ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

Integración de un Sistema de Control para una Plataforma de la Internet de las Cosas (Integration of a Control Dashboard over a Smart City Internet of Things Platform)

Para acceder al Titulo de

Graduado en Ingeniería de Tecnologías de Telecomunicación

Autor: Ignacio Fernández- Regatillo Rubio

Septiembre - 2016

E.T.S DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Ignacio Fernández-Regatillo Rubio
Director del TFG: Luis Sánchez González
Título: "Integración de un Sistema de Control para una plataforma de la Internet de las Cosas"
Title: "Integration of a Control Dashboard over a Smart City Internet of Things Platform"

Presentado a examen el día: 16 de Noviembre de 2016

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre): Sanz Gil, Roberto
Secretario (Apellidos, Nombre): Sánchez González, Luis
Vocal (Apellidos, Nombre): Villa Benito, Enrique

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del TFG (sólo si es distinto del Secretario)

V^{o} B^{o} del Subdirector

Trabajo Fin de Grado N^{o} (a asignar por Secretaría)

AGRADECIMIENTOS

Quisiera agradecer en primer lugar a mi familia tanto por la educación como por los valores que me han inculcado durante todos estos años, así como el apoyo mostrado a lo largo de estos años de carrera.

Mención especial a Marina, por todos estos años que me ha estado aguantando, siempre con una actitud positiva. De verdad, gracias, por apoyarme, cuidarme y sobre todo, confiar siempre en mí.

Igualmente, a todos mis compañeros, con los que he tenido la posibilidad de compartir buenos y malos momentos a lo largo de toda esta etapa universitaria.

Por ultimo, agradecer a todo el cuadro docente de la Escuela Técnica Superior de Ingenieros Industriales y Telecomunicación, por los conceptos y valores que me han ido enseñando a lo largo de la formación como Ingeniero, especial mención a mi tutor, Luis Sánchez, por haberme orientado y ayudado a la hora de realizar este Trabajo, aportándome parte de sus conocimientos, necesarios para la consecución del mismo.

Gracias por estos años.

RESUMEN

El progresivo desplazamiento de la población hacia las zonas urbanas, implica que los retos a los que se enfrentan hoy las grandes ciudades sean cada vez mayores, con el objetivo primordial de proporcionar un entorno de vida sostenible y la necesidad actual de orientar nuestra vida hacia la sostenibilidad. Por ello, el principal objetivo de este Trabajo Fin de Grado no es otro que el de contribuir a la mejora de la eficiencia de los servicios de los entornos urbanos y conseguir un desarrollo más sostenible de los mismos, mediante el diseño e implementación de un Sistema de Control que ofrezca una interfaz visual en la que se concentre toda la información relativa a una infraestructura de dispositivos de IoT. Este sistema se ha probado en particular para la infraestructura IoT desplegada en el proyecto SmartSantander.

Una adecuada gestión de los datos, nos llevará a desarrollar aplicaciones que utilicen toda la información que proporcionan de forma inteligente y con ello dar respuesta a los retos sociales que imperan en la actualidad. Sin embargo, es necesario poder tener un control de la infraestructura subyacente a través del cual el proveedor de la infraestructura pueda en todo momento detectar funcionamientos anómalos y corregirlos oportunamente. Para ello, el objetivo que se aborda, es el de crear una Plataforma de Control (Dashboard) que facilite la gestión de la infraestructura IoT subyacente a una Smart City.

Así, el alcance de este trabajo será conseguir la integración de un sistema de control, que operará sobre la plataforma SmartSantander, con el fin de que los gestores cuenten con un cuadro de control visual en que se concentre la información relativa a toda la infraestructura, facilitando y agilizando de este modo su gestión, mediante una interfaz visual atractiva, a través de la cual se presentará la infraestructura desplegada.

ABSTRACT

The progressive displacement of the population to urban areas implies that the challenges of today's large cities are growing. Its main objective is to supply a sustainable environment and the real necessity to orientate our lives towards sustainability. Due to this, the primary purpose of this end-of-degree Project is to contribute to the improvement of the urban areas efficiency services, and to achieve a more sustainable development for them, through the design and implementation of a Control System that provides a visual interface in which all the information regarding an IoT device infrastructure is gathered. This system has been particularly tested for the IoT infrastructure deployed in the SmartSantander Project.

An adequate data management will lead us to develop applications which use all the information they provide in an intelligent way, and with that, we will be able to provide answers to the social challenges that prevail today. Nevertheless, it is necessary to have control over the underlying infrastructure through which the infrastructure supplier can at all times detect malfunctions and correct them in due course. For that purpose, the addressed objective is to create a Control Platform (Dashboard) that eases the management of the underlying IoT infrastructure of a Smart City.

Thereby, the scope of this Project is to achieve the inclusion of a control system which will operate over the SmartSantander platform, in order to provide the agents with a visual control box that assembles the information related to the entire infrastructure, facilitating and speeding up its management through an attractive visual interface, whereby the deployed infrastructure will be presented.

Índice

Índice de Figuras

Índice de Tablas

Listado de Acrónimos

API	Interfaz de Programación de Aplicaciones
DB	Base de Datos
DSL	Domain Specific Language
GFS	Google File System
HDFS	Hadoop Distributed File System
HTTP	Protocolo de Transferencia de Hipertexto
IaaS	Infraestructura como Servicio
IoT	Internet of Things
IP	Protocolo de Internet
JSON	Notaciones de Objetos JavaScript
M2M	Machine to Machine
MIT	Massachusetts Institute of Technology
PaaS	Plataforma como Servicio
REST	Transferencia de Estado Representacional
SaaS	Software como Servicio
TCP	Protocolo de Control de Transmisión
TIC	Tecnologías de la Información y la Comunicación
XML	Lenguaje de Marcas Extensible

Capítulo 1. **INTRODUCCIÓN**

1.1 Motivación

En las últimas décadas, estamos asistiendo a nivel mundial, a un progresivo desplazamiento de la población hacia las zonas urbanas, o lo que es lo mismo, a una progresiva urbanización. Este hecho, implica que los retos a los que se enfrentan hoy en día las grandes ciudades sean cada vez mayores. En este sentido, estos retos se pueden resumir en un objetivo único, proporcionar un entorno de vida urbano sostenible. Las ciudades inteligentes o Smart Cities, no son sino el resultado de la necesidad actual de orientar nuestra vida hacia la sostenibilidad.

Cada vez más ciudades implementan las infraestructuras basadas en la Internet de las Cosas. El IoT es, entre otros, uno de los novedosos habilitadores de las Smart Cities. El avance en la microelectrónica y la evolución que están alcanzando las tecnologías inalámbricas, han permitido la integración del mundo físico con el mundo digital. El IoT se compone de dispositivos tanto fijos como móviles que interactúan entre ellos. Las infraestructuras basadas en la Internet de las Cosas permiten monitorizar el correcto funcionamiento de sus estructuras, así como adaptar su funcionamiento ante nuevos eventos de forma más flexible.

Los objetivos de este Trabajo Fin de Grado están embebidos en estos grandes objetivos que las grandes ciudades tienen, y que no son otros que mejorar la eficiencia de los servicios de los entornos urbanos y conseguir un desarrollo más sostenible de los mismos.

Una de las claves del concepto de Smart City es la capacidad, que la evolución de las Tecnologías de la Información y las Comunicaciones ha permitido, de acceder a una cantidad cada vez mayor de información acerca del entorno urbano y todos los subsistemas que en él coexisten. Sin embargo, es necesario poder gestionar y manejar tal cantidad de información de tal manera que se pueda sintetizar y seleccionar toda la información y poder hacer un buen uso de la misma. Una adecuada gestión de los datos nos llevará a desarrollar aplicaciones que utilicen toda la información que proporcionan de forma inteligente y con ello afrontar los retos sociales y medioambientales anteriormente referidos.

Si uno de los retos a nivel mundial, es conseguir ciudades sostenibles y eficientes, en las que los ciudadanos pueden interactuar de forma dinámica con la administración y los servicios, uno de los retos de este Trabajo Fin de Grado es contribuir en alguna medida a ello.

El control de la operativa de una plataforma de la Internet de las Cosas, requiere la observación de varios sistemas y componentes que se encuentran distribuidos en una arquitectura de múltiples niveles. Por todo ello, el objetivo del Trabajo que se aborda, es el de crear una Plataforma de Control (Dashboard) que facilite la gestión de la Infraestructura IoT subyacente a una Smart City.

Asumir una tarea de gestión de una infraestructura que puede llegar a contener millares de dispositivos, se puede convertir en un proceso muy engorroso, que no haría sino limitar la capacidad de reacción y análisis del gestor responsable de esta plataforma si se hace de forma manual. La infraestructura desplegada por el Proyecto SmartSantander, nos ofrece la posibilidad de desarrollar e investigar nuevos protocolos y tecnologías de red en el entorno de las redes de sensores. De hecho, el ser la ciudad de Santander probablemente la ciudad que cuenta con más sensores del mundo (se han desplegado más de 20000 dispositivos en la ciudad y entornos para diferentes ámbitos de aplicación) animó a la realización de este proyecto que se presenta.

Con este Proyecto se pretende conseguir la integración de un sistema de control, que operará sobre la plataforma SmartSantander, con el fin de que los gestores cuenten con un cuadro de control visual en el que se concentre la información relativa a toda la infraestructura.

El cuadro de control diseñado será accesible vía web y el sistema de control estará basado en tres herramientas propietarias de Elastic: Logstash(módulo de captación de datos); Elasticsearch (módulo de almacenamiento) y Kibana(módulo de representación).

1.2 Objetivos del proyecto

El principal objetico de este Trabajo es:

Diseñar e implementar un Sistema de Control que ofrezca una interfaz visual en el que se concentre toda la información relativa a la infraestructura de dispositivos IoT desplegada en el proyecto SmartSantander.

Este Sistema de Control permitirá que se opere sobre la plataforma de SmartSantander, que los gestores cuenten con un cuadro de mando que facilite y agilice su gestión, así como una interfaz visual atractiva a través de la cual presentar la infraestructura desplegada.

Para conseguir este objetivo, se ha procedido a:

- Conocer la tipología de datos proporcionada por los sensores. Se ha estudiado el modelo de datos utilizado por SmartSantander a la hora de generar la información. Conocer y comprender el modelado de la información es fundamental para su posterior tratamiento por parte del Sistema de Control desarrollado en este Trabajo.

- Capturar los datos que nos proporcionan para proceder a su análisis. Para ello se han evaluado las diferentes opciones permitidas por la plataforma de SmartSantander y seleccionada la que mejor se ajusta a las tecnologías empleadas por el Sistema de Control que se ha desarrollado.

- Estudiar y comprender cada una de las herramientas integradas en el Sistema de Control. Este estudio ha permitido que se hayan adquirido

los conocimientos necesarios en la implementación y gestión de las mismas, para poder explotar sus posibilidades y su utilización, que nos lleven a una exitosa puesta en funcionamiento de este Trabajo.

- Integrar y desarrollar el Sistema de Control ha sido la última de las tareas desempeñadas.

1.3 Resumen ejecutivo

El Trabajo Fin de Grado que se presenta está dividido en cinco capítulos. En el primer capítulo, aparte del presente resumen ejecutivo, se lleva a cabo una exposición de las motivaciones que han llevado al desarrollo del mismo, así como los objetivos que se pretenden con el mismo.

En el segundo capítulo, se describe el marco tecnológico en el que se ha basado este Trabajo, abordando en primer lugar conceptos y estudios de la IoT, Big Data, así como la descripción del Proyecto SmartSantander. En segundo lugar, se describen y estudian las herramientas Logstash, Elasticsearch y Kibana, culminando el capítulo con la descripción somera de las plataformas cloud.

El tercer capítulo se centra en el Sistema de Control de Infraestructuras IoT, incluyendo la arquitectura del sistema, el acceso a las observaciones de SmartSantander y desarrollando los módulos de captación de datos, el de almacenamiento, así como el módulo de representación. Desarrollando finalmente en este capítulo el despliegue del sistema y los Dashboards de control.

A continuación, en el capítulo cuarto se especifican las directrices para la extensibilidad del sistema mediante un pequeño tutorial explicando las distintas posibilidades de ampliación de los distintos Dashboards.

En el quinto y último capítulo se expondrán todas las conclusiones alcanzadas tras analizar los resultados, así como las posibles futuras líneas de trabajo que quedan abiertas.

Capítulo 2. **Marco del Desarrollo**

2.1 Internet de las Cosas, Big-Data y SmartSantander

2.1.1 Internet de las Cosas

La Internet de las Cosas (IoT, por sus siglas en Inglés) es un concepto que se refiere a la interconexión digital de los objetos cotidianos con internet.

La IoT, se basa en sensores, en redes de comunicaciones y en una inteligencia que maneja todo el proceso y los datos que se generan. Los sensores son los sentidos del sistema y para que puedan ser empleados de forma masiva, deben tener bajo consumo y coste, un reducido tamaño y una gran flexibilidad para su uso en todo tipo de circunstancias.

Figura 1. Descripción Gráfica del Mundo Interconectado

Este concepto, fue propuesto por Kevin Ashton en 1999 en el Auto-ID Center del MIT[1], donde se llevaban a cabo investigaciones en el campo de la identificación por radio frecuencia y tecnologías de sensores.

[1] MIT: "Massachusetts Institute of Technology"

La IoT, en esencia, es un sistema de comunicación de máquina a máquina, pero sus implicaciones van mucho más allá que simplemente la interconexión de dispositivos.

Al igual que ocurre con la Internet, su potencial se desvela cuando se plantean las aplicaciones que puede llegar a soportar. Los datos de la IoT crean oportunidades para analizar la situación de un entorno en un momento dado y actuar en consecuencia, habida cuenta de que no sólo se interconectan dispositivos sensores sino también actuadores.

Desde hace muchos años se viene trabajando con la idea de hacer un poco más interactivos todos los objetos de uso cotidiano. Así la IoT, permite que objetos, que antiguamente se conectaban mediante circuitos cerrados, como comunicadores, cámaras, sensores y demás, (lo que se ha venido a denominar Internet-of-Things) ahora se puedan exportar sus servicios de manera global a través de la red de redes.

Se podría pues definir, como una red que interconecta objetos físicos valiéndose de Internet. Se vale de un hardware especializado que le permite no sólo la conectividad a internet, sino que también programa eventos específicos en función de las tareas que le sean dictadas remotamente.

No hay un tipo específico de objetos conectados a la IoT, en realidad son objetos que funcionan como sensores y objetos que realizan acciones activas. Cada uno de los objetos conectados a Internet, tiene una dirección IP específica y mediante esa dirección IP puede ser accedido para recibir instrucciones. Así mismo, puede contactar con un servidor externo y enviar los datos que recoja.

La IoT, no ha encontrado, al contrario que otras tecnologías, su foco de expansión en el mercado del consumo. Si bien Apple y Google han dado algunos pasos con tecnologías como Home Kit y Android Home, es en el sector privado, donde el internet de las cosas se está haciendo cada vez más popular:

- Cada vez más ciudades implementan las infraestructuras basadas en la IoT: control de semáforos, puentes, vías de tren, permitiendo de este modo monitorear el correcto funcionamiento de sus estructuras, así como adaptar más flexiblemente su funcionamiento ante nuevos eventos. Las aplicaciones IoT que podemos encontrar actualmente en las ciudades inteligentes son variadas, como aparcamientos inteligentes, control de tráfico, alumbrados inteligentes…

- Casas inteligentes, en las que el objetivo es lograr mejores niveles de confort mientras se disminuye el gasto total.

