AT

Plataforma de Fogar Dixital Xunta
Guía técnica de Integración

Elaborado por: AMTEGA

XUNTA DE GALICIA

amtega
Axencia para a Modernización Tecnolóxica

CONTROL DE VERSIÓNS E DISTRIBUCIÓN			
NOME DO DOCUMENTO:	Documento de referencia técnica de integración en la plataforma de Fogar Dixital Sociosanitario de la Xunta de Galicia	**VERSIÓN:**	01.00
COD. DO DOCUMENTO:			
ELABORADO POR:	AMTEGA	**DATA:**	
VALIDADO POR:		**DATA:**	
APROBADO POR:		**DATA:**	

REXISTRO DE CAMBIOS		
Versión	**Causa da nova versión**	**Data de aprobación**

LISTA DE DISTRIBUCIÓN (opcional)		
Nome	**Número de copia**	**Área/Centro/Localización**

Índice

1. INTRODUCCIÓN

El objeto del presente documento es servir de guía a la hora establecer las alternativas de integración posibles de cara a la inclusión de servicios digitales, tanto de carácter social como sanitario, sobre la plataforma integral de "Fogar Dixital de la Xunta de Galicia".

La plataforma de Fogar Dixital de la Xunta de Galicia, posibilita el intercambio de información entre distintos sistemas y dispositivos de forma ágil y segura. Proporcionando además capacidades Big Data y potentes herramientas de inteligencia analítica, permitiendo el almacenamiento y análisis de altos volúmenes de información. Esta plataforma está desarrollada sobre el middleware Sofia2, por lo que a lo largo del presente documento nos referiremos a ella de forma abreviada como "Sofia2.

Se trata de una Plataforma transversal sobre la que se desarrollan soluciones Smart que mejoran la eficiencia de los procesos y mejoran la experiencia de los usuarios.

Este documento, deberá ser tenido en cuenta a la hora de definir propuestas en el marco de la convocatoria "Consultas Preliminares del Mercado para la extensión de la Teleasistencia Avanzada bajo el modelo de Hogar Digital Sociosanitario" de la Xunta de Galicia, siendo un requisito a cumplir que los servicios propuestos se integren en esta plataforma según alguno de los protocolos y tecnologías descritos más adelante, para garantizar un estándar de interoperabilidad.

Indicar que en la actualidad, esta plataforma ya está siendo utilizada como soporte a los servicios de Telemedicina y Teleseguimiento implantados en el Servizo Galego de Saúde.

El contenido del documento se estructura de la siguiente forma:

- Descripción general de la plataforma: Introducción a la plataforma de forma muy general
- Arquitectura de la plataforma: Descripción más completa de los aspectos técnicos de la plataforma: Arquitectura, conceptos técnicos y operativos relevantes y mecanismos de seguridad definidos
- Integración en la plataforma: Guía técnica para comprender las posibilidades de integración y los detalles de implementación/uso de las mismas.
- Desarrollo de las integraciones: Explicación didáctica de los paso en los desarrollos de las integraciones (incluye algún ejemplo)
- Procedimiento regulado de integraciones en la plataforma: Explicación del procedimiento de solicitud e integración de servicios en la plataforma, que ayudará a contextualizar el modo en que los aspectos técnicos de la integración en la plataforma, explicados en los apartados previos, serán ejecutados/articularizados sobre la plataforma de "Fogar Dixital SocioSanitario" de la Xunta de Galicia.

En general a lo largo del documento, se dejarán enlaces a sitios web donde profundizar en los aspectos que describe el documento. Asimismo,. En un apartado final se recoge un resumen de estos sitios web de ampliación de información.

2. DESCRIPCIÓN GENERAL PLATAFORMA

Técnicamente, Sofia2 puede definirse como un middleware y repositorio con capacidades cognitivas que permite la interoperabilidad de múltiples sistemas y dispositivos, permitiendo el procesamiento de miles de eventos por segundo. Su visión semántica y su enfoque open source, multilenguaje y agnóstico de las comunicaciones la convierten en una plataforma abierta e interoperable.

Además de las características mencionadas, Sofia2 dispone de distintos mecanismos out-of-the-box para garantizar su integración con sistemas y dispositivos de naturaleza muy diversa. Entre estos mecanismos cabe destacar:

- La Plataforma ofrece distintos conectores, como REST, actualmente considerado el estándar de comunicación entre sistemas empresariales, también el protocolo Websockets (protocolo HTTP de comunicación bidireccional); MQTT (protocolo bidireccional adecuado para comunicación con dispositivos IoT); Ajax Push (protocolo de comunicación asíncrona desde el servidor al cliente). También soporta el protocolo FIWARE NGSI-10, permitiendo de este modo la compatibilidad directa con componentes y plataformas a través de dicho protocolo.

- La Plataforma es muy flexible en cuanto a que ofrece APIs multilenguaje para facilitar el desarrollo de cualquier cliente que quiera comunicarse con la Plataforma. Cuenta con APIs en lenguajes habituales como Java y C, APIs para el desarrollo de aplicaciones móviles tanto en Android como en iOS y APIs en tecnologías como Javascript, NodeJS, Arduino, Python o .NET.

Además, el enfoque semántico de la Plataforma permite modelar la información en *ontologías*, que posibilitan gestionar la información de manera completamente agnóstica del protocolo tecnológico usado para enviar el dato, lo que se traduce en que independientemente del protocolo, la información se gestiona de la misma forma.

3. ARQUITECTURA DE REFERENCIA DE SOFIA2

3.1. Arquitectura modular

Sofía2 está basada en una arquitectura modular que organiza sus módulos en diferentes etapas del ciclo de vida del dato, ofreciendo diferentes opciones tecnológicas para cada una de dichas etapas. Las propuestas de servicios relacionados con la Tele-asistencia Socio-sanitaria, que se considere integrar sobre la plataforma de "Fogar Dixital Sociosanitario de la Xunta de Galizia, pueden implicar orígenes de datos (Sources), que deben enviarse a la plataforma (Ingest & Process), y / o servicios digitales de consulta o publicación de información en la plataforma (Publish & View), los esquemas y definiciones se centran en estas tres fases, dejando en modo resumido las otras fases.

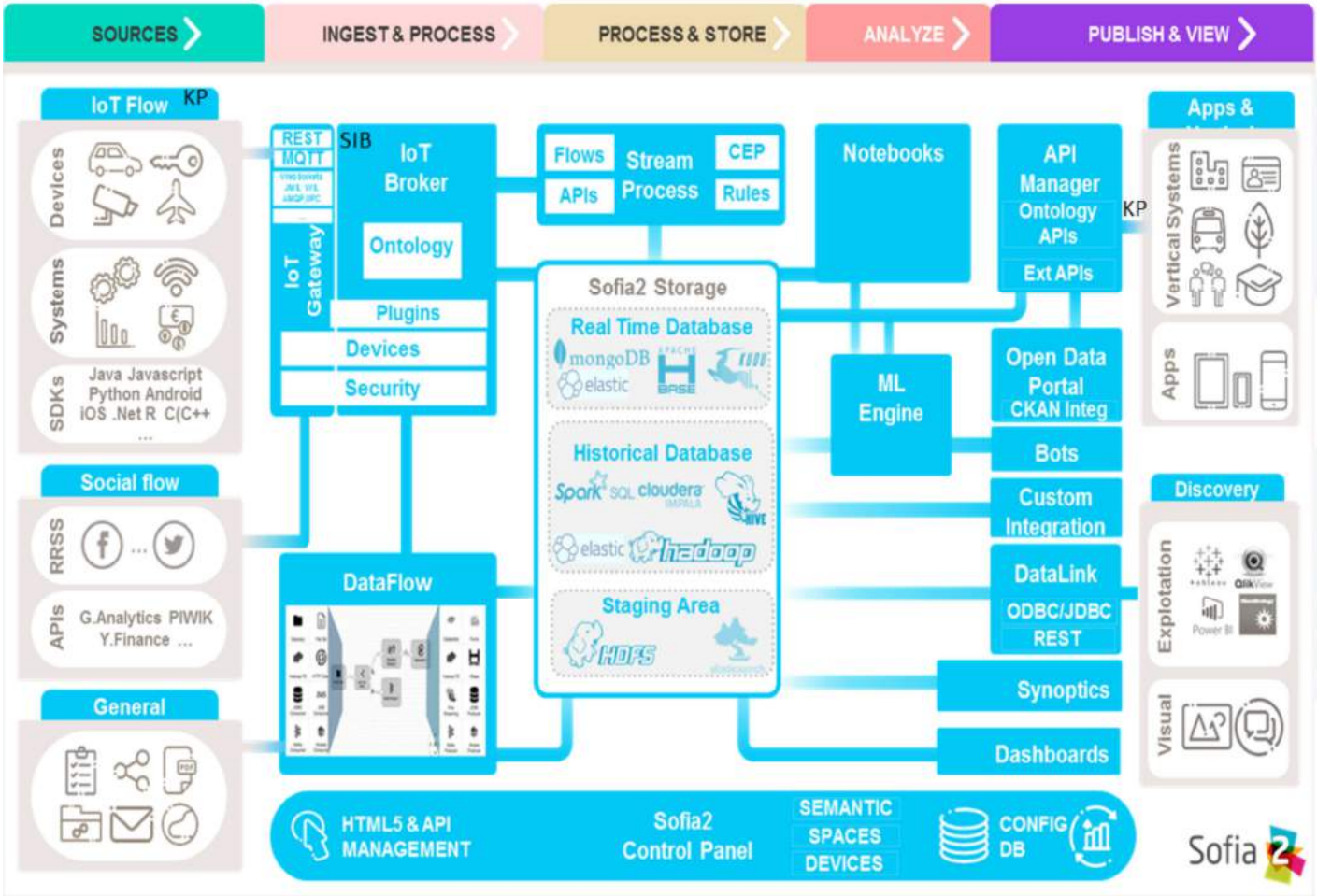


Figura 1 .- Arquitectura modular global de la plataforma

Como se puede comprobar en el diagrama, el flujo digital de información en la Plataforma circula desde las fuentes de datos (a la izquierda) hasta los consumidores de información (a la derecha), teniendo las siguientes etapas:

- **Ingesta y procesamiento:** La plataforma se nutre de la información generada por los llamados "productores de información", que pude ser cualquier sensor, dispositivo, solución, sistema… Dadas sus capacidades multidispositivo y de interoperabilidad, Sofia2 permite la ingesta de información de fuentes en tiempo real de prácticamente cualquier tipo de naturaleza, desde sensores y dispositivos hasta sistemas completos, cubriendo la mayoría de lenguajes estándar en su SDK. **En este punto, las propuestas que incluyan definición de dispositivos (sensores o actuadores) en el hogar, deben contemplar un proceso en el que sus concentradores establezcan comunicación con los posibles sistema de ingesta de datos definidos. A este respecto, se describen más adelante los mecanismos previstos para la solicitud y homologación de estas infraestructuras.**

- **Almacenamiento y procesamiento:** La información absorbida por la plataforma se almacena en el módulo central de almacenamiento, y se puede procesar en tiempo real.
- **Analítica:** Toda la información almacenada en Sofia2 puede tratarse de forma integral con herramientas de analítica avanzada.
- **Publicación y visualización:** Sofia2 pone a disposición de aplicaciones, soluciones, sistemas y usuarios toda la información almacenada en la plataforma a través de distintos mecanismos, como su API manager o su módulo Datalink. También cuenta con herramientas de visualización que permiten explotar de forma sencilla y gráfica la información almacenada dentro de la Plataforma; se pueden crear elementos de visualización unitaria (gadgets), unirlos en una página web (dashboard) o incluso crear complejos sinópticos al estilo SCADA. **En este punto cualquier propuesta servicio de tele-asistencia en el hogar digital tiene que comunicarse con el sistema mediante las opciones de vista y publicación existentes.**

3.2. Conceptos tecnológicos

Para comprender bien la operativa de la Plataforma Sofia2, es importante establecer los siguientes conceptos principales::

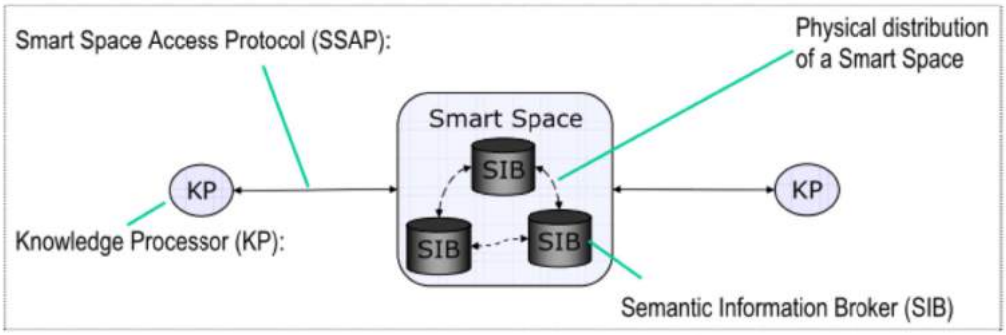


Figura 2 .- Interacción entre los conceptos tecnológicos más importantes

Smart Space

- Es el entorno virtual donde diferentes aplicaciones interoperan por ejemplo definiendo dispositivos (**Things**), modelando sus entidades, aplicando algoritmos o creando visualizaciones.
- El núcleo de un Smart Space es el Semántic Information Broker (SIB).
- En un Smart Space suele existir un único SIB (que puede ser un cluster), aunque en casos concretos pueden existir federaciones de SIBs.
- Los Smart Spaces pueden comunicarse entre ellos estableciendo relaciones de confianza.

