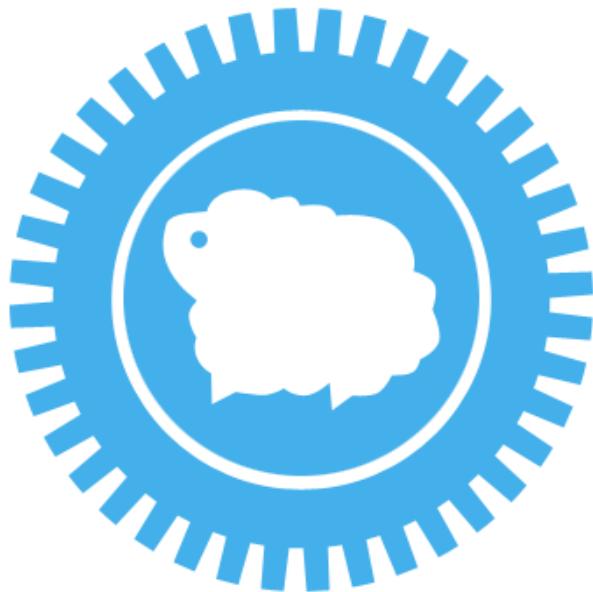


DESARROLLO DE UNA PLATAFORMA DE AUTOMATIZACIÓN DE TAREAS

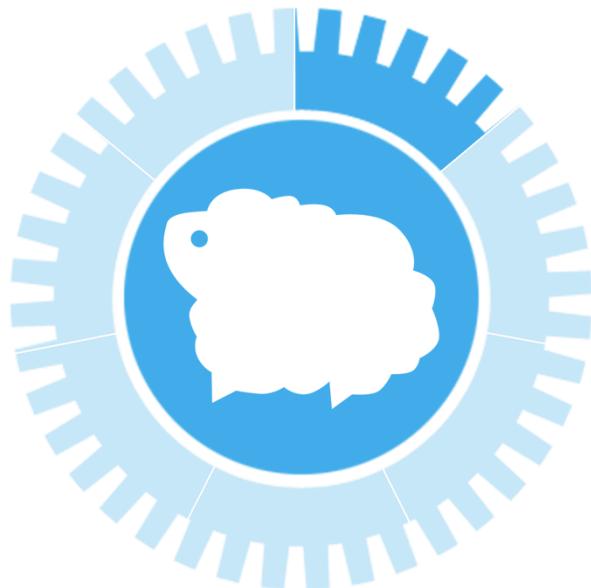
Autor: Carlos Crespo González-Calero

Tutor: Miguel Coronado Barrios

Ponente: Carlos A. Iglesias Fernández

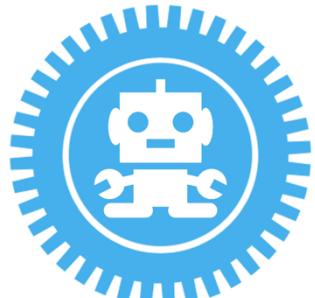


- 1. Introducción**
- 2. Estado del arte**
- 3. Análisis de requisitos**
- 4. Arquitectura**
- 5. Caso de Estudio**
- 6. Conclusiones y trabajo futuro**



- 1. Introducción**
2. Estado del arte
3. Análisis de requisitos
4. Arquitectura
5. Caso de Estudio
6. Conclusiones y trabajo futuro

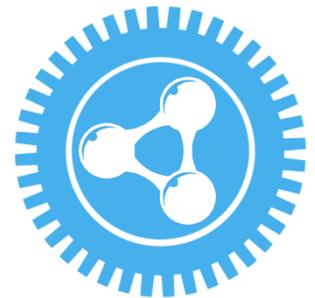
DESARROLLO DE UNA PLATAFORMA INTELIGENTE DE AUTOMATIZACIÓN DE TAREAS



- Plataforma de automatización de tareas



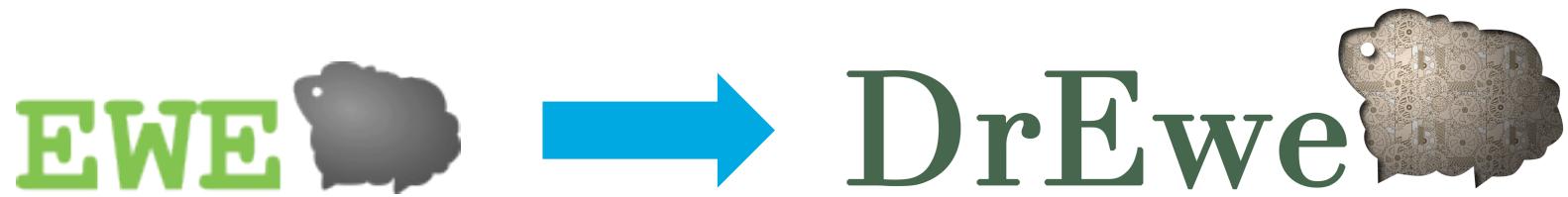
- Procesado de eventos complejo



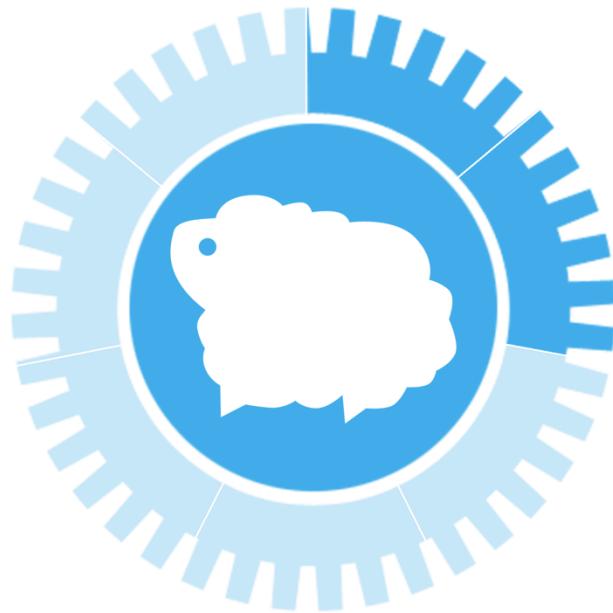
- Modelado semántico

- Creciente número de servicios web          
- Aparición de plataformas de automatización de tareas:
 - *"Cuando sea mencionado en Twitter, envíame un e-mail."*
 - Reglas de evento – condición – acción (ECA)
 - Eventos provenientes de servicios web
- Identificamos una necesidad de estandarización:
 - Modelado a través de una Ontología
 - Interoperabilidad entre plataformas
 - Análisis de datos, filtrado de información (eventos y reglas)

- Ontologa EWE desarrollada en el grupo de sistemas inteligentes:



1. Diseño e implementación de una plataforma de automatización de tareas **extensible** y **escalable** que implemente **procesado de eventos complejo** y soporte la **ontología EWE**
2. Implementar y conectar **servicios externos** existentes a través de una interfaz común
3. Diseñar **sensores físicos** y conectarlos a la plataforma a través de un *middleware*
4. Desplegar una **red de sensores** que satisfaga nuestras necesidades
5. Diseño e implementación de un **motor de reglas**, que permita ejecución de reglas semánticas y razonamiento temporal.
6. Escoger un caso de uso que pueda ser **probado** en el laboratorio del grupo



1. Introducción
2. Estado del arte
3. Análisis de requisitos
4. Arquitectura
5. Caso de Estudio
6. Conclusiones y trabajo futuro

PLATAFORMAS DE AUTOMATIZACIÓN DE TAREAS

IFTTT



- Recipes: reglas simples
evento – condición – acción
- 76 canales
- Orientado al usuario básico de
internet
- Financiación \$8.5M

ZAPIER



- Zaps: reglas simples
evento – condición – acción
- 310 canales (servicios web)
- Más orientado al ámbito
profesional
- Financiación \$1.2M

Motor de reglas de negocio (BRE)

- Componente software que proporciona al desarrollador la funcionalidad Representación de Conocimiento y Razonamiento (KRR)
- Implementa lógica de negocio realizando razonamientos según las reglas definidas y los datos de entrada

CLIPS

- C Language Integrated Production System.
- Desarrollado por la NASA
- Objetivo: modelar el conocimiento humano

Drools

- Anteriormente: Production Rule System
- Desarrollado por JBoss
- Basado en Rete Algorithm
- Drools Fusion

COMPLEX EVENT PROCESSING (CEP)