- En la industria de producción en masa, cada vez más empresas centralizan el control de la infraestructura en lo que se conoce hoy en día como Factoría 4.0

- Además, en el sector de la salud y del control ambiental, también se vislumbra un gran potencial para la IoT (p.ej. telemedicina, control de incendios, etc).

Si bien hay muchas maneras en que la IoT podría afectar a la sociedad y los negocios, estas se pueden condensar en tres grandes áreas que son comunes a cualquier aplicación de las que se intuye tenga la IoT, a saber, comunicación, control y ahorro de costes.

Un punto importante de la IoT, es que requerirá habilidades y conocimientos específicos combinados (tecnológicas, matemáticas o de funcionamiento de las organizaciones), en un perfil que hasta ahora no existía. Esto abrirá nuevas oportunidades de trabajo, como son los llamados gestores de datos que requieren conocer y manejar las nuevas herramientas para la captura, el análisis y el aprovechamiento de los datos.

2.1.2 Big Data

Se denomina Big Data a la gestión y análisis de enormes volúmenes de datos que no pueden ser tratados de manera convencional, al superar los límites y capacidades de las herramientas de software que habitualmente utilizamos para la captura, gestión y procesamiento de datos.

Es un concepto que engloba tecnologías y servicios de infraestructuras que han sido creados para dar solución al procesamiento de grandes conjuntos de datos estructurados, semiestructurados o no estructurados.

Los datos estructurados son aquellos que tienen longitud y formato y que pueden ser almacenados en tablas; los datos no estructurados son aquellos que carecen de un formato determinado y que no pueden ser almacenados en una tabla, pueden ser de tipo texto (documentos Word) y de tipo no texto (audio, video); y los semiestructurados, aquellos que no pertenecen a bases de datos relacionales ya que no se limitan a campos determinados (documentos XML, HTML...).

El objetivo de Big Data, es convertir los datos a información, facilitando así la toma de decisiones, incluso en tiempo real. Pero quizá, como piensan la mayoría de los expertos, más que una cuestión de tamaño, es una cuestión de oportunidad de negocio.

Hablamos de Big Data cuando los volúmenes superan la capacidad de software habitual para ser manejados y gestionados. Cuando se habla de grandes volúmenes, se está hablando de Terabytes o Petabytes. Así, esto permite incluir en este tipo de

proyectos informaciones (logs) que hasta hace poco tiempo no se utilizaba porque la tecnología no permitía procesarlos en un tiempo razonable.

La gestión de una información desestructurada precisa de una tecnología diferente y permite tomar decisiones basadas en información que tienen importantes grados de inexactitud. Muchos de estos algoritmos se relacionan con los tratamientos de sistemas avanzados de lógica difusa.

Pero además del gran volumen de información, existe una gran variedad de datos que pueden ser representados de diversas formas en todo el mundo, por ejemplo de sistemas GPS, dispositivos móviles, video, audio, anemómetros, etc, los cuales pueden medir y comunicar el movimiento, el posicionamiento, la temperatura, la humeda, etc. de tal forma que las aplicaciones que analizan estos datos requieren que la velocidad de respuesta sea lo más rápida posible para lograr obtener la información correcta en el momento preciso.

Si bien existe una amplia variedad de tipos de datos a analizar, la Figura 2 muestra los más utilizados:

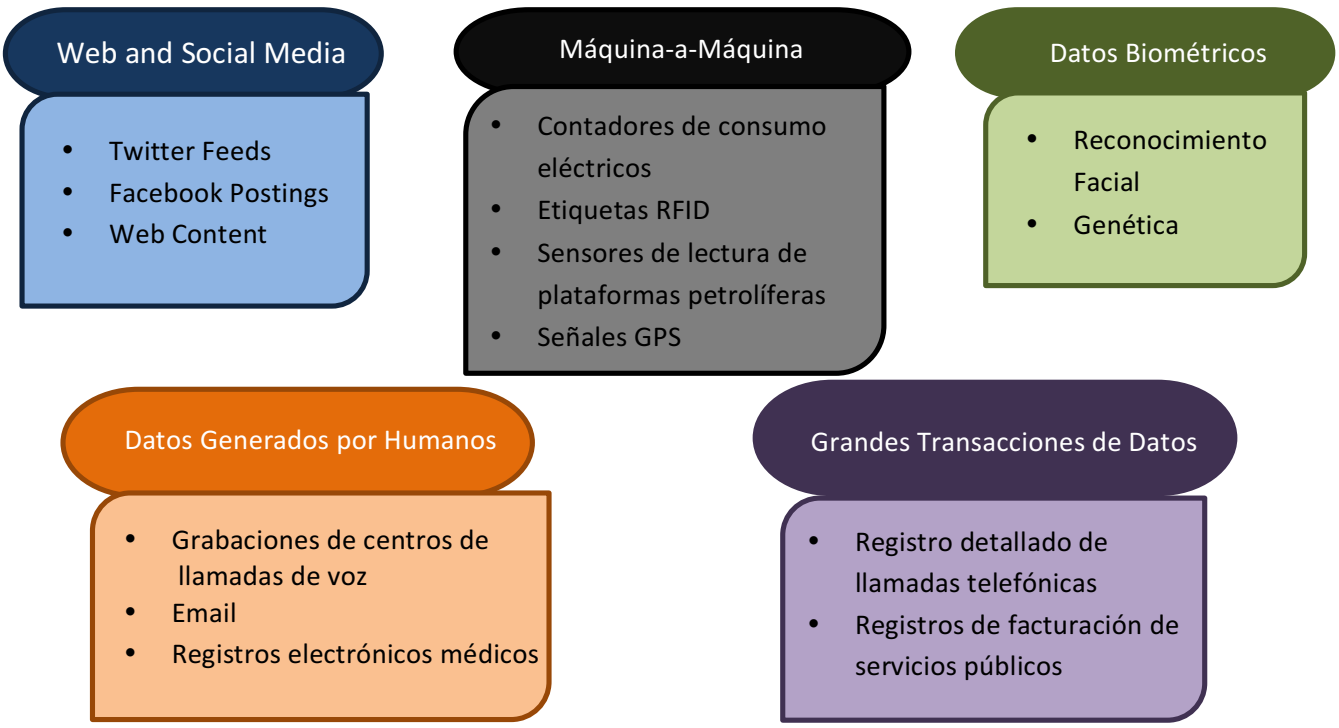


Figura 2. Tipos de Datos de Big Data

1. Web and Social Media, incluye contenido web e información que es obtenida de las redes sociales como Facebook, Twitter, etc.

2. Máquina-a-Máquina (M2M), que se refiere a las tecnologías que permiten conectarse a otros dispositivos.
3. Grandes Transacciones de Datos, que incluye registros de facturación.
4. Datos Biométricos, que es la información biométrica en la que se incluyen huellas digitales, reconocimientos faciales…
5. Datos Generados por Humanos, como notas de voz, correos electrónicos, estudios médicos…

Componentes de una plataforma Big Data

Actualmente quien tiene el liderazgo en términos de popularidad para analizar enormes cantidades de información, es la plataforma de código abierto Hadoop[2].

Hadoop, está inspirado en el proyecto de Google File System (GFS) y en el paradigma de programación MapReduce, el cual consiste en dividir en dos tareas (mapper-reducer) para manipular los datos distribuidos a nodos de un clúster, logrando un alto paralelismo en el procesamiento.

Hadoop está compuesto de tres componentes: Hadoop Distributed File System (HDFS), Hadoop MapRduce y Hadoop Common.

Pero más allá de Hadoop, una plataforma de Big Data, consiste en un ecosistema de proyectos que en conjunto permiten simplificar, administrar, coordinar y analizar grandes volúmenes de información.

2.1.3 SmartSantander

SmartSantander es un proyecto de investigación científica perteneciente al 7ª Programa Marco de la Comisión Europea, en el que se han diseñado, desplegado y validado una plataforma para la experimentación en la IoT compuesta por más de 20.000 dispositivos (sensores, captadores, actuadores, cámaras, terminales móviles, etcétera) por toda la capital cántabra, formando un espacio virtual donde los objetos se comunican entre sí y trasmiten información para las personas con el fin de mejorar su bienestar (calidad de vida).

El campo de pruebas que la ciudad ofrece, crea un esquema de colaboración, entre lo público y lo privado, cuestión que ha sido desde el inicio, uno de los motores del proyecto SmartSantander, no ya sólo en el Proyecto Europeo que le dio nombre, sino en el global de la idea de Santander como ciudad inteligente

[2] Hadoop: “framework de software que soporta aplicaciones distribuidas bajo una licencia libre”.

SmartSantander se encuadra dentro del reto de las Redes Ubicuas y Confiables e Infraestructuras de Servicio, con el objetivo destinado a desarrollar la investigación, en relación a la Internet del Futuro, basada en la experimentación sobre infraestructuras reales.

El núcleo principal de las instalaciones que comprenden más de 20000 dispositivos, se localiza en la ciudad de Santander y sus alrededores, incluyendo puntos singulares de la Comunidad de Cantabria.

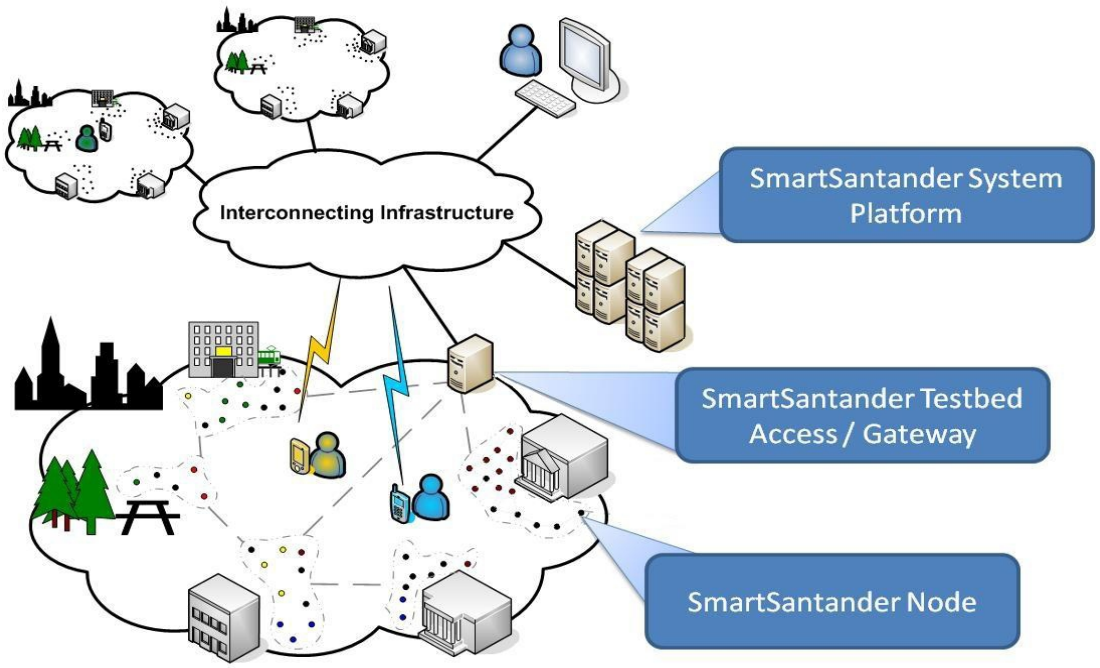


Figura 3. Representación Gráfica de SmartSantander

SmartSantander es una aplicación real y a gran escala de la llamada "computación ubicua", y esta computación es un paradigma de la telemática que tiene por objeto la integración de pequeños dispositivos y sensores en todo los aparatos y objetos de nuestra vida cotidiana, de tal forma que se puedan interconexionarse entre ellos, pudiéndose intercambiar de este modo información útil entre ellos. Lo que se ha venido también a llamar comunicación Máquina a Máquina (M2M).

Como vemos en la Figura 3, los dispositivos que utiliza son de muy pequeño tamaño (nodos), con una capacidad de computación muy limitada, pero a los que es posible acoplar sensores de diversa naturaleza para que puedan captar datos del entorno. Estos datos fluyen a través de la red hasta un punto de concentración donde la información que captan los sensores es tratada con el objeto de obtener un servicio útil.

Para la consecución de los objetivos del proyecto, la plataforma software y la infraestructura física desplegada deben ofrecer una serie de características necesarias para que la futura experimentación pueda realmente suponer un avance cualitativo en la investigación e innovación de los campos de la IoT, las Ciudades Inteligentes y la Internet del Futuro.

Plataforma SmartSantander

La plataforma SmartSantander, se basa en una arquitectura formada por tres niveles:

- Nivel de Dispositivo IoT, que proporciona el sustrato necesario compuesto por los propios dispositivos; estos son recursos limitados y exportan datos fiables que aseguran con una serie de medidas.

- El nivel de Gateway IoT, que enlaza los dispositivos IoT en los bordes de la red a una infraestructura de red central.

- El nivel de Servidor, que dispone de dispositivos de gran capacidad, los cuales son conectados directamente con la infraestructura de red central. Los servidores pueden usarse como repositorios de datos IoT, y como servidores de aplicación que pueden ser configurados para ofrecer una gran variedad de diferentes servicios IoT y aplicaciones.

Análisis de generación de datos de implementación de ciudades

Dada la gran cantidad de datos que se generan y deben ser manejados, almacenados y puestos a disposición , en esta sección se resumirá la forma en que se generan y las cifras aproximadas de datos a tratar.

Patrones de generación de datos

El patrón de generación de datos se define por el servicio para el que está destinado. Podemos identificar dos patrones: el de generación de observación periódica y de basada en eventos.

Los dispositivos IoT programados con el patrón de observación periódica informarán una observación que contiene la información detectada en una base de frecuencia configurable (Dispositivos para servicios de vigilancia ambiental, de intensidad del tráfico y de riesgos de parques y jardines). Sin embrago los dispositivos de IoT que se dedican al estacionamiento al aire libre funcionan en forma de evento, así solo informan sobre la detección de un cambio en el parámetro que están monitorizando (Sensibilización Participativa).

Cantidad de datos generados

Uno de los objetivos de la plataforma SmartSantander era apoyar la experimentación avanzada de la IoT, y para ello y dado que el patrón de generación de observación periódica es configurable, el objetivo era establecer una alta frecuencia.

Teniendo en cuenta sólo las necesidades de servicio, la frecuencia seleccionada conduce a una situación de sobremuestreo. Sin embargo, esto permitió una experimentación más amplia. Para la mayoría de los dispositivos, el período de tiempo utilizado para informar de las nuevas observaciones se fijó en cinco minutos. Para los dispositivos que usan el patrón de generación de observación basado en eventos, el número de observaciones reportadas depende solamente del uso real del servicio.

La Tabla 1 resume el número de observaciones generadas diariamente dentro del lecho de pruebas SmartSantander durante Marzo de 2014.