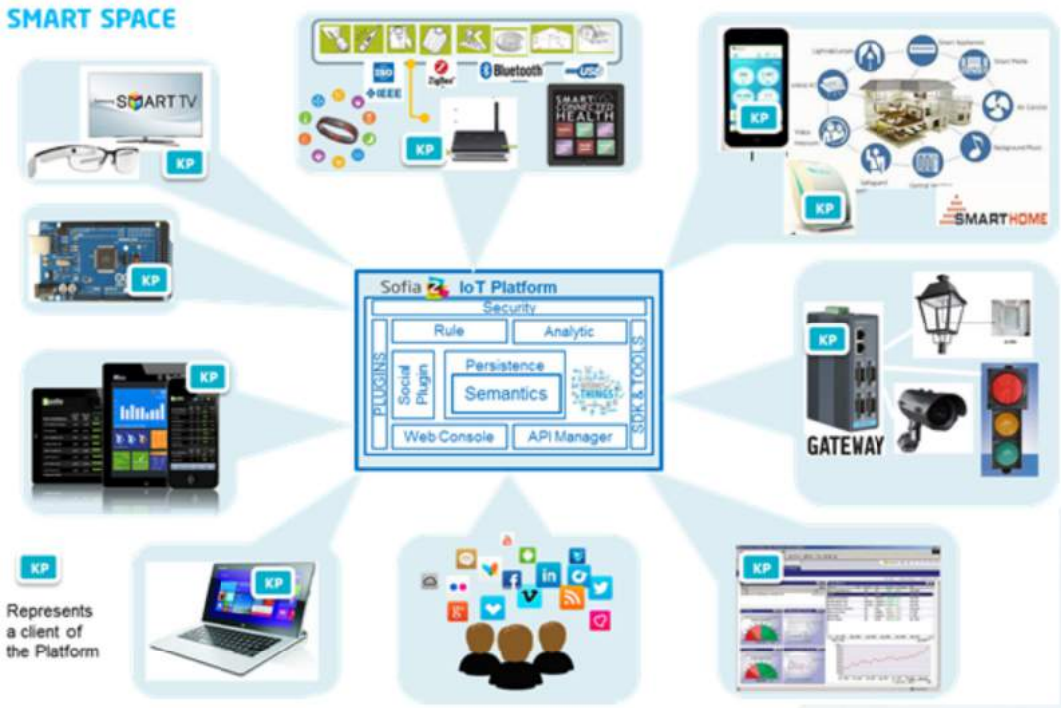


Figura 3 .- Esquema de un Smart Space

Entity/Ontología

- Las ontologías son descripciones semánticas de un conjunto de clases, representan las entidades de mi sistema. Por tanto el Modelo de Dominio que maneja un dispositivo (**Thing** en terminología Sofía2) se basa en ontologías.
- Las Ontologías se representan en **JSON** (con un Schema que valida si la información semántica enviada por el **KP** cumple las reglas de forma de dicha ontología) y pueden representar un modelo básico (como si fuera una Tabla) o un modelo complejo con relaciones (como si tuviésemos un conjunto de tablas relacionadas), , por ejemplo, una ontología que usa **KP** que representa a un sensor de temperatura sería la siguiente:

```
"SensorTemperatura":
{
    "coordenadaGps":
    {
     "altitud": 0,
     "latitud": 40.512274,
     "longitud": -3.675679
    },
    "identificador": "S_Temperatura_00001",
    "medida": 19,
    "timestamp": 1373887443001,
    "unidad": "C"
}
```

- Las Ontologías pueden crearse de diversas formas: visualmente en un diagrama de clases **UML**, a través de un esquema **JSON o XML**, campo a campo o a partir de un **CSV/XLS**.
- Cuando un Dispositivo (**Thing**) envía una medida hablamos de **Instology** (Instancia de Ontología).

ThinKP o KP (Knowledge Processor)

- Es cada uno de los elementos que interopera en la plataforma con el Smart Space a través del SIB, bien publicando y/o consumiendo información en la misma.
- Cada aplicación trabaja con instancias de los conceptos relevantes del dominio (ontología) para la que están diseñada.
- Implementaciones en diversos lenguajes como Java, Javascript, Arduino,…
- Hay 3 tipos de KPs:
  - **Producer**: KP que solo inserta información en el SIB.
  - **Consumer**:KP que solo recupera información del SIB.
  - **Prosumer**: KP que inserta y recupera información del SIB indistintamente
- En Sofia2 se propone el envío de mensajes SSAP en JSON que son más ligeros y adecuados a dispositivos embebidos.
- Una **Thing** puede representar desde un dispositivo sencillo (un Arduino o un iBeacon) a un Gateway (una Raspberry) o un Sistema Empresarial (Backend Java u otro) además puede manejar una o varias Entities (ontologías). También puede al ponerse en ejecución y crear una **instathing** (Instancia de KP), asociado a una **Thing** pueden crearse varias Instancias.
- Todas las comunicaciones con Sofia2 están securizadas. En el caso de las comunicaciones desde una **instathing** a la plataforma, tendremos un **token** de autenticación que garantizará que la **thing** conectada está registrada y autorizada para hacer la operación.

Asset (Think Type)

- Un Asset permite definir las características estáticas de una **Thing**. Puede usarse para definir tipos de dispositivos (p. e. farolas en una ciudad o motores en una planta) o hacer una gestión de activos.

SIB (Semantic Information Broker)

- Es el componente principal de la Plataforma.
- Recibe, procesa y almacena toda la información de las aplicaciones conectadas la plataforma SOFIA, actuando de Bus de Interoperabilidad
- En él se reflejan todos los conceptos existentes en el dominio (reflejados en las ontologías) y su estado actual (instancias particulares de ontologías).
- En Sofia2 se propone el uso de **JSON** para el intercambio de información (protocolo **SSAP**) y para la definición de las ontologías.

```
{
 "body":
 "{
    "query":
    "{
       SensorHumedad.medida: {$gt:18}
    }"
 }",
 "direction": "REQUEST",
 "ontology": "SensorHumedad",
 "messageType": "QUERY",
 "messageId": 121,
 "sessionKey": "88bf5ee7-84d4-4956-98a3-ff290222fd64"
}
```

- Existen implementaciones en diversos lenguajes y plataformas. Sofía2 suministra un SIB JEE que corre sobre cualquier Servidor Web JEE (Tomcat, JBoss,…)
- Gateway soporta manejadores de transporte TCP/IP, HTTP, REST, Bluetooth y Zigbee
- Ofrecer conectores para comunicación desde diversos clientes:

- REST: para clientes Javascript, smartphones,..
- MQTT para comunicaciones bidireccionales y dispositivos limitados
- WebSockets y Ajax Push.
- Otros como Bluetooth, Zigbee,..

- SIB extensible a través de plugins.

SSAP (Smart Space Access Protocol)

- Es el lenguaje de mensajería estándar para comunicar entre los **SIBs** y los **KPs**.
- Lenguaje es independiente de la red subyacente (GPRS, 3G, WIFI, BlueTooth, HFC, Zigbee)
- Existen dos implementaciones:
  - SSAP-XML: formato XML (mayor ancho de banda)
  - SSAP-JSON: mensajes adaptados a este protocolo, pensado para comunicaciones con dispositivos móviles, navegadores,…
- Mensajes de 3 tipos:
  - REQUEST: petición, enviada desde el KP al SIB
  - RESPONSE: Respuesta, enviada desde el SIB al KP en respuesta a un mensaje de REQUEST.
  - INDICATION: Notificación, enviada desde el SIB al KP ante un evento al que el KP está subscrito.
- Las operaciones que se realizan entre el SIB y los KP son las siguientes
  - JOIN: conexión de un KP a un SIB (implica autenticación, autorización y creación de sesión en el Smart Space)
  - LEAVE: desconexión de un KP del SIB
  - INSERT/UPDATE/DELETE: permite a los KPs la inserción/actualización/borrado de información realizada sobre el SIB
  - QUERY: permite a los KPs recuperar información del SIB: Puede ir sobre la Base de Datos de Tiempo Real e Histórica.
  - SUBSCRIBE: permite a los KPs suscribirse a la ejecución de una consulta cada X segundos o bien al desencadenado de un evento en el SIB
  - INDICATION: resultado enviado por SIB a uno o varios KPs para resolver una suscripción
  - UNSUBSCRIBE: Da de baja una suscripción realizada
  - CONFIG: permite al KP solicitar la configuración asociada a su instancia.
- Notificar cambios desde el SIB a suscriptores.

3.3. Seguridad de la Plataforma.

Debido a que los datos que se manejan en la plataforma de Fogar Dixital Sociosanitaria de la Xunta de Galicia, son muy sensibles, en este apartado se describen los mecanismos de seguridad que proporciona la plataforma para garantizar la seguridad y confidencialidad.

La plataforma Sofia2 es un entorno de interoperabilidad controlado, que implementa varios mecanismos de seguridad que operan a lo largo de las diferentes capas ofreciendo autorización, autenticación, consistencia y en su conjunto protección integral de los datos.

La seguridad en la plataforma Sofia2 es implementada a través del mecanismo de plugins que permiten realizar una personalización total de la Autenticación y la Autorización, permitiendo implementar mecanismo basados en diferentes estándares (LDAP, BD, Oauth…).

La Seguridad cubre las diferentes capas del aplicativo:

- Las Comunicaciones: Todos los Gateway implementados en la plataforma permiten la securización del canal a través del uso de Certificados SSL.
- Los Clientes: La plataforma solo permite la conexión de aquellos clientes que han sido autorizados en la plataforma.
- Las Operaciones: La plataforma solo permite realizar las operaciones para las que se tiene permiso.
- La Información: La plataforma solo te permite interactuar con la información para la que has sido autorizado, y valida que la información que se persiste en la plataforma cumple con el modelo de la información esperado.

3.3.1. Seguridad en las Comunicaciones

Sofia2 utiliza el canal seguro de comunicaciones https para todas las comunicaciones con el exterior. Este canal utiliza un protocolo SSL/TSL con un cifrado estándar RSA que necesita de un certificad de seguridad X.509. Es el estándar de cifrado más extendido y aceptado en el mundo del www.

La clave privada se instala en el SIB Sofia2 y su parte pública (conocida como certificado) deberá instalarse en los navegadores o ser incorporado en las apps que necesiten acceder a la plataforma desde la extranet. La seguridad mediante certificados prevendrá filtrado o manipulación de datos entre cliente y servidor mediante ataques del tipo man-in-the-middle.

Hay que destacar que el sistema de autenticación será mediante certificado de servidor, no de cliente-servidor (como podrían ser los certificados de la fnmt). Esta tipo de encriptado garantiza que el cliente se está conectando con el servidor requerido y no a un posible servicio impostor, pero no autentica al cliente.

3.3.2. Seguridad en los Clientes

Un usuario deberá registrar en la plataforma sus KPs (Cliente), de lo contrario, la plataforma rechazará la conexión de los mismos.

Para ello necesitará de un **Token** de Autenticación que el usuario podrá crear asociado al KP (según las restricciones de su rol). Los usuarios con rol Colaborador o rol Usuario únicamente podrán dar de alta tokens para KP's de los que sean propietarios.

Los KP's tienen también una clave de cifrado que sólo es necesaria si quiero usar XXTEA como protocolo de encriptación. Se utiliza en dispositivos que no soportan HTTPs, como por ejemplo Arduinos.

Un KP podrá hacer uso de una o varias ontologías, siendo esta la información que producirá o consumirá de la plataforma.

Una vez registrado en la plataforma, el KP ya podrá establecer conexiones con la misma.

Tras la creación de un KP podemos dejar definida una instancia KP a través de la Consola Web.

Una instancia KP identifica al cliente que se va a conectar a la plataforma Sofia2.

La conexión de un KP con la plataforma debe ser vista como dos tipos de conexión

Conexión Física: Establecida por el protocolo de trasporte utilizado para la conexión por un KP. La manera de realizar esta tipo de conexión depende en gran medida del API de KP utilizado (Java, Javascript, Arduino, C++...).

Conexión Lógica: Establecida por el protocolo SSAP (Smart Space Access Protocol) de mensajería definido en SOFIA. Es común a todos los APIs de KP.

Nos vamos a centrar en la seguridad a nivel de conexión Lógica que debe mantener un KP con la plataforma, pues la seguridad a nivel Física es cubierta por la seguridad en la capa de Comunicaciones.

Para que un KP pueda conectarse a la plataforma y producir/consumir datos e interoperar con otros KP, es necesario que abra una sesión con un SIB de la plataforma.

El protocolo SSAP proporciona dos operaciones en este sentido:

JOIN: Donde un KP informa a la plataforma el usuario y password de su propietario así sus datos de instancia, de manera que si todo es correcto, la plataforma autentica al KP y abre una sesión con el mismo (devolviendo una session key que podemos usar durante un tiempo preestablecido y configurable).

LEAVE: Donde un KP informa a la plataforma que va a cerrar la sesión.

Mientras exista una sesión entre el KP y la plataforma, el KP podrá utilizar el resto de operaciones del protocolo SSAP para producir/consumir información.

3.3.3. Seguridad de Acceso

La plataforma está dividida en dos áreas con independencia en su modelo de seguridad.

La Administración.

En la plataforma se han definido tres tipos de Roles, que definen las funcionalidades que dispondrán los usuarios a nivel administrativo:

- Rol Administrador.
- Rol Colaborador: este rol permite operar con la información de la plataforma, volcando y consumiendo información y crear nuevas estructuras de información además de realizar tareas de procesado de información (Reglas, Script, Informes).
- Rol Usuario: este rol permite operar con la información de la plataforma, volcando y consumiendo información de estructuras de información existentes en las que ha sido autorizado.

La Operación.

Los permisos, para los que también existen 3 tipos, definen las funcionalidades de los usuarios a nivel operativo sobre la información.

Estos permisos se aplican a nivel de Ontología – Usuario.

Los administradores tienen Permiso Total sobre todas las Ontologías.

Los Colaboradores tienen Permiso Total sobre las ontologías de las que son Propietarios:

- Permiso de Lectura: Permite a un usuario o colaborador realizar únicamente operaciones de tipo Query sobre las ontologías para las que se le ha asignado este permiso.
- Permiso de Inserción: Permite a un usuario o colaborador realizar únicamente operaciones de tipo Insert sobre las ontologías para las que se le ha asignado este permiso.
- Permiso Total: Permite a un usuario o colaborador realizar todas las operaciones sobre las ontologías para las que se le ha asignado este permiso.

3.3.4. Seguridad en los Datos

Todas las operaciones son validadas a nivel de Autenticación, para lo que la plataforma comprueba si el Cliente se ha autenticado con la plataforma.

Una vez que se ha comprobado la Autenticación del Cliente se comprueba su autorización en dos niveles:

Primero, se valida que el usuario puede operar con la Ontología para la que quiere realizar la operación.

Segundo, se valida que la operación (Query, Insert, Delete, Update) que quiere realizar el usuario, la puede realizar sobre esa ontología (Tiene los permisos adecuados).

Si todos los pasos anteriores han sido correctamente comprobados y la operación es Insert o Update todavía se ha de realizar una tercera validación, que consisten en comprobar que la información que se inserta cumple escrupulosamente con el Esquema que se ha definido, a través de la validación del JSON Schema, casando la información que está insertando con la estructura de la Ontología.

3.3.5. Implementación de Referencia

La implementación de referencia de la Seguridad está basada en tres plugins:

plugin-sofia-user

Este plugin (Usado únicamente a nivel de Administración) es el encargado de recuperar la información de los usuarios. En la implementación de referencia la recupera de la base de datos de configuración.

Tiene la capacidad de trabajar con Password encriptada o en claro, permitiendo configurar el Algoritmo de encriptación.

plugin-console-security

Este plugin es el encargado de gestionar la Autenticación y Autorización en la consola de Administración y se basa en Spring Security. En la implementación de referencia hace uso de la base de datos de configuración.

Hace uso de **plugin-sofia-user** para recuperar la información de los usuarios.

plugin-sib-security

Este plugin es el encargado de gestionar la Autenticación y Autorización a las operaciones del SIB y está basado en un mecanismo de Token – SessionKey.

Hace uso de plugin-sofia-user para recuperar la información de los usuarios.

4. PROCESO DE INTEGRACIÓN CON SOFIA2

Una vez conocidas las características de Sofia2, se van a detallar los distintos módulos que permiten la integración de plataforma con productos de terceros, explicando los distintos mecanismos para que cualquier empresa pueda realizar la integración con Sofia2, así como construir nuevas soluciones aprovechando las capacidades de la plataforma.

4.1. Mecanismos de integración

La Plataforma ofrece múltiples capacidades de integración, de modo que las distintas empresas que quieran integrar sus productos con la Plataforma puedan elegir el modelo de integración más adecuado, y en el siguiente gráfico se pueden ver los 4 módulos más importantes para esta integración que deberán elegirse en función de las necesidades de cada propuesta.