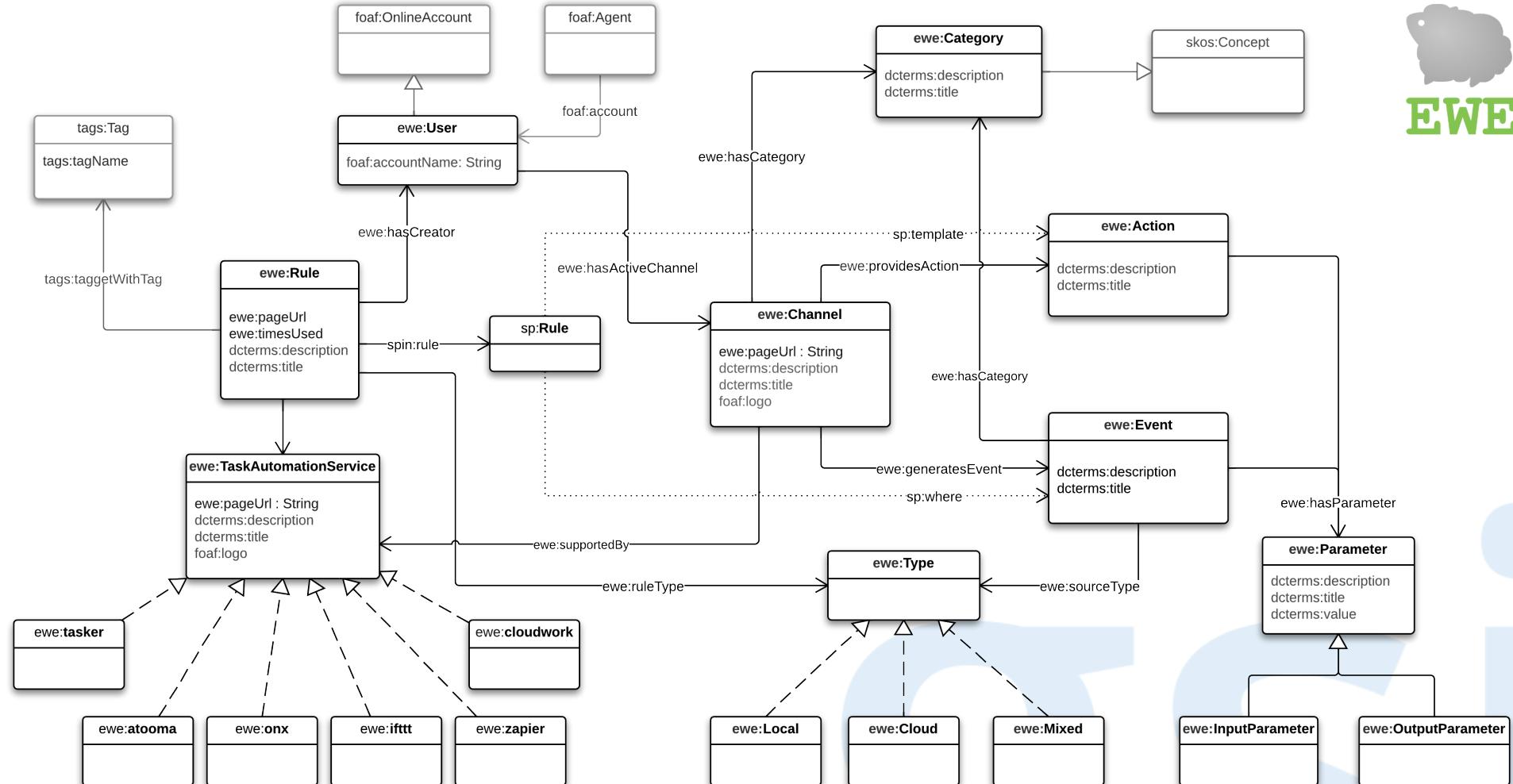
- Procesado de eventos complejo:
 - Procesar múltiples eventos con el objetivo de identificar el significado correspondiente al flujo de eventos.
 - Razonamiento temporal, causalidad, jerarquización.
- Aplicaciones:
 - *Trading*
 - Seguridad
 - Actividad de compañías
 - Detección de fraudes en tarjetas de crédito
- Ejemplo:



- Expert: motor de reglas de la suite Drools. Basado en Java.
- Fusion: módulo de la suite Drools que permite procesado de eventos complejos
- Herramienta elegida para la parte de procesado de eventos complejos en el motor de reglas: eventos de bajo nivel para producir eventos de alto nivel.

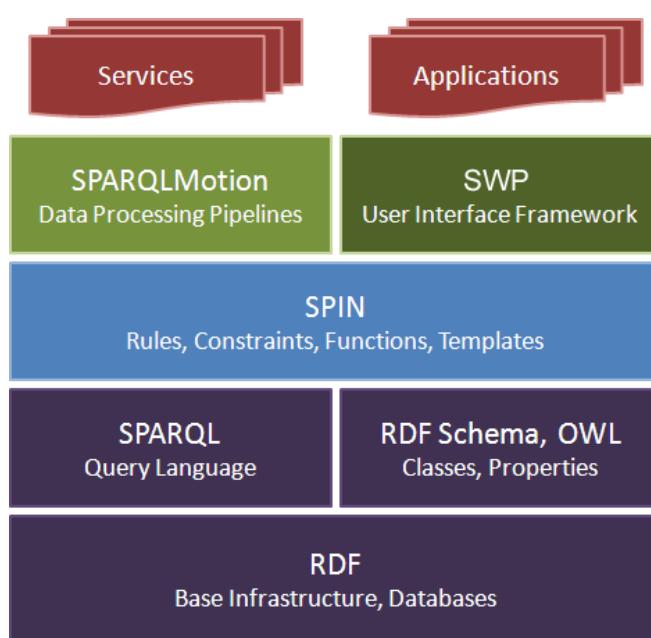
```
rule "RULE NOAH"
when
    Number( $count : intValue,intValue>=40)
        from accumulate ($w : WeatherEvent(currentCondition ==
WeatherCondition.HEAVYRAIN ) from window CuarentaDias,
                        count($w))
then
    EmailAction ea=new EmailAction("Noah","Diluvio universal","Lleva lloviendo 40 días");
    ea.send();
end
```

TECNOLOGÍAS SEMÁNTICAS: ONTOLOGÍA EWE



SPIN: SPARQL INFERRING NOTATION

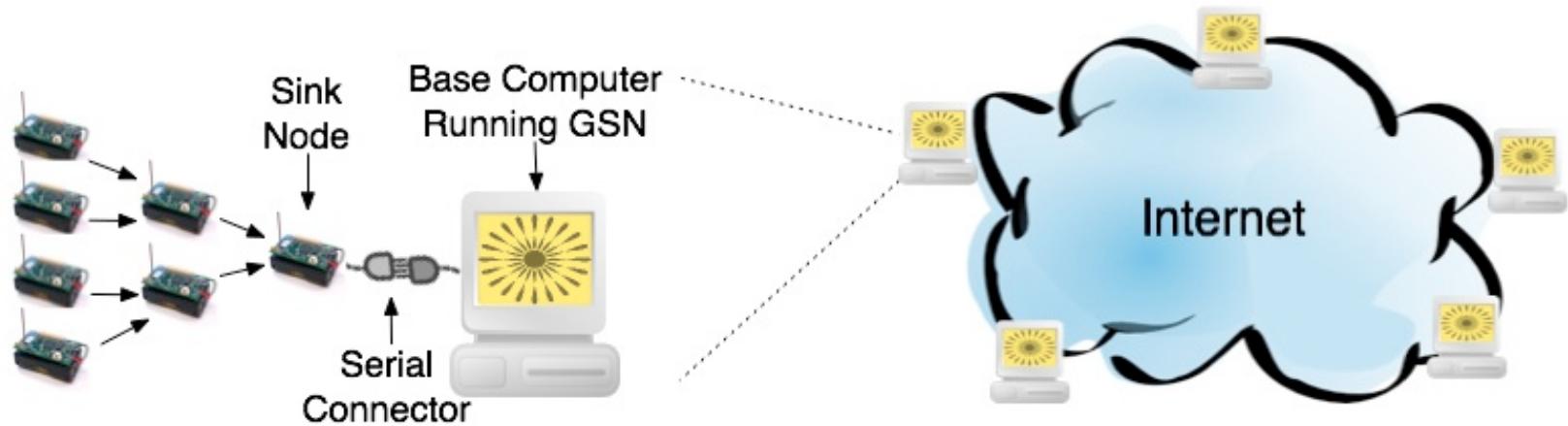
- Estándar de-facto para representar reglas SPARQL y restricciones en modelos semánticos. Incluido en el W3C.
- Usado en este proyecto para el modelado de reglas EWE y en la parte semántica del motor de reglas



- Modelado semántico de reglas de inferencia
- Motor de inferencias
- API en Java para traducir reglas SPIN/SPARQL en RDF

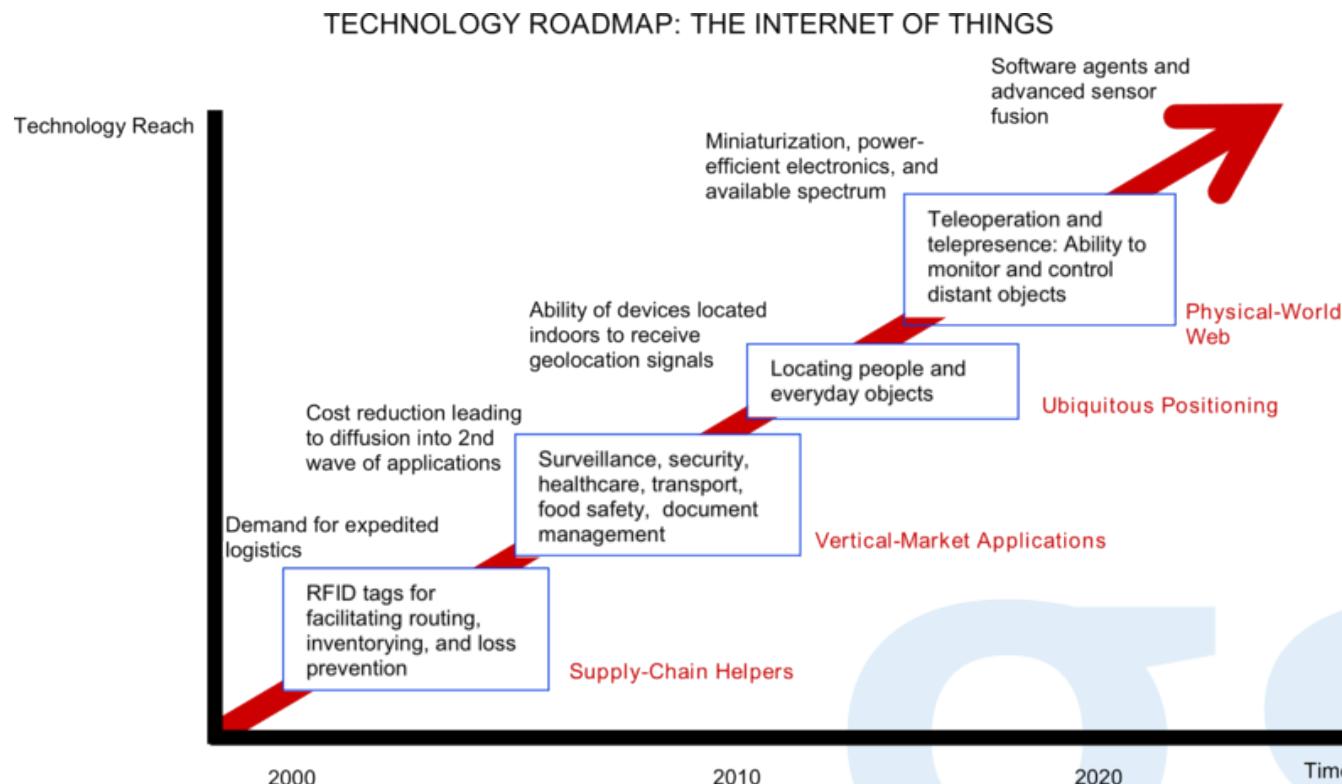
GLOBAL SENSOR NETWORK: GSN

- *Middleware flexible en java para manejar redes de sensores*
- Proyecto europeo desarrollado en el ETH (Zurich)
- Definir un xml por sensor: virtual sensor
- Interconectividad entre distintas redes de sensores a través de la red



