de parámetros, de configurar a través de una página web reglas adaptables a un gran número de situaciones pudiendo generar acciones de alto impacto. Considerando que las acciones a disparar pueden ser diversas, en virtud de la masificación de los medios en la actualidad, puede llegar a un público amplio en poco tiempo, o ser tan específico como enviar una orden a otro sistema para que se tome una acción concreta.

La diversidad de las fuentes de datos, así como los dominios, pueden acercar a un gran número de expertos de dominio a simular sus necesidades de esta forma, siendo la utilidad de este sistema tan amplia como se pueda imaginar.

Analizando la implementación del prototipo y del adaptador, también se puede concluir que el verdadero potencial de utilizar un motor CEP se hace evidente a medida que los datos que ingresan se hacen en mayor cantidad, mayor diversidad y en tiempos distintos. Para poder extender el uso de esta herramienta y explotar sus características, se han identificado puntos a desarrollar más allá del prototipo planteado, de los cuales algunos de ellos se detallan en la siguiente sección.

7.3. Trabajo futuro

Se identificó que la herramienta se puede extender de varias formas para crear un sistema con mayor potencial. A continuación, se describen algunas posibles extensiones:

Geolocalización: integrar a la herramienta información geográfica podría dar la opción de ingresar latitud, longitud y un área a partir de un radio como nuevos parámetros de las reglas configuradas. Con esta mejora se podría indicar para qué localidad deseamos se evalúen las reglas. Asociado a potenciar la integración con servicios de información geográfica, también se podría relacionar con fuentes de datos internacionales de la región. De esta forma, se podrían generar reglas por zona o región y no necesariamente atenerse a las lecturas de cierta área.

Creación de reglas compuestas por más de dos reglas simples: a futuro se podría permitir que el usuario experto sea quién determine la cantidad de reglas simples que conformen cada regla compuesta que se configure. Dependerá del adaptador creado para el motor CEP al cual se integre el sistema, la forma en la que se resuelva esta problemática. En el caso de Drools y el adaptador creado para este prototipo, se podría recorrer la lista de reglas simples que tiene la regla compuesta y dar por cumplida la compuesta cuando todas las simples se cumplen.

Retroalimentación a la base de datos de reglas disparadas: Para tener control y seguimiento de las reglas que se cumplieron y por ende de las alertas disparadas, el adaptador podría actualizar el estado de estas. Esto se lograría exponiendo otros Web Services para que el adaptador los invoque con los datos actualizados. Esto enriquecería los datos presentados en el Dashboard.

Datos categóricos: la herramienta podría también permitir al usuario experto utilizar eventos categóricos, como por ejemplo el índice UV que es alto, medio, bajo etc.

8. Glosario

.Net: es un framework de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.

Algoritmo Rete: El algoritmo Rete es un algoritmo de reconocimiento de patrones eficiente para implementar un sistema de producción de reglas

API: Application Programming Interface. Es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas.

API Java: Como el lenguaje Java es un Lenguaje Orientado a Objetos, la API de Java provee de un conjunto de clases utilitarias para efectuar toda clase de tareas necesarias dentro de un programa. La API Java está organizada en paquetes lógicos, donde cada paquete contiene un conjunto de clases relacionadas semánticamente.

Base de conocimiento: Es la parte del sistema experto que contiene el conocimiento sobre el dominio.

Casos de uso: son la herramienta más aplicada para la especificación de requerimientos funcionales. Por ser expresados textualmente resultan simples de comprender. Un actor es un agente externo (humano o no) que interactúa directamente con el sistema. Un caso de uso narra la historia completa (junto a todas sus variantes) de un conjunto de actores mientras usan el sistema.

Ciudades inteligentes: Son el resultado de la necesidad cada vez más imperiosa de orientar nuestra vida hacia la sostenibilidad. Así, estas ciudades se sirven de infraestructuras, innovación y tecnología para disminuir el consumo energético, reducir las emisiones de CO2 entre otras cosas [18,19].

Diagrama de secuencia: es un tipo de diagrama usado para modelar interacción entre objetos en un sistema según UML.

Diagrama entidad relación: Un diagrama o modelo entidad-relación (a veces denominado por sus siglas en inglés, E-R "Entity relationship", o del español DER "Diagrama de Entidad Relación") es una herramienta para el modelado de datos que permite representar las entidades relevantes de un sistema de información, así como sus interrelaciones.