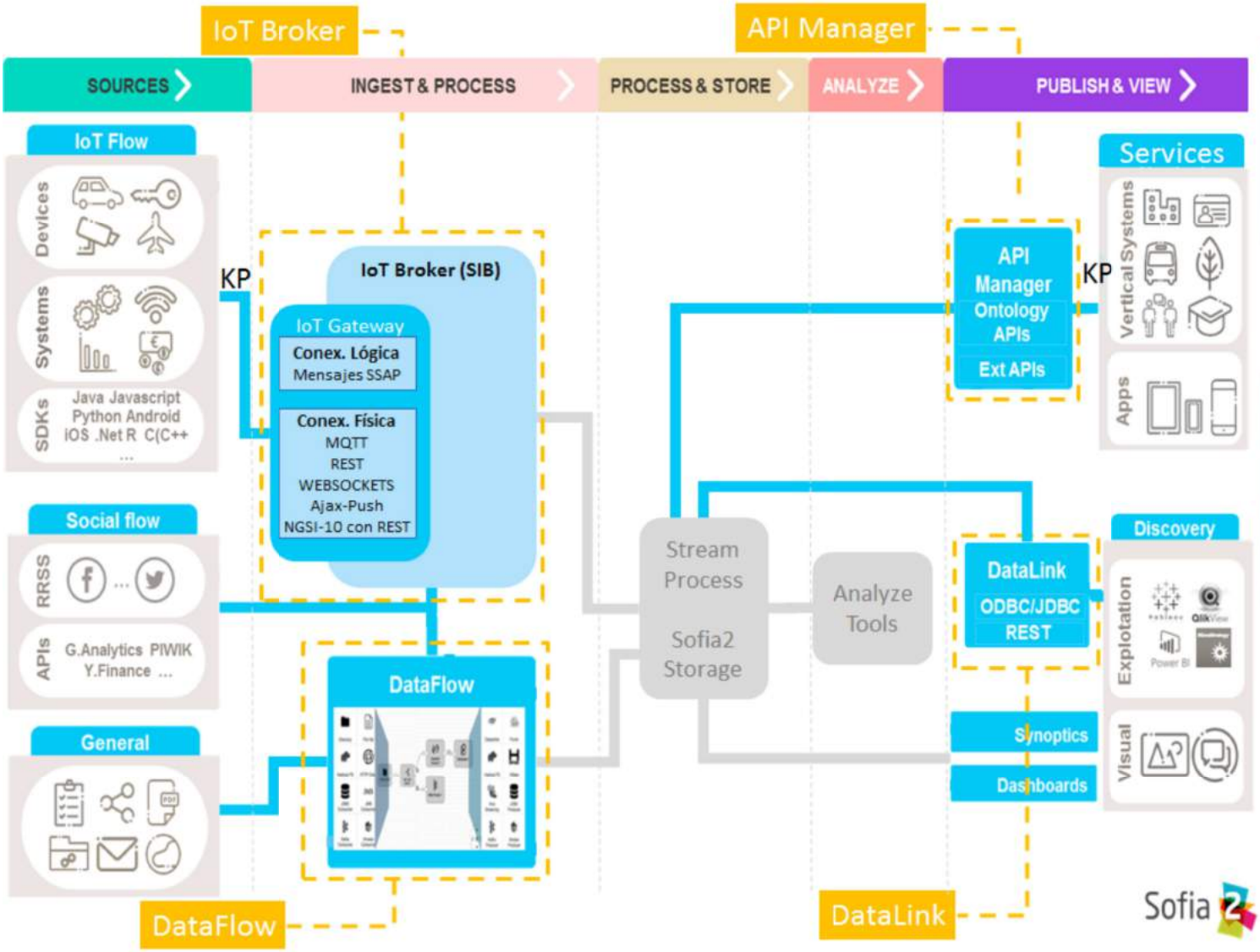


Figura 4 .- Identificación de los módulos que permiten la integración

Los módulos que proporciona la plataforma para esta integración son:

- **IoT Broker:** Es el Broker IoT de la plataforma, ofrece Gateways multiprotocolo (**REST, MQTT, WebSockets, Ajax Push (Javascript), FIWARE NGSI-10 sobre REST**) con comunicación bidireccional con los clientes de la plataforma. Es el mecanismo por defecto de integración. La plataforma ofrece además de interfaces REST, APIS multilenguaje (**Java, JS, C,Android, iOS,**

.NET, Python, Node.js, Arduino,…) que permiten que cada vertical desarrolle sobre su lenguaje y plataforma preferidos.

Es el mecanismo con el que deben integrarse los dispositivos, de modo que la plataforma permita la gestión de estos a través del Panel de control de la plataforma. Además de los dispositivos, los sistemas y aplicaciones que requieran algunas de estas capacidades (**suscripciones, comunicación bidireccional, Near Real Time, actualización SW y configuración…**) utilizarán también este módulo.

- **API Manager:** este módulo permite disponibilizar la información gestionada por la plataforma con interfaces REST. Estas APIs se crean desde el panel de control y pueden ser de consulta o inserción.

  Es el mecanismo típico con el que integran las aplicaciones web, portales y aplicaciones móviles con la plataforma. También pueden usarlo las herramientas BI.

- **DataFlow:** este módulo permite de forma muy sencilla configurar flujos de datos entre orígenes y destinos, donde los orígenes y destinos pueden ser la plataforma, ficheros, base datos…

  Es el mecanismo a usar cuando sea la plataforma la que va a recoger información del sistema origen y no este el que lo envía a la plataforma. Tras esto será necesaria una conversión en la plataforma.

- **DataLink:** este módulo disponibiliza interfaces **ODBC y JDBC** de los repositorios de la plataforma.

  Es el mecanismo tipo de integración para herramientas **BI y Data Discovery**, aunque estas también pueden usar el API Manager para comunicar con la plataforma.

4.2. Tipos de Integración

La Plataforma soporta diversos modelos de integración:

- **Integración base unidireccional:** en este modelo el sistema a integrar mantiene su base de datos y habilita los mecanismos para enviar (ThinKP que envía al IoT Broker) o que la plataforma recoja datos (vía DataFlow).
  - En esta integración el sistema puede usar o no la semántica (ontología) de la plataforma.
  - El sistema a integrar, en este caso, no recibe información de la plataforma.
- **Integración base bidireccional:** en este modelo, el sistema mantiene su base de datos, pero además de volcar información en la plataforma está preparado para que la plataforma pueda comunicarse con él.

  Se puede modelar de diversos modos, con un ThinKP que además de enviar está suscrito a ciertas notificaciones de la plataforma o habilitando un interfaz REST que se invoca por la plataforma.

- **Integración de repositorios:** en este modelo el sistema usa las capacidades de la plataforma como repositorio, y no mantiene una base de datos aparte.

  Al estar integrado con la plataforma, la infraestructura necesaria para desplegar el vertical es mucho menor (no requiere capa persistencia y requiere menor capa Back), en algunos casos una solución puede ser un simple Front HTML5 atacando a la plataforma vía el API Javascript. El sistema, por tanto, usa la semántica de la plataforma.

  Al estar integrado, la plataforma ofrece al sistema diversas facilidades como:

- **Dashboards y sinópticos** que pueden construirse sin programar
- **Motores de reglas** para reaccionar ante cualquier evento determinado.
- **Publicación** de cierta información como APIS.
- Capacidades para **ejecutar algoritmos** complejos.
- **Ejecución** del sistema vertical si este es puro HTML5 dentro de la plataforma.

4.3. Pasos para realizar la integración

A continuación se presentan los diferentes pasos que se deben seguir para integrar distintos sistemas y soluciones con la Plataforma.

4.3.1. Integración a través de IoT Broker

El procedimiento que cualquier empresa/desarrollador puede seguir para realizar la integración entre la Plataforma y su sistema o dispositivo utilizando el IoT Broker sería el siguiente:

1. Registrarse como usuario de la Plataforma de Hogar Digital disponible durante la fase de desarrollo.

   Una vez creada la cuenta, el usuario deberá solicitar el paso a rol *Colaborador*, con el fin de tener permisos de escritura.

2. Configuración de una serie de ontologías que modelen la información que se vaya a compartir entre el sistema externo y la Plataforma. Aunque la configuración de ontologías se realiza desde la Consola web, que ofrece un conjunto de herramientas que facilita la creación de este modelo semántico, si existen muchas iniciativas diferentes deberá establecerse un procedimiento más reglado en el que esta tarea la realicen unos responsables, para que no se produzca información redundante.
   Se puede visitar esta guía interactiva para ver paso a paso cómo crear una ontología.

3. Conexión del dispositivo a la Plataforma. Para permitir que cualquier sensor, aplicación o sistema pueda enviar/consumir información con Sofia2 se debe dar de alta el dispositivo en la Plataforma. Para ello es necesario la creación de un **ThinKP** a través del *menú ThinKP* de la Consola Web de Sofia2, al que se le asignarán las ontologías creadas en el apartado anterior.
   Se puede visitar esta guía interactiva para ver paso a paso cómo crear una ThinKP.

4. Conexión física del dispositivo. Consiste en el desarrollo de la aplicación que enviará/consumirá información de la Plataforma. Para ello el desarrollador deberá elegir y descargar la API más adecuada en función de las características del dispositivo: Java, Javascript, Android, iOS, Python, Node.js, C, Arduino, .Net.
   Todas las APIs de Sofia2 se pueden descargar en esta página.

Se puede encontrar información detallada sobre el modelo de datos, la creación y configuración de ontologías y la conexión de dispositivos en el documento Primeros Pasos con Sofia2 [1].

4.3.2. Integración utilizando el API Manager

Como ya se ha explicado, Sofia2 pone a disposición de otros sistemas la información gestionada por la plataforma mediante la configuración de servicios REST a través del API Manager. Este módulo se gestiona también a través de la Consola Web y permite definir operaciones de lectura, escritura (POST), actualización (PUT), borrado (DELETE) búsqueda básica o búsquedas avanzadas (GET).

Para ello los pasos son los siguientes:

1. Registrarse como usuario de la Plataforma y solicitar el paso a rol *Colaborador*.

2. Configuración de las ontologías que modelen la información que se vaya a compartir a través de la Consola web.

3. Publicación de APIs de acceso a los datos. Para crear un conjunto de operaciones sobre las ontologías definidas en el punto anterior se accede desde la Consola Web al menú *API Manager*. Desde ahí se pueden crear fácilmente nuevas APIs pinchando en *Nueva API* y seleccionando las ontologías y las operaciones que se desee.

Se puede encontrar más información sobre la publicación de APIs a través del API manager en el apartado *5.2 Publicando APIs de acceso a los datos* del documento Gestión de dispositivos en Sofia2 [2].

4.3.3. Integración con DataFlow

Como se ha comentado, Sofia2 habilita la ingesta masiva de información a través de su módulo DataFlow, para lo que es necesario la creación de un Pipeline desde la Consola Web de la Plataforma. Para ello hay que dirigirse a la opciones de *Menú Analytics --> Mis Pipelines.*

Tras seleccionar *Crear*, es sencillo generar un nuevo pipeline definiendo el origen, destino y las operaciones de procesamiento de la información.

Se puede encontrar información detallada de cómo utilizar el módulo DataFlow en el apartado *3. Ingesta de Datos* del documento Taller de Analytics [3].

4.3.4. Integración con DataLink

Sofía2 también permite la exportación de la información existente en la Plataforma mediante conectores ODBC y JDBC. El módulo actúa de interfaz con productos de analítica, ofreciendo conectores estándar JDBC, ODBC y REST y una capa de abstracción que permite operar a través de SQL, independientemente del origen de los datos. Ofrece el resultado de las consultas en un formato tabular, y en base a dichas consultas permite definir vistas.

Existe un tutorial en el blog de Sofia2 que explica paso a paso cómo publicar información existente en la Plataforma utilizando este módulo, que está disponible en este enlace.

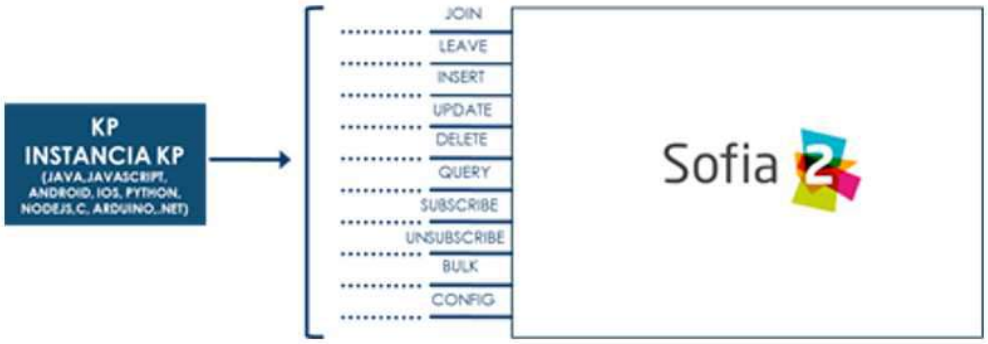
4.3.5. Desarrollo de servicios directamente sobre la Plataforma

Otra posibilidad es, en vez de integrar un sistema/dispositivo existente, desarrollar un nuevo servicio sobre la Plataforma. Para ello Sofia2 incorpora diferentes capacidades, así como un SDK, que permite que a cualquier empresa o desarrollador aproveche toda la potencia de la Plataforma para construir nuevos productos, que se pueden descargar en diferentes plataformas como describe en el apartado "Desarrollos sobre Sofia2" de este documento.

La Guía Primeros Pasos Desarrollador Soluciones [4] explica en detalle todos los pasos que debe seguir un desarrollador de soluciones para desarrollar diferentes piezas sobre la plataforma.

4.4. Gestión de Things o Dispositivos IoT

Para la integración de dispositivos IoT con la plataforma, teniendo en cuenta los conceptos tecnológicos explicados en apartados anteriores podremos comunicar nuestras "**things**" (dispositivos) con la plataforma, siguiendo el protocolo SSAP nativo de Sofía2.



Para ello, Sofia2 provee una serie de APIs de desarrollo (disponibles en nuestra web), de tal manera que esta comunicación se pueda implementar tanto en distintos lenguajes de programación, como en distintos protocolos de comunicación (**REST, MQTT, WebSockets, Ajax Push (Javascript), FIWARE NGSI-10 sobre REST**).

Una vez puesto en marcha el proyecto, la plataforma permite controlar la actividad de las conexiones desde varios puntos de vista:

KPs Activos

Desde el menú ThinKP Sofia2, submenú '***ThinKPs conectados'*** se pueden visualizar las conexiones activas, junto con sus datos (identificación, SessionKey y fecha de activación) .

Gestión de conexiones

Por otra parte, desde el menú de Administración, submenú '***Gestión de Conexiones***', se pueden visualizar las conexiones tanto desde el punto de vista físico como lógico, pudiendo hacer búsquedas, y cerrar conexiones o incluso bloquear clientes específicos.