THE INTERNET OF THINGS

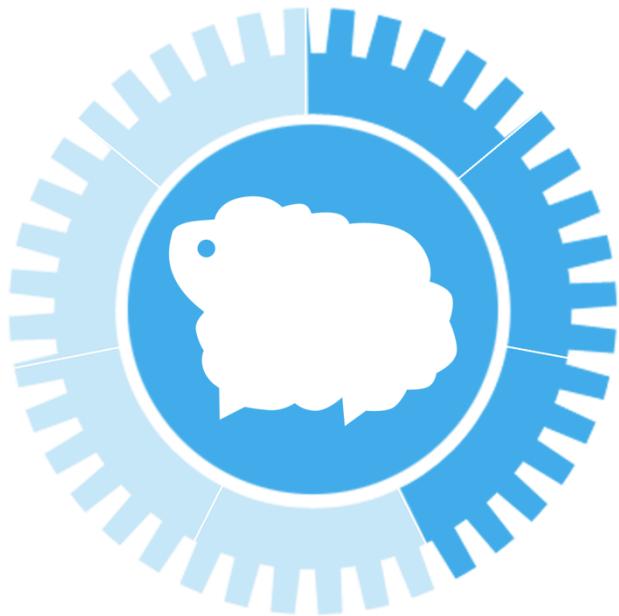
- “Todos los objetos y personas conectados a la red con un identificador único”



Source: SRI Consulting Business Intelligence

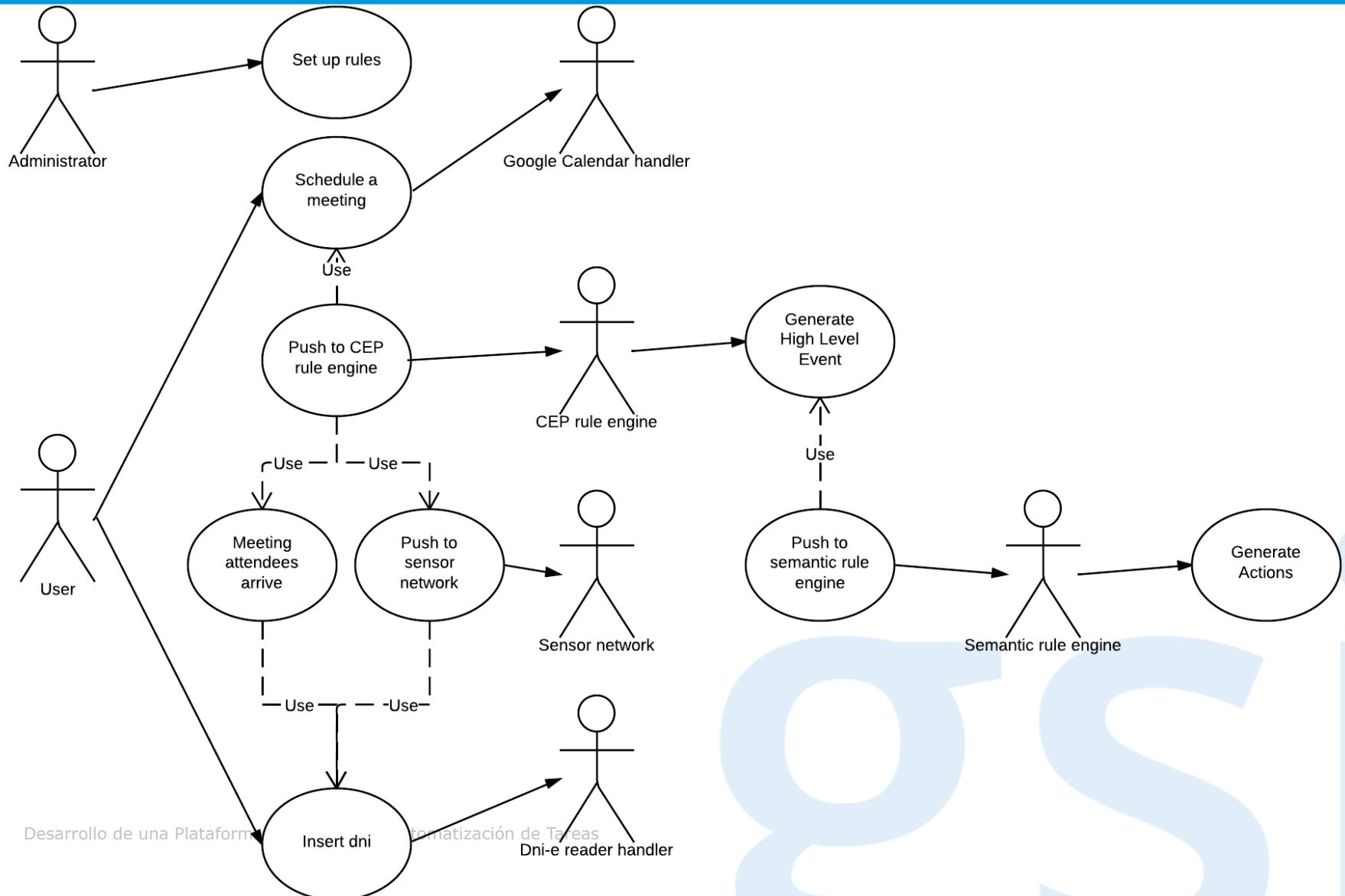
Desarrollo de una Plataforma Inteligente de Automatización de Tareas

ANÁLISIS DE REQUISITOS



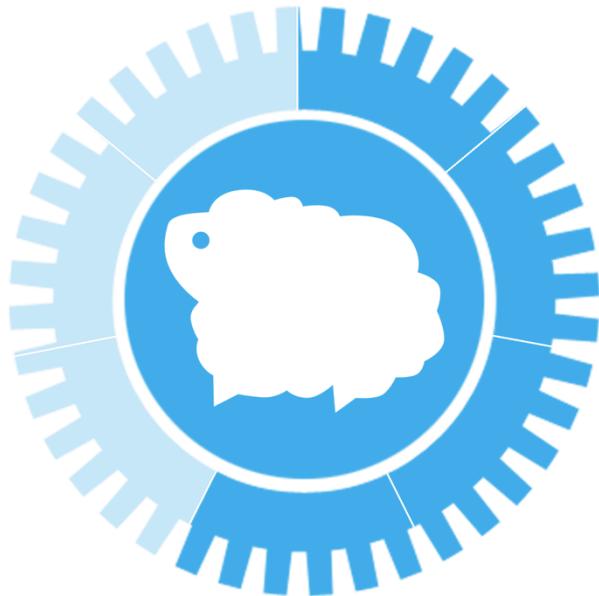
- 1. Introducción**
- 2. Estado del arte**
- 3. Análisis de requisitos**
- 4. Arquitectura**
- 5. Caso de Estudio**
- 6. Conclusiones y trabajo futuro**

