GENERALITAT VALENCIANA

IVACE INSTITUTO VALENCIANO DE COMPETITIVIDAD EMPRESARIAL

UNIÓN EUROPEA
Fondo Europeo de Desarrollo Regional

*Una manera de hacer Europa*

# SAIN4

**Sistemas Avanzados de eficiencia productiva para la Industria 4.0**

**PROGRAMA:** PROYECTOS DE I+D EN COLABORACIÓN

**ACTUACIÓN:** IMDECA-Proyectos de I+D en colaboración

**Fecha de concesión**: 1 de julio de 2016

## Entregable E2.2 (E2.2)

## Diseño de la arquitectura Big Data Analytics y Captura de Datos

Perteneciente al paquete de trabajo: PT2

Participante responsable: ITI

Mes estimado de entrega: Mes 12

GENERALITAT VALENCIANA
IVACE INSTITUTO VALENCIANO DE COMPETITIVIDAD EMPRESARIAL
UNIÓN EUROPEA Fondo Europeo de Desarrollo Regional
Una manera de hacer Europa

# RESUMEN

SAIN4 es un proyecto financiado con el Instituto Valenciano de Competitividad Empresarial (IVACE) y la Unión Europea a través del Fondo Europeo de Desarrollo Regional (FEDER).

El presente documento tiene el objetivo de especificar el diseño de la arquitectura global de la infraestructura Big Data Analytics y Captura de Datos. La problemática está divida en tres niveles: (i) la arquitectura conceptual o de referencia en la que se detalla la organización de los módulos, componentes e interfaces que componen la arquitectura, (ii) la arquitectura software en la que se describan los diferentes servicios y tecnologías que componen la infraestructura, y (iii) la arquitectura hardware que describe los sistemas físicos necesarios para el despliegue arquitectónico planteado.

# ABSTRACT

SAIN4 is a project funded by the Valencian Institute for Business Competitiveness (IVACE) and the European Union through the European Regional Development Fund (FEDER).

This document aims to specify the design of the global architecture of the Big Data Analytics and Data Capture infrastructure. The problem is divided into three levels: (i) conceptual architecture, which details the organization of the modules, components and elements interfaces, (ii) the software architecture in which the different services are described and technologies that make up the infrastructure, and (iii) the hardware architecture that describes the physical systems needed for the proposed architectural deployment.

# Tabla de Contenidos

**1 Glosario de términos ..... 6**

**2 Introducción ..... 7**

*2.1 Objetivos del Paquete de Trabajo 2 ..... 7*

*2.2 Objetivo del Presente Documento ..... 7*

**3 Visión General de la Arquitectura ..... 8**

*3.1 Requisitos Arquitectónicos ..... 8*

*3.2 Arquitectura Conceptual de Referencia ..... 9*

3.2.1 Plant Floor ..... 10

3.2.2 Factory ..... 10

3.2.3 Cloud ..... 11

**4 Arquitectura Software ..... 11**

*4.1 Sensoring Layer ..... 12*

*4.2 Operation Data Layer ..... 13*

*4.3 Factory Layer ..... 14*

*4.4 End-User Applications Layer ..... 15*

*4.5 Data & Analytics Layer ..... 16*

**5 Arquitectura Hardware ..... 16**

*5.1 Plant Floor ..... 17*

*5.2 Factory ..... 17*

*5.3 Cloud ..... 17*

**6 Tecnologías Involucradas ..... 18**

*6.1 Infraestructura ..... 19*

6.1.1 CDH (Cloudera Distribution Including Apache Hadoop) ..... 19

6.1.2 IaaS basada en OpenStack ..... 20

6.1.3 OPC-UA ..... 20

*6.2 Repositorio de Información ..... 24*

6.2.1 Apache Cassandra ..... 24

6.2.2 Apache HBase....25
6.2.3 Bases de Datos Relacionales....25
*6.3 Comunicación....26*
6.3.1 RabbitMQ (AMQP)....26
6.3.2 Mosquitto (MQTT)....27
*6.4 Analytics....27*
6.4.1 Spark....28
6.4.2 Hive....28
**7 Conclusiones....29**

Figura 1: Arquitectura conceptual ..... 10
Figura 2: Arquitectura Software basada en Capas ..... 12
Figura 3: Visión general de OPC-UA ..... 21
Figura 4: Arquitectura de OpenStack ..... **¡Error! Marcador no definido.**
Figura 5: Arquitectura de CDH ..... **¡Error! Marcador no definido.**
Figura 6: Arquitectura de YARN ..... **¡Error! Marcador no definido.**
Figura 7: Arquitectura de Cassandra ..... **¡Error! Marcador no definido.**
Figura 8: Arquitectura de Spark ..... **¡Error! Marcador no definido.**
Figura 9: Diagrama de petición-respuesta en ZeroMQ ..... **¡Error! Marcador no definido.**

Tabla 1: Requisitos Arquitectónicos y Soporte Propuesto ..... 29

# 1 Glosario de términos

En el presente documento se utilizará algunos conceptos que deben ser definidos con anterioridad a su lectura para aclarar ambigüedades y facilitar el entendimiento.

- **Industria 4.0:** Evolución de los procesos de fabricación industriales en los cuales a través de la sensorización de los sistemas es posible explotar la información de forma inteligente
- **Big data:** Conjunto de tecnologías que permiten el manejo de grandes volúmenes de datos que no pueden ser tratados con las tecnologías convencionales.
- **Machine Learning:** Conjunto de técnicas que permiten crear algoritmos capaces de generalizar comportamientos a partir de información no estructurada.
- **Big Data Analytics:** Disciplina enfocada a la aplicación de técnicas estadísticas y analíticas con el fin de extraer conocimiento a partir del *Big Data*.
- **Overall Equipment Efficiency:** Indicador ampliamente utilizado en entornos industriales que indica mediante una razón porcentual la disponibilidad, rendimiento y calidad de un equipamiento industrial determinado.
- **Data cluster:** conjunto de servidores interconectados para gestionar grandes volúmenes de información de forma distribuida.
- **PLC:** Controlador lógico programable utilizado para automatizar procesos industriales y capturar señales de las maquinas involucradas
- **SCADA: S**upervisory **C**ontrol **A**nd **D**ata **A**cquisition, software que permite la gestión y monitorización de procesos industriales a través de la información ofrecida por PLCs y otros dispositivos.
- **Industrial Internet of Things:** extensión del paradigma “Internet of Things” en el cual el ecosistema de dispositivos interconectados entre si lo componen elementos del entorno industrial como sensores, equipamiento, PLCs o SCADAs.
- **Sistema:** contempla el total de componentes, paquetes, servicios y aplicaciones que permiten alcanzar los objetivos generales del proyecto.

# 2 Introducción

## 2.1 Objetivos del Paquete de Trabajo 2

El objetivo de este paquete de trabajo (**PT2-Infraestructura Big Data Analytics y Captura de Datos)** es el de diseñar e implementar las infraestructuras de captura, almacenamiento, procesamiento y análisis de grandes volúmenes de datos que son necesarias en el marco del proyecto SAIN4. Esta infraestructura será diseñada teniendo en cuenta el ciclo de vida del proceso productivo que integra la denominada Industria 4.0. Con dicho fin, será necesario realizar un conjunto de tareas para:

- Identificar y evaluar las características de los sistemas industriales que producirán información teniendo en cuenta sus mecanismos de comunicación
- Analizar los estándares, protocolos y arquitecturas industriales de comunicación para seleccionar los más adecuados para los objetivos del proyecto y establecer un marco común de interoperabilidad de los diferentes sistemas productores de información.
- El diseño y desarrollo de la infraestructura de captación de datos a partir de sensores, dispositivos industriales y otros sistemas informáticos, que sea adaptable a los diferentes modelos de producción presentes en la industria metal-mecánica.
- Diseño arquitectónico y desarrollo de una infraestructura Big Data Analytics que permita realizar tareas de procesamiento paralelo de grandes volúmenes da datos.
- Integración y validación de los sistemas de captura de datos con la infraestructura Big Data Analytics para implementar posteriormente un motor de prognosis.

El resultado final del paquete de trabajo será la implementación de una infraestructura que permita el desarrollo del paquete PT3, Motor de Prognosis para la eficiencia productiva, y del paquete P4, la implementación del Sistema de Gestión Avanzada para la eficiencia productiva.

## 2.2 Objetivo del Presente Documento

El objetivo del entregable E2.2 es un informe descriptivo de la arquitectura global de la infraestructura Big Data Analytics y Captura de Datos. Este informe se presenta en base a tres niveles de abstracción: (i) la arquitectura conceptual o de referencia en la que se detalla la organización de los módulos, componentes e interfaces que componen la arquitectura, (ii) la arquitectura software en la que se describan los diferentes servicios y tecnologías que componen la infraestructura, y (iii) la arquitectura hardware que describe los sistemas físicos necesarios para el despliegue arquitectónico planteado.