Gestión de Configuraciones SW

Además, se pueden controlar las versiones de los clientes desplegados en nuestros things, y su configuración, mediante la gestión de configuraciones, donde se pueden asociar SW y parámetros de configuración a nuestros ThinKPs o a instancias de ThinKPs. (Figura 47)

De esta manera, si se desea actualizar la versión del SW con la que se conectan nuestros dispositivos o things, se puede actualizar la configuración de SW asignada, y la próxima vez que el dispositivo compruebe la versión de SW, se le informará que hay una nueva versión, pudiendo lanzar la descarga y actualización en cliente de manera automática.

Esta funcionalidad es muy útil en escenarios en que se tienen cientos de dispositivos conectados a la plataforma (por ejemplo, una smartCity, o una fábrica), y se desean hacer actualizaciones remotas de todos ellos.

Gestión de assets

A todas estas capacidades, y como funcionalidad adicional, podemos añadir la gestión de los assets (los elementos del mundo real conectados a través de nuestros ThinKPs), con funcionalidades como la geolocalización de cada uno de ellos, categorización y gestión de sus propiedades.

4.5. Modelado de Información en SOFIA2

Si bien este punto se incluye como parte de la explicación, para entender de forma global las características implicadas en la integración, dado que este proceso es critico de cara a garantizar la homogeneidad e interoperabilidad de la información, el modelado de las ontologías será gestionado por la administración, según se define en el flujo explicado en el apartado 6 de este documento.

4.5.1. Modelado de Ontologías

Las propuestas deberán reflejar un modelo de datos que defina la información que se intercambiará entre los diferentes elementos de la solución con la plataforma, de forma que permita en la fase de desarrollo convertir esos modelos de información en ontologías. No es necesario que las propuestas incluyan ya un modelo de ontologías, pero es recomendable una documentación suficiente para poder crearlas (la creación como se indica en el apartado 6 es responsabilidad de la Xunta de Galicia).

4.5.1.1. Un primer vistazo

Como hemos dicho en Sofia2 una ontología representa una entidad en mi sistema (SmartSpace), y esta se define en JSON. La Plataforma ofrece una Web (+API REST) de Configuración en la que los usuarios con permisos (colaboradores y administradores) pueden crear sus ontologías, aunque como también hemos mencionado puede ser que este proceso se delegue en el equipo responsable del gobierno de ontologías.

4.5.1.2. Tecnologías Implicadas

JSON: es un formato ligero para el intercambio de datos (como XML pero menos verboso)

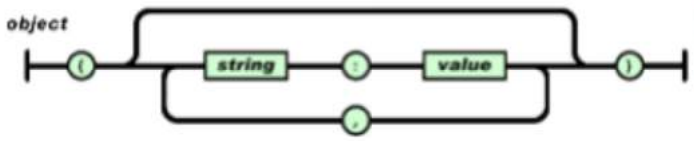
JSON-Schema: un esquema JSON es un documento JSON que permite especificar cómo es un documento JSON al que se refiere (si hay atributos obligatorios, si son de tipo number, si pueden ser nulos). En la equivalencia XML correspondería con un esquema XML o con un DTD.

Los tipos de datos más habituales para este formato son:

- **string** : Cadena de texto
- **number**: Numérico
- **object**: Objeto
- **char**: Caracteres Unicode válidos
- **array**: Colección de valore
- **null**: Nulo
- **boolean**: Valores true o false

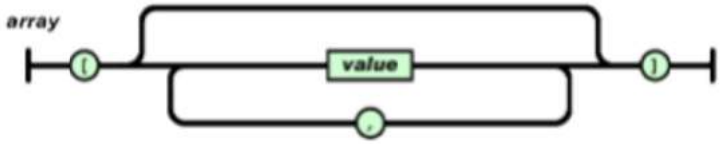
4.5.1.2.1. Tipos de datos de JSON

Un objeto es un conjunto sin ordenar de pares clave-valor. Comienza por "{" y termina con "}". Cada nombre estará seguido por ":", los pares clave-valor estarán separados por ",".




```
{
  "address" : {
    "line1" : "555 Main Street",
    "city" : "Denver",
    "stateOrProvince" : "CO",
    "zipOrPostalCode" : "80202",
    "country" : "USA"
  }
}
```

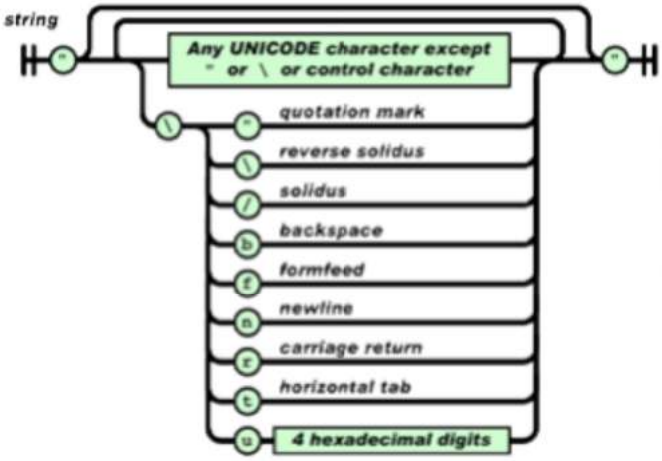
Un array es una colección de valores. Comienza por "[" y finaliza con "]". Los valores se separan por ","



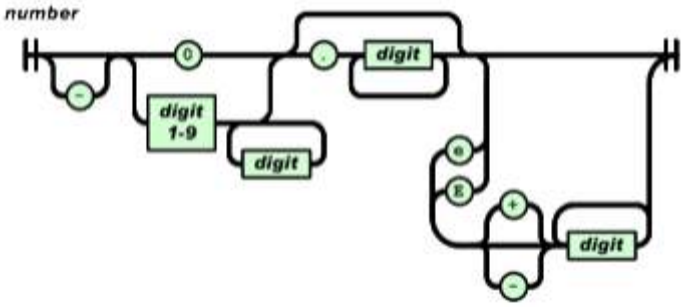

```
{
  "people" : [
    { "firstName": "John", "lastName": "Smith", "age": 35 },
    { "firstName": "Jane", "lastName": "Smith", "age": 32 }
  ]
}
```

Un valor puede ser una cadena de caracteres con comillas doble, un número, true, false, null, un objeto o un array. Esta estructuras pueden anidarse:

Un string es una secuencia de cero o más caracteres Unicode, encerrados entre comillas dobles (")



Un número es como un número decimal en Java.



4.5.1.2.2. Atributos de JSON-Schema

Se puede ver la referencia completa de la especificación del esquema de JSON aquí:
http://jsonschema.org/latest/json-schema-core.html

4.5.2. Gobierno de Ontologías

La Plataforma Sofia2 ofrece todos los mecanismos necesarios para interconectar cualquier tipo de red de sensores ofreciendo diversos mecanismos para ello. El presente documento, en aras de establecer una base común de trabajo para todas las aplicaciones y verticales, establece una serie de recomendaciones sobre reglas y procedimientos para la utilización de los mismos. El objetivo es que a priori los desarrolladores y sistemas para explotar la información conozcan las estructuras de datos y tipos que maneja la Plataforma y puedan hacer un uso homogéneo de los mismos independiente de la implementación concreta y particular de los desarrollos de integración. Se recomienda encarecidamente la lectura previa de la documentación de referencia para entender todos los conceptos manejados en el presente documento. Plataforma obtiene, registra y gestiona toda la información de carácter sensorial (medidas, comandos, alertas, eventos, etc…) usando ontologías json para la representación de los datos. Dado que los mecanismos ofrecidos por Sofia2 para la gestión de ontologías son muy flexibles y adaptables, para garantizar que toda la información en la Plataforma tiene una estructura homogénea entre los diferentes aplicaciones y verticales, y que permite hacer un uso horizontal de toda la información generada, se han establecido las reglas de gobernanza fijadas en el presente documento. La responsabilidad y gestión del gobierno de ontologías, como se indica en el apartado 6 será responsabilidad de la Xunta de Galicia

4.5.2.1. Entidades Sofia2

La Plataforma Sofia2 gestiona dos tipos de entidades:

- **Assets**. Un asset es todo elemento (físico o virtual) capaz de generar o consumir información de carácter sensorial y gestionarla a través de la Plataforma Sofia2
- **Ontologías**. Los assets generan información y dicha información se modela por medio de ontologías (JSON) en la Plataforma.

4.5.2.1.1. Assets.

Dependiendo de su naturaleza y de cómo se conectan dichos assets a la Plataforma se establece una clasificación de los mismos en:

- Assets del Inventario de la Plataforma. Son Assets directamente gestionados por la Plataforma por lo cual se debe de conocer a priori toda la información sobre los mismos que permita conectarse a ellos directamente. Todos los assets de inventario tienen que estar dados de alta en el Inventario de Assets de la Plataforma. Se dividen a su vez en dos tipos:
  - Managed. Se trata de los Assets conectados a la Plataforma es decir, gestionados por los KPs. Su característica principal es que los KPs que los gestionan necesitan conocer toda la información de acceso físico a los mismos.
  - Unmanaged. Se trata de Assets conocidos en el inventario de la Plataforma, que generan y consumen información de la misma, pero para los cuales no es responsabilidad del módulo de interconexión su gestión final, es decir, son gestionados por otros sistemas de información que se integran en la Plataforma.
- Assets Virtuales. Los assets virtuales son a priori no conocidos, generan información y esta llega a la Plataforma a través de integraciones, normalmente contra otros servicios de información en la nube, como por ejemplo las redes sociales.

4.5.2.1.2. Ontologías

Los assets conectados generan y consumen información sensorial en forma de ontologías json. A la hora de crear una ontología se pueden usar los catálogos (plantillas) siguientes:

- **Feed** (medidas). Se trata de la información de medida emitida por los assets conectados. Pueden ser medidas instantáneas, agregaciones por períodos de tiempo, cálculos, etc… Un feed puede recoger varias medidas simultáneamente y todos los feeds deben de estar georreferenciados (puntos, líneas o áreas) pudiendo ser éstos móviles.

- **Command** (comandos u órdenes). Se trata de las órdenes enviadas a los assets con capacidades de actuación. Dichas órdenes son asíncronas (no existe respuesta inmediata) por lo que existen dos tipos de comandos:
  - Comandos request. Las instrucciones enviadas a un Asset.
  - Comandos response. Las respuestas o ACKs enviadas por los Assets a dichos comandos. Las respuestas a los comandos únicamente indican la confirmación de la recepción de dicha instrucción.
- **Alert** (alertas). Las alertas son mensajes de notificación de situaciones (alertas, incidencias, mensajes, notificaciones, etc…) y tienen la particularidad de que su estado así como su nivel de criticidad cambia a lo largo del tiempo desde un mensaje inicial que genera la alerta hasta el cierre de la misma. Las alertas no se conocen a priori, es decir, se generan en el momento en el que suceden.
- **Schedule** (eventos). Los eventos reflejan situaciones conocidas en el tiempo y en el espacio, es decir, ocurren durante un periodo de tiempo en un lugar concreto y, normalmente, se conocen a priori.
- **Audit** (log). Los mensajes de auditoría o log son internos a la Plataforma y se utilizan para hacer explícitas y dejar gestionadas situaciones concretas particulares de los Assets. Por ejemplo, el encendido de una farola, el mal funcionamiento de un sensor, etc...
- **KPI** (indicador). Los indicadores son un tipo especial de medida y su característica principal es que no son generados por un único Asset. Se trata de cálculos realizados sobre múltiples datos, series históricas e incluso capas de datos espaciales que se almacenan como medidas en Plataforma.

Reglas especiales a tener en cuenta:

- Cada vez que se conecta un nuevo conjunto de assets (del mismo tipo) a la Plataforma deben de darse de alta al menos las ontologías feed y audit y si corresponde command para el mismo.
- Inmediatamente a la recepción de un comando y su correcta ejecución el Asset (el KP que lo gestiona) debe de generar una nueva medida para reflejar en la Plataforma su nuevo estado.

4.5.2.2. Reglas de nomenclatura

A continuación se establecen las reglas básicas de nomenclatura:

- Se utilizarán nombres en inglés y se seguirá el estándar Java (aka “camel”).
- Para la definición de plantillas:
  - Nombres cortos, autoexplicativos. Primera letra en mayúsculas. Las primeras letras identifican el tipo de ontología:
    - Feed: Feed
    - Command: Cmd
    - Alert: Alrt
    - Schedule: Schdl
    - Audit: Adt
    - KPI: Kpi
- Para la definición de ontologías:
  - Primera letra en minúscula, empiezan siempre por el tipo de ontología. Ejemplo para una Espira: feedEspira, cmdEspira, adtEspira …
- Para la definición de atributos de las ontologías:
  - Primera letra en minúsculas, sin espacios, sin caracteres especiales.
- Para la definición de constantes utilizadas como valores posibles para los atributos de las ontologías: todas las letras en mayúsculas. Por ejemplo: MOBILE, FIXED, VIRTUAL

4.5.2.3. Tipado y Formatos

Para la definición de ontologías se utilizarán cadenas de texto UTF-8 siguiendo el esquema json establecido por la correspondiente plantilla (actualmente siguiendo JSON Schema 0.4).

A continuación se establecen las reglas de tipado y formato para los diferentes tipos soportados:

UUIDs: o Cadena de texto. Standard Universally Unique Identifier.

- Números enteros:
  - o Entero Largo de 64 bits
  - o Ejemplo: {'contador' : 10}
- Números flotantes:
  - o Notación simple. Decimal con punto. 64 bits
  - o Ejemplo: {'valor' : 10.5}
- Cadenas de texto:
  - o Cadena de texto. UTF-8. Caracteres especiales escapados
  - o Ejemplo: {'comment' : 'next station'}
- URLs y URIs:
  - o Cadena de texto. Codificadas siguiendo estándar RFC-1738
  - o Ejemplo: {'url' : 'http%3A%2F%2Fwww.coruna.es%2Fmedioambiente%2F'}
- Timestamps:
  - o Fecha. Cadena de texto siguiendo formato ISO-8601. RFC 3339
  - o Objeto conteniendo atributo “$date”
  - o Ejemplo: {"timestamp":{"$date":"2014-01-27T11:14:00Z"}}
- Fechas e intervalos de fechas:
  - o Cadena de texto siguiendo formato ISO-8601.RFC 3339
  - o Objeto con atributo "$date"
  - o Ejemplo de fecha: {"created":{"$date":"2014-01-27T11:14:00Z"}}
  - o Ejemplo de intervalo entre dos fechas: {“period”:{“$date” : “2010-07-02T11:44:09Z/2010- 07-02T11:47:00Z”}}
- Direcciones:
  - o Notificación simplificada para facilitar las tareas de integración:

```
{"address": {
            "location": "cadena de texto",
             "number": "cadena de texto"
            }
}
```

- Unidades de medidas
  - o Cadena de texto string siguendo notación JScience library (http://jscience.org/api/javax/measure/unit/SI.html) (http://jscience.org/api/javax/measure/unit/NonSI.html)
  - o Ejemplo {'unit': 'A'} # Amperios
- Coordenadas geográficas:
  - o Siguen la definición OGC GeoJson. Esquema de coordenadas WGS84. No hay coordenada Z. Orden [longitud, latitud]
  - o Puntos:
    - GeoJson Point
    - Ejemplo:

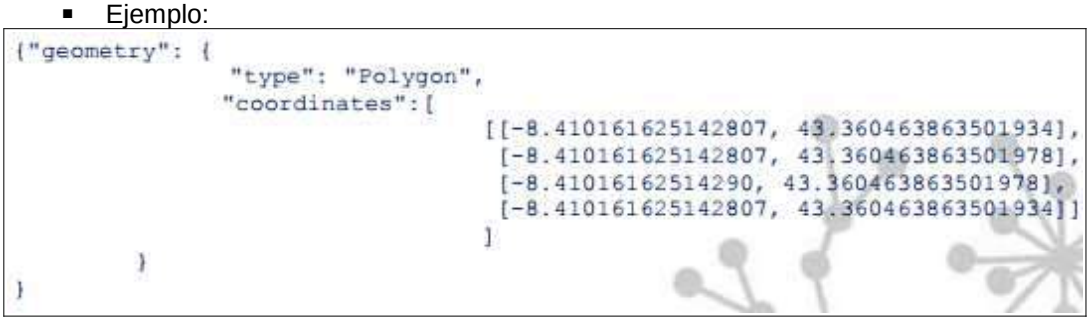
```
{"geometry": {
              "type": "Point",
             "coordinates": [-8.410161625142807, 43.360463863501934]
        }
}
```

  - o Líneas:
    - GeoJson LineString
    - Ejemplo:

```
{"geometry": {
              "type": "LineString",
             "coordinates":[
                            [-8.410161625142807, 43.360463863501934],
                            [-8.410161625142807, 43.360463863501978]
                            ]
        }
}
```

  - o Áreas:
    - GeoJson Polygon

- Ejemplo:



```
{"geometry": {
                "type": "Polygon",
               "coordinates":[
                                [[-8.410161625142807, 43.3604638635019 34],
                                 [-8.410161625142807, 43.3604638635019 78],
                                 [-8.41016162514290, 43.3604638635019 78],
                                 [-8.410161625142807, 43.3604638635019 34]]
                                ]
        }
}
```

[NOTA]: Para cerrar el polígono el primer y el último valor de cada anillo deben de ser idénticos.
[NOTA]: Un polígono puede tener 2 anillos (el exterior y el interior).