CASO DE USO



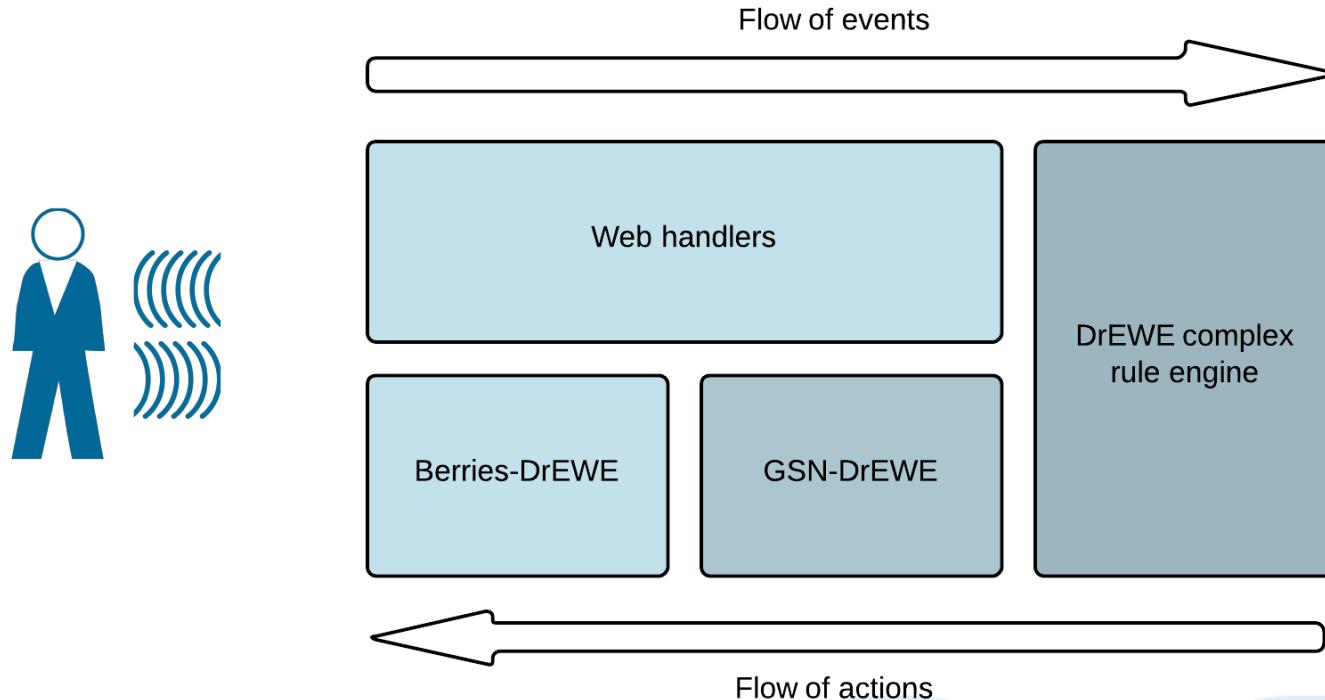
CAPTURA DE REQUISITOS

1. La arquitectura debe permitir la comunicación entre un moderado número de módulos **heterogéneos**
2. La arquitectura debe estar dividida en **capas** para separar el elevado número de funcionalidades.
3. La plataforma debe tener un motor **CEP** para procesar eventos de bajo nivel y generar eventos de alto nivel
4. La plataforma debe tener un motor **semántico** para manejar los eventos de alto nivel y generar acciones
5. Red de **sensores** para manejar los eventos de los sensores físicos
6. Plataforma **escalable**

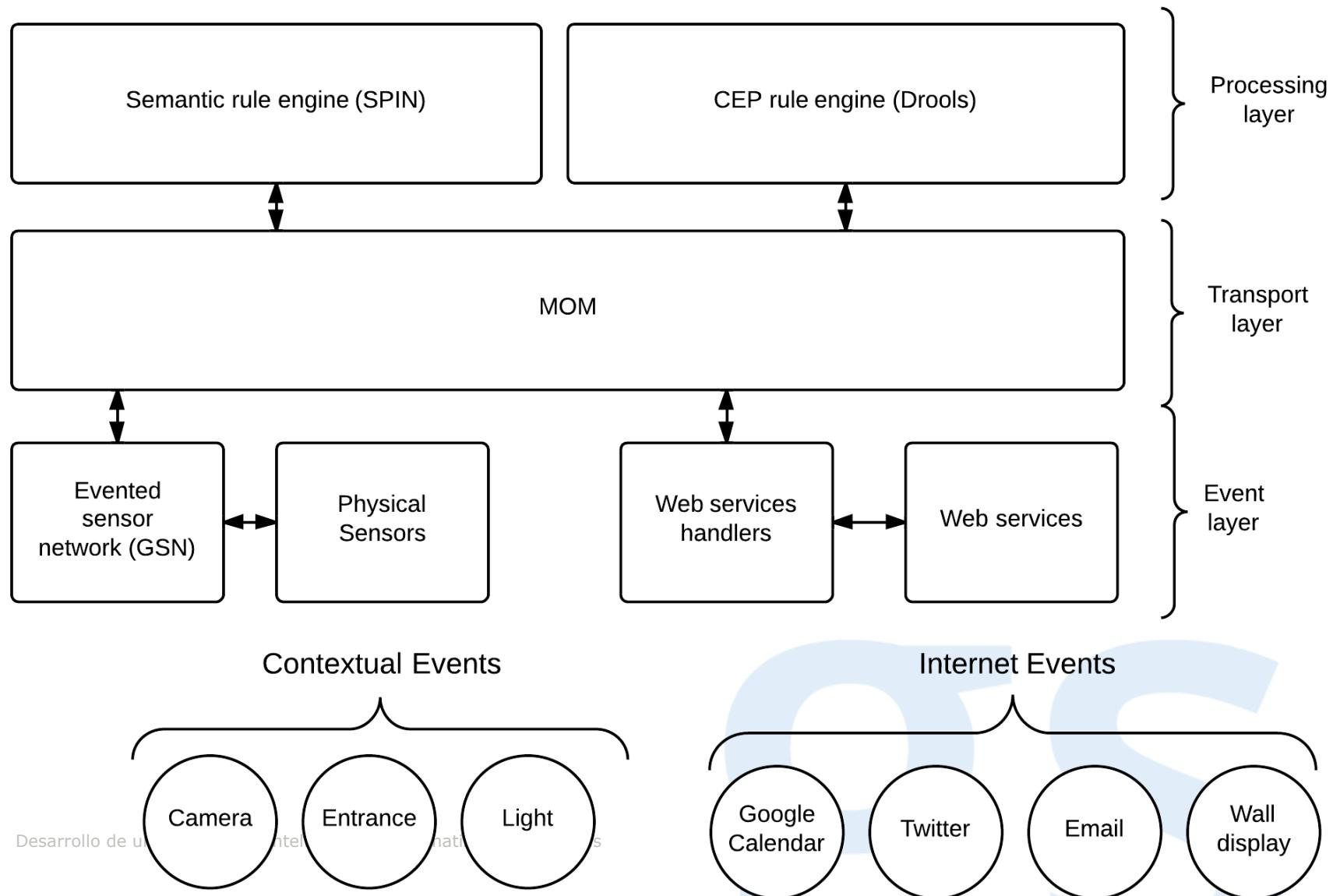


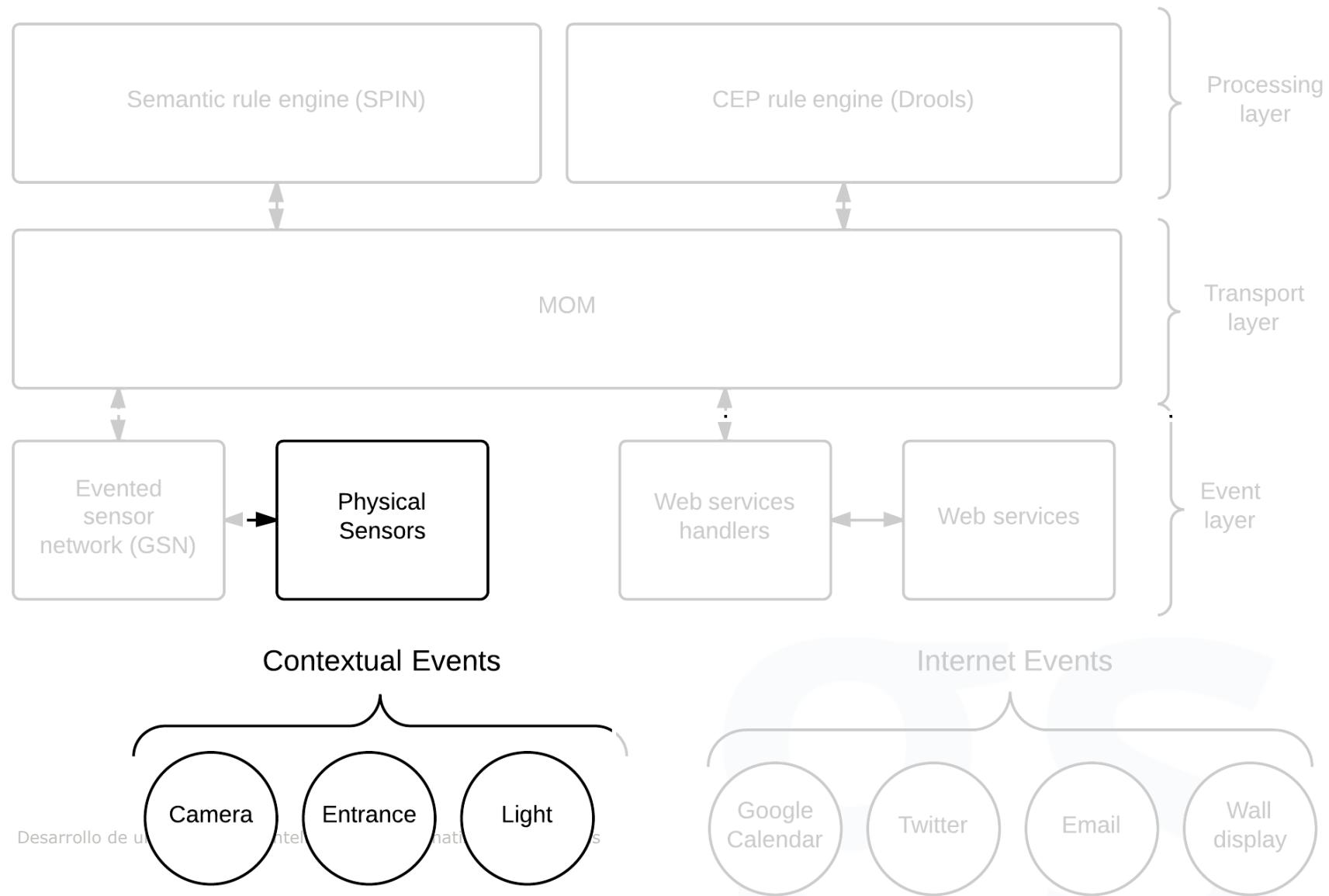
- 1. Introducción**
- 2. Estado del arte**
- 3. Tecnologías Habilitadoras**
- 4. Arquitectura**
- 5. Caso de Estudio**
- 6. Conclusiones y trabajo futuro**