Drools: es un sistema de gestión de reglas de negocio (BRMS, por las siglas en inglés de Business Rule Management System) que utiliza un motor de reglas basado en inferencia de encadenamiento hacia adelante(forward chaining) y de encadenamiento hacia atrás (backward chaining), más conocido como sistema de reglas de producción, y que utiliza una implementación avanzada del algoritmo Rete.Es software libre distribuido según los términos de la licencia Apache.

Eclipse: Plataforma, típicamente usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse).

Evento: Un evento es algo que ocurre o que no ocurre y que tiene interés potencial para el negocio en un momento determinado [17].

Evento Complejo: El evento resultante del acontecimiento de varios eventos [17].

Framework: La palabra “framework” define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

Http: HyperText Transfer Protocol (Protocolo de transferencia de hipertexto) es el método más común de intercambio de información en la world wide web, el método mediante el cual se transfieren las páginas web a un ordenador.

Ide: Un IDE es una herramienta que nos ayuda a desarrollar de una manera amigable nuestras aplicaciones, brindándonos ayudas visuales en la sintaxis, plantillas, wizards, plugins y sencillas opciones para probar y hacer un debug.

Interfaz de usuario: Es el medio con que el usuario puede comunicarse con una máquina, equipo, computadora o dispositivo, y comprende todos los puntos de contacto entre el usuario y el equipo.

IoT: Es un concepto que se refiere a una interconexión digital de objetos cotidianos con internet.

Jboos: es un servidor de aplicaciones Java EE de código abierto implementado en Java puro, más concretamente la especificación Java EE. Al estar basado en Java, JBoss puede ser utilizado en cualquier sistema operativo para el que esté disponible la máquina virtual de Java. JBoss Inc., empresa fundada por Marc Fleury y que desarrolló inicialmente JBoss, fue adquirida por Red Hat en abril del 2006.

Lenguaje de procesado de eventos: o EPL (Event Processing Language en inglés) es un lenguaje de consulta similar a SQL. Las cadenas reemplazan a las tablas como fuente de datos y los eventos reemplazan a las filas como unidad básica de datos

Memoria de trabajo: Parte de un sistema experto que contiene los hechos del problema que se descubren durante la sesión.

Motor CEP: El sistema experto modela el proceso de razonamiento humano con un módulo conocido como el motor de inferencia. Dicho motor de inferencia trabaja con la información contenida en la base de conocimientos y la base de hechos para deducir nuevos hechos.

Multitenant: Se refiere a un principio en la arquitectura de software donde una única instancia del software se ejecuta en un servidor y al servicio a múltiples clientes.

MySql: Es un sistema de administración de bases de datos (Database Management System, DBMS) para bases de datos relacionales. Así, MySQL no es más que una aplicación que permite gestionar archivos llamados de bases de datos.

Protocolos: En informática, un protocolo es un conjunto de reglas usadas por computadoras para comunicarse unas con otras a través de una red. Un protocolo es una convención o estándar que controla o permite la conexión, comunicación, y transferencia de datos entre dos puntos finales.

Sistemas expertos: Los sistemas expertos son llamados así porque emulan el razonamiento de un experto en un dominio concreto y en ocasiones son usados por éstos. Con los sistemas expertos se busca una mejor calidad y rapidez en las respuestas dando así lugar a una mejora de la productividad del experto.

WCF: Windows Communication Foundation (WCF) es un marco de trabajo para la creación de aplicaciones orientadas a servicios. Con WCF, es posible enviar datos como mensajes asincrónicos de un extremo de servicio a otro. Un extremo de servicio puede formar parte de un servicio disponible continuamente hospedado por IIS, o puede ser un servicio hospedado en una aplicación.

Web service: describe una forma estandarizada de integrar aplicaciones WEB mediante el uso de XML, SOAP, WSDL y UDDI sobre los protocolos de Internet. XML es usado para describir los datos, SOAP se ocupa de la transferencia de los datos, WSDL se emplea para describir los servicios disponibles y UDDI para conocer cuáles son los servicios disponibles. Uno de los usos principales es permitir la comunicación entre las empresas y entre las empresas y sus clientes. Los Web Services permiten a las organizaciones intercambiar datos sin necesidad de conocer los detalles de sus respectivos Sistemas de Información.