Como punto de partida, se plantea una visión genérica que sea coherente con los principios establecidos por la filosofía de la Industria 4.0. Al tratarse de una primera aproximación metodológica, el presente documento no contempla los requisitos específicos que pueda tener el dominio de aplicación, en este caso el sector metalmecánico y del mueble. Por lo tanto, la arquitectura aquí propuesta evolucionará, conforme se detecten las necesidades específicas, para dar soporte al despliegue de los respectivos proyectos pilotos que se ejecutarán en posteriores anualidades. En líneas generales los requisitos arquitectónicos a contemplar son: 1) la recopilación de la información proveniente del equipamiento industrial en distintas

localizaciones de una fábrica, 2) la agregación de toda la información de forma histórica, 3) la monitorización de la evolución de los indicadores a partir de los modelos analíticos desarrollados y de eventos relevantes.

El resto del documento se organiza de la siguiente forma. En la sección 3 se presentan los requisitos arquitectónicos a contemplar y que guían el diseño global de la arquitectura. Esta sección también presenta la visión abstracta de la arquitectura y de los distintos niveles y capas que la conforman. A continuación, las secciones 4 y 5 detallan respectivamente la arquitectura software y hardware a desplegar. En la sección 6 se detallan las tecnologías candidatas a implementar la arquitectura propuesta. Por último, las conclusiones presentan brevemente el alineamiento entre los requisitos planteados y la propuesta arquitectónica desarrollada.

# 3 Visión General de la Arquitectura

Como punto de partida para la definición de la arquitectura, a continuación, planteamos una serie de requisitos arquitectónicos a considerar por la plataforma a desarrollar. Cabe reseñar, que estos requisitos se han planteado teniendo en cuenta los objetivos iniciales del proyecto SAIN4 y la experiencia previa en proyectos de índole similar. Durante la evolución del proyecto se definirá el alcance de forma más concreta en función de los casos de uso específicos a desarrollar.

## 3.1 Requisitos Arquitectónicos

**Interoperabilidad**: En el marco de los sistemas de producción industriales, un reto que continua presente es como intercambiar datos entre la amplia variedad de dispositivos involucrados como maquinaria industrial, sensores, aplicaciones SCADA o HMI, etc. Considerando el contexto del proyecto SAIN4, es de esperar la utilización de equipamiento industrial de diversos fabricantes y con distintos protocolos de comunicación. Por lo tanto, la arquitectura debe garantizar la interoperabilidad entre los servicios y aplicaciones para que cada servicio pueda consumir datos de otros servicios o equipamientos industriales. Satisfacer la interoperabilidad descrita implica que la arquitectura debe soportar las transformaciones necesarias, para adaptar los datos a un formato o esquema que sea comprensible para los distintos servicios involucrados.

**Elasticidad:** Por la naturaleza del proyecto es de esperar que la cantidad de datos a considerar evolucione desde las etapas iniciales hasta la fase de validación de los demostradores. Para asegurar la continuidad del sistema a largo plazo y, su adaptación a las necesidades de una amplia gama de usuarios u organizaciones, la arquitectura tiene que ser inherentemente escalable. La escalabilidad se considerará con respecto a los recursos tanto hardware como software que deberá proveer.

**Disponibilidad:** En los entornos industriales es habitual que el equipamiento se encuentre en funcionamiento durante largos periodos de tiempo e, incluso, de manera ininterrumpida. En consecuencia, una caída o problema en el sistema de captura de datos, puede ocasionar

pérdidas de información con un alto impacto en la organización. La arquitectura propuesta deberá ser robusta en el marco de la disponibilidad, garantizando un determinado *uptime* en función de los procesos industriales a soportar.

**Alto volumen de cómputo bajo demanda:** la realización de modelos estadísticos y predictivos utilizando Big Data, son tareas que requieren de un alto consumo de recursos de computación. Teniendo en cuenta el contexto de las PyMES involucradas en el proyecto SAIN4, es de esperar que no se posea de la capacidad de computo necesaria como para ofrecer resultados con tiempos de respuesta aceptables. Debido a esta restricción, la arquitectura debe soportar la inclusión de unidades de cómputo, bien a través de clúster físicos o virtualizados, con el fin de dar soporte a las tareas algorítmicas de mayor complejidad. El uso de estas unidades adicionales será puntual para la realización de una tarea específica, pudiendo ser liberadas para otras tareas tan pronto como finalicen.

**Visualización de grandes volúmenes de datos**: en el marco del proyecto será necesario desarrollar un conjunto de visualizaciones gráficas para dar soporte a actividades de Business Intelligence o Overall Equipment Efficiency (OEE) que involucren a los usuarios finales. En concreto la información será utilizada para la representación gráfica y el cálculo de indicadores de eficiencia productiva. Para la construcción de dichas visualizaciones será necesario recuperar una gran cantidad de información, la cual será transferida desde el sistema histórico que la almacene. Por lo tanto, la arquitectura tendrá que considerar aspectos como la latencia, con el fin de proporcionar al usuario final una interacción fluida con la visualización de la información.

## 3.2 Arquitectura Conceptual de Referencia

El término “Arquitectura Conceptual de Referencia” representa una descripción abstracta de un sistema en términos de su organización, las funciones de sus componentes clave y los tipos de interacciones válidos entre ellos. Para la establecer la referencia arquitectónica de la propuesta hemos establecido una descomposición en niveles y capas. Un nivel hace referencia a la localización de la infraestructura que se encarga de dar soporte a un proceso industrial concreto. Hemos definido una jerarquía desde el nivel más cercano físicamente al proceso industrial, como es la propia maquinaria industrial, hasta el nivel superior que hace referencia al nivel más abstracto, que son los sistemas de información. Modelos más elaborados en el marco de la Industria 4.0 (como RAMI 4.0) establecen una jerarquía de niveles con mayor detalle y alcance. Sin embargo, ya que el marco de las PyMEs es más reducido que el de una gran empresa, hemos optado por una jerarquía más sencilla en aras de facilitar su comprensión.

Cada nivel se subdivide en capas funcionales que hacen referencia al conjunto de componentes software de la plataforma que son desarrollados en el proyecto para implementar un nivel. Por lo tanto, las capas proporcionan una descomposición funcional de la arquitectura a nivel software y hardware, mientras que los niveles lo hacen a nivel organizativo.

A continuación, describimos con mayor detalle la descomposición en niveles, resumida en la siguiente figura, mientras que las distintas capas son detalladas en las secciones de arquitectura software y hardware.

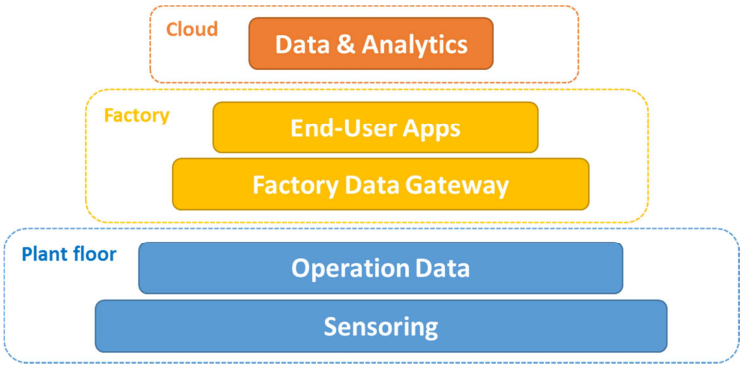



**Figura 1:** Arquitectura conceptual

### 3.2.1 Plant Floor

El nivel de planta o *Plant Floor* hace referencia al equipamiento y procesos que se desarrollan en el marco de la cadena de producción. Los actores principales en este nivel son por un lado el equipamiento industrial que da soporte a la producción y, por el otro lado, los operarios involucrados en las tareas de monitorización y mantenimiento. Desde el punto de vista de la arquitectura a desarrollar, en este nivel nos encontramos los dispositivos encargados de captar lecturas e información en "bruto" proveniente del equipamiento industrial y la red de comunicaciones. Ejemplos de estos dispositivos pueden ser sensores de temperatura, sensores acústicos, motores de control de movimiento, PLCs, etc.

También forma parte de este nivel la red de comunicaciones de planta tanto entre los distintos dispositivos industriales, como con los niveles superiores. La comunicación y el intercambio de datos a este nivel tradicionalmente se realizan mediante el uso de una red Ethernet Industrial, que es lo que se espera en el ámbito del proyecto.

Este nivel se compone de dos capas. Una capa de sensorización o *Sensoring* que aglutina aquellos dispositivos, sensores y protocolos que proporcionan datos "en bruto" a la plataforma, y una capa de datos de operación, *Operation Data,* que organiza, normaliza, y almacena la información de sensorización para su consumo por parte de las capas o niveles superiores.

### 3.2.2 Factory

Las distintas plantas o cadenas de producción se agrupan mediante el nivel superior de *Factory* o fábrica. En este nivel se agregan los resultados obtenidos de forma independiente en cada una de los niveles de planta con el fin de proporcionar una visión unificada. Inicialmente en el proyecto SAIN4, se espera una relación uno a uno entre el nivel de planta y fábrica, puesto que se tendrá en cuenta en un único proceso industrial. Sin embargo, es necesario considerar la

posibilidad de tener que organizar varios procesos industriales que no se encuentra físicamente interconectados. Por lo tanto, en este nivel se gestiona el intercambio de información que se produce entre los dispositivos de una planta, operación u otra distribución organizativa en el marco de una fábrica.