DESCRIPCIÓN GLOBAL



MODELO FUNCIONAL

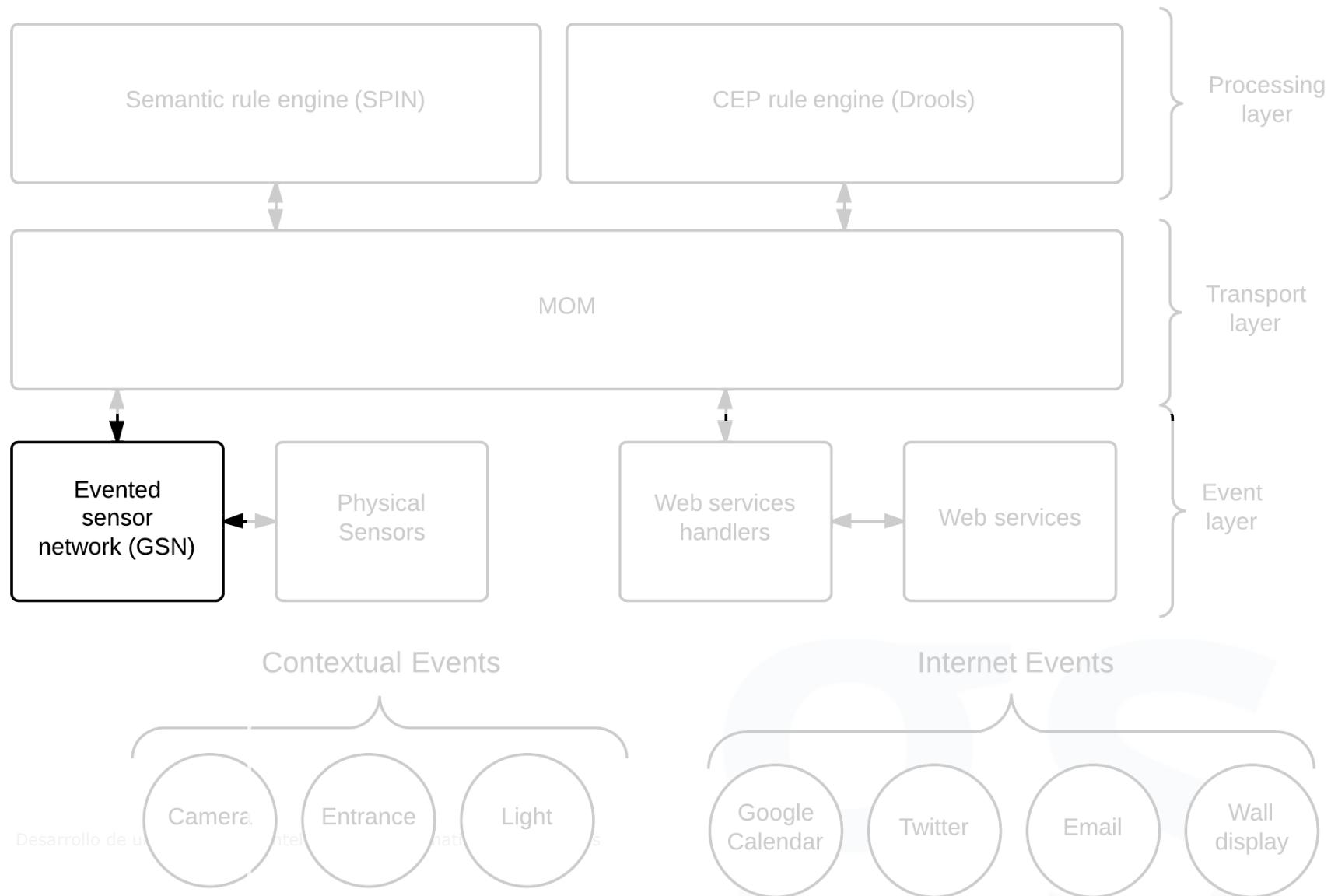




- Módulos de software encargados de recoger información del entorno y enviarla a la red de sensores. Implementados dentro de una raspberry.
- Enviar datos a la red:
 - Petición HTTP PUT a `http://<gsn-url>:22001/streaming/`
 - Parámetros:
 - `Notification-id`: identificador único por punto de entrada
 - `Data`: formato definido por la red de sensores.

```
1 <stream-element timestamp='2013-12-16 03:24:44 CET'>
2   <field name='number' type='numeric'>70588202</field>
3   <field name='name' type='string'>Carlos Crespo</field>
4 </stream-element>
```

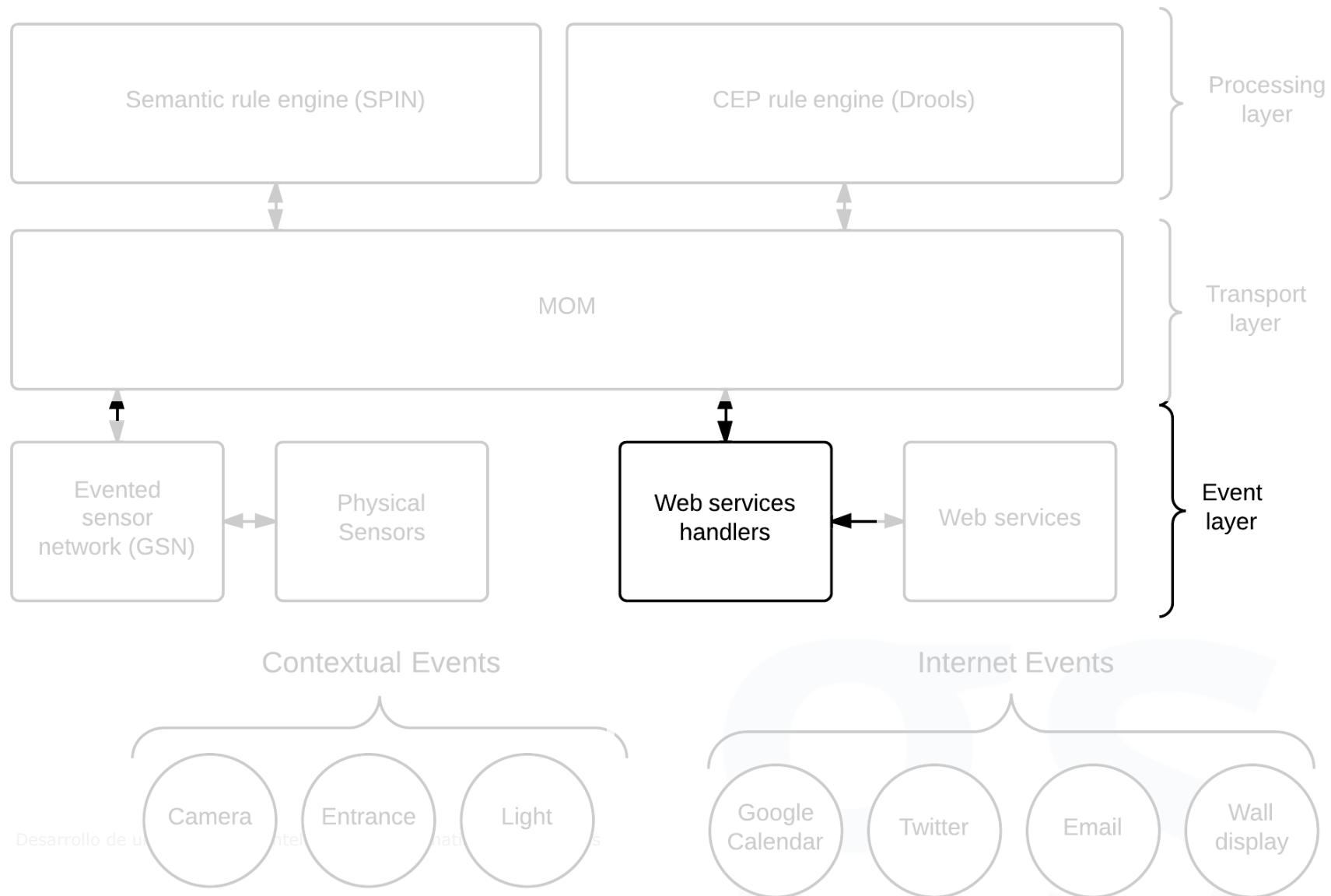
MÓDULO GSN



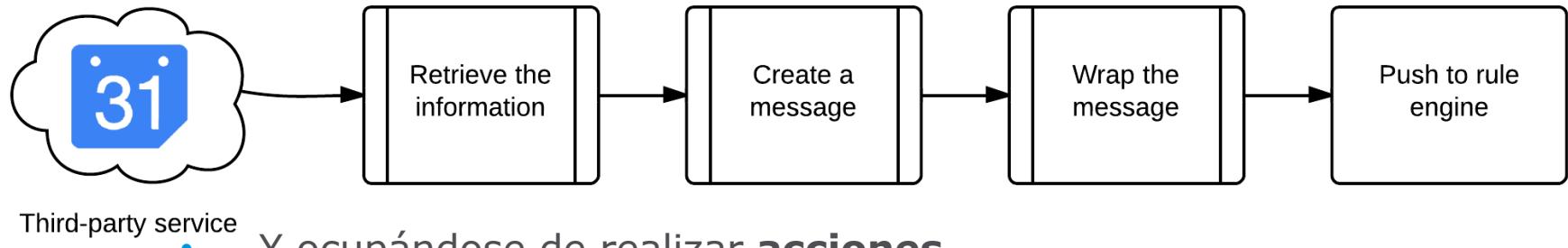
- Red de eventos a partir de red de sensores:
 - Puntos de entrada direccionados
 - Puntos de salida accesibles por el motor de reglas
 - Base de datos con *timestamps*
- Virtual sensors: xml para definir salida
 - Definimos uno por sensor físico
- Wrappers: clase java para procesar la información
 - Direct Remote Push Wrapper