Servicio	Observaciones diarias
Monitoreo ambiental	139.370
Irrigación de Parques y Jardines	8,365
Monitorización Ambiental Móvil	82.726
Ocupación de aparcamiento	13.489
La gestión del tráfico	54.720
Detección Participativa	6.352
Realidad aumentada	1.489

Tabla 1. Número de Observaciones generadas diariamente en el banco de pruebas de SmartSantander

Este Proyecto, es único en el mundo y ofrecerá un excelente campo de experimentación a la comunidad científica europea, puesto que SmartSantander establecerá las bases para la comunicación entre elementos heterogéneos, permitiendo la federación con otras redes de similar naturaleza en el resto de Europa y el mundo. Este Proyecto, hace posible que con la gran cantidad de sensores de que dispone, se cumpla el objetivo de construir una “Ciudad Inteligente”.

2.2 Logstash, Elasticsearch y Kibana

2.2.1 Logstash

Logstash es un potente motor de recopilación de datos de código abierto con capacidades de pipelining en tiempo real. Permite unificar dinámicamente los datos de diferentes fuentes y normalizar los datos a un único modelo, así como hacer que converjan al destino o destinos que se elijan.

Logstash, aunque inicialmente nació para la recopilación de registros, posee una capacidad que va mucho más allá de ese uso. Cualquier tipo de evento puede ser enriquecido y transformado mediante distintos tipos de plug-ins (entrada, filtrado y salidas).

Principales funcionalidades de Logstash

- Procesamiento de datos escalable horizontalmente con una fuerte sinergia con ElasticSearch y Kibana.
- Arquitectura de tuberías conectable: recolectar, mapear y organizar diferentes entradas, filtros y salidas.
- Comunidad extensible y ecosistema de plug-ins para los desarrolladores: más de 200 plug-ins disponibles, abierta la posibilidad de crear, desarrollar y compartir con la comunidad.

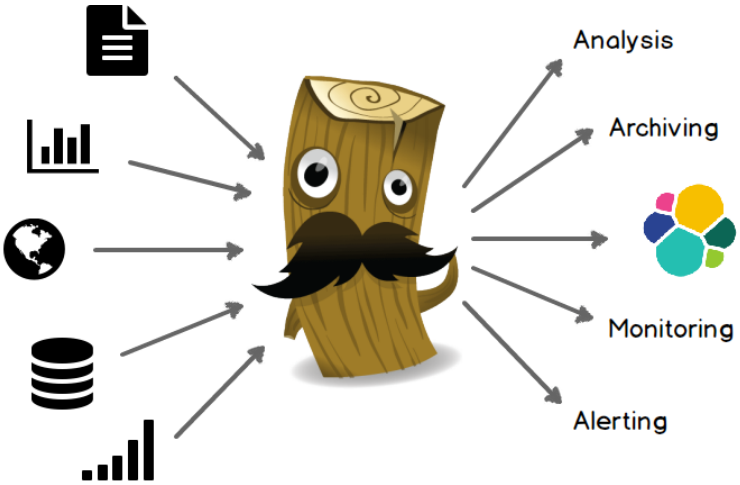


Figura 4. Funcionalidades de Logstash

Fuentes de datos para Logstash

Logstash, fue diseñado para recopilar datos provenientes de ficheros de logs con el fin de pre-procesarlos y enviarlos a un sistema de almacenamiento. Pero la herramienta ha evolucionado mucho y existe multitud de entradas que nos permite obtener datos desde fuentes de datos de lo más variado.

Las fuentes de datos para Logstash son múltiples, y lo más importante es que son ampliables. Los ejemplos más desarrollados por la comunidad son:

- Logs y Métricas:
  - Maneja todo tipo de datos de registro
    - Registros web como Apache, Registros de aplicaciones como log4j para Java…
    - Captura otros formatos de registros como syslog, eventos de Windows, de red y cortafuegos, y muchos más…
  - Recoge las métricas de Ganglia[3], collectd[4], NetFlow[5], JMX[6] y muchas otras plataformas de infraestructuras y aplicaciones sobre TCP y UDP
- La Web
  - Transforma las peticiones http en eventos.
    - Captura de métricas y otros tipos de datos de las interfaces de aplicaciones Web.
- Sensores e IoT

  Explora una anchura expansiva de otros datos. Logstash puede formar la columna vertebral de recopilación de eventos para la ingestión de los datos enviados desde dispositivos móviles a las casas inteligentes, vehículos, y muchas aplicaciones específicas de la industria.

[3] Glanglia: "software que provee monitoreo en tiempo real y ejecución de ambientes. Enlaza líneas de Cluster y computación".

[4] Collectd: "programa informático que recoge del sistema las aplicaciones métricas periódicamente y proporciona mecanismos para almacenar los valores".

[5] Netflow: "protocolo de red desarrollo por Cisco Systems para recolectar información sobre tráfico IP".

[6] JMX: "tecnología que define una estructura de gestión, la API, los patrones de diseño y los servicios para la monitorización de aplicaciones basadas en JAVA".

CONFIGURACION DE UNA LOGSTASH PIPELINE

La operativa de Logstash se organiza alrededor del concepto de Pipeline Logstash. Una Pipeline Logstash, en la mayoría de los casos, tiene una o más entradas, una serie de filtros y uno o varios plug-ins de salida. La cantidad, tipo y operativa de cada uno de estos elementos dentro de una Pipeline Logstash se define a través de un archivo de configuración.

Tal y como se ve en la Figura 5, en un Pipeline Logstash, las fuentes de datos se canalizan a través de un Input plug-in. Es posible tener varias fuentes utilizando el mismo plug-in o usando distintos Input plug-ins

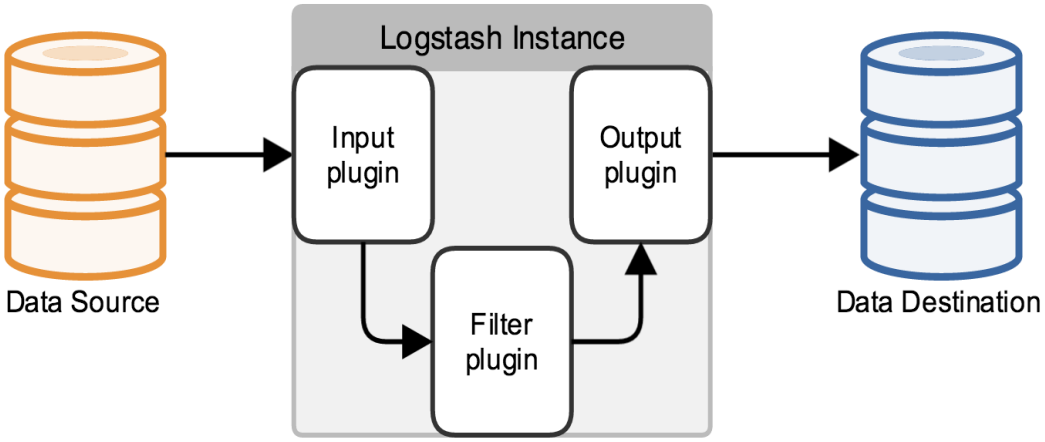


Figura 5. Configuración Avanzada de una Logstash Pipeline

Los filtros operan sobre los eventos que se producen desde las fuentes de datos moldeándolos, adaptándolos o interpretándolos según sea necesario. Por último, los plug-ins de salida permiten almacenar los eventos una vez modificados.

- Algunas entradas más utilizadas son:
  - File: se lee de un archivo en el sistema de archivos, al igual que el comando UNIX "tail -0F"
  - Syslog: escucha en el puerto 514 y lo analiza de acuerdo con el formato RFC 3164.
  - Redis: lee desde un servidor Redis[7], usando ambos canales y listas redis. Redis es utilizado con frecuencia como intermediario en una instalación centralizada Logstash.
  - Beats: procesa eventos enviados por Filebeat.

[7] Redis: "se trata de un servidor de caché de memoria que conoce la estructura de los datos que alberga".

- Algunos filtros útiles son:
    - Grok: analiza y estructura arbitrariamente el texto. Es actualmente la mejor manera en Logstash para analizar los datos de registro no estructurados en algo estructurado y consultable.
    - Mutate: realiza transformaciones generales en los campos de los eventos. Puede renombrar, eliminar, reemplazar y modificar los campos en tus eventos.
    - Drop: omite un evento, por ejemplo, depuración de eventos.
    - Clone: realiza una copia de un evento, añadiendo o quitando campos.
    - Geoip: añade información sobre la ubicación geográfica mediante las direcciones IP.

- Las salidas hacen referencia a la fase final. Un evento puede pasar a través de varias salidas, pero una vez que el proceso de salida se ha completado, el evento ha finalizado su ejecución. Algunas salidas incluyen:
    - Elasticsearch: envía los datos de los eventos a Elasticsearch.
    - File: escribe los datos de un evento en un fichero del disco.
    - Graphite: envía los datos de los eventos a Graphite[8].
    - StatsD: envía los datos de los eventos a StatsD[9].

Existe una cuarta funcionalidad dentro de Logstash que ofrece una forma de manejar los eventos más allá de los filtros. Los codecs, son filtros que pueden funcionar como parte de una entrada o salida. Permiten separar fácilmente el transporte de sus mensajes desde el proceso de publicación. Destacan: JSON, multiline y plain.

2.2.2 Elasticsearch

Elasticsearch es un motor de búsqueda y análisis de texto de código abierto de gran escalabilidad. Se utiliza generalmente como tecnología subyacente para funciones complejas. Nos permite almacenar, buscar y analizar grandes volúmenes de datos de

[8] Graphite:"una popular herramienta de código abierto para el almacenamiento y representación de métricas".

[9] StatsD: "es una simple herramienta desarrollada por Etsy para agregar y resumir las métricas de apliaciones".

forma muy rápida y en tiempo casi real. Va más allá de la búsqueda por texto gracias a un Lenguaje de Dominio Específico (DSL, por sus siglas en Inglés, Domain Specific Language) y una Interfaz de Programación (API, Application Programming Interface) para búsquedas más complicadas.

Por defecto, se puede esperar un retraso de un segundo desde el momento en el que el índice se actualiza o borra sus datos hasta el momento en que aparecen los resultados de búsqueda. Esta es una distinción importante con otras plataformas.

Se basa en los siguientes conceptos:

- ***Cluster***: Es un conjunto de uno o más nodos que mantienen toda la información de manear distribuida e indexada. Cada cluster está identificado por un nombre, que por defecto se llama Elasticsearch.

- ***Nodo:*** forma parte de un cluster, almacenando información y ayudando a las tareas de indexación y búsqueda del cluster.

- ***Index:*** es una colección de documentos que tiene características similares.

- ***Sharding y Réplicas:*** cuando la información que estamos indexando sobrepasa el límite de una sola máquina, Elasticsearch nos ofrece distintas maneras de saltarnos esta limitación. El Sharding nos permite dividir estos índices en distintas piezas, ofreciéndonos la posibilidad de escalar horizontalmente, además de paralelizar y distribuir las distintas operaciones que hagamos sobre estos índices. La replicación nos ofrece un mecanismo para que en caso de fallo el usuario no se vea afectado.

Hay que destacar como características de Elasticsearch:

- Es una plataforma de búsqueda en tiempo casi real (NRT)

- Elasticsearch proporciona una API muy completa y potente REST que se puede utilizar para interactuar con el clúster.

- Además de ser capaz de reemplazar documentos, se pueden actualizar, eso sí cada vez que queramos hacer una actualización, Elasticsearch elimina el documento y luego lo actualiza en una sola toma.

- Es capaz de realizar múltiples operaciones de forma rápida, evitando la red de ida y vuelta mediante una API concisa y simplificada.

- Elasticsearch no requiere que le creen de forma explícita un índice antes de poder indexar documentos en él, Elasticsearch creará automáticamente el índice si no existiera de antemano.

- Elasticsearch, proporciona un lenguaje especifico de dominio de estilo JSON, que se puede utilizar para ejecutar consultas. Esto se conoce como el DSL de consulta.

- Elasticsearch está construido usando Java, y requiere al menos Java 7, ya que las versiones anteriores podrían dañarlo. Únicamente Java de Oracle y el OpenJDK son compatibles.

2.2.3 Kibana

Kibana es una plataforma de análisis y visualización de código abierto diseñada para trabajar sobre índices de Elasticsearch. Permite buscar, ver e interactuar con los datos almacenados en los índices Elasticsearch. Puede realizar fácilmente análisis avanzado de datos y visualizar los datos en una amplia variedad de gráficos, tablas y mapas.

Kibana facilita la interpretación de grandes volúmenes de datos. Contiene una sencilla interfaz gráfica basada en un navegador web, permitiendo crear de forma rápida distintos paneles dinámicos de las consultas de Elasticsearch en tiempo real.

Se puede instalar Kibana y comenzar a explorar los índices Elasticsearch en cuestión de minutos, sin ninguna infraestructura adicional requerida.

La configuración de Kibana es simple, tendremos que indicar donde se encuentra Elasticsearch.

Conceptos básicos del funcionamiento de Kibana:

- *Kibana y Elasticsearch. Asignación dinámica:*

De forma predeterminada, Elasticsearch, permite el mapeo dinámico para los campos. Kibana necesita de dicho mapeo para manejar los distintos campos en la creación de las visualizaciones, así como para poder gestionar búsquedas, visualizaciones y cuadros de mando.

- *Conexión Kibana-Elasticsearch:*

Antes de empezar a utilizar Kibana, hay que configurar en Elasticsearch, qué índices se desean explorar. Así se nos pedirá definir un patrón de índices.

- *Visualización:*

Las herramientas de visualización permiten mostrar los aspectos de los conjuntos de datos de diferentes maneras. Las visualizaciones dependen de Elasticsearch "agregaciones", en dos tipos diferentes: agregaciones de tubo y agregaciones métricas.

- *Acceso a Kibana:*

Kibana ofrece sus visualizaciones a través de una aplicación web. El punto en el que escuda el servidor web que soporta esta aplicación es fácilmente configurable. De esta forma el usuario solo tiene que utilizar su navegador web para acceder a estas visualizaciones.

- *Mecanismos de desbordamientos:*

Se pueden explorar interactivamente los datos de la página Discover. Así se tendrá acceso a todos los documentos que coinciden con el patrón índice seleccionado. Se pueden enviar consultas de búsqueda, filtrar los resultados de búsqueda y ver los documentos. También podemos ver el número de documentos que coinciden con la consulta de búsqueda y obtener estadísticas de valor de campo.

La pantalla de Discover está compuesta por:

  - Search Bar
  - Time filter
  - Field Selector
  - Date Histogram
  - Log View

- Tiempo de Filtrado:

El tiempo de filtrado restringe los resultados de búsqueda a un periodo de tiempo específico. Así, se puede establecer un filtro de tiempo si el índice contiene eventos basados en el tiempo y un campo de hora si está configurado para el patrón de índice seleccionado. Por defecto, el filtro de tiempo se establece en los últimos 15 minutos.

- Búsqueda de índices:

Se pueden buscar los índices que coinciden con el patrón actual, mediante una búsqueda específica que se encuentra en la página Discover. Así se pueden introducir cadenas de consultas simples, utilizando el Lucene, o usar la completa, basada en JSON.

- Visualización:

A través de "visualize", se pueden diseñar las visualizaciones de datos. Se pueden guardar, utilizar de forma individual o combinarlas en un tablero de mandos.