En este nivel, se definen dos capas arquitectónicas. Por una parte, el *Factory Data Gateway* se encarga de agregar la información proveniente desde el nivel de planta, y ponerlo a disposición del resto de servicios o aplicaciones que la requieran. Por otro lado, la capa *End-user Apps* hace referencia a aquellas aplicaciones a partir de las cuales los usuarios finales pueden interactuar con la información disponible.

### 3.2.3 Cloud

Este nivel define una infraestructura de alta computación externa a los sistemas informáticos de la fábrica para dar soporte a las necesidades de persistencia y *Big Data Analytics* de la organización. El hecho de denominarla *Cloud* hace referencia a la necesidad de escalabilidad bajo demanda que define este nivel. A efectos prácticos, este nivel puede componerse de los sistemas o servidores informáticos que posea la empresa, junto con los servicios externos que sean necesarios para soportar las necesidades de computación o almacenamiento adicionales. Esta infraestructura informática se diferencia de la presentada en los anteriores niveles por su alto nivel de sofisticación: mientras que en los niveles de planta y fábrica se utilizan PCs de carácter industrial, en este nivel es de espera un clúster o servidor avanzado. Este nivel se compone de una única capa, *Data & Analytics*, que se encarga de aglutinar toda la información disponible en la organización para realizar los análisis oportunos.

# 4 Arquitectura Software

En la presente sección detallamos los distintos componentes software que implementan cada una de las capas introducidas en la sección anterior. La siguiente figura resume de forma visual la interacción entre las distintas capas y los distintos módulos que conforman cada una.

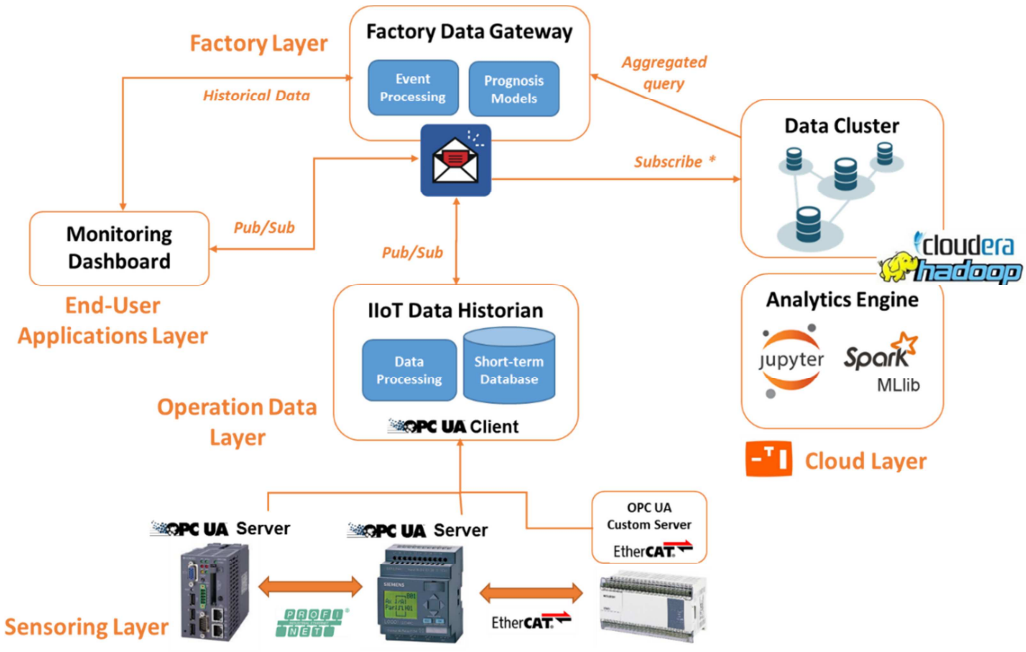



**Figura 2:** Arquitectura Software basada en Capas

## 4.1 Sensoring Layer

La información que se genera en esta capa se compone fundamentalmente de lecturas provenientes de los distintos dispositivos industriales implicados en el proceso de producción. Los datos suelen seguir un esquema preestablecido en el que se indica el dispositivo que originó el dato y la marca temporal o *timestamp* en la cual se originó, los cuales son los datos descriptivos fundamentales además del valor obtenido en sí. Si bien el tamaño de cada lectura o registro suele ser pequeño, en el marco de la Industria 4.0 se generan datos en el orden de segundos a milisegundos. Por esta razón la capa de sensorización tiene que transmitir registros de pequeño tamaño, pero en grandes volúmenes.

La comunicación y el intercambio de datos a este nivel tradicionalmente se realizan mediante los denominados *field bus,* siendo los más habituales los basados en Ethernet como Modbus TCP, PROFINET IO o EtherCAT. Estos buses permiten el intercambio de información a bajo nivel, es decir codificada en una trama TCP o similar, lo que complica su posterior explotación a nivel de aplicación. Una arquitectura que quisiera consumir esta información, debería entender el protocolo subyacente, extraer de la trama el valor concreto (normalmente codificado en forma binaria) y almacenarlo de forma accesible (por ejemplo, en una base de datos). Esta aproximación resulta inviable no sólo en el marco del presente proyecto, sino además en organizaciones que cuenta con un mayor número de recursos.

Mediante la popularización del estándar OPC-UA, descrito con detalle en el entregable E2.1, cada vez son más los dispositivos industriales con la capacidad de actuar de forma nativa como un OPC-UA Server. El estándar define una serie de modelos de datos, como DA-Access o DA-

Historian, para exponer la información de dispositivos industriales en un formato y APIs comunes. Este hecho implica que los datos pueden ser obtenidos según los modelos de información y servicios que aporta el estándar, simplificando su explotación.

Por lo tanto, asumiremos que esta capa está formada por dispositivos OPC-UA que actúen como servidores y que cumplan al menos con el modelo *Data Access,* que permite recuperar la información de lecturas sensorizadas con un esquema que facilita su procesamiento. Tal y como se muestra en la figura, los distintos dispositivos ofrecerán una interfaz *OPC-UA Server* a las capas superiores. Sin embargo, entre ellos, podrán continuar intercambiando datos mediante protocolos más específicos. Esta dualidad es necesaria puesto que OPC-UA implica una carga de recursos, que desaconseja su utilización en procesos que requieren interoperabilidad en tiempo real. Además, esta aproximación es compatible con comunicaciones de datos previamente establecidas en la cadena de producción, que no son relevantes para los niveles superiores.

Es de esperar que existan dispositivos que no cumplan dicho estándar, por ejemplo, porque fueron implantados con bastante anterioridad, y que no sea sencillo y/o económica su sustitución. En estos casos se desplegará un *OPC-UA Server* personalizado para el tipo de protocolo con el que el dispositivo pueda comunicarse. Existen en el mercado soluciones software comercial para un amplio número de protocolos, por ejemplo, los servidores OPC ofrecidos por la empresa Matrikon. Si el protocolo no se encuentra soportado o se necesitan características específicas, es posible utilizar SDKs, tanto comerciales como libres, para el desarrollo del servidor. También es posible introducir soluciones *hardware* que actúan como pasarelas de varios dispositivos para traducir la información conforme al estándar, por ejemplo, el *echocollect* de Softing para conectar con PLCs.

Podríamos resumir que el fin de la capa de sensorización es exponer toda la información relevante en términos de OPC, bien de forma nativa o bien definiendo intermediarios que transformen la información.

## 4.2 Operation Data Layer

Esta capa implementa el software necesario para recopilar la información que se produce entre los dispositivos de una planta, operación u otra distribución organizativa de una fábrica. La persistencia se implementa mediante un *Data Historian*, es decir una base de datos histórica a corto plazo con los datos de todos los dispositivos bajo un esquema común. Esta base datos se complementa con un módulo para el procesamiento de datos en “bruto”. La combinación de ambos componentes implementa un *IIoT Data Historian*, especializado en el procesamiento y almacenamiento de información proveniente de redes industriales.

La comunicación con la capa de sensorización se implementa mediante un *OPC Client*, que se encarga de recopilar la información que expone cada uno de los dispositivos involucrados siguiendo el mismo estándar. Está información es previamente procesada mediante un módulo *Data Processing*, que se encarga no sólo de transformar el formato de los datos, sino de también de normalizar la información, eliminar “ruido” y, en definitiva, las tareas asociadas

a un proceso de *Data Wrangling.* Este proceso será implementado mediante scripts para el procesamiento de datos en lenguajes como Java o Python.