GET `http://<gsn-server>:22001/gsn?request=114&name=RemoteDniVS&window=40`

WEB HANDLERS

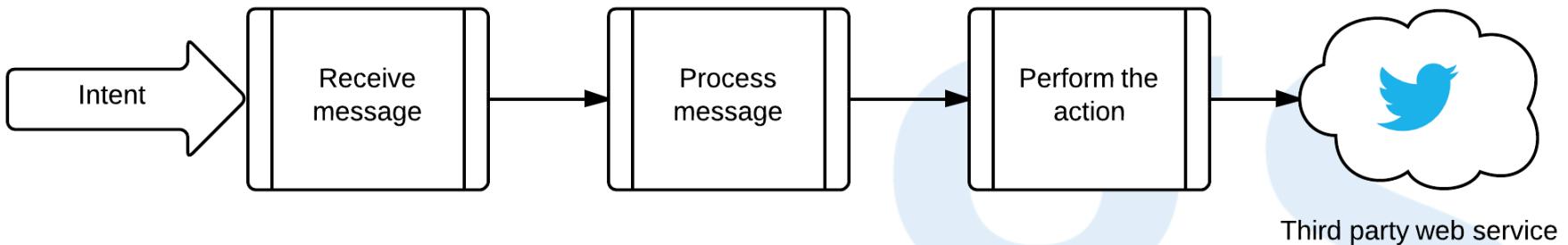


- Módulos encargados de conectar servicios externos a nuestro sistema.
 - Empaquetando información de estos servicios en forma de **eventos**