La visualización puede basarse en uno de los siguientes tipos de fuentes de datos:

  - Una nueva búsqueda interactiva.
  - Una búsqueda guardada.
  - Una visualización guardada existente.

- Dashboard:

Un Dashboard (Figura 6) muestra un conjunto de visualizaciones guardados en grupos que se pueden organizar libremente. Puede guardar un cuadro de mandos para compartir o volver a cargar en un momento posterior.

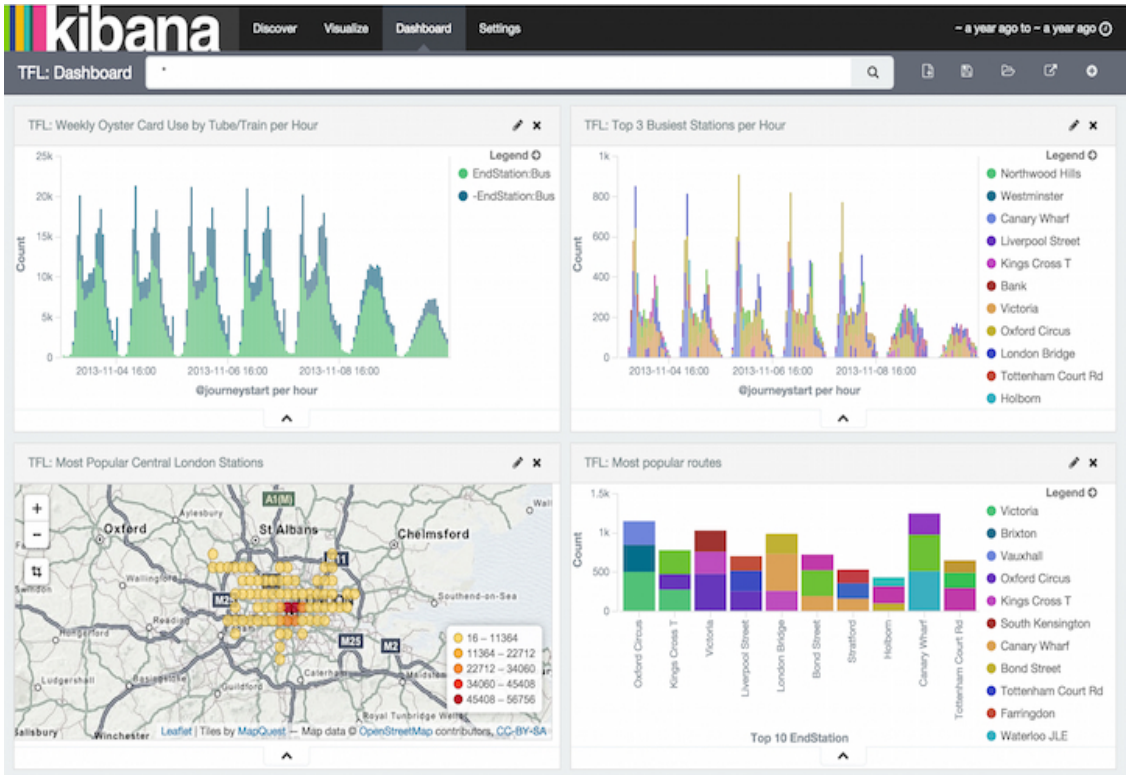


Figura 6. Ejemplo de un Dashboard

2.3 Plataformas Cloud: Infrastructure, Platform and Software as a Service

La computación en la nube (cloud computing) es un término que muchos encuentran confuso. Sin embargo, no es tan confuso como suena. De hecho, la mayor parte de los que no comprenden el concepto son parte de la mayoría que la utilizan a

diario.

En términos sencillos, la computación en la nube es el concepto utilizado para describir los distintos escenarios en los que los recursos de computación se ofrecen como un servicio a través de una conexión de red (por lo general, se trata de Internet).

Por lo tanto, la computación en la nube es un tipo de computación que se basa en el intercambio de un conjunto de recursos físicos y/o virtuales, en lugar de implementar hardware y software local o personal. No deja de ser sinónimo del término "utility computing" ya que los usuarios son capaces de acceder a una oferta de recursos de computación en lugar de gestionar el equipo necesario para generarlos por sí mismos; de la misma manera que un consumidor aprovecha el suministro eléctrico nacional, en vez de ejecutar su propio generador.

Una de las características fundamentales de la computación en la nube es la flexibilidad que ofrece y ellos es debido a su escalabilidad. Se refiere a la capacidad de un sistema para adaptarse y escalar los cambios en la carga de trabajo. La tecnología en la nube permite de forma automática proveer y desproveer de recursos como y cuando sea necesario, asegurando así que el nivel de recursos disponibles sea muy similar a la demanda requerida tanto como sea posible. Esto lo diferencia de otros modelos de computación donde los recursos se entregarán en bloques (por ejemplo, servidores individuales, aplicaciones de software, aplicaciones de descarga de software), la mayoría con capacidades fijas y costos iniciales. Mediante la computación en la nube, el usuario final paga sólo por los recursos que consuma, evitando así los gastos e ineficiencia de cualquier recurso que no utilice.

Sin embargo, no solo una de las ventajas de la nube es la flexibilidad. La empresa también puede beneficiarse (en diversos grados) de las economías a escala creadas por la creación de servicios en masa con los mismos entornos informáticos, y la fiabilidad de albergar servicios físicamente a través de múltiples servidores en que los fallos de los sistemas individuales no afecten a la continuidad del servicio.

Centrándonos en las distintas clases de Cloud, podemos distinguir:

- Clientes que la usan
- Servicios ofrecidos

En función de los clientes que la usan (Figura 7), se distinguen:

- ***Nube Pública***: es una nube en la que los servicios y la infraestructura están alojados en una organización externa y gestionados por ella misma. En este tipo de nube se mezclan tanto tipo de datos como procesos de distintos clientes en los distintos servidores, sistemas de almacenamiento y otras infraestructuras requeridas por la propia nube, accediendo a ellos el usuario final a través de manera remota, como Internet. El beneficio de este tipo de nubes es el abanico de precios y la redundancia, sin embargo, son más vulnerables que las configuraciones de nubes privadas, debido a sus altos niveles de accesibilidad.

- ***Nube privada***: la gran diferencia respeto a la pública es la autogestión de la misma, aumentando los niveles de control y seguridad. Esto repercute en los costes, debido a la creación más personalizada de los servicios demandados por el cliente (software y hardware especifico).

- ***Nube Híbrida:*** como su propio nombre indica, es una combinación de ambas (pública y privada). Esto permite al cliente maximizar su eficiencia, mediante la utilización de la nube pública para las operaciones no sensibles mientras que para las operaciones sensibles o críticas usaría la nube privada, asegurando así al cliente una configuración de computación acorde a sus necesidades y sin elevar los costes.

Figura 7. Clasificación de las Cloud

Por otra parte, tenemos la clasificación en función de los distintos modelos de servicios de computación, clasificados en:

- IaaS (Infraestructura como Servicio).
- PaaS (Plataforma como Servicio).
- SaaS (Software como Servicio).

2.3.1 IaaS

Infraestructura como Servicio, es uno de los tres modelos fundamentales en el campo de las cloud computing, proporcionando acceso a recursos informáticos situados en un entorno virtualizado, "la nube", a través de una conexión pública que suele ser Internet, abarcando aspectos como el espacio en servidores virtuales, conexiones de red, ancho de banda, direcciones IP y balanceadores de carga. Podríamos destacar como ventaja su escalabilidad.

2.3.2 PaaS

Plataforma como Servicio, es una categoría de servicios cloud que proporciona una plataforma y un entorno que permiten la creación de aplicaciones y servicios que funcionan a través de Internet.

Los servicios se alojan en la nube y se puede acceder a ellos a través de un navegador permitiendo crear aplicaciones de software utilizando las herramientas suministradas por el proveedor. Podemos destacar como ventajas su flexibilidad y su adaptabilidad.

2.3.3 SaaS

Software como Servicio, es cualquier servicio cloud, en el que los consumidores pueden acceder a aplicaciones de software a través de Internet.

Estas aplicaciones están alojadas en la nube y pueden utilizarse para una amplia variedad de tareas, tanto para particulares como para organizaciones. Twitter, Facebook y Google, son algún ejemplo de Saas, en las que los usuarios pueden acceder a los servicios a través de cualquier dispositivo que pueda conectarse a Internet. Este modelo, es también conocido como "software a demanda", y las aplicaciones se compran y utilizan a través de Internet y los archivos se guardan en la nube, no en el ordenador del usuario.

A continuación, como se muestra en el gráfico (Figura 8), veremos la separación de las responsabilidades de un abonado de las de un proveedor de servicios.

Separación de Responsabilidades

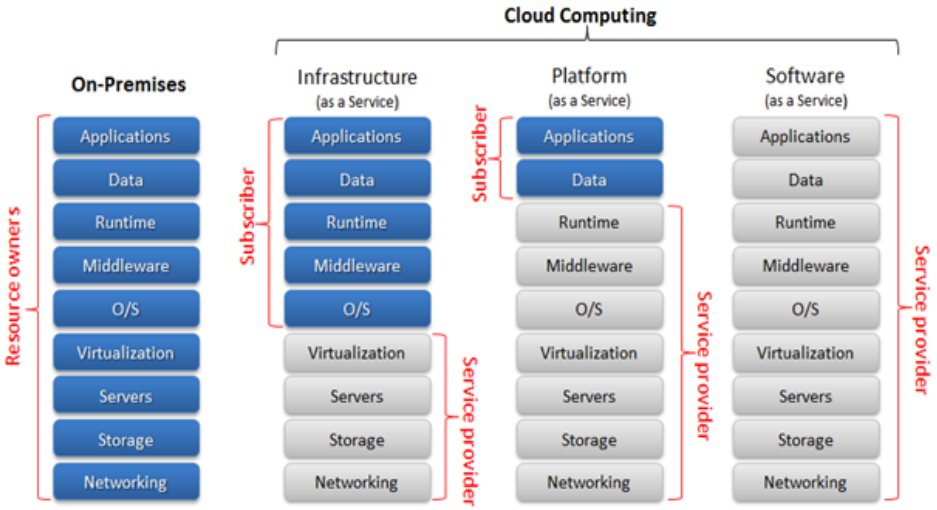


Figura 8. Separación de Responsabilidades de la Computación en la Nube

Capítulo 3. **Sistema de Control de Infraestructuras IoT**

3.1 Arquitectura del Sistema

En la actualidad, la Internet de las Cosas, vierte constantemente información de todo tipo a la red. Genera cantidades inmensas de datos. Pero el Internet de las cosas, no alude únicamente a las distintas formas en las que pueden interconectarse los dispositivos en red, sino a "una transformación de la visión y aplicación del procesamiento, análisis, almacenamiento y comunicación". En efecto, esta amplia información demanda mayor almacenamiento y procesamiento de datos.

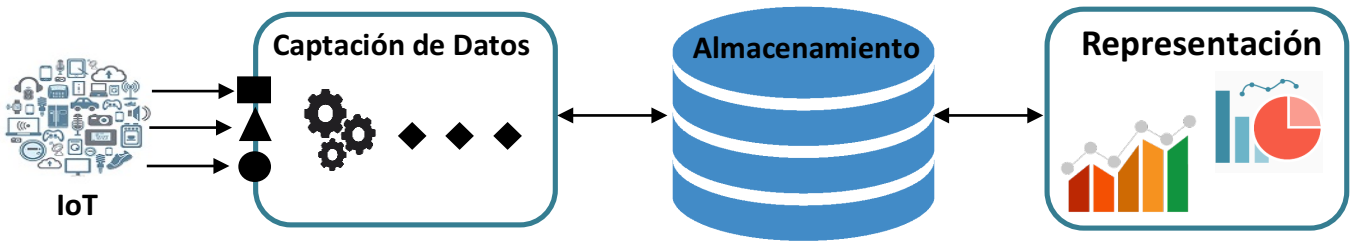


Figura 9. Arquitectura del Sistema

En este capítulo describiremos el diseño y la arquitectura del sistema de control que se ha desarrollado en este TFG. Así, como vemos en la Figura 9, éste consta de 3 partes diferenciadas:

El primer módulo representa la captación de datos, con dicha captura, simplificaremos todos los datos e información que recibimos de la infraestructura subyacente. Se reciben múltiples entradas de distintos formatos, y mediante unos filtros, se llegará a darle un formato único.

Posteriormente, estos datos son almacenados en una base de datos, creando unas tablas con sus respectivos índices. Desplegaremos la tecnología adecuada para agrupar y analizar estos datos y crear una información de valor, con el diseño de una arquitectura adecuada para el almacenamiento de dicha información, que facilite su posterior acceso para su análisis.

Finalmente, a través de su representación gráfica, ilustraremos y presentaremos este conjunto de datos, de manera que se facilite su comprensión, comparación y análisis. Con la representación gráfica de estos datos ofrecemos mensajes más claros, facilitando de este modo la extracción de conclusiones.

3.2 Acceso a las Observaciones de SmartSantander

Tal y como se ha descrito en la anterior sección, la fuente de datos del sistema desarrollado en este TFG son las observaciones que se generan en la plataforma de SmartSantander.

Desde la perspectiva del usuario, las suscripciones son el elemento básico del servicio asíncrono del sistema SmartSantander. Para recibir notificaciones push en tiempo real de SmartSantander se tiene que configurar el sistema mediante el uso de las subscripciones asíncronas. Esto instruye a la plataforma de que envíe al subscriptor las observaciones que cumpla con los criterios definidos en la subscripción.

Para ello es necesario autenticarse. SmartSantander IO API es compatible actualmente con dos mecanismos de autenticación diferentes:

- Esquema de autenticación básica basada en la provisión de una clave de API válida.
- Esquema de autenticación mediante certificados PCKS#12.

En este Trabajo hemos empleado el primer método para subscribirnos a las observaciones.

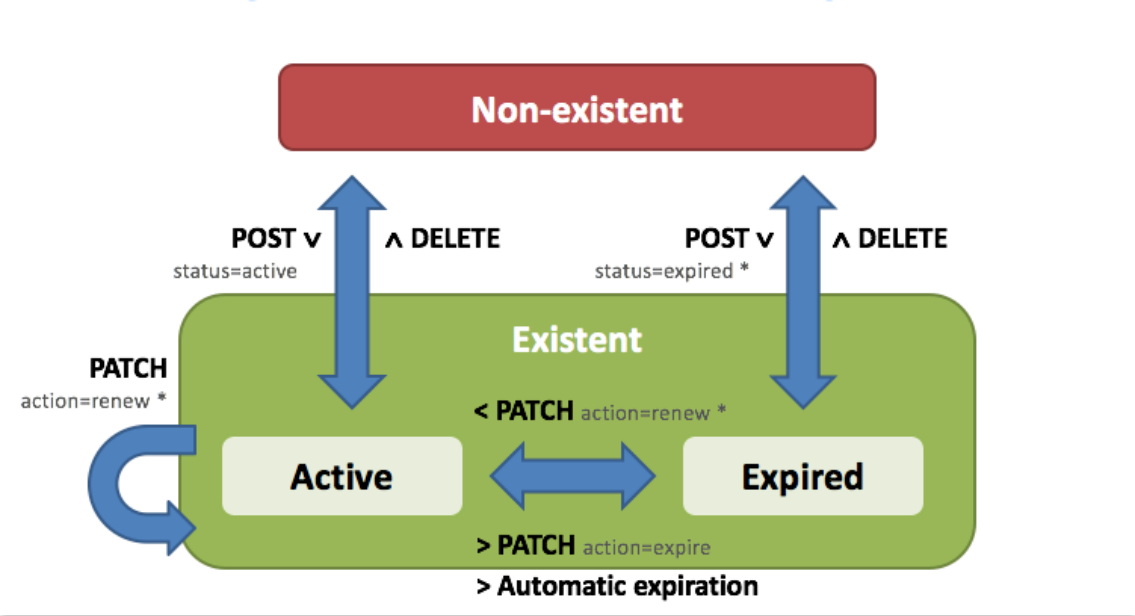


Figura 10. Gráfico de la Sistemática de una Subscripción a SmartSantander

A continuación, explicaremos la sistemática que hay que seguir para la creación de una subscripción, así como la activación, modificación o eliminación de la misma.