5. DESARROLLO SOBRE SOFIA2

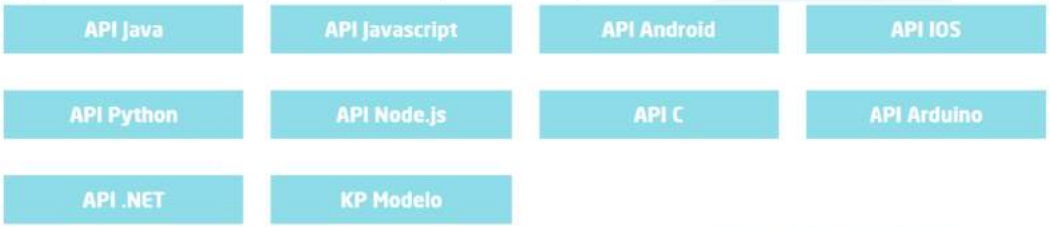
5.1. SDKs de Sofia2

Para aquellos casos en los que no existan productos y haya que desarrollarlos, se podrían hacer directamente integrados con la plataforma Sofía2, para ello existen diferentes entornos de desarrollo para las plataformas más comunes que pueden descargarse.

5.1.1. Elementos que componen el SDK de Sofia2

Este SDK se compone de varios elementos como son los siguientes:

- **IDE**: Entorno de desarrollo basado en Eclipse y que debe configurarse como para cualquier otro proyecto basado en Maven.
- **Consola de arquitectura**: Ofrece herramientas de productividad para crear proyectos Sofia2 de forma sencilla. Ejecutando en la Consola el comando S:\>arqspring Se abrirá la consola de Arquitectura iTR .
- **API**: por último y según los tipos de dispositivos (things) con los que trabajemos, necesitaremos descargar un API compatible con ellos. Sofía dispone de diferentes APIs para los lenguajes de programación más comunes en la mayoría de dispositivos:



5.2. Pasos en el desarrollo de aplicaciones Sofia2

5.2.1. Registro en la plataforma

Cada uno de los proyectos aprobados deberá registrarse en la plataforma como un SmartSpace, cada uno de los cuales podrán interactuar entre ellos.

5.2.2. Alta de usuario en la plataforma

La primera vez será necesaria una cuenta de usuario exclusiva para este proyecto. Tras la creación el usuario tendrá privilegios de ROL_USUARIO, y necesitará solicitar los privilegios de ROL_COLABORADOR.

5.2.3. Ontologización de la información

Sofia2 soporta múltiples métodos de creación de ontologías (interfaz gráfico de definición de JSON Schemas, generación automática desde Excel, Wizard de creación de ontologías, etc). En este caso se mostrará la ***creación guiada de la ontología,*** aunque como se ha comentado, este paso puede ser que lo realice un equipo experto y centralizado para asegurar la optimización de ontologías creadas en la

plataforma de Hogar Digital, para evitar redundancias y permitir la fácil interoperabilidad de sistemas o servicios que utilicen las mismas ontologías.

1. **Identificación de los conceptos de información**: Consiste en identificar los conceptos que contiene la ontología (medidas, persona, etc.)
2. **Identificación de los atributos de los conceptos**: Consiste en identificar los datos agrupados por los conceptos de información y que serán relevantes para los KPs
3. **Modelado en formato JSON-Schema**: Identificados los datos a intercambiar, el siguiente paso es estandarizarlos para que tengan una definición unívoca para los KPs en la plataforma. En esto consiste la ontologización de la información, donde cada concepto relevante se define de acuerdo a un schema JSON
4. **Alta o solicitud de alta de la ontología en la plataforma**: Una ontología deberá ser registrada en la plataforma para quedar operativa y poder ser utilizada por los KPs para insertar/consumir la información descrita por la misma.

5.2.4. Desarrollo clientes Sofia2 (KP)

Para el desarrollo de aplicaciones cliente (dispositivos o servicios digitales de terceros) que interactúan con la plataforma, en la mayoría de casos será necesario registrar los siguientes pasos:

1. **Alta de permisos para usuario**: Para que los KPs de un usuario puedan producir o consumir datos de una determinada ontología, el usuario deberá disponer de los permisos adecuados sobre dicha ontología.
2. **Alta de KP en la plataforma**: Un usuario deberá registrar en la plataforma sus KPs, de lo contrario, la plataforma rechazará la conexión de los mismos. Para registrar un KP, la plataforma proporciona la sección Gestión KPs, donde un usuario podrá crear un nuevo KP o administrar los que ya tiene dados de alta.
3. **Conexión del KP con la plataforma**: La conexión de un KP con la plataforma debe ser vista como dos tipos de conexión
   - **Conexión Física**: Establecida por el protocolo de trasporte utilizado para la conexión de dispositivos físicos a través de un KP. La manera de realizar esta tipo de conexión depende en gran medida del API de KP utilizado (Java, Javascript, Arduino, C++...). En Sofía2 se implementa esta conexión con protocolos de comunicación "Out the Box" entre un KP y el SIB:
     - **MQTT** (Message Queue Telemetry Transport) es un protocolo de conectividad enfocado a M2M (machine-to-machine) y al IoT (Internet of Things).
     - **RESTful**. Servicios web de tipo **Representational State Transfer ,** más livianos y ágiles que el modelo SOAP.
     - **Ajax Push** (JavaScript). Comunicación asíncrona, en la que el origen de la comunicación surge en el servidor.
     - **WebSocket**. Protocolo HTTP bidireccional.
     - **FIWARE NGSI-10.** Iniciativa open source para un estándar de interoperabilidad en aplicaciones Smart.
   - **Conexión Lógica**: Establecida por el protocolo **SSAP** (Smart Space Access Protocol) de mensajería definido en SOFIA. Es común a todos los APIs de KP. Aparte del protocolo de mensajería **SSAP** usado internamente por la plataforma.

     El protocolo SSAP proporciona dos operaciones en este sentido:

     - **JOIN**: Donde un KP informa a la plataforma el usuario y password de su propietario así sus datos de instancia, de manera que si todo es correcto, la plataforma autentica al KP y abre una sesión con el mismo.
     - **LEAVE**: Donde un KP informa a la plataforma que va a cerrar la sesión.

     Mientras exista una sesión entre el KP y la plataforma, el KP podrá utilizar el resto de operaciones del protocolo SSAP para producir/consumir información.

4. **Captación/Explotación de la información.**

   Constituye parte de la lógica de negocio de un KP y es independiente de la plataforma. Depende exclusivamente de la naturaleza y propósito del KP el modo de captar la información de las distintas fuentes si es productor de información, así como su explotación una vez recibida la información si se trata de un KP consumidor.

5. **Transformación de la información a formato Ontológico**

   Como ya se ha comentado en el presente documento la información que envíe un KP productor a la plataforma debe cumplir con el formato definido en la ontología que la representa. De manera que con tal información se deberá construir mensaje JSON que agrupe tales datos cumpliendo el JSONSchema de la ontología correspondiente, convirtiéndose de este modo los datos en una instancia de la ontología

6. **Envío a la plataforma según protocolo SSAP**

   Una vez construido el mensaje JSON con los datos a enviar a la plataforma. Se deberá construir el mensaje SSAP de INSERT correspondiente y que integrará tales datos. La plataforma validará que el usuario propietario del KP tiene el correspondiente permiso sobre la ontología que representan tales datos, así como que los datos cumplen con el Schema JSON de la ontología. Si hay algún problema, se notificará al KP, si todo va bien, tales datos se agregan a la base de datos de tiempo real del SIB, quedando disponible para el resto de KPs. Al igual que todas las operaciones SSAP, la operación INSERT está contemplada en todos los API de KP proporcionados.

7. **Consulta/Suscripción de la información según protocolo SSAP**

   La información enviada a la plataforma por los KPs, puede ser consultada por otros KPs, bien explícitamente mediante la operación QUERY del protocolo SSAP, bien en modo suscripción a futuras entradas de información mediante la operación SUBSCRIBE. En ambas operaciones se indican a la plataforma los criterios de la consulta. En la operación QUERY, nos serán devueltas las instancias existentes actualmente en la base de datos de tiempo real que cumplen con los criterios de la consulta. En la operación SUBSCRIBE, la plataforma nos enviará en el futuro nuevas instancias cada vez que un KP las inserte y cumplan con los criterios de la consulta. SOFIA permite que las operaciones de QUERY puedan ser:

   - **Query de tipo Nativo**: cuando la query es resuelta por el motor de BDTR subyacente, siendo en la implementación de referencia MongoDB.

     ```
     db.SensorTemperatura.find().limit(3);
     ```

   - **Query de tipo SQL-Like**, cuando la query es transformada por SOFIA al lenguaje de query subyacente.

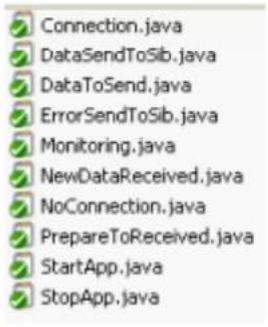
     ```
     select * from SensorTemperatura limit 3;
     ```

8. **Recepción de la información a formato Ontológico**

   Del mismo modo que un KP envía la información a la plataforma de acuerdo a una ontología, cuando un KP recibe información de la plataforma, esta información también viene en formato JSON según la ontología correspondiente, de modo que una vez extraída del mensaje SSAP correspondiente, el KP puede tratar dicha información según la definición de la ontología en el JSONSchema que la define.

5.2.5. Desarrollo de un KP para un Gateway.

1. Desde la consola de la plataforma lanzamos el comando Sofia2 crearAppModelo --id [NOMBRE APPMODELO] –paquetebase [PAQUETE APLICACION]
2. Implementamos el método readDataSensor de la clase PrepareToReceived, donde deberemos de establecer la conectividad con los sensores y convertir la información obtenida de estos a un objeto SensorMessage.
3. En la clase PrepareToReceived podemos leer de todos los sensores o crear nuevas clases, extendiendo de PrepareToReceivedWorkerImpl para gestionar cada uno de los sensores conectados.
4. Implementaremos el método generateSSAPMessage de la clase NewDataReceived, donde debemos convertir los datos que hemos recogido de los sensores en un mensaje SSAP.
5. A través de la implementación de la clase DataSendToSib obtenemos el mensaje de respuesta del SIB junto con el mensaje original que enviamos al SIB.
6. Si queremos suscribirnos al SIB debemos implementar la clase SubscriptionListener en el método init, podremos a través del método subscribe(ontology, query, SSAPQueryType), suscribirnos al SIB, y a través del método onEvent, definir las operaciones a realizar cada vez que nos llega una notificación del SIB asociada a la suscripción realizada.
7. Adicionalmente podemos implementar el resto de clases que nos permitirán controlar aspectos como el:



   a) Arranque y Detención del servidor de la Aplicación KP. StartApp y StopApp (Este último deberá implementar el comportamiento que deseamos ante una detención solicitada por el contenedor del KP).
   b) Conexión con el SIB y perdida de esta conectividad.
   c) Preprocesamiento y postprocesamiento de los mensajes enviados al SIB.
   d) Error al enviar mensajes al SIB

   Los logramos implementando cada uno de los Workers definidos en el diagrama anterior.
8. Es posible que los desarrolladores necesiten en ocasiones Registrar nuevos workers: Estos deben ser registrados a través del bean KPWorkerCfg.
   a) Obtener una referencia a este objeto:

```
@Autowired

protected KPWorkerCfg kPWorkerCfg;
```

   b) Invocar al método siguiente:

```
public void addEvent(String event, Class message, Subscriber  worker):
  kPWorkerCfg.addEvent("MI_EVENTO", LifeCicleMessage.class, new StartAppWorker())
```

9. Para publicar un evento existente o creado por lo desarrolladores debemos, hacerlo a través del método.

   `publish (String topic, Object message)` que encontramos en el bean `KPWorkerCfg`.

   Los eventos disponibles de facto por la plataforma son los que encontramos en las clases enumeradas:
   a) SibEvents
   b) SensorEvents
   c) LifeCicleEvents
   d) InfraestructureEvents

5.3. Ejemplo práctico

Este tutorial pretende servir como una referencia paso a paso para la evaluación de la gestión de dispositivos IoT por la plataforma Sofia2 Smart IoT Platform.

En concreto esta guía construirá un entorno de demostración con la estructura definida en la Figura 1.

Figura 5 .- Esquema de conectividad de los dispositivos involucrados en la demostración

En este caso en concreto, se utiliza el dispositivo SensorTag de Texas Instruments como dispositivo de adquisición de datos. Este dispositivo dispone de múltiples sensores y una interfaz de comunicación a través de Bluetooth Low Energy. Usando un Smartphone, se establecerá una conexión Bluetooth con el SensorTag, y comenzará un mapeo de los valores medidos por los sensores. El Smartphone será el encargado de encapsular la información y transmitirla a Sofia2 usando redes de telefonía móvil o una red WiFi.

Gracias a las capacidades de Sofia2, la información se puede almacenar, tratar y representar de manera sencilla. En este tutorial iremos explicando paso a paso cómo crear un proyecto en la plataforma FEEP IoT Enablement Platform Sofia2, en el que configuraremos un Dashboard mostrando diversos valores recibidos desde los dispositivos, además de un Sinóptico para mostrar también esta capacidad. Además usaremos el motor de reglas, utilizando Groovy, que evaluará si el valor de una de las magnitudes críticas medidas ha excedido un valor máximo, lanzando en consecuencia un SMS de alerta.