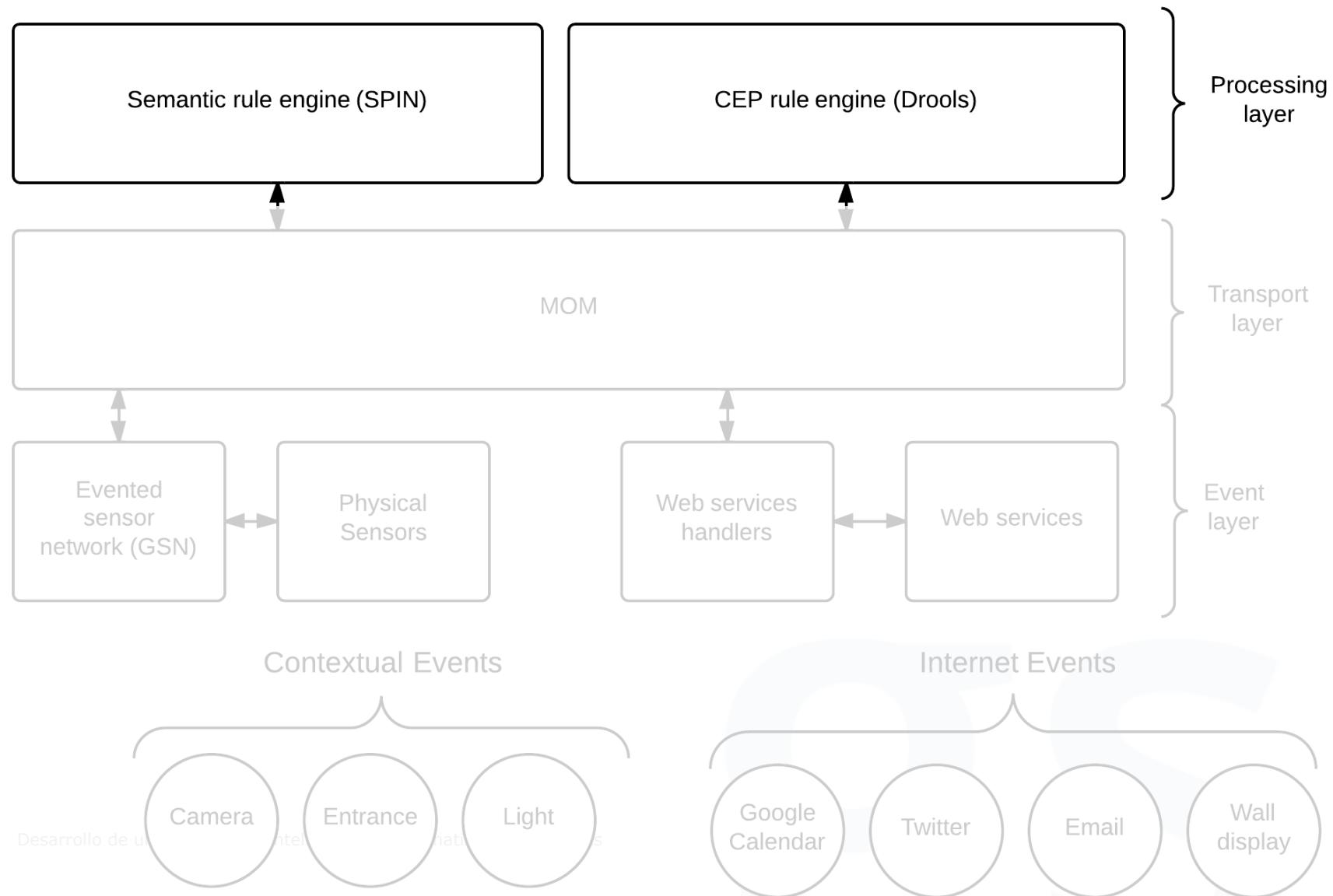
Third-party service

- Y ocupándose de realizar **acciones**



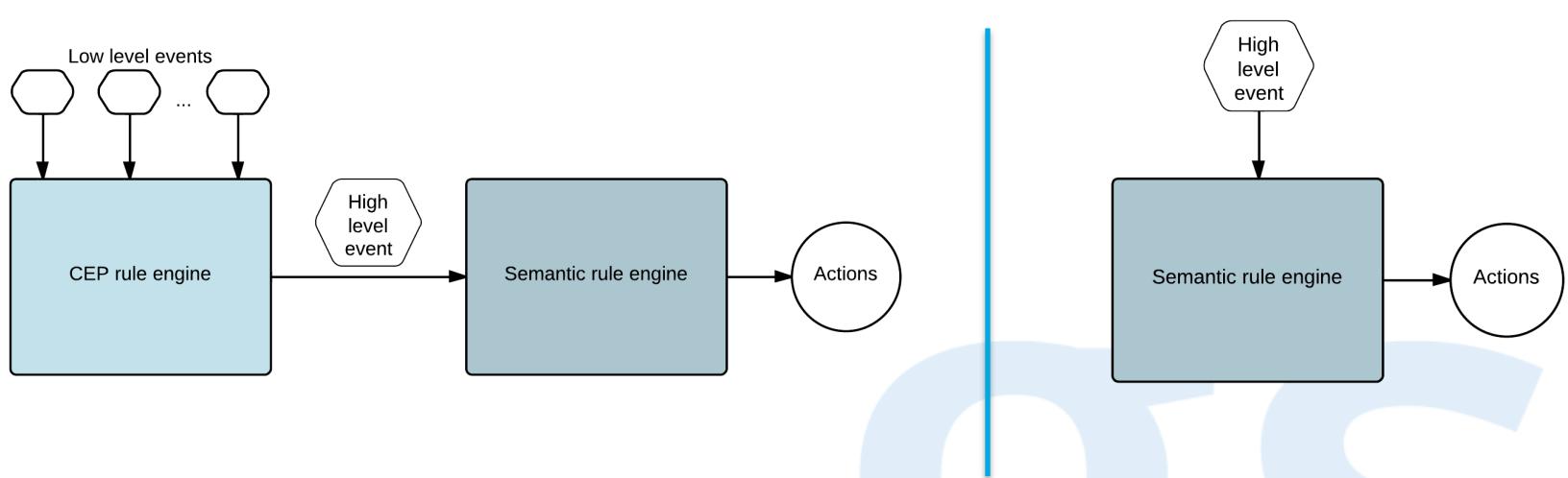
Third party web service

DrEWE COMPLEX RULE ENGINE



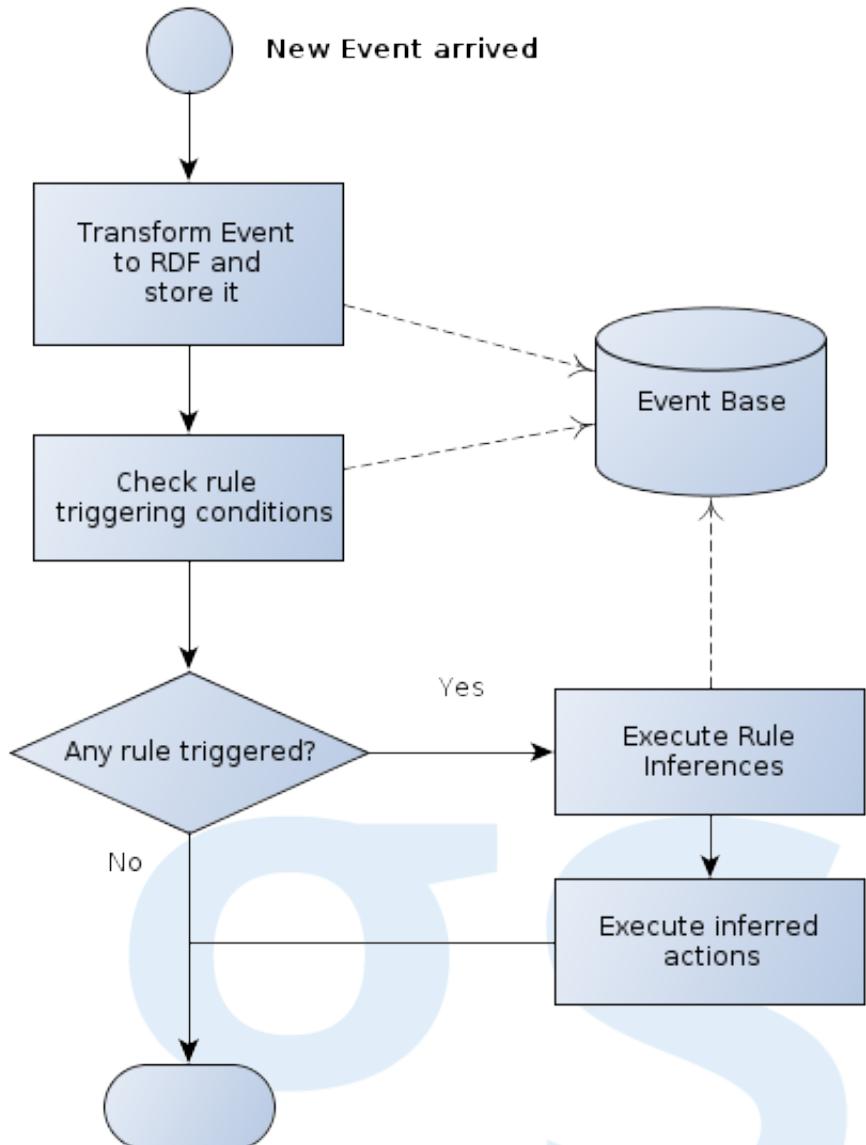
DrEWE COMPLEX RULE ENGINE

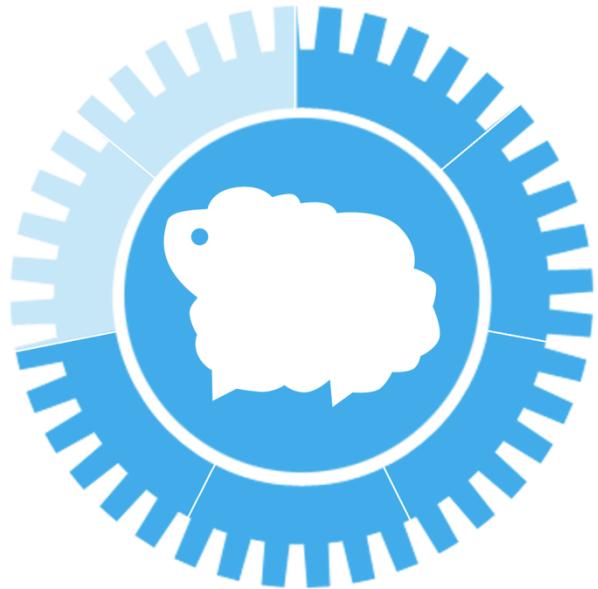
- Módulo en el que dos motores de reglas (CEP y Semántico) procesan los eventos, evalúan y disparan las reglas y lanzan *intents* para realizar acciones.
- Dos posibles escenarios de funcionamiento:



MOTOR DE REGLAS SEMÁNTICO

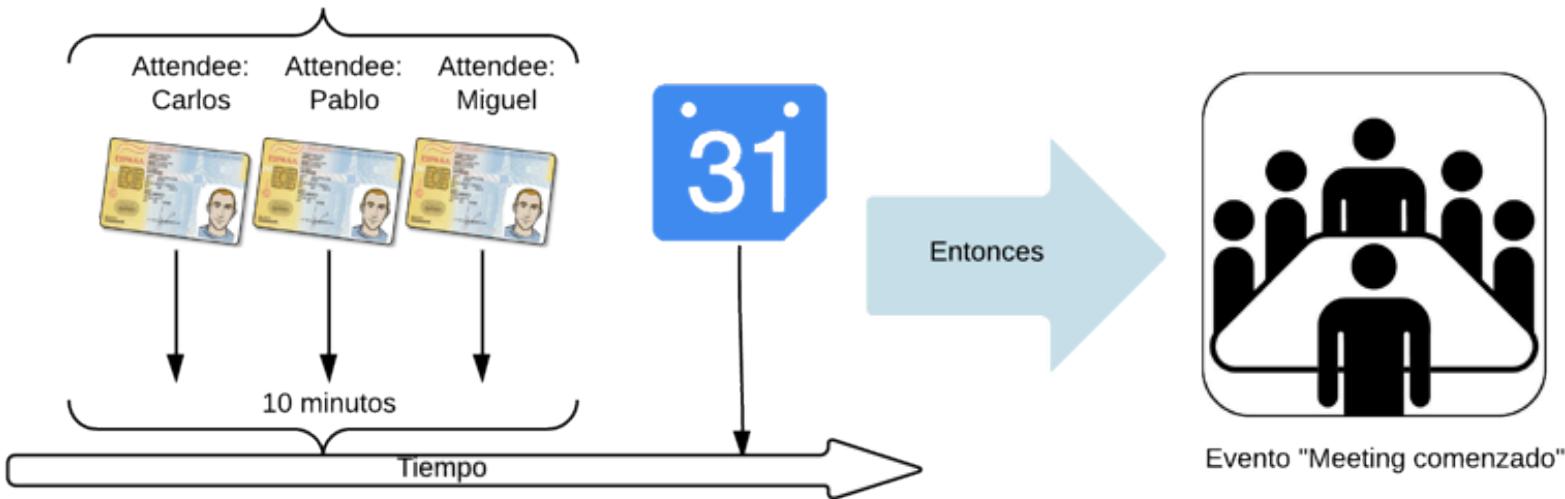
1. Transformar evento en instancia rdf
2. Almacenarlo en nuestro modelo en forma de instancia
3. Ejecutar inferencias
4. Almacenar las instancias inferenciadas en un segundo modelo (acciones)
5. Comprobar el segundo modelo en busca de nuevas instancias
6. Procesar instancias para ejecutar acciones





- 1. Introducción**
- 2. Estado del arte**
- 3. Tecnologías Habilitadoras**
- 4. Arquitectura**
- 5. Caso de Estudio**
- 6. Conclusiones y trabajo futuro**

CASO DE ESTUDIO

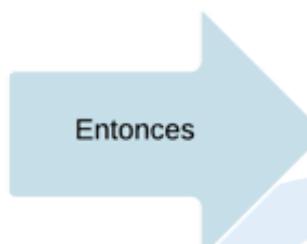


CEP

Semántico



Evento "Meeting comenzado"