La Figura 10, muestra los diferentes estados por los que puede pasar una subscripción. Desde el estado Non-existent, en el cuál básicamente no hay subscripción, se puede generar una mediante una petición POST al API. En esta petición se incluirá como parte del body la descripción de la subscripción. Por defecto las subscripciones se crean inactivas y es necesario activarlas mediante una petición PATCH. En esta petición

PATCH es necesario incluir el identificador de la subscripción que se ha obtenido en su creación. La petición PATCH permite activar y desactivar la subscripción tantas veces como se desee. Por último, se puede eliminar la subscripción definitivamente mediante una petición DELETE.

Creación(POST):

Figura 11. Creación de una Subscripción

En la Figura 11, vemos las distintas opciones en el envío de la petición POST para la creación de la subscripción.

En primer lugar, nos encontramos con el método de autenticación, explicado anteriormente.

A continuación, vemos los parámetros de la consulta:

- Profile: podemos activar un perfil de usuario especial cuando se llame al método de la API, por defecto, el cuadro está vacío.
- Status: permite seleccionar si se desea o no activar la subscripción después de su creación. Si no le indicamos nada, dejándolo vacío, la subscripción se creará, pero no estará activada, teniendo que activarla posteriormente con el envío de una petición HTTP PATCH.

Y por último, el cuerpo de la petición. En él podemos indicar los parámetros que queremos que tenga nuestra subscripción, siempre siguiendo el formato descrito en la documentación de SmartSantander.

Mediante la Figura 12, explicaremos los aspectos básicos de la generación de una subscripción y las posibilidades que se tienen para fijar los criterios que deben cumplir las observaciones que queremos que nos sean notificados:

```
{
  "target": {
    "technology": "http",
    "parameters": {
      "url": "[a valid endpoint]"
    }
  },
  "query": {
    "what": {
      "format": "measurement",
      "_anyOf": [
        {
          "phenomenon": "temperature:ambient",
          "filter": {
            "uom": "degreeCelsius",
            "value": {
              "_gt": 5
            }
          }
        }
      ]
    },
    "where": {
      "_anyOf": [
        {
          "area": {
            "type": "Circle",
            "coordinates": [
              -3.810011,
              43.462403
            ],
            "radius": 0.5,
            "properties": {
              "radius_units": "km"
            }
          }
        }
      ]
    }
  }
}
```

Figura 12. Elementos de una Subscripción

El elemento target contiene la especificación de la forma en que se desea que las observaciones sean notificadas. En nuestro caso hemos utilizado el target denominado HTTP que instruye el sistema para en enviar las observaciones embebidas en peticiones HTTP POST al endpoint específico.

Por su parte dentro de la query tenemos:

- "what": en este campo le indicaremos a qué queremos subscribirnos. En el campo "format" tenemos dos posibilidades: measurement u observation. La diferencia radica en que si elegimos la primera, measurement , nos suscribimos a las medidas de los distintos fenómenos que le indiquemos a continuación, en el campo "phenomenon", sin embargo si elegimos la segunda, observation, nos subscribimos a la observación generada por el sensor, que puede contener una o más medidas.
- "where": en este campo le indicaremos el área que queremos delimitar para nuestra subscripción, teniendo multitud de posibilidades: Point, Multipoint, Linestring, Multilinestring, Area…

También existe la posibilidad de especificar los identificadores de los dispositivos de los cuales queremos obtener las medidas, pero esta opción sólo se empleó durante la fase de pruebas.

Las subscripciones tienen un tiempo de vigencia por defecto de una semana, teniendo que reactivarlas mediante un PATCH pasado este tiempo.

Una vez enviada la petición POST, recibiremos en el campo "status" un valor indicándonos lo siguiente:

- 201: la subscripción se ha creado con éxito.
- 400: el cuerpo del mensaje no contiene una definición de subscripción válida
- 401: la solicitud no puede procesarse porque no está autenticado en el sistema.
- 403: la solicitud no puede ser procesada debido a que no se le permite realizar esta operación, debido a restricciones de autorización.

También recibiremos un identificador de nuestra suscripción, que usaremos como ya hemos comentado para su activación, modificación o eliminación.

Modificación (PUT):

Para la modificación o actualización de la subscripción se utiliza el método PUT de HTTP. Este no modifica ni el estado de la subscripción ni la fecha de caducidad. El cuerpo de la solicitud tiene que incluir la definición de subscripción JSON completa, en

un formato similar al utilizado cuando se ha creado la subscripción original a través de la solicitud POST. También tendremos que indicarle el "id" de la subscripción a modificar.

Activación (PATCH):

La petición PATCH, permite al usuario gestionar el ciclo de vida de la subscripción ya existente. Esto no afecta de modo alguno al contenido de la subscripción. El cuerpo de la petición no se utiliza y ha de dejarse vacío. En cambio, el parámetro "action" se ha de utilizar para especificar la operación que se desea realizar, teniendo dos opciones, "renew": para renovar la subscripción una semana más y "expire": para que expire la subscripción.

Otra de las funcionalidades de esta petición es activar la subscripción, debiendo indicarle el "id" de la subscripción.

Estado (GET):

Mediante la solicitud GET, obtendremos la información relacionada con una suscripción o varias, en función si le indicamos el "id" de la misma o lo omitimos (mostrando todas las subscripciones realizadas). Esto incluye todos los datos de la suscripción, incluyendo su estado y tiempo de caducidad.

Eliminación (DELETE):

Mediante esta petición, el usuario podrá eliminar una subscripción ya existente, indicando el "id" de la misma. Si la subscripción estuviera activada, se desactivaría antes de la eliminación.

3.3 Módulo de Captación de Datos

En esta sección vamos a detallar la herramienta utilizada para la captación de datos en nuestro sistema.

La herramienta utilizada es Logstash, forma parte de la plataforma Elasticsearch. La elección de la misma, ha sido por la integración que tiene con las demás herramientas implementadas en este Trabajo. Es un potente motor de recopilación de datos, permitiéndonos unificar y estructurar los datos provenientes de SmartSantander, así como haciéndolos converger al módulo de almacenamiento (Elasticsearch).

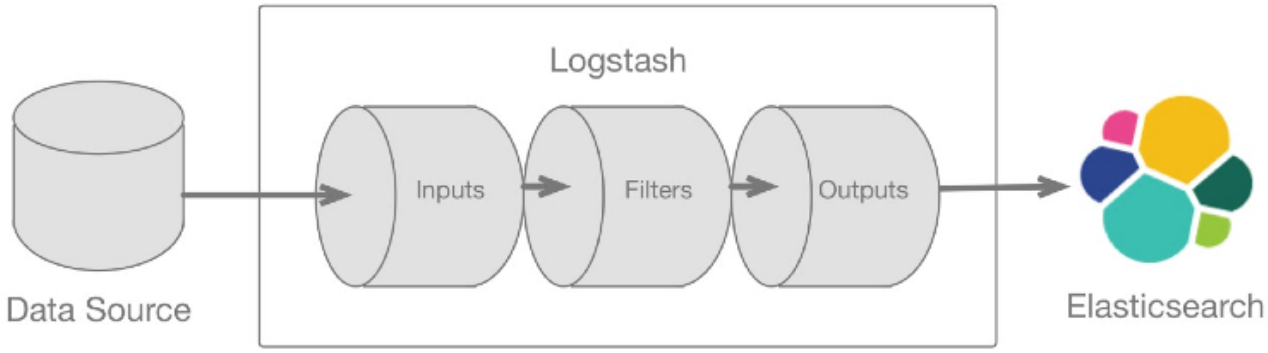


Figura 13. Arquitectura de Logstash

En la Figura 13 mostramos las partes de las que se compone el archivo de configuración de nuestra herramienta de captación de datos, que explicaremos a continuación.

En el apartado inputs, hay que indicarle el tipo de entrada de datos. En un primer instante, para familiarizarnos con la herramienta y ver de manera más legible la sistemática utilizamos "stdin", que nos permite introducir los datos por teclado. Una vez que vimos la estructura que seguían los mismos, adecuamos el formato de entrada a SmartSantander, permitiéndonos indicar el formato de peticiones HTTP. El uso de este tipo de entrada permite recibir eventos a través de peticiones HTTP(s). SmartSantander enviaría una solicitud HTTP POST con el cuerpo del mensaje y Logstash se encargará de transformarlo a un formato común para su posterior procesamiento.

```
    Input {
        http {
        additional_codecs          =>
{"application/json" => "plain"}
        }
    }
```

Figura 14. Código del Input de Logstash

Como vemos en la Figura 14, hemos añadido a HTTP el plugin "additional_codecs" para indicarle el formato de entrada, para una correcta decodificación posteriormente. En este caso, dado que las observaciones provenientes de SmartSantander estaban serializadas en formato JSON, el tipo MIME indicado es application/json.

Una vez configurada la entrada de nuestro sistema de captación de datos, procedemos a configurar la parte más importante de este módulo, el filtro.

En un principio se probaron las distintas opciones de plug-in de filtrado que nos permite Logstash con las entradas de objetos "JSON[10]". De este análisis se concluyó que la decodificación y conversión realizada no se adecuaba a la estructura de almacenamiento necesaria para su posterior análisis. Investigando las distintas posibilidades de plug-ins, se decidió utilizar la posibilidad que permite Logstash de utilizar scripts de Ruby para conseguir una transformación de los datos completamente programables.

A continuación, vemos en la Figura 15 y Figura 16, el código del filtro implementado.

```
    filter{
        grok { match => [ "message",
"%{HTTPDATE:[@metadata][timestamp]}" ] }
        date { match => [ "[@metadata][timestamp]",
"dd/MMM/yyyy:HH:mm:ss Z" ] }
    ruby {
          init => "
            require 'json'
            def parse_json obj, pname=nill, event
              obj = JSON.parse(obj) unless obj.is_a? Hash
              obj = obj.to_hash unless obj.is_a? Hash

              obj.each {|k,v|
                p = pname.nil?? k : pname
                if v.is_a? Array
                  v.each_with_index {|oo,ii|
                    if oo.is_a? Numeric

parse_json_array_num(oo,ii,p,event)
                    else
                        parse_json_array(oo,ii,p,event)
                    end
                  }
                elsif v.is_a? Hash
                  parse_json(v,p,event)
```

[10] JSON: " Java Script Object Notation, es un formato de texto ligero para el intercambio de datos".

```
                else
                  p = pname.nil?? k : [pname,k].join('.')
                  event[p] = v
                end
              }

            end


            def parse_json_array obj, i,pname, event
              obj = JSON.parse(obj) unless obj.is_a? Hash
              pname_ = pname
              if obj.is_a? Hash
                obj.each {|k,v|

                  p=[pname_,i,k].join('.')
                  if v.is_a? Array
                    v.each_with_index {|oo,ii|
                      parse_json_array(oo,ii,p,event)
                    }
                  elsif v.is_a? Hash
                    parse_json(v,p, event)
                  else
                    event[p] = v
                  end
                }

              else
                n = [pname_, i].join('.')
                event[n] = obj
              end
            end
    def parse_json_array_num obj, i,pname, event
              pname_ = pname
              if obj.is_a? Hash
                obj.each {|k,v|

                  p=[pname_,i,k].join('.')
                  if v.is_a? Array
                    v.each_with_index {|oo,ii|
                      parse_json_array(oo,ii,p,event)
                    }
                  elsif v.is_a? Hash
                    parse_json(v,p, event)
                  else
                    event[p] = v
                  end
                }
```

Figura 15. Código del Filtro de Logstash

```
    else
                n = [pname_, i].join('.')
                event[n] = obj
              end
            end
          "
          code=>
"parse_json(event['message'].to_s,nil,event)if
event['message'].to_s.include? ':'"
        }
    ##cambiar el . por _
    ruby {
        init => "
            def remove_dots hash
                new = Hash.new
                hash.each { |k,v|
                    if v.is_a? Hash
                        v = remove_dots(v)
                    end
                    new[ k.gsub('.','_') ] = v
                    if v.is_a? Array
                        v.each { |elem|
                            if elem.is_a? Hash
                                elem = remove_dots(elem)
                            end
                            new[ k.gsub('.','_') ] = elem
                        } unless v.nil?
                    end
                } unless hash.nil?

                return new

            end

        "
        code => "

event.instance_variable_set(:@data,remove_dots(event.to_has
h))
        "
      }
      mutate {
                add_field       =>       [       "[geo]",
"%{[location][coordinates][0]}" ]
                        add_field     =>     [      "[geo]",
"%{[location][coordinates][1]}"]
                        convert => [ "[geo]", "float" ]
        }
      }
```

Figura 16. Código del Filtro de Logstash

En primer lugar, hemos utilizado el plug-in “grok” para hacer corresponder el timestamp de nuestro evento con el timestamp que se crea al añadirlo en las tablas. Posteriormente, se ejecuta el filtro “ruby” estructurando cada campo del evento, en distintos subcampos. Debido a la imposibilidad de indexar los distintos subcampos con la herramienta de representación utilizando el separador “.”, un segundo filtro “ruby” nos permite cambiar los “.” Por “_” que si son válidos a la hora de hacer la representación. Finalmente, utilizamos el plug-in “mutate”, ya que la estructura del formato generada por los eventos de localización de SmartSantander no es compatible con el formato de nuestra herramienta de representación.

Por último, la última parte del código de configuración de Logstash es el “output”, indicando los distintos plug-ins para darle formato a la salida.

```
output{
        elasticsearch {
                index => "m"
        }
stdout {
        code => rubydebug
        }
}
```

Figura 17. Código de la Salida de Logstash

Vemos en la Figura 17 los dos plug-ins que hemos utilizado. Por una parte, el de “elasticsearch”, para almacenar los registros en la herramienta de almacenado (Elasticsearch), debiendo de indicarle el nombre del índice “index”, que deberá de coincidir con las dos herramientas posteriores (Elasticsearch y Kibana) para un correcto funcionamiento. El otro pulg-in utilizado es “stdout”, con el “code” => rubydebug, para poder ir viendo la salida estructurada por pantalla.

En el Anexo se puede encontrar el archivo .conf de Logstash al completo.