Todas estas capacidades quedarán englobadas en un proyecto sobre Sofia2. A modo de vista general del proyecto, cuando todos los elementos estén conformados se podrá visualizar como en la Figura 2, donde podremos distinguir los distintos componentes que conforman la solución,

5.3.1. Alta de usuario en la plataforma

El primer paso será generar una cuenta de usuario exclusiva para este proyecto. Tras la creación el usuario tendrá privilegios de ROL_USUARIO, y necesitará solicitar los privilegios de ROL_COLABORADOR, para poder acceder a todas las capacidades que se pretenden desarrollar en este escenario.

5.3.2. Definición de ontologías a utilizar

Para el escenario de esta demostración se creará una sola ontología con el objetivo de recoger las distintas magnitudes obtenidas tanto del dispositivo SensorTag como del Smartphone.

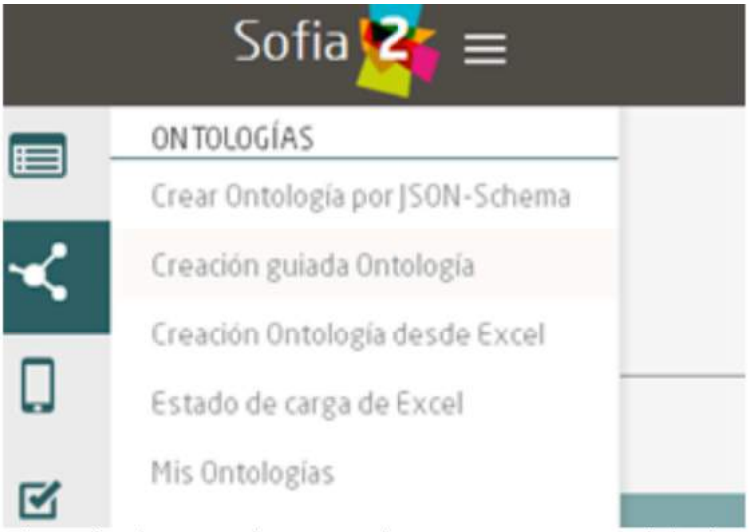


Figura 6 .- Acceso a los comandos para operar con ontologías

En la creación guiada de Ontología se pueden configurar múltiples parámetros (Figura 7).

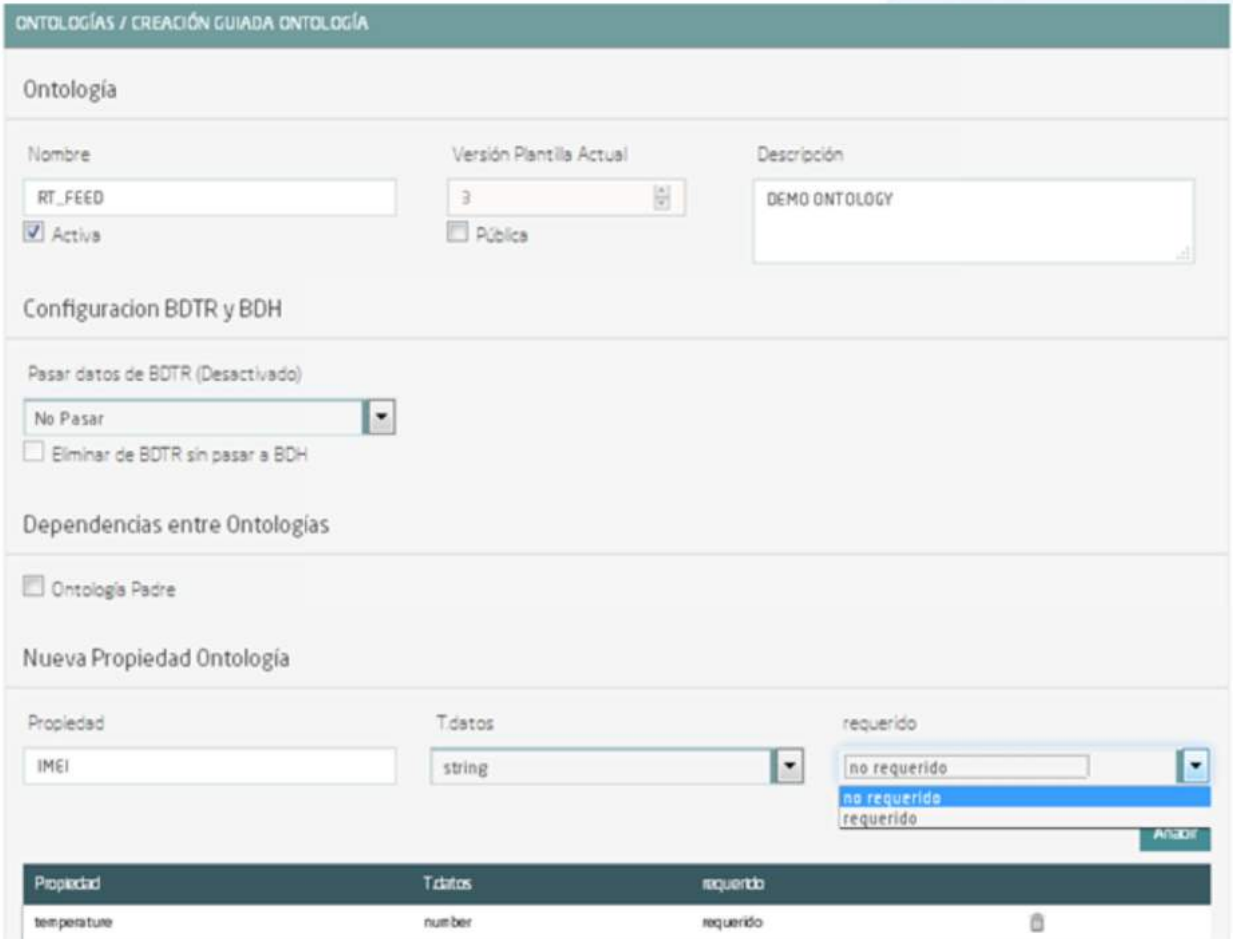


Figura 7 .- Menú de creación guiada de ontologías

De inicio hay que definir un nombre que identificará a la ontología de aquí en adelante, y existe un campo de descripción para anotar las particularidades y usos de la misma. Justo debajo del campo de nombre existe un campo para activar la ontología.

A continuación aparece la configuración de las bases de datos, en cuanto al trasvase de información desde la base de datos en tiempo real (BDTR), a la base de datos histórica (BDH). Para este escenario de demo, se mantendrán los datos en la BDTR.

El apartado de dependencia entre ontologías no aplica a este escenario. A continuación aparece el apartado de añadir nueva propiedad a una ontología y que será el que se use en esta demo para añadir los datos que se desean manejar. Para este caso, se crearán los siguientes campos:

ampo	Descripción	Fuente	Tipo
deviceID	Código IMEI del dispositivo gateway	Smartphone	String
date	Fecha de generación de la trama	Smartphone	Date
accelX	Aceleración en G's sobre el eje X	SensorTag	Number
accelY	Aceleración en G's sobre el eje X	SensorTag	Number
accelZ	Aceleración en G's sobre el eje X	SensorTag	Number
gyroX	Velocidad de giro en rad/s en el eje X	SensorTag	Number
gyroY	Velocidad de giro en rad/s en el eje X	SensorTag	Number
gyroZ	Velocidad de giro en rad/s en el eje X	SensorTag	Number
temperature	Temperatura ambiente en ºC	SensorTag	Number
humidity	Valor de porcentaje de humedad relativa	SensorTag	Number
geometry	Coordenadas de geoposición	Smartphone	Geometry

El resultado final es la ontología completamente definida y lista para recibir información. En el escenario de demo, hemos denominado a la ontología como **demoDispositivos_RTFrame**. En la Figura 7 se muestra un ejemplo de instancia de esta ontología.


```
{
    "demoDispositivos_RTFrame": {
        "deviceID": "string",
        "date": {
            "$date": "2014-01-30T17:14:00Z"
        },
        "accelX": 28.6,
        "accelY": 28.6,
        "accelZ": 28.6,
        "giroX": 28.6,
        "giroY": 28.6,
        "giroZ": 28.6,
        "temperature": 28.6,
        "humidity": 28.6,
        "geometry": {
            "type": "Point",
            "coordinates": [9, 19.3]
        }
    }
}
```

Con esta definición y manteniendo la ontología activa, en el lado de la plataforma Sofia2 sólo faltaría definir el ThinKP que se usará para interactuar con los datos, y tras este paso, ya se podrá enviar y/obtener datos de la plataforma.

5.3.3. Conectando el dispositivo

5.3.3.1. Creación del ThinKP asociado

En este apartado se creará un ThinKP para este usuario de demo. Para ello hay que pulsar sobre el tercer icono del menú de comandos de la izquierda de la pantalla, y seleccionar Mis ThinKPs, tal y como se muestra en la siguiente imagen.

Figura 8.- Acceso al menú de ThinKPs

En la parte derecha de la siguiente pantalla aparecerá el botón de creación de un nuevo ThinKP y tras pulsar el botón se desplegará el cuadro de creación del nuevo ThinKP (siguiente imagen). La creación es muy sencilla y tan solo requiere la introducción de un identificador y una breve descripción.

Figura 9.- Cuadro de creación de un nuevo ThinKP

Además será necesario asociar al menos una ontología asociada al ThinKP. En este caso tan solo se accederá a la ontología que creamos en los apartados anteriores, demoDispositivos_RTFrame, por lo que habría que seleccionarla y pulsar el botón de creación para que nos quede como se ve en la siguiente imagen:

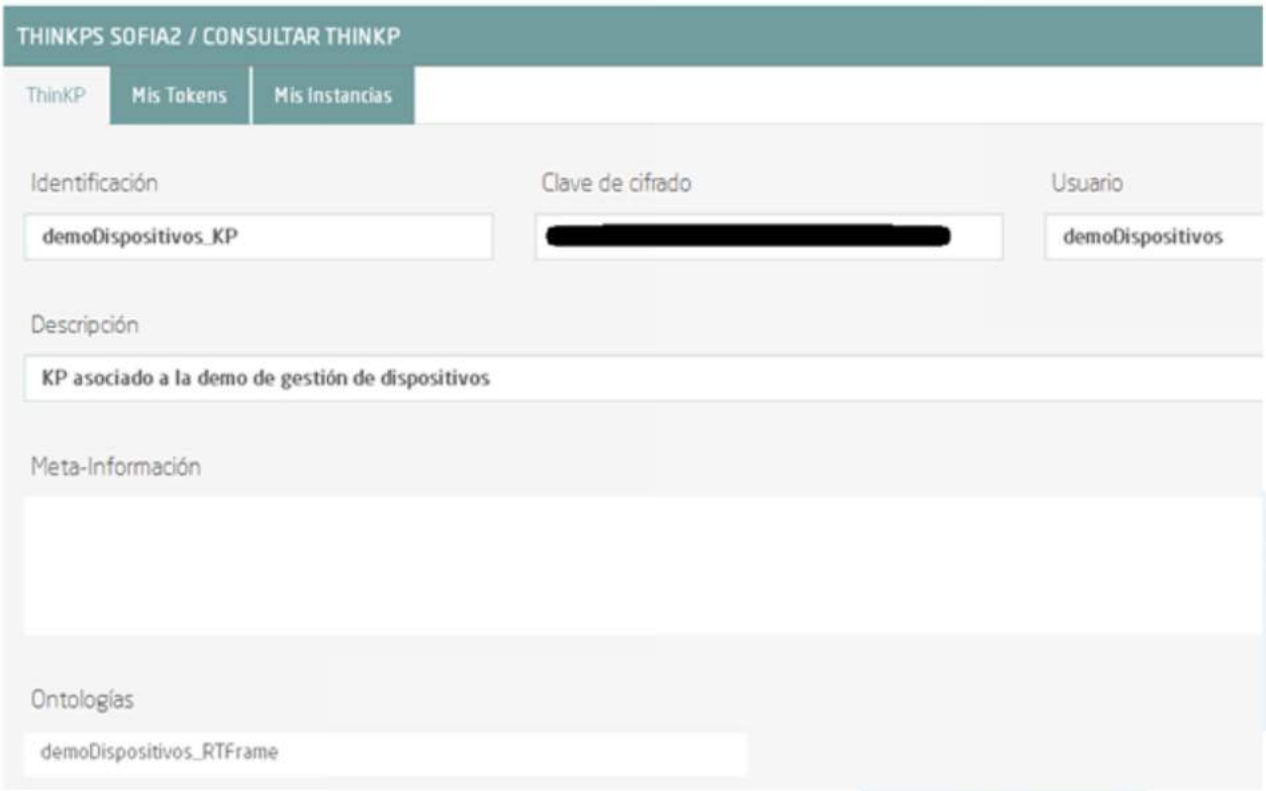


Figura 10.- Visor de ThinKPs

5.3.3.2. Uso de la instancia del ThinKP en dispositivos IoT

Tras finalizar la definición del ThinKP, quedaría listo para que distintas instancias del mismo pudieran interactuar con la plataforma. En esta demostración, se utilizará una instancia de ThinKP en el dispositivo que posee el rol de Gateway entre la placa de sensores y la plataforma Sofia2, el smartphone. Utilizando una instancia de ThinKP, se habilitará la inserción, lectura y en definitiva uso de las capacidades de Sofia2 desde el dispositivo Android.

En la siguiente imagen se muestra como simplemente bastaría con introducir los valores de los parámetros asociados al ThinKP recién creado en Sofia2.

```
private final static String TOKEN = "609[redacted]a95684";
private final static String KP_INSTANCE = "demoDispositivos_KP:demoDispositivos_KP01";
private final static String ONTOLOGY_NAME = "demoDispositivos_RTFrame";
```

Figura 11 .- Definición de parámetros de ThinKP en Android

En esta demostración se enviarán los datos de sensores hacia la plataforma, utilizando el protocolo REST que otorga una gran simplicidad a la inserción de datos utilizando operaciones POST. En la siguiente imagen, se muestra un extracto del método de envío de tramas a Sofia2, en donde se produce el mensaje de JOIN para abrir una sesión en Sofia2, realizando un POST que utiliza los parámetros de la instancia de ThinKP asociada.

```
public synchronized  boolean send(Context context, String TOKEN, String KP_INSTANCE, String ONTOLOGY_NAME, LinkedList<String> frames ) {
    URL url = null;
    try {
        url = new URL(SERVICE_URL); //"http://sofia2.com/sib/services/api_ssap/v01/SSAPResource/"

        HttpURLConnection connection = (HttpURLConnection)url.openConnection();
        connection.setDoInput(true);
        connection.setDoOutput(true);
        connection.setRequestMethod("POST");
        connection.setConnectTimeout(10000);
        connection.setReadTimeout(10000);
        connection.setAllowUserInteraction(false);
        connection.setUseCaches(false);
        connection.setRequestProperty("Content-Type","application/json");

        connection.connect();

        DataOutputStream dStream = new DataOutputStream(connection.getOutputStream());
        dStream.writeBytes("{\"join\": true,\"instanceKP\": \"" + KP_INSTANCE + "\",\"token\": \"" + TOKEN + "\"}");

        dStream.flush();
        dStream.close();
        int responseCode = connection.getResponseCode();

        Log.d(TAG,"Sending 'POST' JOIN request to URL : " + url);

        if(responseCode!=200){
            Log.i(TAG,"Error Join " + responseCode);
            return false;
        }

        Log.d(TAG,"Join OK");
```

Figura 12 .- Ejemplo de método de JOIN usando REST y ThinKP en Android

Con esto se obtendría un conector con Sofia2, a través del cual se pueden introducir datos en la ontología asociada siendo en este caso demoDispositivos_RTFrame (ejemplo de instancia en la Figura 7).