Una vez la información es procesada, esta es almacenada en una base de datos histórica de corto-plazo (*Short-term Historian*) que almacenará únicamente la información generada en un rango determinado, por ejemplo, un mes, en función de los recursos disponibles y las necesidades organizativas. Automáticamente, la información que expire será eliminada de este espacio de almacenamiento, con lo cual su labor es ser un almacén temporal. Por sus características se utilizará una base de datos que siga el paradigma de "columna", bien relacional, como MySQL, o bien NO-SQL, como Cassandra, ya que no existen requisitos de recursos específicos y diversas tecnologías de bases de datos cumplen este rol. En el caso de ser necesaria una ingesta de datos elevada, se evaluará la introducción de una *Timeseries Database*. El objetivo de esta base de datos es también imponer un esquema previamente definido a los datos para simplificar su posterior análisis.

Todos los datos procesados y almacenados, son distribuidos y publicados a través del bróker que se detalla en el siguiente apartado para su difusión al resto de la planta o fábrica. Se asume que esta capa estará replicada en diversos PCs industriales para monitorizar un conjunto específico de equipamiento industrial desplegado en una planta u operación.

## 4.3 Factory Layer

El objetivo de esta capa es aglutinar y distribuir los datos con los cuales se monitoriza el estado de la cadena de producción. Se compone fundamentalmente de un *Factory Data Gateway*, que realiza la labor de *proxy* para la recuperación de información de los diversos *IIoT Data Gateways* desplegados en el entorno de la fábrica. La idea es que el *Factory Data Gateway* establece un punto de acceso único a través del cual el resto de servicios acceden a los datos, sin importar si pertenecen al mismo nivel arquitectónico, capa funcional o estructura organizativa.

Esta capa se implementa mediante un *bróker* basado en el patrón publicar/suscribir para distribuir la información generada por los distintos *IIoT Data Gateways* al resto de dispositivos de la fábrica. Este bróker sigue las recomendaciones planteadas en la futura implementación del protocolo de transporte soportado por OPC-UA, que enfatiza el uso de este tipo de patrón de comunicación. De forma simplificada, se publicarán las lecturas de datos convenientemente etiquetados con su fuente de origen y con un esquema establecido en la base de datos histórica. Cada mensaje será asociado a un *tópico* estableciendo un canal de comunicación específico para un tipo de mensaje y una cola de mensajes. Ejemplos de tópicos son alertas, información de control, monitorización de un dispositivo determinado, etc.

Las distintas aplicaciones y servicios se suscribirán a aquellos tópicos que resulten de su interés para las tareas a desempeñar. Los protocolos utilizados en soluciones similares son AMQP (RabbitMQ) y MQTT (Apache ActiveMQ), ya que son abiertos y están estandarizados, sin embargo, otras tecnologías como Apache Kafka o ZeroMQ son completamente válidas para la

implementación de este patrón. Idealmente existirá un único *bróker* por fábrica, salvo que por necesidades de rendimiento y escalabilidad sugiera su replicación.

A partir de los datos recibidos en esta capa se realizan dos tareas relacionadas entre sí y que se implementan mediante dos módulos adicionales. El módulo de *Event Processing*, basado en tecnologías de *Complex Event Processing* (CEP) como WSO2 o Apache Camel, se encarga de detectar eventos relevantes a través de reglas que tienen en cuenta los datos recibidos. Por ejemplo, que se reciba un valor de temperatura anómalo por parte de un componente industrial crítico. Los eventos generados son a su vez publicados en el bróker para su distribución. Esta capa también incluye un módulo de *Prognosis Models*, que se compone de un conjunto de modelos analíticos para establecer predicciones en tiempo real a partir de los datos. Estos modelos, implementados en lenguajes como Python o R, son previamente generados por la infraestructura de alta computación disponible. Al igual que en el caso de los eventos, el resultado de los modelos es publicado en el *bróker* con la periodicidad que se establezca.

## 4.4 End-User Applications Layer

Esta capa representa el conjunto de aplicaciones como MES, ERPs HMIs o Dashboards que hacen uso de los datos producidos por las distintas operaciones de la fábrica, para mostrarlos al usuario final. Para la obtención de los datos en tiempo real, o siendo más específicos con un horizonte temporal breve, estas aplicaciones deberán subscribirse utilizando el *bróker* a los eventos y datos que resulten de relevancia. Sin embargo, las peticiones de datos históricos deberán ser solicitadas a través de una API al *Factory Data Gateway*. Este componente determinará, en función del rango temporal solicitado, si la información se encuentra "localmente" en alguno de los *IIoT Data Historian* desplegados, o si por el contrario de solicitarse al nivel Cloud. Ilustremos el funcionamiento de esta capa con un ejemplo práctico:

Supongamos que una aplicación tiene que generar un gráfico dinámico de las temperaturas que está registrando un determinado sensor desde que iniciamos la aplicación. En este caso nos suscribiríamos a las temperaturas que estuviese recibiendo el bróker e iríamos construyendo el gráfico conforme se fuesen recibiendo. Sin embargo, si queremos consultar la evolución anual de la temperatura, es decir información histórica, lanzaríamos una petición al *Factory Da Gateway* en el cual dicho sensor se encuentra registrado. El *Gateway* determinaría si el rango de fechas solicitado corresponde con la información que mantiene alguno de los *IIoT Data Historian* que gestiona. De no ser así, solicitaría la información directamente al nivel Cloud. Se asume que, por motivos de rendimiento, este tipo de consultas serán de tipo agregado (medias, desviaciones típicas, sumatorio, etc.) para evitar la transferencia de grandes volúmenes de datos.

Además de las aplicaciones que ya se encuentren actualmente desplegadas, esta capa también incluirá un *Monitoring Dashboard* en el marco del proyecto SAIN4. Este *dashboard*, el cual será implementado mediante tecnologías Web, ofrecerá indicadores y predicciones en función de los modelos analíticos desplegados. Esta aplicación será el punto de entrada de los usuarios finales y los operarios para interactuar con el resto de la arquitectura, además de ser

notificados de los eventos relevantes. Debido al volumen de datos esperado, se deberán elaborar visualizaciones avanzadas bien mediante *frameworks* de gráficas en Javascript, como Cytoscape.js o D3.js, o la integración con Tableau.

## 4.5 Data & Analytics Layer

Está capa define los componentes software para dar soporte a las necesidades de *Big Data Analytics* de la organización y a la persistencia de la información. Por la estrecha relación entre ambas tareas, se ha decidido que formen parte de una misma capa.

Como capa de persistencia se ha optado, por un *Data Cluster* escalable, desplegado sobre Hadoop y con una tecnología NO-SQL como base de datos, para almacenar toda la información histórica generada. Este clúster estará suscrito todos los *topics* generados por el *broker*, almacenando la información con un esquema similar al que soportan los *IIoT Data Historian* y el *Factory Data Gateway*. Utilizando un esquema de datos unificado entre los distintos niveles se simplifica su explotación. Por las necesidades de volumen se optará por un almacenamiento distribuido que garantice la escalabilidad. El acceso a los datos se realizará a través de una API que abstraiga la implementación tecnológica en la cual se haya desarrollado el clúster. Para la ingesta masiva de datos debe existir un mecanismo alternativo al *bróker,* bien porque el alto volumen desaconseja su uso o bien por qué no han sido generados a través de él. Por lo tanto, esta capa también define un mecanismo de volcado automatizado desde ficheros de datos o bases de datos legadas.

El segundo componente en un motor de *analytics* para entrenar modelos y aplicar algoritmos de *Machine Learning* a partir de los datos disponibles en el *Data Cluster*. Para ello se encarga de entrenar un modelo estadístico a partir del histórico almacenado. Este componente da soporte a distintos modelos estadísticos, así como a su reentrenamiento con una periodicidad escogida. Este componente utilizará librerías específicas como MLib, scripts en Python o R y *Spark* para realizar los cálculos necesarios de forma distribuida con varias unidades de computación. El resultado de este proceso será un modelo que será desplegado en el módulo de *Prognosis Models* del *Factory Data Gateway*.

Esta capa puede ser desplegada tanto mediante un Cloud Privado, utilizando un PaaS como Cloudera, o mediante servicios de Cloud Público, es decir, utilizando tecnologías como Azure o AWS, dependiendo de la infraestructura tecnológica de la organización.

# 5 Arquitectura Hardware

En la presente sección detallamos el hardware requerido en los distintos niveles definidos. Los requisitos y restricciones planteadas a lo largo del documento, implica la necesidad de seguir una aproximación elástica de la arquitectura hardware del sistema.

Cabe reseñar, que algunos aspectos de esta arquitectura vendrán determinados en gran medida por el equipamiento industrial que deba soportarse en el proyecto. De ahí que se proporcione una arquitectura flexible ante futuros cambios.