3.4 Módulo de Almacenamiento

Una vez que hemos visto el módulo de captación de datos, pasaremos a describir en esta sección la herramienta para el almacenamiento de los mismos, llamada Elasticsearch.

Elasticsearch es un potente motor de búsqueda y análisis de datos en tiempo real de código abierto construido encima de Apache Lucene[11]. En este trabajo, no hemos modificado el archivo de configuración, se ha dejado por defecto, ya que la modificación y creación de la estructura la hemos realizado en el filtrado de la herramienta de captación de datos. Lo que sí que hemos tenido que modificar, es el mapeo de los datos de localización y realizar unos ajustes, acordes con nuestros valores de Longitud y Latitud.

Los valores de localización de SmartSantander siguen la estructura indicada en la Figura 18:

```
"location":{
        "type": "Point",
        "coordinates":[Lon, Lat]
}
```

Figura 18. Formato Coordenadas de Localización de SmartSantander

Para la correcta interpretación, hemos tenido que realizar el siguiente mapeo:

```
curl -XPUT 'localhost:9200/x/_mapping/logs' -d
'{"properties":{"geo":{"type":"geo_point","geohash":true}}}'
```

Figura 19. Mapping de ElasticSearch

El significado del mapeo, (Figura 19), es el de definir en Elasticsearch, la estructura del campo de localización que hemos modificado en el filtro de Logstash, puesto que de origen no se ajusta con el que nos permite la herramienta de representación.

Quedándonos finalmente la estructura en ElasticSearch como vemos en la Figura 20:

```
{
"m":{
  "mappings":{
    "logs":{
      "properties":{
          "@timestamp":{
              "type":"date",
           "format":"strict_date_optional_time||epoch_millis"
                        },
          "@version":{
```

[11] Apache Lucene: “es una API de código abierto para recuperación de información, originalmente implementada en JAVA por Doug Cutting”.

```
                    "type":"string"
                          },
                "geo":{
                    "type":"geo_point",
                    "geohash":true
                     },
                "headers":{
                    "properties":{
                        "content_length":{
                                "type":"string"
                                      },
                        "content_type":{
                                "type":"string"
                                      },
                        "http_accept":{
                                "type":"string"
                                      },
                        "http_accept_charset":{
                                "type":"string"
                                      },
                        "http_accept_encoding":{
                                "type":"string"
                                      },
                        "http_accept_language":{
                                "type":"string"
                                      },
                        "http_authorization":{
                                "type":"string"
                                      },
                        "http_cache_control":{
                                "type":"string"
                                      },
                        "http_connection":{
                                "type":"string"
                                      },
                        "http_host":{
                                "type":"string"
                                      },
                        "http_if_modified_since":{
                                "type":"string"
                                      },
                        "http_pragma":{
                                "type":"string"
                                      },
                        "http_proxy_connection":{
                                "type":"string"
                                      },
                        "http_user_agent":{
                                "type":"string"
                                      },
                        "http_version":{
                                "type":"string"
                                      },
                        "request_method":{
                                "type":"string"
                                      },
                        "request_path":{
                                "type":"string"
                                      },
```

```
                    "request_uri":{
                            "type":"string"
                                }
                            }
                        },
            "host":{
                "type":"string"
                    },
            "location":{
                "type":"geo_point",
                "geohash":true
                        },
            "location1":{
                "type":"geo_point",
                "geohash":true
                        },
            "location_0":{
                "type":"double"
                        },
            "location_1":{
                "type":"double"
                        },
            "location_type":{
                "type":"string"
                        },
            "message":{
                "type":"string"
                    },
            "phenomenon":{
                "type":"string"
                        },
            "timestamp":{
                "type":"date",
             "format":"strict_date_optional_time||epoch_millis"
                        },
            "uom":{
                "type":"string"
                  },
            "urn":{
                "type":"string"
                    },
            "value":{
  "type":"double"
                    }
                }
              }
          }
      }
  }
```

Figura 20. Estructura de ElasticSearch

3.5 Módulo de Representación

Para finalizar la explicación en detalle de la arquitectura del sistema implementado, describiremos la potente herramienta de representación Kibana.

Kibana es una plataforma de análisis y visualización de código abierto diseñado para trabajar con Elasticsearch. Se utiliza para buscar, visualizar e interactuar con los datos almacenados los índices de Elasticsearch. Puede realizar fácilmente avanzados análisis de datos y visualizaciones de los mismos en una amplia variedad de gráficos, tablas y mapas. Hace que sea más fácil de representar y entender grandes volúmenes de datos. Su sencilla interfaz Web, le permite crear y compartir de forma rápida e intuitiva paneles dinámicos que muestran los cambios en las distintas consultas de Elasticsearch a tiempo real.

Figura 21. Interfaz Web de Kibana

Como vemos en la Figura 21 , la interfaz Web de Kibana se compone de distintos campos, destacando los siguientes:

- En la parte superior del tablero de instrumentos vemos 4 opciones:
  - Discover: encontraremos información acerca de las búsquedas y el filtrado de los datos.

  - Visualize: encontraremos información sobre los distintos tipos de visualización que ofrece Kibana.
  - Dashboard: encontraremos las distintas opciones para crear y visualizar los mismos.
  - Settings: encontraremos los distintos campos de configuración de Kibana, así como la creación de índices, modificaciones de búsquedas creadas, etc.
- Tiempo de Filtrado: en él podremos restringir los resultados de búsqueda especificando un periodo de tiempo concreto, pudiendo establecer un filtro de tiempo si el índice contiene los eventos basados en el tiempo. También podremos establecer un filtro de tiempo a partir del histograma, seleccionando la zona con los eventos deseados. Por defecto, el filtro de tiempo se establece en 15 min.
- Barra de Herramientas: en ella podremos generar distintas búsquedas de los eventos, modificándolas y guardándolas en función de los criterios añadidos.
- Patrón de Índices: le indicaremos el índice creado, coincidiendo con el de ElasticSearch.
- Listado de Campos: en esta columna, vemos los distintos campos que se crean al indexar los eventos, pudiendo seleccionar uno o varios para ver la información de los mismos en la tabla de contenidos.
- Tabla de contenidos: en esta sección de la interfaz, vemos con más detalle la información de cada evento.

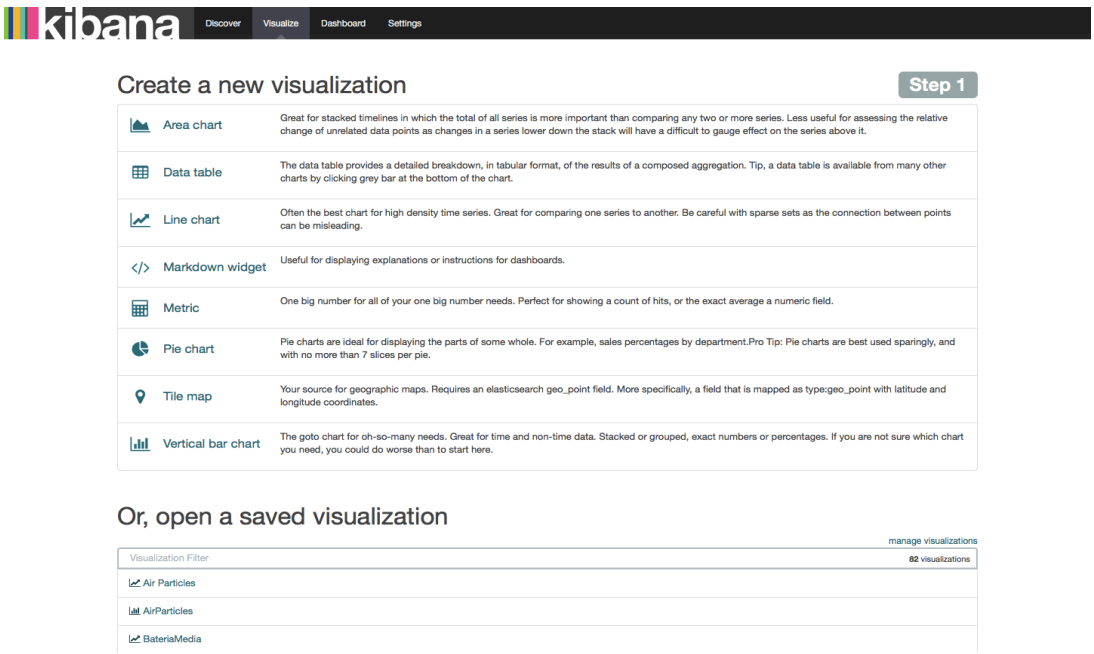


Figura 22. Interfaz Web de Kibana (Visualize)

En la Figura 22, vemos la interfaz Web que corresponde a la parte de visualización de Kibana. En ella podemos crear distintas visualizaciones, desde tabla de datos, gráfico de tablas, circulares hasta un mapa de localización. También nos permite crear visualizaciones a partir de una búsqueda realizada anteriormente, acotando los distintos campos del evento.

3.6 Despliegue del Sistema

Una vez presentados los diseños e implementación de los distintos módulos del sistema, procederemos en esta sección a detallar cómo se ha llevado a cabo el despliegue del sistema sobre una plataforma Cloud real.

Para comenzar, hemos utilizado Amazon Web Services para alojar todas las herramientas de la plataforma Elastic (Logstash, ElasticSearch y Kibana). El hecho de utilizar un servicio de cloud computing es por la escalabilidad que nos ofrece, permitiendo una buena integración con la plataforma SmartSantander. Hemos utilizado un sistema Linux de 64 bits, en el que hemos instalado las herramientas de la siguiente manera:

1. Instalación de Logstash: en primer lugar, debemos comprobar la versión de Java de nuestro sistema, puesto que sólo funciona con Java 7 o superior. Instalaremos mediante los repositorios disponibles por APT con los siguientes comandos:
   - Descargamos e instalamos la clave de firma pública:

   wget -qO - https://packages.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add –

   - Añadimos el repositorio Logstash:

   echo "deb https://packages.elastic.co/logstash/2.4/debian stable main" | sudo tee -a /etc/apt/sources.list

   - Por último, ejecutamos la siguiente línea de comandos y el repositorio estará listo para usarse:

   sudo apt-get update && sudo apt-get install logstash

2. Instalación de ElasticSearch: al igual que ocurre con Logstash, hay que comprobar la versión de Java. Los pasos para la instalación son:

- Descargamos el fichero de Elastic:

curl -L –O https://download.elastic.co/elasticsearch/release /org/elasticsearch /distribution/tar /elasticsearch/2.4.1/elasticsearch-2.4.1.tar.gz

- Una vez extraido, estará listo para su utilización:

tar -xvf elasticsearch-2.4.1.tar.gz

3. Instalación de Kibana: la sistemática es la misma que hemos empleado en Logstash:
   - Descargamos e instalamos la Firma de Clave Pública:

wget -qO - https://packages.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -

   - Añadimos el repositorio:

echo "deb https://packages.elastic.co/kibana/4.6/debian stable main" | sudo tee -a /etc/apt/sources.list.d/kibana.list

   - Por último, corremos apt-get y estará listo para un correcto funcionamiento:

sudo apt-get update && sudo apt-get install kibana

Una vez explicados todos los pasos para la instalación, explicaremos a continuación las acciones que hay que realizar para poner en funcionamiento todo el Sistema.

En primer lugar, realizaremos una subscripción a SmartSantander como vimos en la Figura 11 y Figura 12, indicando en el target de nuestro body, la url que corresponde al servidor Web que exporta Logstash a través del plug-in HTTP, así como los eventos a los que nos queremos subscribir.

Después, pondremos a correr ElasticSearch y Kibana:

- *cd elasticsearch-<version>*

  ./bin/elasticsearch

- *cd kibana-<version>*

  ./bin/kibana

Por último, arrancamos nuestro archivo de configuración de Logstash:

- *cd logstash- <versión>*
  bin/logstash –f <nombre del archivo>.conf

Una vez realizados todos estos pasos, debemos activar la subscripción mediante la petición PATCH (explicado en el capítulo 3.2) y comenzaremos a ver por pantalla la salida de nuestro filtro empleado en Logstash, así como en Kibana veremos los distintos eventos creados.

3.7 Dashboards de Control

En esta sección vamos a describir los distintos Dashboards creados para la visualización de nuestro sistema de control.

Para la creación de un dashboard, primero hay que crear las visualizaciones individualmente y posteriormente, dirigirse a la pestaña “dashboard” y añadirlas una por una, diseñando el cuadro gráfico acorde a nuestras necesidades.

En este trabajo, de los distintos fenómenos que miden los sensores de Santander, hemos hecho un estudio para determinar cuál de estos fenómenos representar, orientándolo a la información más útil para nuestra ciudad. Llegando a la conclusión de representar información relativa a los siguientes fenómenos:

- Temperatura ambiente
- Nivel de batería de los sensores
- Luminosidad
- Dióxido de Carbono (CO2)
- Ozono (O3)
- Dióxido de Nitrógeno (NO2)
- Nivel de partículas en el aire
- Nivel acústico
- Campo electromagnético
- Nivel de llenado de los contenedores de basura

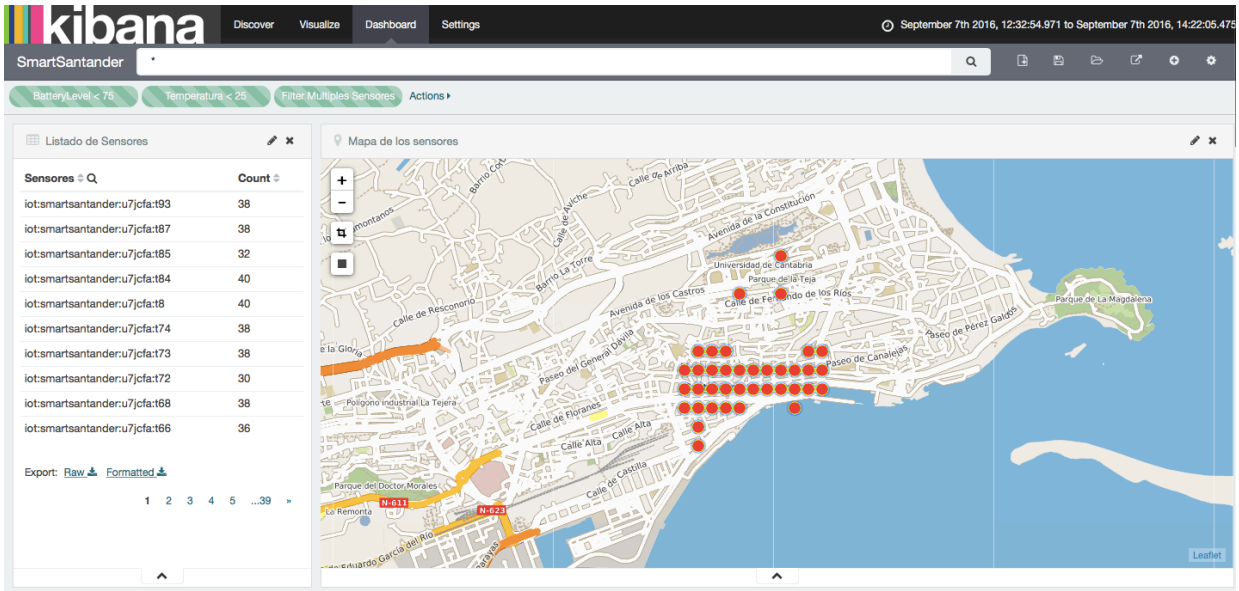


Figura 23. Captura del Dashboard General

En la Figura 23 vemos una captura del dashboard general de nuestro sistema. En él podemos diferenciar distintas partes, como un mapa de la ciudad de Santander identificando la posición de los distintos sensores, así como a su izquierda, un listado de todos los sensores disponibles. Estas dos visualizaciones las veremos en todos los dashboards, tanto en el general como en los de cada fenómeno. Otros campos a destacar, son los pequeños botones verdes que vemos a continuación, permitiéndonos establecer distintos filtros.