En cuanto a la toma de datos, en esta demostración se conecta el smartphone con el dispositivo SensorTag a través de BLE (Bluetooth Low Energy). Las características de los servicios disponibles para esta placa en concreto se pueden encontrar en la web asociada de Texas Instruments (http://processors.wiki.ti.com/index.php/CC2650_SensorTag_User's_Guide).

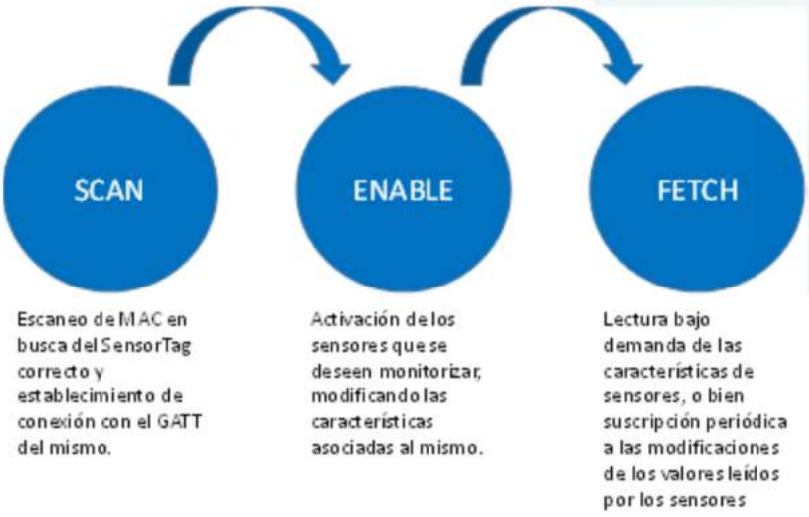


Figura 13 .- Fases de captura de datos

La captura de datos del dispositivo SensorTag se puede estructurar en 3 bloques principales, reflejados en la imagen anterior.

En la fase de **SCAN**, basta con utilizar el API de BLE de Android. En este ejemplo en concreto se ha desarrollado la aplicación para que sea soportada desde la versión KitKat de Android hasta las actuales. Para el escaneo se utiliza la llamada del sistema onLeScan, que se ejecuta cada vez que una nueva MAC de un dispositivo BLE ha sido detectada por el smartphone. En esta aplicación en concreto, simplemente se filtra la dirección del SensorTag y se lanza un Runnable para conectar con el dispositivo.

```
private BluetoothAdapter.LeScanCallback mLeScanCallback =
        new BluetoothAdapter.LeScanCallback() {

            @Override
            public void onLeScan(final BluetoothDevice device, int rssi, byte[] scanRecord) {
                    if(device.getAddress().equals(pref_sensor_id)&& !FOUND_FLAG){
                        Log.i(TAG,"Found!");
                        FOUND_FLAG = true;
                        mDeviceAddress = device.getAddress();
                        bleConnHandle.post(connect);
                    }
            }
        };
```

Figura 14 .- Escaneo BLE de direcciones MAC

Para iniciar/pausar el escáner basta con llamar a las funciones startLeScan/stopLeScan, mostradas en la figura, pasándoles la referencia del callback de escaneo definido anteriormente.

```
private Runnable runnableCode = new Runnable() {
    @Override
    public void run() {
        if(SCAN_FLAG){
            FOUND_FLAG = false;
            loadPreferences();
            mBluetoothAdapter.startLeScan(mLeScanCallback);
            bleScanHandle.postDelayed(runnableCode, 20000);
            SCAN_FLAG = false;
            //bleConnHandle.post(connect);

        }
        else{
            mBluetoothAdapter.stopLeScan(mLeScanCallback);
            bleScanHandle.postDelayed(runnableCode, 40000);
            SCAN_FLAG = true;
        }
    }
};
```

Figura 15 .- Inicio y parada de escaneo

Una vez se establece la conexión con el equipo, se pasa a la fase de **ENABLE**, donde hay que activar los sensores que se deseen monitorizar, siguiendo las directrices de la wiki de SensorTag.

El servidor GATT del SensorTag presenta un servicio para cada sensor de los que monta, y que a su vez constan de 3 características principales:

- **Configuración**: Sirve para encender/apagar el sensor
- **Datos**: Característica donde se almacena el valor capturado por el sensor
- **Periodo**: Característica que almacena el valor de la resolución de lectura del sensor.

Si se desea recibir notificaciones cuando varíen los datos de la característica de datos, habrá que activarlas siguiendo las indicaciones, y la aplicación recibirá un callback con el nuevo valor.

En esta demostración se utilizan los sensores de temperatura a través de IR (con capacidad de leer temperatura ambiente, y temperatura de un objeto a corta distancia) y el de movimiento (con capacidad de leer datos de acelerómetro, giróscopo y magnetómetro). En la siguiente figura se presenta un extracto de la información necesaria para interactuar con el sensor. En la fase de **ENABLE**, habría que escribir '0x01' en la característica de configuración del equipo, mientras que en la fase **FETCH**, se puede o bien leer directamente la característica de datos, o activar las notificaciones periódicas (usado en el proyecto).

IR Temperature Sensor

Type	UUID	Access	Size (bytes)	Description
Data	AA01*	R/N	4	Object[0:7], Object[8:15], Ambience[0:7], Ambience[8:15]
Notification	2902	R/W	2	Write 0x0001 to enable notifications, 0x0000 to disable.
Configuration	AA02*	R/W	1	Write 0x01 to enable data collection, 0x00 to disable.
Period	AA03*	R/W	1	Resolution 10 ms. Range 300 ms (0x1E) to 2.55 sec (0xFF). Default 1 second (0x64)

Figura 16 .- Tabla de valores para interactuar con el servicio de temperatura IR en SensorTag

Con los datos de sensores obtenidos, bastará con encapsularlos en base a la ontología creada, por ejemplo conformando un String como el de la Figura 18. En este ejemplo en concreto, se reporta también el código IMEI del dispositivo móvil a modo de indicador, y se añade la localización por GPS del smartphone para geo-localizar las medidas.

```
private String composeFrame(Location location){
    if(location.getAccuracy()>maxAccuracy){
        return "";
    }
    else{
        return "{ \"demoDispositivos_RTFrame\": { \"geometry\": { \"coordinates\": [ "+
                location.getLongitude()+", "+location.getLatitude()+" ], \"type\": \"Point\" }, \"" +
                "deviceID\": \""+IMEI+"\", \"" +
                "temperature\": "+temperatureValue+", \"" +
                "humidity\": "+humidityValue+", \"" +
                "accelX\": "+accXValue+", \"" +
                "accelY\": "+accYValue+", \"" +
                "accelZ\": "+accZValue+", \"" +
                "giroX\": "+gyrXValue+", \"" +
                "giroY\": "+gyrYValue+", \"" +
                "giroZ\": "+gyrZValue+", \"" +
                "date\": { \"$date\": \""+sdf.format(new Date())+"\"}}}";
    }
}
```

Figura 17 .- Ejemplo de construcción de trama en Android

5.3.4. Visualización de datos

Una vez realizados el diseño y la configuración de la ontología, en conjunto con la integración de los dispositivos IoT con Sofia2, dispondremos en la plataforma de todos estos datos, que se podrán utilizar de diversas maneras. Por ejemplo, representándola en tiempo real en un dashboard o un sinóptico, o procesándola mediante el motor de reglas.

El uso de estas dos capacidades de Sofia2 será lo que describamos en este apartado.

5.3.4.1. Composición de Dashboards

Sofia2 tiene la capacidad de configurar gadgets y dashboards sobre la información disponible. Para ello accederemos al menú de Visualización, submenú de Gadgets tal y como aparece en la siguiente imagen.

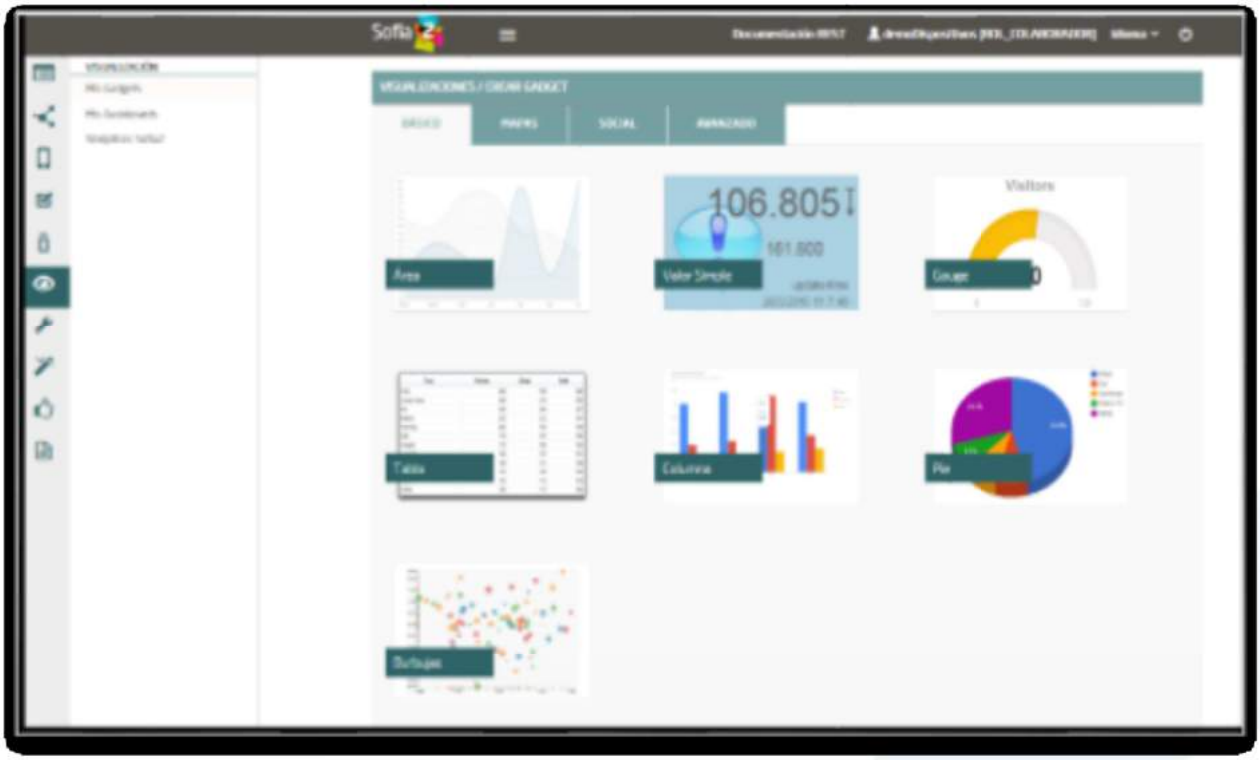


Figura 18 .- Menú de creación de Gadgets en Sofia2

Para nuestro ejemplo, crearemos un par de gadgets de valor simple, para visualizar los datos de temperatura del sensor, y un par de gadgets de columna, para visualizar los ejes x, y, z del giroscopio y el acelerómetro de los sensores de nuestro SensorTag.

Para cualquiera de los dos casos, lo primero que tenemos que hacer es dar un nombre al gadget y seleccionar nuestro ThinKP, que nos dará visibilidad a la conexión con la ontología que hayamos configurado.

Figura 19 .- Selección de ThinKP a representar

Una vez seleccionado el ThinKP (Figura 20), tendremos dos opciones para obtener los datos:

- Obtener los datos en directo: Esto es, el gadget se mantendrá suscrito a la ontología, actualizando el valor representado en el mismo momento en que un nuevo valor de ésta entra en el repositorio (Figura 21).
- Obtener datos por query: Definiremos un intervalo de tiempo para el refresco del gadget, transcurrido el cual se lanzará la consulta que definamos contra la base de datos en tiempo real o bien contra la base de datos histórica.

En el caso de los valores simples, elegiremos la segunda opción, lanzando cada 20 segundos la siguiente query a la base BDTR (que nos devuelve el último registro insertado en la ontología):

```
db.demoDispositivos_RTFrame.find().sort({'demoDispositivos_RTFrame.date':1}).sort({'contextData.timestamp':-1})
```

Figura 20 .- Obtención de datos en directo para representar en Gadget

Con los datos seleccionados en nuestro gadget, solo necesitaremos seleccionar cuál de los campos de la instancia de ontología recuperada queremos representar, asignarle un nombre en la gráfica y opcionalmente una transformación del dato recuperado de la ontología, tal y como se muestra en la siguiente imagen:

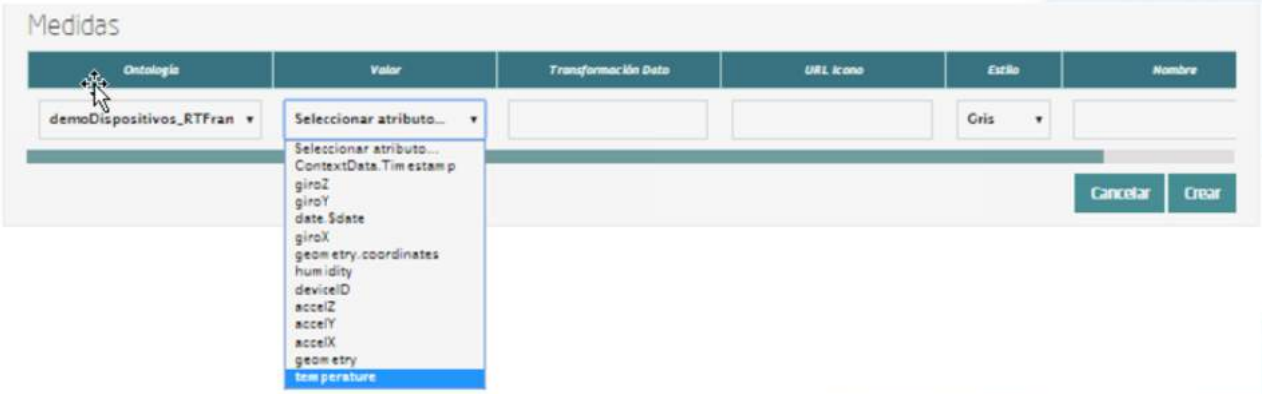


Figura 21 .- Selección/Definición de magnitudes a representar

Con todos estos pasos, queda seleccionar un token de seguridad de los disponibles en el ThinKP y guardar el gadget creado

Figura 22 .- Token de seguridad

Ya con el conjunto de gadgets creados, podremos componer nuestro dashboard de una manera sencilla, accediendo al menú de visualización, submenú dashboards.