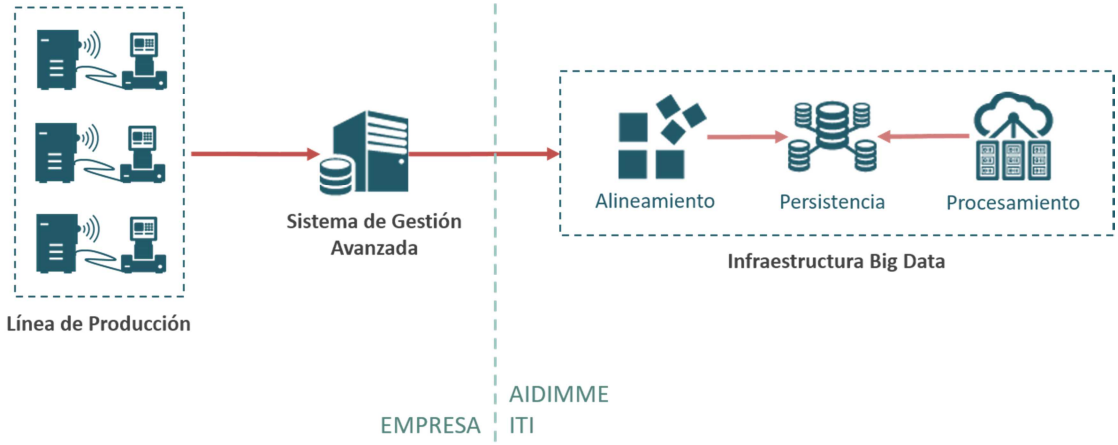



**Figura 3:** Arquitectura Hardware

La figura muestra cómo se organizan los diferentes elementos hardware que intervienen en el ciclo de vida del Sistema de Gestión Avanzada.

## 5.1 Plant Floor

El hardware principal que componen este nivel es el equipamiento de interconexión entre la maquinaría industrial. Este equipamiento puede consistir en PLCs o dispositivos más especializados para la transmisión de información. En principio no es posible determinar con exactitud, el equipamiento involucrado debido a las características específicas de cada proceso industrial. Como red de comunicaciones se asumirá el uso de Industrial Ethernet como tecnología base, si bien otros protocolos de bus de campo pueden estar involucrados en procesos determinados. En cada uno de esto niveles se usará al menos un PC industrial para monitorizar y obtener los datos en bruto, procedentes del equipamiento industrial. Además, al menos uno de estos PCs tendrá desplegado el *IIoT Data Historian* para establecer el almacenamiento local. No se descarta la utilización de dispositivos que actúen como intermediarios para procesar la información y proporcionarla en formatos más adecuados.

## 5.2 Factory

El hardware que del cual se compone este nivel es un servidor centralizado situado en un lugar interconectado con el resto de las plantas, y con la red externa que se comunique con el nivel Cloud. Este servidor requiere de cierta capacidad de cómputo, para gestionar adecuadamente el *bróker* de mensajes, pero no son necesarios grandes recursos de almacenamiento, puesto que localmente los datos se encuentran en los distintos *IIoT Data Historian.* Será en este servidor dónde también resida la aplicación *Monitoring Dashboard*.

## 5.3 Cloud

Los requisitos planteados a lo largo del documento requieren proporcionar elasticidad a la arquitectura hardware del sistema. En concreto, la filosofía de Infraestructura como Servicio (IaaS) se ajusta perfectamente a los requerimientos relacionados con la escalabilidad, disponibilidad y alta capacidad de cómputo. De esta forma la arquitectura propuesta puede evolucionar en función de las necesidades de datos que requieran los distintos demostradores del proyecto SAIN4.

Este nivel seguirá una estrategia basada en virtualización, es decir, se crearán de forma dinámica instancias o máquinas virtuales para dar soporte a los servicios requeridos. Estas instancias contarán con los mismos servicios y características basándose en un despliegue base de tecnologías. En el marco del proyecto, se distinguen dos posibles tipos de instancias:

- **Instancias de almacenamiento:** especializadas en el almacenamiento y tratamiento de los datos, son instancias con una amplia capacidad de almacenamiento y redundancia. Estas instancias formarán el *Data Clúster* distribuyendo la información a través de múltiples tablas y ficheros. El hecho de utilizar varias instancias de almacenamiento en vez un repositorio único, no sólo mejora la disponibilidad y la escabilidad, sino que también las consultas a la información son más eficientes.
- **Instancias de cómputo:** son instancias compuestas de uno o varios *cores* de procesamiento especializadas en la ejecución de algoritmos complejos. Su función es la de crear varios hilos de ejecución de los algoritmos que procesan los datos, haciendo también un uso intensivo de la memoria principal. Su función es dar soporte al módulo de *Analytics.*

Por otra parte, el proyecto se desarrollará de forma coordinada con el ITI, que aportará su infraestructura física (clúster de computo) para dar soporte al nivel Cloud. En dicho clúster se encuentra desplegado el software necesario para realizar las labores de análisis y minería de datos. Sus características hardware son las siguientes:

- 10 nodos físicos con 400 cores de CPU
- 3TB de RAM y 100 TB de Disco Duro
- Red de 10 GB
- Clúster virtualizado con OpenStack y con los servicios de Cloudera disponibles

# 6 Tecnologías Involucradas

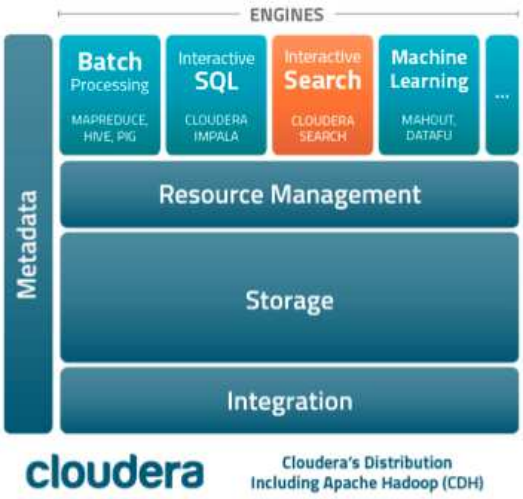
En la presente sección realizamos una breve introducción de las tecnologías y frameworks que estarán involucrados en la implementación de la arquitectura anteriormente descrita. Hemos subdivido la categorización de las tecnologías en cuatro bloques. En primer lugar, introducimos aquellas tecnologías involucradas en la definición de la infraestructura que da soporte al resto de componentes arquitectónicos. A continuación, introducimos las tecnologías para la creación de los repositorios de datos que contendrá la información de los distintos niveles. El tercer bloque presenta los protocolos para el intercambio de datos entre los dispositivos y sistemas

en la *Factory Layer*. Por último, introducimos los frameworks que serán utilizados para la implementación de las técnicas de *Analytics* y *Machine Learning* necesarias.

## 6.1 Infraestructura

### 6.1.1 CDH (Cloudera Distribution Including Apache Hadoop)

A la hora de desplegar los distintos servicios que conforman una plataforma *Big Data*, uno de los grandes problemas es instalarlos de forma adecuada y configurarlos acorde con las necesidades del problema. CDH (Cloudera Distribution Including Apache Hadoop) es una distribución de código abierto que simplifica el despliegue de un ecosistema de servicios basado en Apache Hadoop. Esta distribución cuenta con el respaldo de Cloudera, uno de las empresas más relevantes en el área del *Big Data*, proporcionando una de las soluciones más extendidas a nivel empresarial. Desde el punto de vista arquitectónico, tal y como muestra la siguiente figura, CDH aglutina un conjunto de tecnologías para abordar desde la integración de la información, su almacenamiento, la gestión de recursos de procesamiento y, por último, el análisis de la información. Se compone de más de una docena de tecnologías, también de código abierto, como Apache Spark o Hive ya pre configuradas para ejecutarse correctamente sobre un clúster Hadoop.




En el marco del proyecto SAIN4 utilizaremos CDH como la distribución base para la selección de los componentes que implementan fundamentalmente la capa de *Data & Analytics*. Las tecnologías seleccionadas serán presentadas en las siguientes secciones, optando por una tecnología alternativa si mejora la propuesta de CDH. Esta decisión se fundamenta en experiencias previas que han demostrado su robustez en entornos de Industria 4.0. También abre la posibilidad de incluir servicios y herramientas empresariales, como Cloudera Manager o Cloudera Enterprise, que permiten garantizar tanto el soporte como facilitar la adopción de

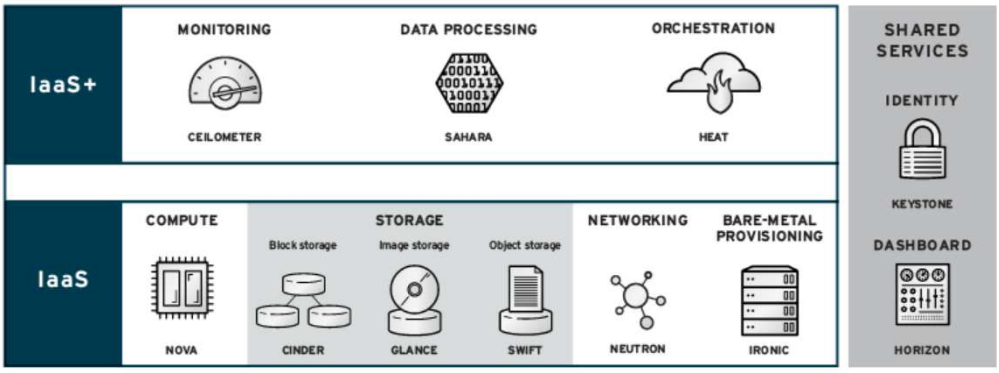
la plataforma. El resultado esperado será el de un cluster Hadoop desplegado utilizando CDH que permitirá el despliegue elástico de los servicios que requiera el proyecto.