En este ejemplo, nos encontramos con un filtro, que nos acota los sensores cuya medida del nivel de batería es menor al 75%, otro que acota la temperatura, siendo esta menor de 25 °C, así como otro que nos permite mostrar múltiples sensores creando una comparativa entre ambos. Si ponemos el cursor encima de los botones, vemos lo siguiente:

- Activar Filtro : siempre que esté el tic, el filtro permanecerá activado, si no, el botón del filtro tendrá el color verde a rayas.
- Pin : este icono nos permite anclar el filtro para todos los dashboards.
- Conmutador : haciendo clic sobre él, revertimos la acción, quedando el filtro de color rojo.
- Eliminación : mediante él, eliminamos el filtro.
- Personalización : mostrará un campo de texto como en la Figura 24, para modificar la representación JSON del filtro y añadirle un alias.

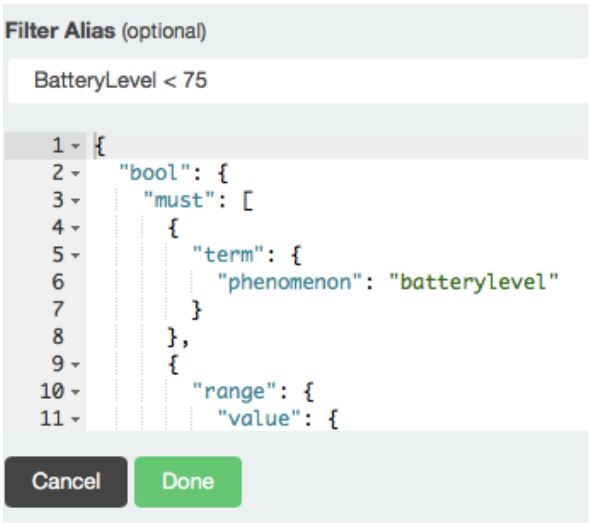


Figura 24. Personalización del Filtro de los Dashboards

Otras visualizaciones que se verán en los distintos dashboards son las siguientes:

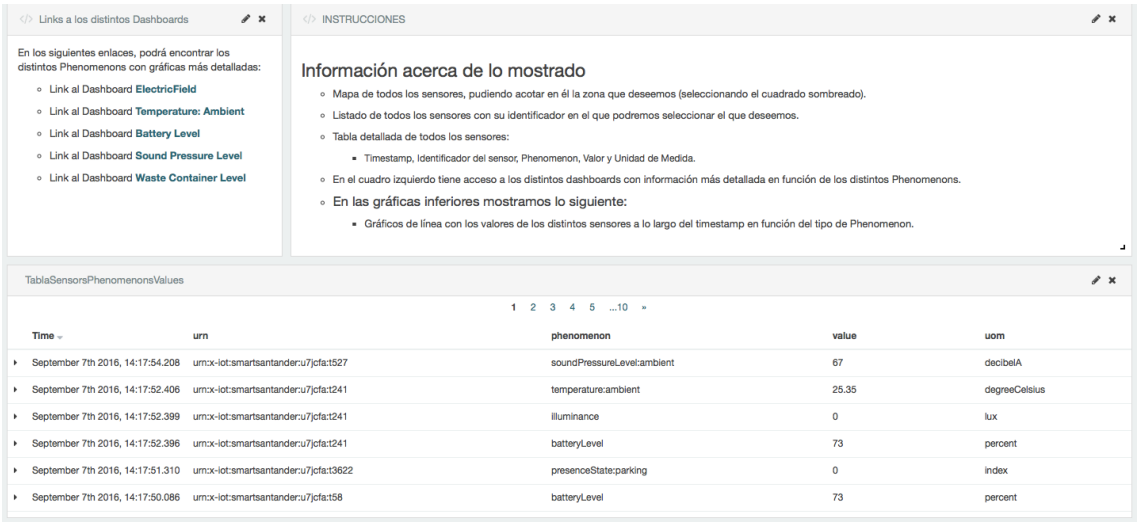
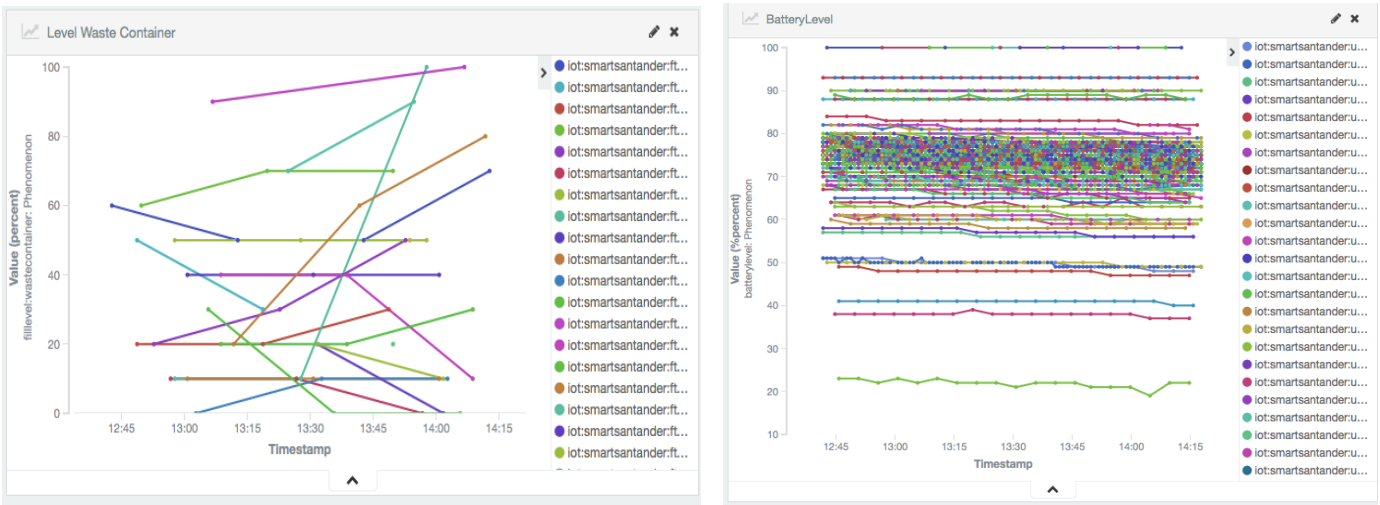


Figura 25. Visualizaciones del Dashboard General

Consta de dos cuadros de información, uno de ellos con enlaces a otros dashboards, con una tabla de los sensores, detallando el fenómeno, valor y unidad de medida de cada una de las medidas que provee dicho sensor.

Tanto el mapa como el listado, son interactivos, pudiendo en el mapa acotar una zona seleccionada sobre él, limitando el número de sensores mostrados, así como seleccionar en el listado, el identificador de unos de los sensores, para mostrarlo en el mapa.

En este dashboard general, como vemos en la Figura 26, hemos realizado un gráfico lineal de todos los fenómenos, mostrando en el eje “X” el timestamp y en el eje “Y” el valor medido.



a) b)

Figura 26. Gráficas de Línea del Dashboard General. (a) Nivel de llenado de los contenedores de basura (b) Nivel de carga de las baterías

En la Figura 26 a, vemos el nivel de llenado de los distintos contenedores de basura distribuidos en Santander,por otra parte, en la Figura 26 b, nos muestra el porcentaje de nivel de batería restante de los distintos sensores. Estas dos gráficas son análogas a las demás mostradas en el dashboard general sobre los diferentes fenómenos.

Una vez visto esto, profundizaremos en las gráficas más relevantes de los distintos dashboards específicos.

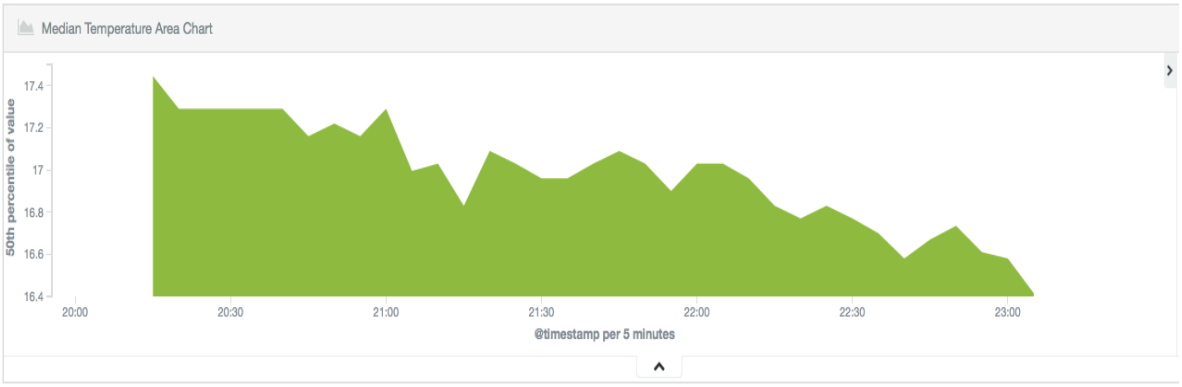


Figura 27. Gráfica de Área de la Temperatura Media

En la Figura 27, vemos un gráfico de área, mostrando la temperatura media acumulada de todos los sensores a lo largo de un periodo de tiempo.

Otro ejemplo de visualización, es la que vemos en la Figura 28, mostrándonos unos gráficos circulares en función del porcentaje de batería del sensor, divididos en 5 franjas de batería y a su vez, indicando las 10 últimas medidas de los sensores.

Figura 28. Gráficos Circulares y Lineales del Nivel de Batería

Otra representación circular, respecto al fenómeno de batería se puede ver en la Figura 29:

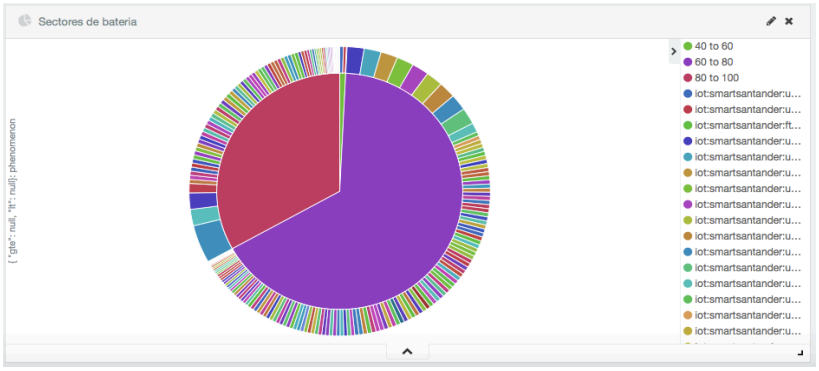


Figura 29. Gráfico Circular de la Batería

Este gráfico, divido en los 5 rangos de porcentaje de batería anteriores, nos indica qué porcentaje de los nodos está en dichos rangos de batería, ayudándonos a ver anomalías en las baterías y pudiendo detectar nodos que tengan problemas con su alimentación eléctrica.

Por último, destacar el gráfico de la Figura 30 en el cual se ha dividido el eje "y" en cuatro valores de frecuencia, mostrando una comparativa a lo largo del tiempo del campo electromagnético para cada banda de frecuencias medida por los sensores desplegados por la ciudad.

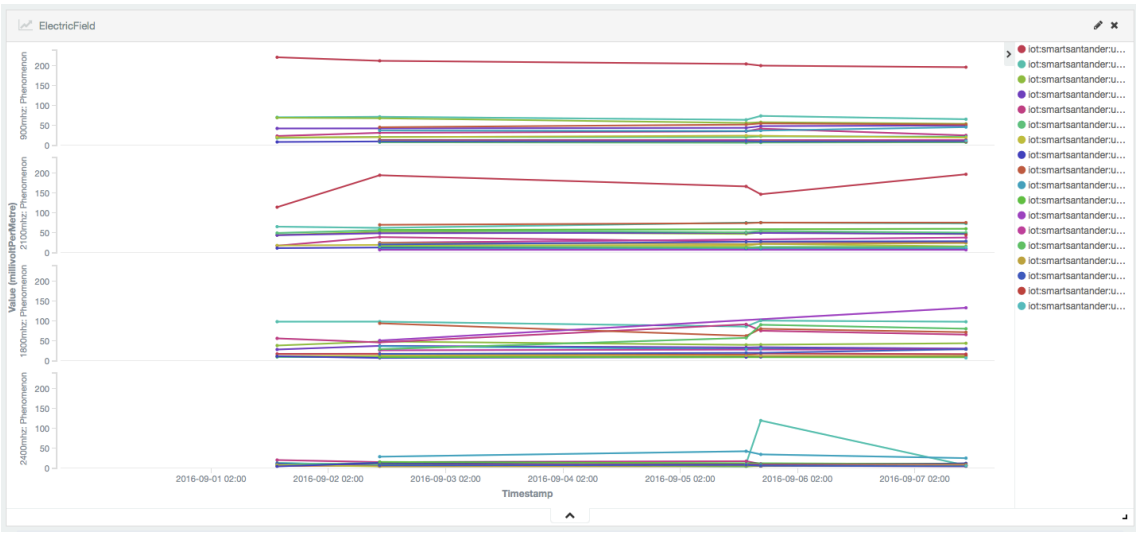


Figura 30. Gráfica del Campo Electromagnético

Capítulo 4. **Directrices para la Extensibilidad del Sistema**

Una vez explicado todo el desarrollo e implementación del Sistema de Control para el caso particular de SmartSantander, procederemos a definir las pautas que habrá que seguir para extender o adaptar la plataforma de representación a otro marco u otras necesidades dentro de SmartSantander.

En primer lugar, antes de proceder con la creación de nuevas visualizaciones y dashboards, es recomendable crear en el apartado "Discover", búsquedas en función de los distintos datos que se pretende representar, puesto que posteriormente en la pantalla "visualize", existen dos opciones de generar gráficos, la primera, seleccionando los datos en función del "index" y la segunda, en función de la búsqueda previamente realizada, acotando de esta manera los datos a representar.