Primero configuraremos el estilo general, icono, tipo de menú y crearemos una primera página, como se muestra en la siguiente imagen.

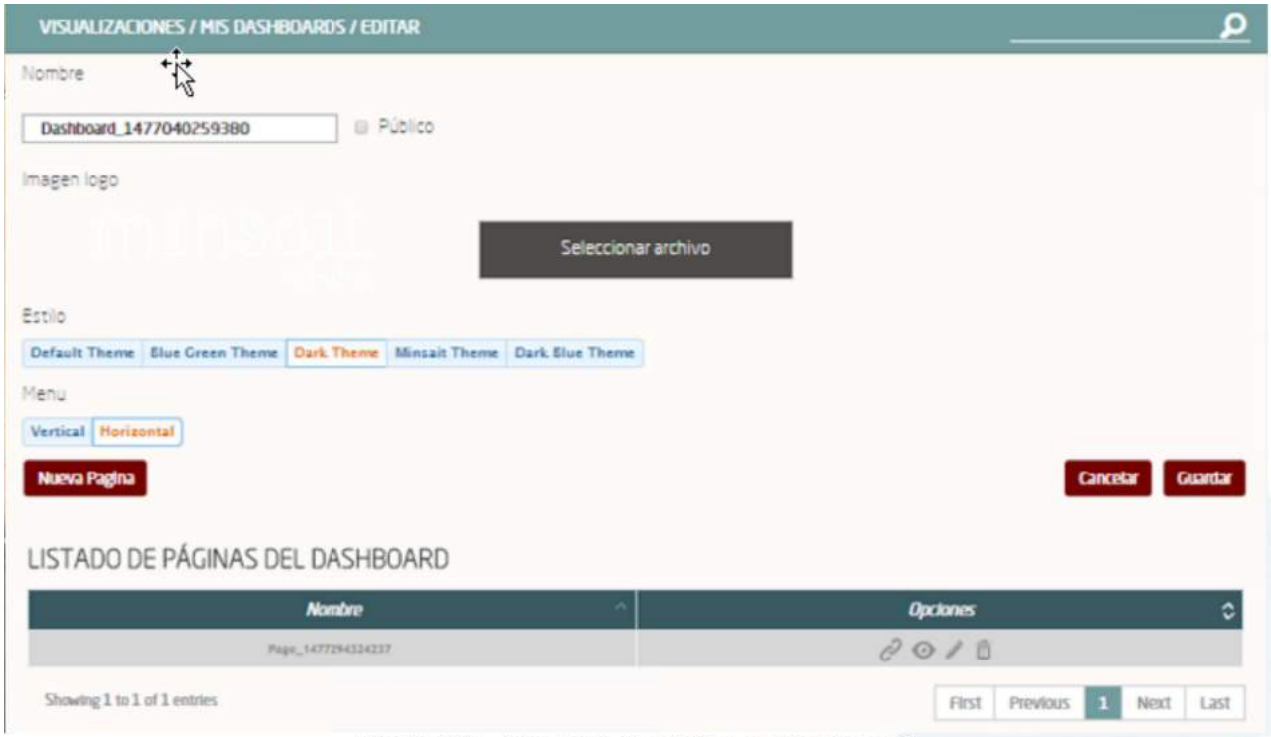


Figura 23 .- Creación de página de Dashboard

Accediendo a la nueva página recién creada del dashboard, podremos añadir los gadgets creados, y arrastrarlos al área donde queramos que se visualice, quedando algo similar al siguiente ejemplo:

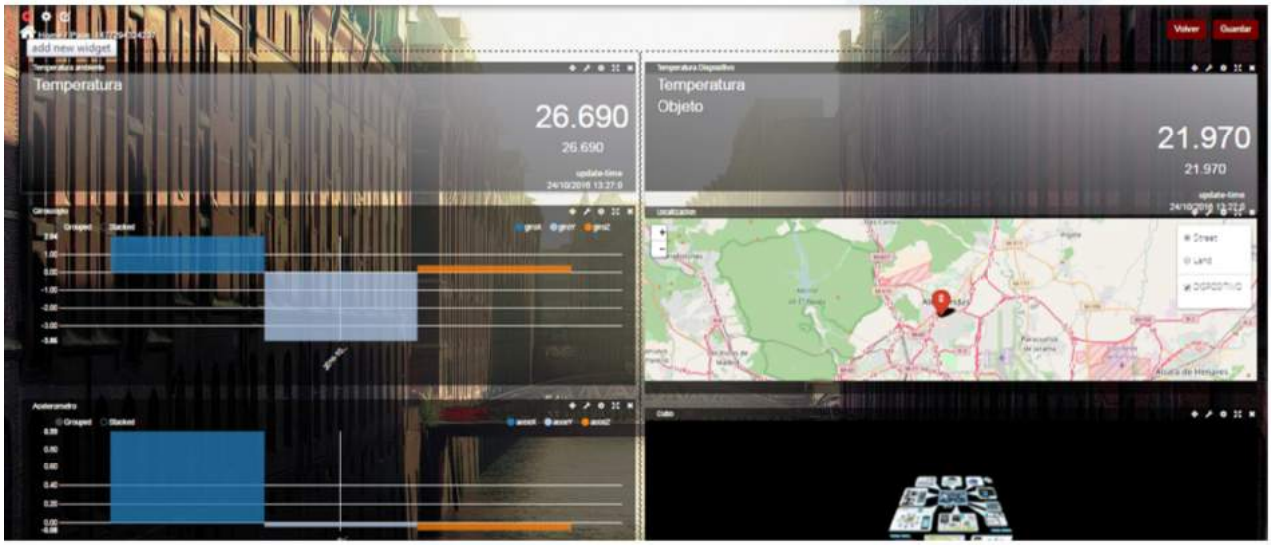


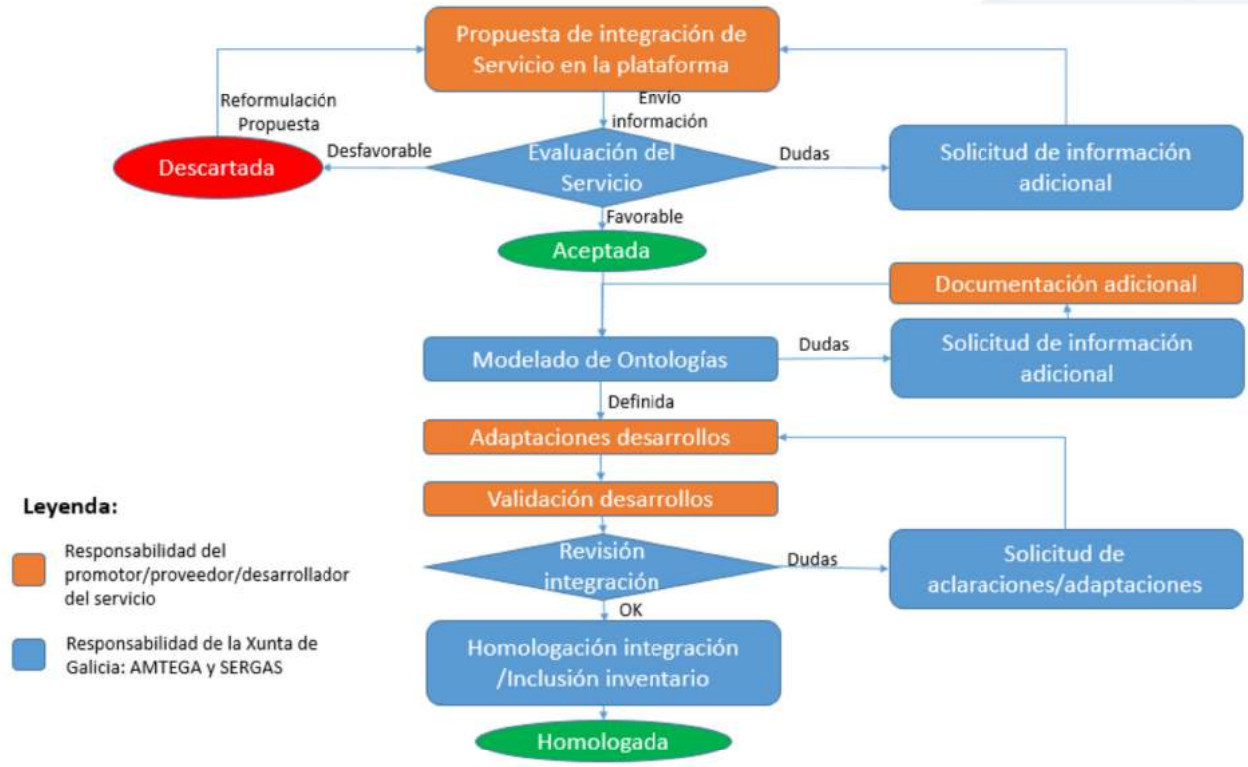
Figura 24 .- Aspecto final de Dashboard

6. PROCEDIMIENTO REGULATORIO DE INTEGRACIONES EN LA PLATAFORMA

Como se ha adelantado en la explicación de los diferentes pasos asociados a la integración de un servicio en la plataforma, desde el punto de vista técnico, de cara a mantener una coherencia y control sobre los servicios integrados en la plataforma de "Fogar Dixital SocioSanitario" de la Xunta de Galicia, se ha establecido un procedimiento de para la solicitud y homologación de integraciones sobre la plataforma.

Es importante aclarar que este procedimiento, está definido para regular el proceso de inclusión de servicios en la plataforma a futuro, y no es de aplicación en el proceso actual de consultas preliminares a mercado, pero se ha incluido en este documento aclaratorio sobre la integración de servicios en la plataforma para entender mejor el proceso que se seguirá en la solicitud e implantación de integraciones en la plataforma. Creemos que la información técnica descriptiva de las posibilidades de integración, conjuntamente con este procedimiento que describe la operativa que regirá esas integraciones, proporcionan una visión general de todos los aspectos relacionados con las futuras integraciones en la plataforma de "Fogar Dixital SocioSanitario" de la Xunta de Galicia.

El flujo general definido para este procedimiento es el siguiente:



Como se puede apreciar en la leyenda asociada al flujo presentado, las acciones recogidas en el mismo se identifican con dos posibles responsables de realización de las mismas:

- Bloques naranja: Se corresponden con acciones cuya responsabilidad de ejecución será del promotor, proveedor o desarrollador (dependiendo del caso) del servicio objeto de la integración.
- Bloques azules: Se corresponden con acciones que deben ser ejecutadas por diferentes entidades u organizaciones pertenecientes a la administración. A este objeto, se establecerá una "Oficina de interoperabilidad" que estará formada por personal tanto de la AMTEGA como del Servizo Galego de Saúde, que será la encargada de coordinar todas los aspectos relativos a la integración de servicios sobre la plataforma de "Fogar Dixilal SocioSanitario" de la Xunta de Galicia.
- Estados verdes: Se corresponde con los estados por las que pasa el flujo de la integración que son favorables.
- Estados rojos: Se corresponde con los estados por las que pasa el flujo de la integración que son desfavorables.

Indicamos a continuación una pequeña descripción de las acciones definidas en este flujo:

- Propuesta de integración de Servicio en la plataforma: El promotor/proveedor de un servicio que se desee integrar sobre la plataforma de “Fogar Dixital SocioSanitario” deberá remitir una propuesta detallada de la integración que propone, comprendiendo información relativa, entre otros, a los siguientes aspectos:
  - Identificación de los elementos tecnológicos en el fogar para la prestación de los servicios de teleasistencia avanzada
  - Identificación de la conectividad necesaria en el hogar para la prestación de dichos servizos.
  - Previsión del coste de la provisión, instalación, mantenimiento y operación de esto elementos tecnológicos en el hogar
  - Que modelos de financiamiento de estos costes son posibles

- Evaluación del servicio/integración propuesto: La administración evaluará la propuesta de servicios propuesta, de cara a estimar la idoneidad y viabilidad de la misma. En caso de que fuese necesario podría solicitar información adicional que ayude a la evaluación de la integración.

  Como resultado de la evaluación la propuesta podría ser aceptada, en cuyo caso se desencadenarían las acciones precisas para su puesta en marcha (esto podría llevar aparejada la puesta en marcha de un concurso publico para la presentación de ofertas, si las condiciones lo requieren).

  Si la propuesta se considera no apropiada, se hará una notificación al solicitante conforme la misma es descartada. Ante esta respuesta la solicitud podría ser reformulada para atender a los argumentos negativos de la valoración esgrimidos por la administración.

- Modelado de ontologías; Como se ha adelantado en partes anteriores de este documento, es muy importante la correcta administración semántica de la información a integrar en la plataforma, es por ello, que la oficina de interoperabilidad será la encargada de definir las ontologías que soportarán el servicio. Esta definición será realizada en coordinación con el solicitante de la integración para asegurar que se cubren las necesidades del servicio, pudiendo requerirse en este proceso información adicional.
- Implementación/adaptación de los desarrollos: Una vez que todos los detalles del servicio están identificados, el promotor/proveedor/desarrollador del servicio, deberá realizar los desarrollos, o adaptaciones que sean precisos en las herramientas que sustentarán el servicio para permitir la integración.
- Validación de los desarrollos: Una vez que las herramientas (aplicaciones, dispositivos,…) relacionados con el servicio cumplan con los requerimientos establecidos para la integración, deberán ser realizadas pruebas que validen dicha integración en los entornos de validación que la Xunta de Galicia definirá a este efecto.
- Revisión de la integración: La oficina de interoperabilidad será la responsable de coordinar a todas las áreas precisas para que se realice una revisión/validación completa de la integración en los entornos definidos. Si fuese preciso solicitará las correcciones/aclaraciones que sean precisas, hasta que de alcance una valoración favorable de la integración.
- Homologación de la integración: La validación de forma favorable de la integración por parte de la “Oficina de interoprabilidad” dará como resultado la homologación de las entidades objeto de la integración (Aplicaciones, dispositivos,…). Estas entidades homologadas pasarán a forma parte del inventario de la plataforma.

7. DOCUMENTOS DE REFERENCIA

A continuación se recogen algunos de los documentos de esta sección que aportan información relevante para algunos apartados recogidos en el presente documento:

- Primeros Pasos con Sofia2: Documento explicativo de cómo empezar a utilizar Sofia2, que ilustra cómo modelar los datos, cómo conectar dispositivos a la plataforma y la creación de aplicaciones.

  También se puede visitar la versión interactiva en este enlace.

- Gestión de dispositivos en Sofia2: Tutorial de referencia para la evaluación de la gestión de dispositivos IoT en Sofia2.
- Taller de Analytics: Documento explicativo de cómo utilizar las distintas herramientas analíticas de Sofia2.
- Guía Primeros Pasos Desarrollador Soluciones: Documento guía para el desarrollador que quiere construir nuevas soluciones sobre Sofia2.
- Taller IoT: Documento explicativo de cómo utilizar las distintas herramientas del flujo IoT de Sofia2.
- Conceptos Sofia2: Documento que describe los conceptos básicos de la plataforma Sofia2.
- APIs Sofia2: Documento que describe las diferentes APIs que proporciona la Plataforma SOFIA para el desarrollo de KPs.