### 6.1.2 IaaS basada en OpenStack

Una de las aproximaciones más habituales a la hora de desplegar una infraestructura de captura de datos de altas prestaciones, es optar por una aproximación basada en el paradigma *IaaS* (*Infrastructure as a Service*). Optando por esta alternativa, el *hardware* en el cual se ejecutan los distintos servicios no se encuentra disponible en la propia organización, sino que esta externalizado en una infraestructura sobre la cual se asocian un conjunto de recurso (de almacenamiento, computo, etc. ) bajo demanda. Una de las tecnologías más habituales para la implementación de una plataforma IaaS es OpenStack.

Openstack es proyecto de código abierto para el soporte al *Cloud Computing* en concreto siguiendo el paradigma IaaS. De forma similar a CDH, OpenStack se compone de varios proyectos interrelacionados que permiten gestionar distintos aspectos de un centro de datos, como el almacenamiento o asignación de nodos de computación, a través de un panel de administración web. Como base principal, hace uso de la virtualización para gestionar bajo demanda los recursos a través de la creación de máquinas virtuales. También ofrece sistemas de almacenamiento tanto a nivel de objeto, como por ejemplo un fichero, o por bloque, de forma similar al almacenamiento en disco, que abstraen la configuración de discos física del servidor al usuario final.

En el marco del proyecto SAIN4, se ha optado por la utilización de OpenStack para garantizar la futura evolución de la solución planteada. En el contexto del proyecto, el ITI ofrecerá su clúster bajo la aproximación IaaS para proporcionar recursos que sean necesarios para el procesamiento y análisis de la información. El hecho de utilizar tecnologías abiertas, permite la futura migración a otro clúster basado en Openstack, o bien a plataformas Cloud como las ofrecidas por Amazon o Google.




### 6.1.3 OPC-UA

Actualmente OPC-UA es el estándar de interoperabilidad, creado por la *OPC Foundation*, para la comunicación de los datos provenientes de sensores y dispositivos de campo a los sistemas

de control y planificación industrial. A la hora de diseñarse, se ha tenido en cuenta el soporte al mayor número de dispositivos posibles, desde PLCs hasta servidores empresariales. OPC-UA ha sido diseñado para ser totalmente independiente del fabricante, lenguaje de programación o cualquier tecnología propietaria con lo que se considera un estándar totalmente abierto. Además, es la aproximación elegida para la implementación de la capa de comunicación en el modelo RAMI 4.0. Por su amplio respaldo se considera un estándar "de facto", de tal forma que muchos de los protocolos y modelos de información propuestos por otras organizaciones buscan alinearse con OPC-UA.

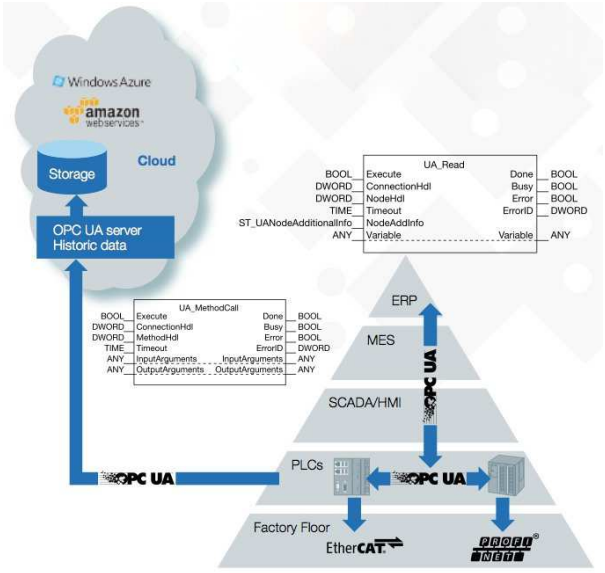



**Figura 4:** Visión general de OPC-UA

En el marco de la Industria 4.0 existe una clara tendencia en la inclusión de la semántica a la hora de definir el proceso de estandarización. Algunos ejemplos son los estándares ISA 88, para definir procesos industriales por lotes, o el ISA 95, para definir la información gestionada por un MES (*Manufacturing Execution System)*. Siguiendo estas pautas OPC-UA, también publicado como estándar IEC 62542, permite el intercambio de modelos de información independientemente de su grado de complejidad. Por lo tanto, complementa a los estándares anteriores incluyendo la interoperabilidad en el nivel semántico. OPC-UA cuenta con un amplio respaldo por parte por la de industria de la producción, con empresas como National Instruments o General Electric, y empresas tecnológicas, como Microsoft o SAP

Podemos dividir el estándar en tres grandes bloques:

- **Meta-modelo y modelo de información:** especifica las reglas, tipos y componentes que debe cumplir un modelo de información siguiendo el estándar OPC-UA.
- **Servicios:** define la interfaz entre un servidor que provee la información y los usuarios que consumen dicha información.

- **Comunicaciones y transporte:** OPC-UA incorpora diferentes protocolos para el intercambio de información como son TCP, cuando se necesita un mayor rendimiento, y HTTP + SOAP, para favorecer la integración entre aplicaciones.

A continuación, detallamos estos tres componentes.

#### *6.1.3.1 El modelo de información de OPC UA*

Este modelo describe un conjunto de entidades, en la terminología OPC-UA nodos, y tipos estándar que pueden ser utilizados para describir objetos del mundo físico en base a un espacio de direccionamiento, del inglés *Address Space*, común. El modelo representa los objetos con sus propiedades o variables, métodos, eventos y relaciones con otros objetos de forma similar a un modelo entidad-relación. De esta forma es posible definir datos de producción, alarmas, eventos etc. de un proceso industrial y soportar dicha información mediante un servidor OPC-UA. Por ejemplo, un sensor de temperatura es representado como un objeto con un valor que mide la temperatura y un conjunto de alarmas si la temperatura rebasa unos límites establecidos. Tanto el concepto de "temperatura" como de "límites" vienen definidos por el modelo de información de OPC-UA. De esta forma, las aplicaciones que utilicen OPC-UA conocen de antemano la información que está representada y como recuperarla. El modelo se estructura de forma jerárquica por lo que las entidades de los niveles superiores son comunes a cualquier aplicación OPC-UA y, a partir de ellas, es posible crear entidades especializadas.

Las propiedades de cada nodo o entidad del modelo son descritas mediante atributos definidos por OPC-UA. Los atributos son los únicos elementos del servidor que pueden contener valores conformes a un tipo simple o complejo. OPC UA permite modelar cualquier objeto, tipo o relación necesaria, mientras que la semántica es definida en el servidor y seleccionada por los clientes. Los tipos pueden ser estándar o específicos de un fabricante, siendo en este caso una organización la responsable de su definición.

El estándar proporciona un conjunto de modelos genéricos para soportar el intercambio de la información industrial más habitual. No obstante, es posible definir modelos más especializados para soportar una tecnología o estándar concreto. Algunos ejemplos son el modelo para soportar los sistemas de control basados en el estándar ISA 95 o para la conectividad entre máquinas (MTConnect que se describe en la siguiente sección). Sin embargo, debe tenerse en cuenta que habitualmente los clientes se encuentran programados para consumir los modelos genéricos, con lo cual no son capaces de procesar la totalidad de un modelo especializado. A continuación, se presentan brevemente los modelos genéricos ofrecidos por el estándar:

- **Data Access:** permite modelar datos en tiempo real, o de forma más concreta, datos que representan el estado actual de un proceso industrial o de negocio. Por ejemplo, incluye la definición de variables análogas o discretas, unidades de medición o códigos de calidad. También incluye la definición de fuentes de datos

que representan sensores, monitores de posición etc. que se encuentran conectados directamente a un dispositivo o mediante buses de campo.

- **Alarms and Conditions:** este modelo define como se gestionan los cambios de estado a través de alarmas. Los clientes pueden registrarse para recibir el evento que se produce cuando cambia un estado y recibir información al respecto.
- **Historical Access:** este modelo permite a un cliente recuperar información histórica sobre valores de variables y eventos. La información puede encontrarse en distintos medios de almacenamiento y el servidor proporciona funciones de agregación para su pre-procesamiento.
- **Programs:** Un programa representa una tarea compleja, como la gestión de un proceso por lotes. Cada programa se representa como una máquina de transición de estados, en la que cada transición envía un mensaje al cliente.