Figura 31. Creación de Búsquedas en Kibana

En la (Figura 32),vemos las distintas visualizaciones que nos permite Kibana:

- Gráficos de Área
- Tabla detalla de los diferentes fenómenos
- Gráficos de Línea
- Cuadro de texto para indicar, por ejemplo, instrucciones
- Gráficos Circulares
- Mapa Geográfico
- Gráficos de Barras

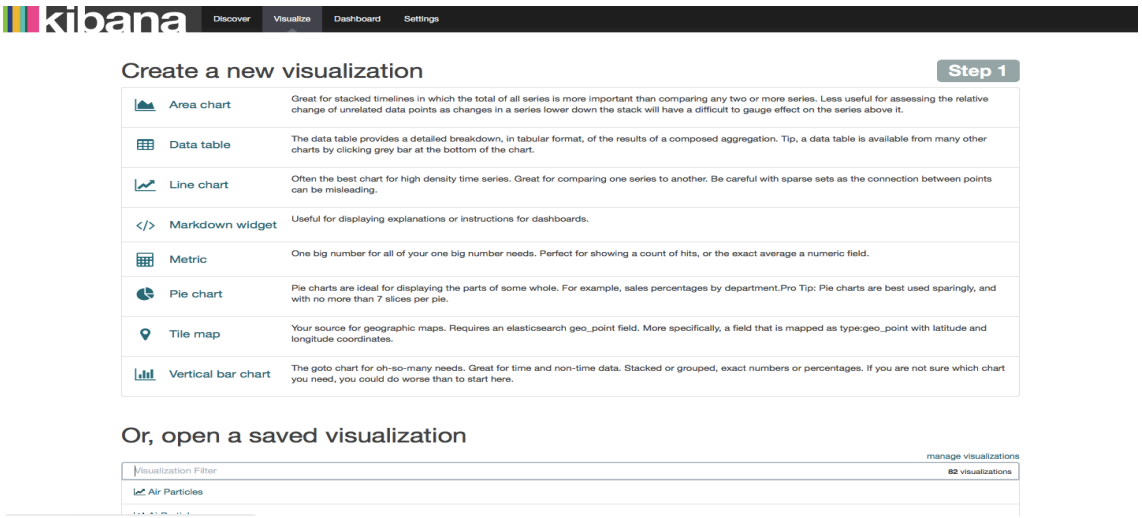


Figura 32. Opciones de Visualización de Kibana

Dentro de cada tipo de gráfico, vemos distintos campos, los mostrados en la Figura 33:

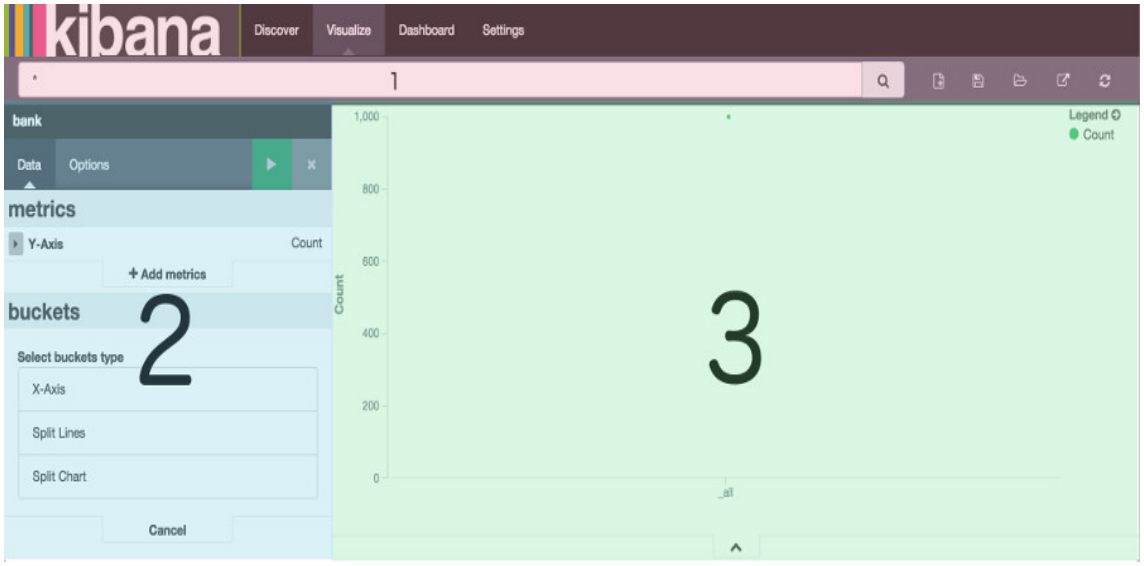


Figura 33. Ilustración de los Distintos Campos en la Creación de Visualizaciones

1. Barra de herramientas: en ella podremos escribir los distintos parámetros de búsqueda que queremos acotar, para su posterior representación.

2. Aggregation Builder: en ella nos encontramos los distintos campos a rellenar, tanto el eje "Y", que es el primero que tenemos que definir, así como las diferentes posibilidades en el eje "X".

3. Pre-visualización: en esta parte de la herramienta, podremos ir viendo las distintas gráficas según se va creando.

Una vez que hayamos generado la visualización acorde a nuestro sistema, deberemos guardarla, mediante el botón que se encuentra en la barra de herramienta con la siguiente ilustración: 

Para finalizar, explicaremos la creación de un dashboard que recoge las visualizaciones generadas anteriormente. En la Figura 34 vemos la barra de herramientas del dashboard.

Figura 34. Barra de Creación de Dashboard

-  Con este botón añadiremos las visualizaciones al dashboard, pudiendo ir colocando y ajustando el tamaño una a una.
-  Con este crearemos nuevos dashboards.
-  Por ultimo, con este compartiremos el enlace de nuestro dashboard.

Capítulo 5. **Conclusiones y futuras líneas de trabajo**

5.1 Conclusiones

El trabajo "Integración de un Sistema de Control para una plataforma de la Internet de las Cosas" corresponde al trabajo de Fin de Grado en Ingeniería de Tecnologías de Telecomunicación en la Universidad de Cantabria.

En este trabajo hemos integrado diferentes herramientas desarrolladas por la plataforma Elastic para la captación, almacenamiento y representación de los datos. El resultado de todo ello ha dado lugar a la creación de diferentes dashboards de control para facilitar la tarea de gestión e interpretación de los diferentes datos aportados por la plataforma SmartSantander.

Para valorar y extraer las conclusiones, debemos analizar los objetivos que nos propusimos al inicio del Proyecto, y de este modo poder obtener el nivel de satisfacción logrado y alcanzado, analizando todos los pasos dados para su obtención.

El objetivo principal era el diseño e implementación de un Sistema de Control que nos ofreciera una interfaz visual en el que concentráramos toda la información relativa a la infraestructura de dispositivos IoT desplegada en el Proyecto SmartSantander.

La consecución de este objetivo ha sido posible mediante:

- El conocimiento de la tipología de datos proporcionada por los sensores, llevando a cabo un estudio del modelo de datos utilizados por SmartSantander, ya que era necesario conocer y comprender el modelado de la información.

- La captura de los datos que nos proporciona para proceder a su análisis, evaluando las diferentes opciones permitidas por la Plataforma SmartSantander y seleccionando la que mejor se ajustaba a las tecnologías empleadas por el Sistema de Control que se ha desarrollado.

- Estudiando y comprendiendo cada una de las herramientas integradas en el Sistema de Control. A través de este estudio hemos podido explotar sus posibilidades y su utilización, llevándonos así a una exitosa puesta en funcionamiento de este Trabajo.

Hemos conseguido integrar y desarrollar el Sistema de Control, iniciando el proceso con la captación de datos con la herramienta Logstash, al ser ésta, un potente motor de recopilación de datos, lo que nos ha permitido unificar y estructurar los datos

provenientes de SmartSantander. Pero quizá, la parte más importante haya sido la configuración del filtro de Logstash, de este modo e investigando las distintas posibilidades de plug-ins, decidimos utilizar el plug-in ruby de Logstash para conseguir una trasformación de los datos completamente programables.

Se ha conseguido crear los distintos Dashboards para la visualización de nuestro sistema de control. Destacando fundamentalmente la creación de :

- Dashboard del campo electromagnético
- Dashboard de la temperatura ambiente
- Dashboard del nivel de batería
- Dashboard del nivel de llenado de los contenedores de basura
- Dashboard del nivel de sonido acústico

Por otra parte, la compresión tanto de la filosofía de la plataforma Elastic como de sus diferentes secciones es un objetivo que se ha logrado cumplir. Es notorio en el trabajo el tiempo que se ha tenido que invertir en la lectura y comprensión de diferentes documentos albergados en la plataforma para conseguir llevar a cabo el trabajo. Además, en este documento se ha descrito de una forma clara aspectos de esta plataforma para así facilitar, en la medida de lo posible, la comprensión de la plataforma Elastic a los lectores del mismo. Uno de los inconvenientes que se puede relacionar con este objetivo es la constante actualización de documentos técnicos que ha sufrido la plataforma a lo largo de este trabajo. Puede que esto haya sido un inconveniente a la hora del desarrollo, pero sin duda esto otorga mayor empaque al uso de esta plataforma, demostrando que este trabajo se enmarca dentro de una plataforma en continua expansión y cuenta con el apoyo de gigantes de las telecomunicaciones.

En relación con las bases teóricas, se ha alcanzado el objetivo de adquirir los conocimientos necesarios así como un repaso de aquellos que habíamos estudiado a lo largo de estos años universitarios en el Grado de Ingeniería de Tecnologías de Telecomunicación.

La satisfacción de haber logrado crear una herramienta propia hace que mis objetivos personales se hayan visto totalmente conseguidos, y se hayan visto compensada la gran cantidad de horas de estudio y de trabajo que lo han hecho posible.

5.2 Futuras Líneas de Trabajo

En base a las distintas herramientas implementadas en este Proyecto, se pueden realizar modificaciones en las mismas, para una ampliación y mejora de los distintos Dashboards de Control.

En primer lugar, con las constantes actualizaciones y mejoras que reciben las tres herramientas por parte de Elastic, una de las mejoras que han desarrollado actualmente es su propia nube, Elastic Cloud, por tanto, el uso de ella para la implementación y despliegue de Logstash, Elasticsearch y Kibana, sería recomendable para una mayor simplicidad, integración y convergencia de las mismas.

También, otra de las mejoras ha implementar en un futuro, sería añadirle la pila de Elastic, X-Pack, que incluye distintas herramientas para la mejora en los campos de seguridad, permitiendo la autenticación y la definición de roles y permisos, en la creación de alertas, pudiendo definir diferentes patrones de avisos y en el monitoreo a tiempo real, para la creación de distintos informes sobre el correcto o no funcionamiento de la misma e informes gráficos sobre los distintos dashboards.

Capítulo 6. **Bibliografía**

[1] *API de SmartSantander.* https://api.smartsantander.eu.

[2] *Documentación de ElasticSearch.* https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html.

[3] *Documentación de Kibana.* https://www.elastic.co/guide/en/kibana/current/index.html.

[4] *Documentación de Logstash.* https://www.elastic.co/guide/en/logstash/current/index.html.

[5] Domotys. *SMART CITY, HACIA LA GESTIÓN INTELIGENTE.* Madrid: Mancobo, 2014.

[6] Fernández, Manu. *DESCIFRAR LAS SMART CITIES: ¿QUE QUEREMOS DECIR CUANDO HABLAMOS DE SMART CITIES?* Madrid: MEGUSTAESCRIBIR, 2016.

[7] *GitHub de Elastic.* https://github.com/elastic.

[8] IE Businnes School, Gildo Seisdedos. «Informe sobre Smart Cities de PWC, IE Business School y Telefónica.»

[9] *Información sobre las peticiones HTTP.* http://code.tutsplus.com/tutorials/http-the-protocol-every-web- developer-must-know-part-1--net-31177.

[10] Jorge Lanza, Pablo Sotres, Luis Sánchez, José Antonio Galache, Juan Ramón Santana, Verónica Gutiérrez, Luis Muñoz. «Managing Large Amount of Data Generated by a Smart City Internet of Things Deployment.» *International journal on Semantic Web and information systems.* Santander: Information Resources Management Association, IGI Global, 2016.

[11] Peterson, Armando Fox & David. *Desarrrollando Software como Servicio.* Strawberry Canyon LLC, 2015.

[12] ECMA-404. ECMAScript® Language Specification. *"The JSON Data Interchange Format".* Octubre 2013.

Anexo 1. Archivo de configuración de Logstash

```
input {

            http {

            additional_codecs => {“application/json” => “plain”}

            }

      }

      filter{
          grok       {       match       =>       [       "message",
"%{HTTPDATE:[@metadata][timestamp]}" ] }
          date    {    match    =>    [    "[@metadata][timestamp]",
"dd/MMM/yyyy:HH:mm:ss Z" ] }
      ruby {
            init => "
              require 'json'
              def parse_json obj, pname=nil, event
                obj = JSON.parse(obj) unless obj.is_a? Hash
                obj = obj.to_hash unless obj.is_a? Hash

                obj.each {|k,v|
                  p = pname.nil?? k : pname
                  if v.is_a? Array
                    v.each_with_index {|oo,ii|
                      if oo.is_a? Numeric
                          parse_json_array_num(oo,ii,p,event)
                      else
                          parse_json_array(oo,ii,p,event)
                      end
                    }
                  elsif v.is_a? Hash
                    parse_json(v,p,event)


                  else
                    p = pname.nil?? k : [pname,k].join('.')
                    event[p] = v
                  end
                }

              end


              def parse_json_array obj, i,pname, event
                obj = JSON.parse(obj) unless obj.is_a? Hash
                pname_ = pname
                if obj.is_a? Hash
                  obj.each {|k,v|

                    p=[pname_,i,k].join('.')
                    if v.is_a? Array
                      v.each_with_index {|oo,ii|
                        parse_json_array(oo,ii,p,event)
```

```
                    }
                  elsif v.is_a? Hash
                    parse_json(v,p, event)
                  else
                    event[p] = v
                  end
                }

              else
                n = [pname_, i].join('.')
                event[n] = obj
              end
            end
      def parse_json_array_num obj, i,pname, event
              pname_ = pname
              if obj.is_a? Hash
                obj.each {|k,v|

                  p=[pname_,i,k].join('.')
                  if v.is_a? Array
                    v.each_with_index {|oo,ii|
                      parse_json_array(oo,ii,p,event)
                    }
                  elsif v.is_a? Hash
                    parse_json(v,p, event)
                  else
                    event[p] = v
                  end
                }

              else
                n = [pname_, i].join('.')
                event[n] = obj
              end
            end
          "
           code => "parse_json(event['message'].to_s,nil,event) if
event['message'].to_s.include? ':'"
        }
      ##cambiar el . por _
      ruby {
          init => "
              def remove_dots hash
                  new = Hash.new
                  hash.each { |k,v|
                      if v.is_a? Hash
                          v = remove_dots(v)
                      end
                      new[ k.gsub('.','_') ] = v
                      if v.is_a? Array
                          v.each { |elem|
                              if elem.is_a? Hash
                                  elem = remove_dots(elem)
                              end
                              new[ k.gsub('.','_') ] = elem
                          } unless v.nil?
                      end
                  } unless hash.nil?
```

```
                    return new

                end

            "
            code => "

event.instance_variable_set(:@data,remove_dots(event.to_hash))
            "
          }
          mutate {
                    add_field          =>          [          "[geo]",
"%{[location][coordinates][0]}" ]
                          add_field        =>        [        "[geo]",
"%{[location][coordinates][1]}"]
                          convert => [ "[geo]", "float" ]
            }
          }
       output{
             elasticsearch {
                   index => "m"
             }
       stdout {
             code => rubydebug
             }
       }
```