Toma una foto



Muéstralala vía
muro del meeting

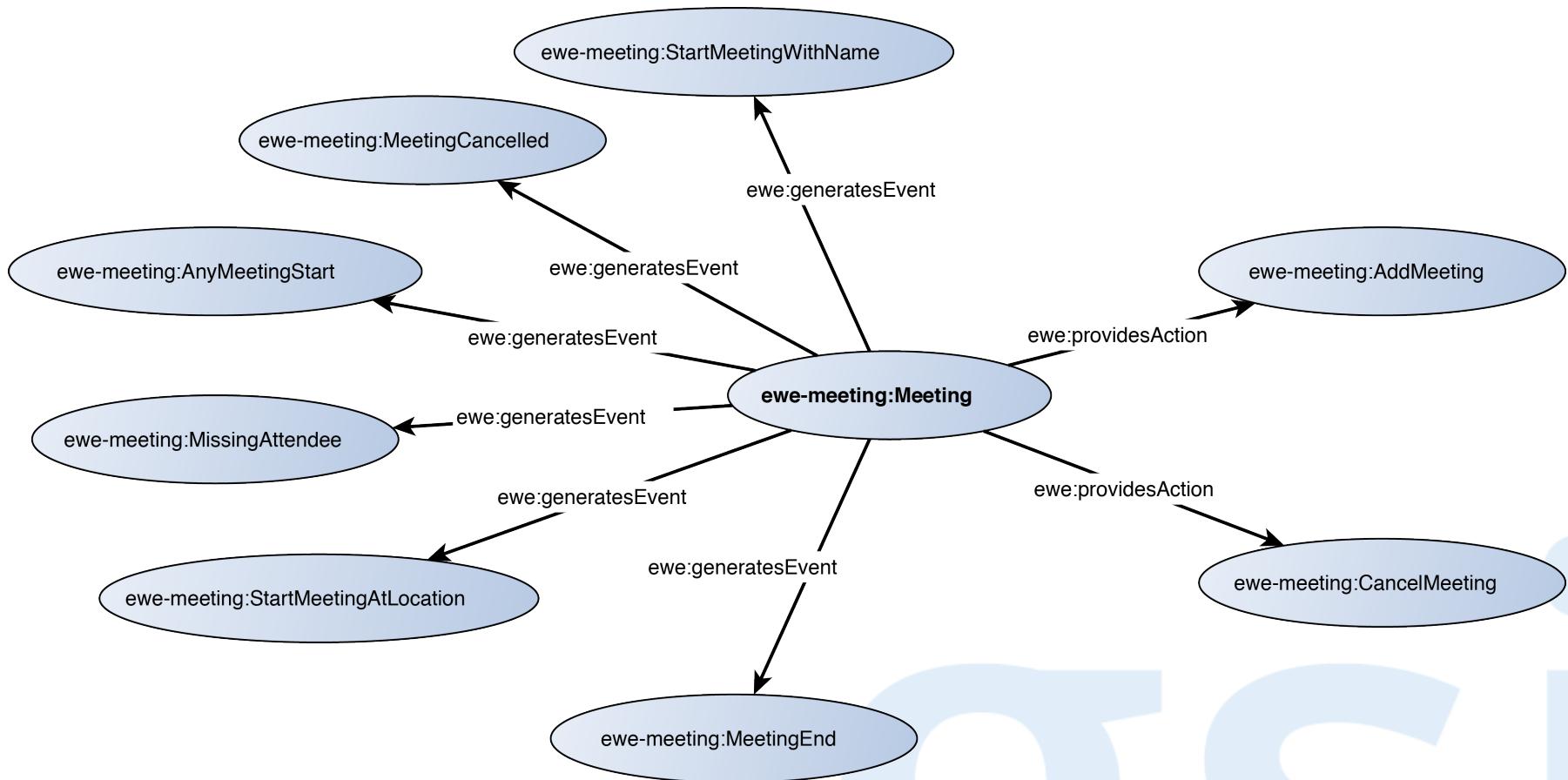


Envía un
tweet



Envía un email

DISEÑO DE CANALES PARA EWE



DEMO

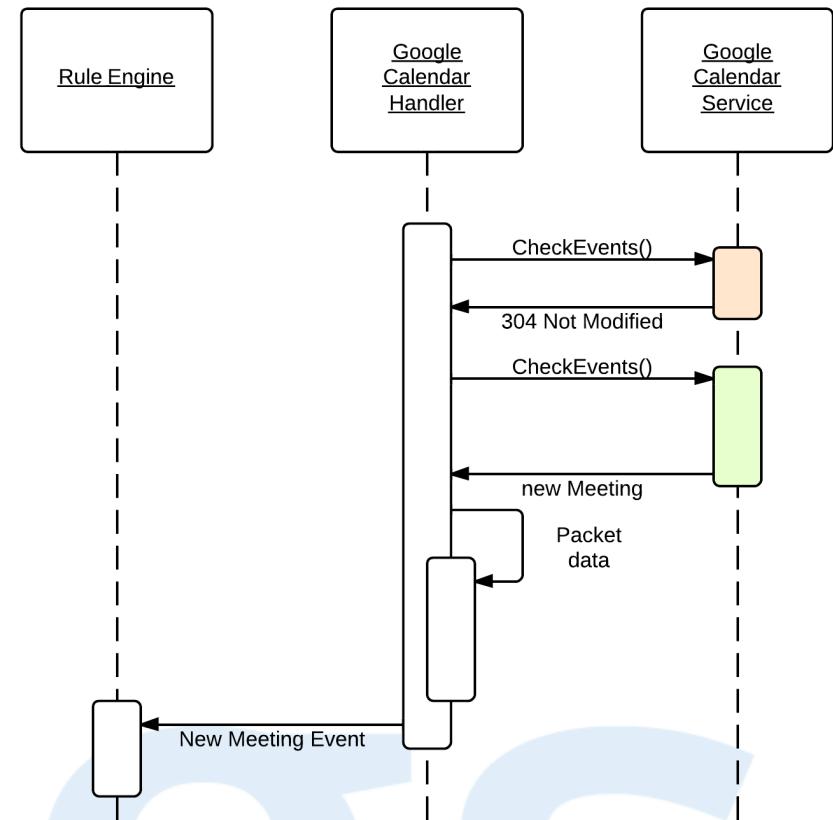


gsi

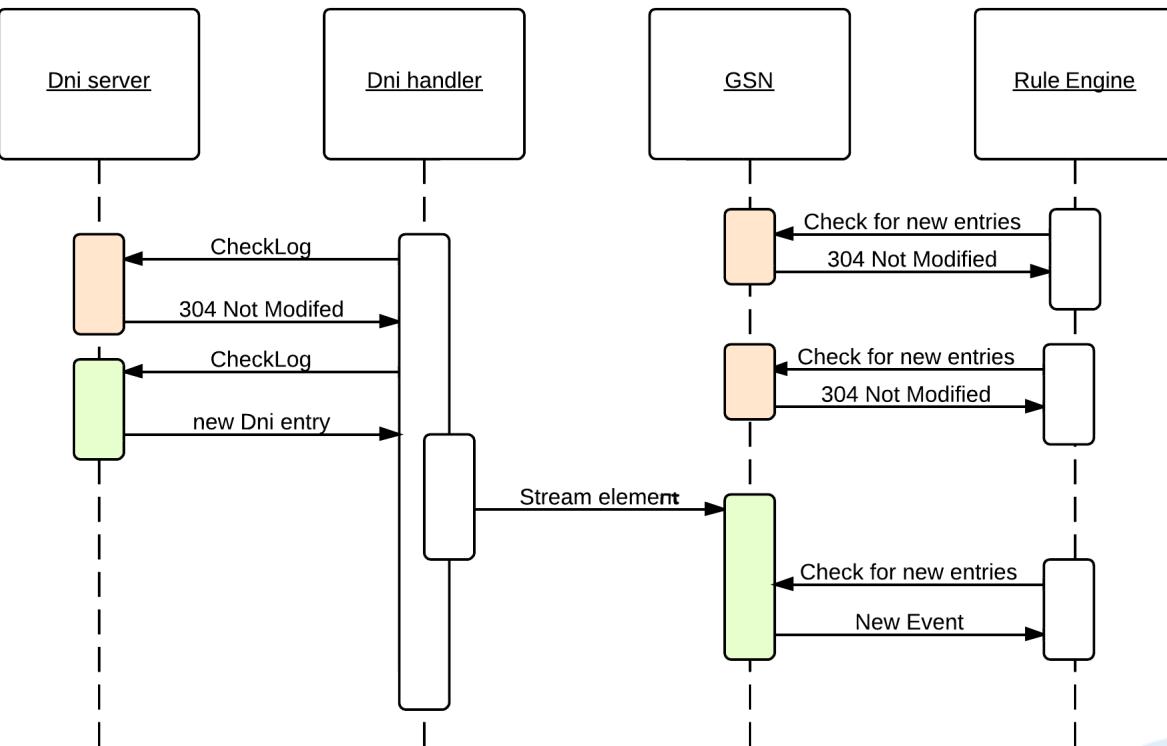
GOOGLE CALENDAR HANDLER

1. Comprobar continuamente a través de la API de google si hay nuevas entradas en el calendario

2. Cuando encuentres una nueva entrada:
 - a. Empaquetarlo en forma de evento
 - b. Enviarlo al motor de reglas



DNI EVENT HANDLER



- Servidor conectado al lector de dnis del laboratorio
- 1. Comprobar el log del servidor en busca de nuevas entradas
- 2. Si encuentra una nueva entrada, la empaqueta para ser leída por GSN
- 3. Enviar a GSN
- 4. El motor de reglas comprueba las nuevas entradas en GSN
- 5. Si hay una nueva entrada el motor de reglas la introduce en su base de creencias

CEP RULE ENGINE (DROOLS)

- Procesado de eventos de bajo nivel para generar eventos de alto nivel
- Comprobar si las 3 últimas entradas de dni se corresponden con algún meeting programado:

```
rule "Use case meeting 3 people"
    when
        $newEventReunion: CalendarEvent() from entry-point entrada
        $newEventDni : DniEvent(this during $newEventReunion) from entry-point entrada
        $newEventDni2: DniEvent(this after[ 1s,20m ] $newEventDni ) from entry-point entrada
        $newEventDni3: DniEvent(this after[ 1s,20m ] $newEventDni2 ) from entry-point entrada

        eval(checkAttendees($newEventReunion,$newEventDni,$newEventDni2,$newEventDni3))

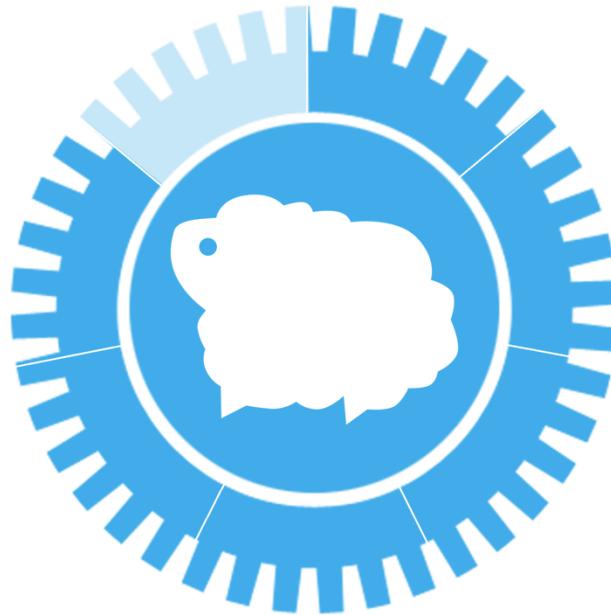
    then
        insert(new SPINEvent("meeting","Meeting started with "+$newEventDni.getName()+" , "+
        $newEventDni2.getName()+" and "+$newEventDni3.getName()));

        System.out.println("Use case rule triggered");
end
```

SEMANTIC RULE ENGINE (SPIN)

- Reglas SPIN
- Inferencias a partir de queries SPARQL
- Cuando veas una nueva instancia de tipo meeting, crea una instancia de ewe:action e introduceo al nuevo modelo:

```
CONSTRUCT{  
    ewe:action dcterms:title "wallDisplay" .  
    ewe:action dcterms:description ?description .  
}  
WHERE {  
    ?event dcterms:title "meeting" .  
    ?event dcterms:description ?description .  
}
```



- 1. Introducción**
- 2. Estado del arte**
- 3. Análisis de requisitos**
- 4. Arquitectura**
- 5. Caso de Estudio**
- 6. Conclusiones y trabajo futuro**

CONCLUSIONES

1. Plataforma de automatización de tareas que admite reglas CEP y reglas semánticas
2. Arquitectura modular y escalable
3. GSN como red de sensores
4. Soporte para reglas EWE
5. Probada arquitectura para nuestro caso de uso
6. Modelo multiusuario

1. Editor visual de reglas para DrEWE
2. Integrar más servicios web y más sensores físicos
3. Implementarla dentro de smartphones
4. Reglas que implementen *Context awareness*
5. Extender el modelo de reglas multiusuario
6. Reglas sociales
7. Comprobar colisión entre reglas
8. Incorporación de un asistente personal (agente proactivo)
9. Interacción con la plataforma a través de lenguaje natural

PREGUNTAS