#### *6.1.3.2 Servicios*

OPC-UA define un conjunto de servicios para navegar a través de los modelos, leer o escribir variables o subscribirse a eventos. Los servicios se implementan como un conjunto de mensajes que son intercambiados entre el servidor el cliente. Los servicios se encuentran agrupados de forma lógica a través de 9 *Service Sets* que se describen a continuación:

- **Security Channel:** estos servicios determinan la configuración de seguridad del servidor para establecer un canal de comunicación que garantice la confidencialidad e integridad de los mensajes intercambiados.
- **Session:** estos servicios definen una conexión con la capa de aplicación que representa a un usuario específico.
- **View:** permiten explorar las entidades definidas en el modelo, así como navegar a través de la jerarquía definida y las relaciones.
- **Node Management:** permiten configurar el servidor añadiendo, modificando o borrando los nodos que componen el modelo.
- **Attribute:** permiten leer y escribir valores de los atributos del modelo.
- **Method:** los métodos representan llamadas a funciones definidas en los objetos. Estos servicios definen los protocolos para realizar las invocaciones pertinentes.
- **Monitored Item:** mediante estos servicios se define que atributos deben ser monitorizados o en cuales eventos está interesado un cliente.
- **Susbscription:** estos servicios están relacionados con los anteriores permitiendo generar, modificar o eliminar los mensajes de monitorización generados.
- **Query:** mediante estos servicios un cliente puede seleccionar los nodos o entidades del modelo utilizando criterios de filtrado.

#### *6.1.3.3 Transporte*

Los servicios definidos mediante OPC-UA pueden ser implementados mediante dos aproximaciones. Mediante UA-TCP se garantiza un mayor rendimiento aprovechando la pila de tecnologías de Ethernet. Por otro parte, mediante HTTPS + SOAP se simplifica tanto la integración de la información como la invocación desde distintas plataformas tecnológicas (.NET, Java, etc.). Cabe destacar que, independientemente de la aproximación utilizada, los servicios han sido diseñados para proporcionar un rendimiento alto, por ejemplo, una única lectura puede acceder a miles de valores.

Además del modelo de comunicación cliente-servidor, el estándar ha incorporado recientemente la implementación del patrón de comunicación *Publish/Subscribe.* Mediante este patrón es posible distribuir datos y eventos a todos aquellos dispositivos suscritos dentro de un entorno de producción industrial. Actualmente, se hace uso del protocolo ISO/IEC AMQP 1.0 utilizando JSON para la codificación de la información, de tal forma que se habilita el soporte a sistemas de *Analytics* para el proceso masivo de información. En un futuro se espera incorporar conectividad con plataformas Cloud, como Microsoft Azure.

## 6.2 Repositorio de Información

### 6.2.1 Apache Cassandra

Cassandra es una de las bases de datos NO-SQL más populares debido a su capacidad para crear un clúster utilizando hardware de bajo coste. Su modelo físico se basa en una gran tabla con un número indeterminado de columnas que, unido a un esquema no estricto, permite escalar con relativa facilidad ante nuevas necesidades de datos. De hecho, su escalabilidad utilizando múltiples nodos es una de sus características más destacables. También destaca por el soporte a la replicación asíncrona, esto es sin la necesidad de un nodo maestro que distribuya las peticiones y aumente la latencia. Su soporte a múltiples nodos la convierten en una base datos altamente tolerable a fallos, ya que replica la información de forma transparente a los usuarios. También cuenta con soporte a Hadoop, con lo que permite integrarse con Cloudera CDH. Para la realización de consultas ofrece el Cassandra Query Language (CQL), un subconjunto del lenguaje SQL que simplifica la interacción con la base de datos. Debido a su popularidad y su alto rendimiento, suele ser la referencia sobre la cual otras bases de datos NO-SQL se comparan.

Cassandra es una base de datos que se adapta fácilmente a las necesidades del proyecto SAIN4 y en concreto de la gestión de datos para la Industria 4.0. El paradigma columnar permite una transición sencilla desde los datos almacenados siguiendo un esquema relacional. Además, lenguaje CQL es una ayuda para comenzar a trabajar con la base datos para aquellos que tienen experiencia con el modelo relacional.  Ya que cuenta con soporte al tipo de datos timestamp, es posible definir como índice principal de la información cuando esta ha sido recibida o generada desde los distintos sistemas industriales.

El diseño físico de las tablas tiene implicaciones en las posteriores consultas, en concreto con la definición de la partition key y la clustering key, que serían conceptos próximos al de clave primaria. La partition key, determina como los datos son distribuidos entre los nodos mientras que la clustering key, se encarga de ordenar o agrupar los datos dentro de una misma partición. Resulta por lo tanto esencial definir estas keys de tal forma que se mejore la eficiencia de las consultas. En nuestro caso, el timestamp es un candidato claro para ser definido como primary key mientras que como clustering key, se deberá utilizar bien un identificador del sistema industrial que lo genero o bien su situación en la planta de producción.

### 6.2.2 Apache HBase

A diferencia de Cassandra, HBase es un base de datos distribuida especialmente diseñada para soportar tablas del orden de billones de filas y millones de columnas. Catalogada como No-SQL, su arquitectura se basa en el paradigma *Big Table* introducido por Google, pero construida sobre el ecosistema Hadoop y haciendo uso de HDFS como sistema de almacenamiento distribuido. Siguiendo el modelo de datos de *Big table,* la información se almacena en filas dentro de una tabla determinada. Cada fila tiene una clave de búsqueda y puede tener un número de columnas totalmente arbitrario con respecto al de otra fila almacenada en la misma tabla. A continuación, los datos son replicados a través de un número de nodos, de forma similar a como la información se distribuye en un conjunto de discos configurados en Raid. Por sus características HBase es especialmente indicada en el proyecto para la integración de un amplio número de información histórica. Fundamentalmente porque es posible realizar búsquedas de información generada en un punto temporal concreto.

### 6.2.3 Bases de Datos Relacionales

El modelo relacional es actualmente el más utilizado en el marco de las base de datos empresariales, fundamentalmente por su soporte de transacciones que cumplen con las propiedades ACID: Atomicidad, la transacción se realiza completamente o se descarta en su conjunto; Consistencia, al finalizar una transacción la base de datos se encontrará en un estado válido; Aislamiento, que evita que la base datos se encuentre en un estado invalido antes transacciones concurrentes; Durabilidad, una vez la transacción ha sido confirmada los cambios permanecen ante cualquier circunstancia.

En el marco del proyecto se harán uso de bases de datos relacionales para aquellos volúmenes de datos que se adapten a los requisitos de tamaño soportados por estas tecnologías. De esta forma las bases de datos relacionales serán intermediarias entre las bases de datos No-SQL, Cassandra y HBase, desplegadas en el proyecto. El uso de estos dos niveles de almacenamiento simplifica el posterior desarrollo de aplicaciones que hacen uso de la información ya preprocesada y filtrada desde el clúster de datos. Fundamentalmente, por el amplio soporte a nivel de *frameworks* que existe con las tecnologías relacionales al ser su grado de madurez e implantación mayor que las No-SQL.

Se barajan dos posibles tecnologías en el ámbito del proyecto para mantener los sistemas relacionales. Por un lado, MySQL, base de datos soportada por Oracle, con una amplia

utilización en el ámbito del desarrollo de aplicaciones Web y un fácil despliegue. Por otro lado, PostgreSQL base de datos de código abierto con características más avanzadas a las ofrecidas por MySQL con respecto al soporte del estándar SQL. En función de los requisitos específicos del componente a desarrollar se optará por una u otra aproximación.

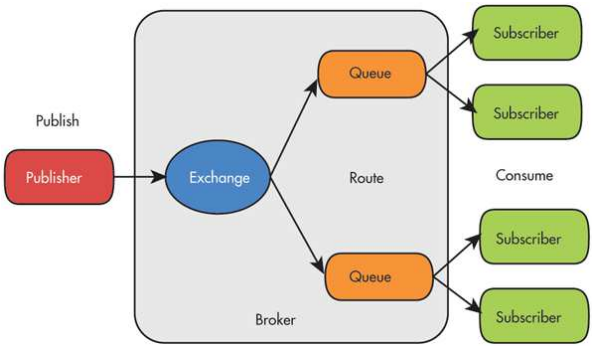
## 6.3 Comunicación

### 6.3.1 RabbitMQ (AMQP)

*Advanced Message Queying Protocol* (AMQP) es un protocolo abierto y estandarizado, características que le otorgan un papel relevante en la capa de comunicaciones en el marco de la Industria 4.0 e impulsado fundamentalmente por Microsoft. RabbitMQ es una implementación de dicho estándar, soportada por Pivotal, que hace la función de *bróker* de mensajes, es decir, recibe mensajes y los reenvía a los clientes interesados. Para garantizar la interoperabilidad el modelo de AMQ define de forma precisa la semántica de los componentes en este caso implementados por RabbitMQ. Los tres componentes principales son:

- El intercambiador o *Exchange* que recibe los mensajes de un publicador y los reenvía a colas de mensajes en función de unos criterios definidos
- La cola de mensajes (*message queue*) encargada de almacenar los mensajes hasta que son consumidos por los clientes que los solicitan
- El emparejador o *binding* que especifica la relación entre un intercambiador y la cola de mensajes

En el marco del proyecto a RabbitMQ puede ser utilizado como un middleware para gestionar la información producida por los distintos sensores y dispositivos involucrados. Delegando las funciones de comunicación a dicho *middleware* se reduce la sobrecarga de otros sistemas. Por ejemplo, mediante una instalación con cuatro servidores, dicho middleware sería capaz de gestionar unos 100 mensajes por segundo. El uso de este protocolo permite además no sólo la comunicación directa entre un publicador y suscriptor, sino que son posibles otros modos de comunicación: es posible asignar un mensaje a todas las colas asignadas a un intercambiador, asignar los mensajes a un tópico o tema específico o tener en cuenta los atributos de la cabecera del mensaje para decir donde debe ser reenviado. RabbitMQ cuenta con librerías para un amplio número de lenguajes de programación, como Python, Java o C# entre otros.