9. Referencias

[1] Introducción al procesamiento de Eventos Complejos I
https://decidesoluciones.es/introduccion-al-procesamiento-de-eventos-complejos-i/
Último acceso 31/05/2021

[2] Parte II: CEP - Procesamiento de Eventos Complejos
http://blog.vitria.com/es/bid/81509/Parte-II-CEP-Procesamiento-de-Eventos-Complejos
Último acceso 31/05/2021

[3] ESPERTECH
https://www.espertech.com/esper/
Último acceso 31/05/2021

[4] Drools
https://www.drools.org/
Último acceso 31/05/2021

[5] Flinkp
https://flink.apache.org/news/2016/04/06/cep-monitoring.html
Último acceso 31/05/2021

[6] Siddhi
http://siddhi.sourceforge.net/
Último acceso 31/05/2021

[7] Google cloud platform
https://cloud.google.com/solutions/architecture/complex-event-processing
Último acceso 31/05/2021

[8] Apache storm
https://storm.apache.org/
Último acceso 31/05/2021

[9] A model-driven approach for facilitating user-friendly design of complex event patterns, autores: Juan Boubeta Puig, Guadalupe Ortiz y Inmaculada Medina Bulo, publicado en la revista Expert Syst. Appl. 2014
https://rodin.uca.es/xmlui/bitstream/handle/10498/17262/2014-ESWA-Boubeta-AModelDrivenApproach.pdf
Último acceso 31/05/2021

[10] Semantic Complex Event Processing for Social Media Monitoring - A Survey , autor: Robin Keskisarkka, publicado en la conferencia International Semantic Web 2014
http://ceur-ws.org/Vol-1191/paper1.pdf
Último acceso 31/05/2021

[11] CASA and LEAD: Adaptive Cyberinfrastructure for Real-Time Multiscale Weather Forecasting,

autores: Beth Plale, Dennis Gannon, Jerry Brotzge, Kelvin Droegemeier, James F. Kurose, David McLaughlin, Robert Wilhelmson, Sara J. Graves, Mohan Ramamurthy, Richard D. Clark, Sepi Yalda, Daniel A. Reed, Everette Joseph y V. Chandrasekar, publicado en la revista Computer en 2006.
http://kelvin.ou.edu/CASA-LEAD-IEEE.pdf
Último acceso 31/05/2021

[12] Scalable CEP for Smart Cities, autores: Fernando Freire y Kelly Braghetto, publicado en la revista CoRR 2020
https://es.overleaf.com/articles/scalable-cep-for-smart-cities/vhqzrpypfwjm
Último acceso 31/05/2021

[13] Complex event processing and data mining for smart cities, autores: Alexandra Moraru y Dunja Mladenić
https://www.researchgate.net/publication/262062623_Complex_event_processing_and_data_mining_for_smart_cities
Último acceso 31/05/2021

[14] Drools Expert User Guide
https://docs.jboss.org/drools/release/5.2.0.Final/drools-expert-docs/html_single/
Último acceso 31/05/2021

[15] Drools – Getting Started
https://techgrafitti.wordpress.com/2011/02/21/drools-%E2%80%93-getting-started-hello-world-example/
Último acceso 31/05/2021

[16] TararirasHoy – Noticia inundación
https://tararirashoy.blogspot.com/2016/04/luego-del-temporal-quedaron-puentes.html
Último acceso 31/05/2021

[17]¿Qué es CEP?, Objetivos, Evento simple y complejo
http://blog.vitria.com/es/bid/66308/Qu-es-CEP-Objetivos-Evento-simple-y-complejo#:~:text=Desde%20esta%20perspectiva%20definimos%20evento,del%20acaecimiento%20de%20varios%20eventos.
Último acceso 31/05/2021

[18] Smart Cities
https://www.sostenibilidad.com/construccion-y-urbanismo/que-es-una-smart-city-top-5-ciudades-inteligentes/
Último acceso 31/05/2021

[19] Smart Cities
https://panelesach.com/blog/smart-cities-o-ciudades-inteligentes-que-son/
Último acceso 31/05/2021

[20] Esper Runtime
http://esper.espertech.com/release-8.7.0/reference-esper/html/processingmodel.html
Último acceso 31/05/2021

[21] Drools production rule system
https://docs.jboss.org/drools/release/6.5.0.Final/drools-docs/html_single/index.html#d0e4333
Último acceso 31/05/2021

[22] Flink Stream Processing
https://ci.apache.org/projects/flink/flink-docs-release-1.13/docs/learn-flink/overview/
Último acceso 31/05/2021

[23] Siddhi Overview
https://github.com/siddhi-io/siddhi
Último acceso 31/05/2021

[24] Storm Streams
https://storm.apache.org/releases/2.2.0/Tutorial.html
Último acceso 31/05/2021