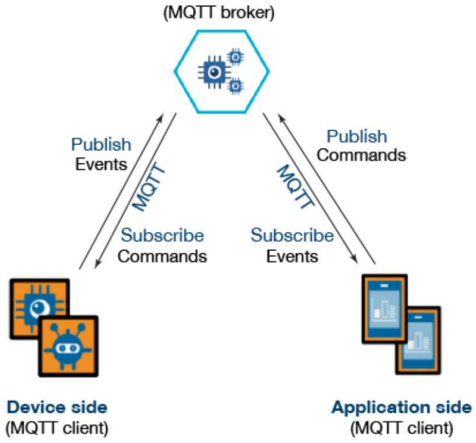


### 6.3.2 Mosquitto (MQTT)

Eclipse Mosquitto es un bróker de mensajes de código abierto que implementa la versión 3.1 del protocolo MQTT. A diferencia de RabbitMQ, Mosquitto implementa un protocolo publicar/suscribir ligero, es decir con menor necesidad de recursos. En concreto debido a que está implementado con un número de líneas de código bajo y minimizando el uso de ancho de banda de red, implica que su consumo de energía es especialmente reducido. Como referencia existen implementaciones del *bróker* del orden de 80Kb. Esta característica lo convierte en una tecnología adecuada para ser desplegada en dispositivos IoT, como sensores de potencia o microcontroladores como Arduino, y en particular, en sistemas embebidos en el marco de la industria 4.0. Mosquitto es completamente agnóstico de los datos ya que transfiere arrays de bytes.

Mosquitto implementa el patrón publicar/subscribir que permite desacoplar la comunicación entre el publicador del mensaje y el cliente o subscriptor que lo recibe. Este hecho implica que ni el publicador ni el subscriptor conocen la existencia el uno del otro. La implementación de este patrón proporciona una escalabilidad mucho mayor que la obtenida mediante una aproximación cliente-servidor. Fundamentalmente por que las operaciones en el bróker pueden ser paralelizadas hasta el millón de conexiones. MQTT utiliza un filtrado de los mensajes a través de *topics* que son los que utiliza el *bróker* para encontrar los mensajes requeridos por un cliente específico. Por último, también implementa distintas estrategias de QoS para garantizar que el mensaje es obtenido por un cliente en particular.

La utilización de Mosquitto o RabbitMQ vendrá determinada por lo tanto por el dispositivo en el cual se pueda desplegar el *bróker* de mensajes. No obstante, ambas alternativas son perfectamente compatibles en el caso de uso a desplegar, utilizando un proxy entre ambas tecnologías.




## 6.4 Analytics

### 6.4.1 Spark

Spark es un framework de código abierto para la implementación de aplicaciones que procesen *Big Data* utilizando una aproximación *Map-Reduce*. Spark introduce una serie de ventajas a la hora de desarrollar este tipo de aplicaciones que lo convierten en una solución ampliamente utilizado en el marco del *Analytics.* En primer lugar, gestiona de forma transparente el tipo de conjunto de datos a tratar, ya sea texto o grafos, y el modo de ingesta de información, en lotes o tiempo real. Su otra característica diferenciadora es el uso de la memoria principal para almacenar resultados intermedios, en vez de escribirlos en disco. Esta característica permite acelerar el procesamiento de datos con respecto a otras aproximaciones basadas en Hadoop. También permite la evaluación "perezosa" o diferida de consultas, es decir previamente a la ejecución de las consultas genera un plan de ejecución más óptimo teniendo en cuenta el contexto de procesamiento. Spark además incluye una serie de librerías que extiende sus capacidades de procesamiento más habituales. Dos de estas librerías resultan de especial relevancia para el proyecto SAIN4, que pasamos a describir a continuación:

**Spark SQL:** es un módulo para Spark que permite trabajar con los datos de forma estructurada utilizando un subconjunto del estándar SQL. Utilizando este módulo es posible acceder de forma unificada a la información almacenada en distintos repositorios como Hive o Avro, además de permitir conexiones a bases de datos a través de JDBC y OBDC. Para alcanzar un alto nivel de rendimiento incluye un optimizador basado en el modelo columnar para generar las consultas y ejecutarlas rápidamente. La utilización de este módulo en el marco del proyecto, permitirá la ejecución de consultas avanzadas para la construcción de las visualizaciones históricas que sean necesarias.

**Spark MLib:** es una librería que contiene un conjunto de algoritmos comunes de *Machine Learning* implementados sobre Spark. Los algoritmos incluyen aproximaciones de clasificación, regresión, clustering, filtrado colaborativo, reducción de la dimensionalidad, así como otras primitivas para la optimización de este tipo de tareas. MLib se utilizará en el marco del proyecto SAIN4 como base para la selección de las técnicas de *Machine Learning* más adecuadas en función de los datos y el resultado deseado.

### 6.4.2 Hive

Apache Hive es un *framework* para la lectura y escritura de grandes conjuntos de datos almacenados en un sistema distribuido basado en Hadoop o HDFS. Hive permite el acceso a dichos datos de forma similar a un sistema de *data warehouse*, con lo que a efectos prácticos su utilización es similar a la de una base de datos distribuida. Para facilitar la interacción con la información proporciona el lenguaje HiveQL, el cual ofrece una sintaxis similar a SQL que es traducida a llamadas de la API en Java ofrecidas por Hadoop o trabajos de Spark. Además, incluye la posibilidad de combinar distintos tipos de almacenamiento como la utilización de ficheros de texto plano con información en HBase.

Por sus características, Hive está especialmente recomendado para tareas de ETL y *Data Mining.* En el marco del proyecto será utilizado para la realización de consultas analíticas sobre el grueso de la información histórica que no pueda integrarse, por volumen o complejidad, en

una tecnología de base de datos. Además, es también posible exponer la información a través de un catálogo de metadatos, HCatalog, o una REST API. Utilizando esta última aproximación es posible acceder a la información de forma independiente de cómo, dónde y con qué formato específico ha sido almacenada. El uso de Hive también resulta interesante en el marco del proyecto por su integración con Spark, la cual permite realizar tareas de procesamiento de forma más eficiente que siguiendo una aproximación *MapReduce* tradicional.

# 7 Conclusiones

A modo de resumen la siguiente tabla establece la relación entre los requisitos arquitectónicos planteados inicialmente, y como la propuesta planteada aborda su cumplimiento.

**Tabla 1:** Requisitos Arquitectónicos y Soporte Propuesto

| Requisito | Soporte |
|---|---|
| **Interoperabilidad** | La incorporación del estándar OPC entre los niveles de Planta y Fábrica, garantiza que los datos se comuniquen con un estándar común. El posterior alineamiento de las aplicaciones con los esquemas de base datos garantiza la posterior interoperabilidad a nivel de aplicación. |
| **Elasticidad** | La externalización de la alta demanda de cómputo y persistencia a un nivel ajeno a la fábrica con un modelo *Cloud,* da soporte a la elasticidad. También resulta relevante el uso de un *bróker* para la distribución rápido de un gran volumen de información entre los distintos componentes. Por otra parte, el uso de varios *IIoT Data Gateway*, por cada operación industrial específica, es una aproximación fácilmente escalable. |
| **Disponibilidad** | La arquitectura descrita establece diversos puntos en los cuales la replicación es posible. Si bien el *Factory Data Gateway* es único, es posible definir una replicación del mismo para balancear la carga. Por las características inherentes al *Cloud,* este nivel también garantiza la disponibilidad. El punto crítico lo conforma los *IIoT Data Historian*, que pesar de ser PCs de carácter industrial, deberán mantenerse periódicamente si bien al basarse en el mismo software se simplifica su sustitución. |
| **Computo** | La inclusión del nivel *Cloud* introduce la externalización de esta tarea bien a una infraestructura privada de alto rendimiento, o bien a plataformas *Cloud* industriales. Mediante cualquiera de las dos alternativas las necesidades de cómputo de SAIN4 quedan satisfechas. El resto de tareas únicamente requieren de un PC industrial estándar para su realización. |

| | |
|---|---|
| **Visualización** | Para garantizar la recuperación efectiva de la información para su visualización, se han utilizado un conjunto de esquemas de datos y APIs unificados. A efectos de rendimiento, se han definido distintos niveles de almacenamiento para dar un soporte específico tanto a la visualización en tiempo real, a través del *broker*, a la visualización de datos actuales, a través del *IIoT Data Historian*, y a la visualización de datos históricos, a través del *Data Cluster*. |