UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Desarrollo de una solución para la monitorización, control e integración del sistema de suministro del pegamento en una fábrica de embalajes

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Rubén Vergara Arroyo

Tutor: Joan Fons Cors

Director experimental: Tomás Sinarcas Fernández

2020-2021

Resumen

En este documento se resuelve un problema de integración de sistemas para optimizar las líneas de producción en una empresa de embalajes, donde se pretende monitorizar el proceso de fabricación de cantoneras, en concreto el pegamento/cola que se usa en la fabricación, para poder realizar estudios de calidad del producto.

Se muestran los pasos seguidos, desde que se analiza el problema viendo a que nos enfrentamos y las restricciones que tenemos que afrontar en este ámbito industrial, pasando por el diseño de una solución adecuada que cubra los objetivos establecidos en la etapa de análisis, haciendo uso de una estrategia IAP que nos facilita la labor y su posterior implementación mediante colas de mensajería como método de comunicación, prototipos Java de los sistemas y nuevos elementos como monitores de temperatura o módulos de control del sistema.

Por otra parte, se ven también el resto de opciones sobre tecnologías y estrategias que hay actualmente para abordar este tipo de problemas y sus características, lo que ayuda a entender por qué escogemos una solución y no otra, ya que nos permite ser críticos basándonos en lo que nos ofrece cada una y lo que necesitamos cubrir.

Al final del documento se encuentra una conclusión donde se habla de los objetivos alcanzados, un posible futuro en este proyecto y ampliaciones, ya que hablamos de una empresa industrial donde se cuenta con un entorno muy dinámico donde se añaden nuevos dispositivos a menudo y una valoración personal, viendo lo que nos ha aportado la realización de este proyecto.

Palabras clave: integración, colas, mensajería, IAP, embalajes, industrial.

Resum

En aquest document es resol un problema d'integració de sistemes per optimitzar les línies de producció en una empresa d'embalatges, on es pretén monitoritzar el procés de fabricació de cantoneres, en concret la cola / cua que s'usa en la fabricació, per poder realitzar estudis de qualitat del producte.

Es mostraran els passos seguits, des que s'analitza el problema veient a que ens enfrontem i les restriccions que hem d'afrontar en aquest àmbit industrial, passant pel disseny d'una solució adequada que cobreixi els objectius establerts en l'etapa d'anàlisi, fent ús d'una estratègia IAP que ens facilita la tasca i la seva posterior implementació mitjançant cues de missatgeria com a mètode de comunicació, prototips Java dels sistemes i nous elements com a monitors de temperatura o mòduls de control de sistema.

D'altra banda, es veuran també la resta d'opcions sobre tecnologies i estratègies que hi ha actualment per abordar aquest tipus de problemes i les seves característiques, el que ajudarà a entendre per què escollim una solució i no una altra, ja que ens permetrà ser crítics basant-nos en el que ens ofereix cadascuna i el que necessitem cobrir.

A la fi del document es troba una conclusió on es parla dels objectius assolits, un possible futur en aquest projecte i ampliacions, ja que parlem d'una empresa industrial on es compta amb un entorn molt dinàmic on s'afegeixen nous dispositius sovint i una valoració personal, veient el que ens ha aportat la realització d'aquest projecte.

Paraules clau: integració, cues, missatgeria, IAP, embalatges, industrial.

Abstract

This document solves a system integration problem to optimize production lines in a packaging company, where it is intended to monitor the manufacturing process of corner pieces, specifically the glue used in manufacturing, in order to carry out product quality studies.

The steps followed will be shown, since the problem is analyzed, seeing what we are facing and the restrictions that we have to face in this industrial field, going through the design of an adequate solution that covers the objectives established in the analysis stage, making use of an IAP strategy that facilitates the work and its subsequent implementation through messaging queues as a communication method, Java prototypes of the systems and new elements such as temperature monitors or system control modules.

On the other hand, the rest of the options on technologies and strategies that currently exist to address this type of problems and their characteristics will also be seen, which will help to understand why we choose one solution and not another, and it will allow us to be critical based on what each one offers us and what we need to cover.

At the end of the document there is a conclusion that talks about the objectives achieved, a possible future in this project and extensions, because we are talking about an industrial company where there is a very dynamic environment where new devices are added often and an assessment personal, seeing what the realization of this project has brought us.

Keywords : intagration, queues, messaging, IAP, packaging, industrial.

Tabla de contenidos

Índice de figuras

1. Introducción

La integración de sistemas se define como: un proceso de construcción complejo que conecta las funciones de una organización desde diferentes sistemas, racionalizando sistemas dispares, incluyendo hardware, software (personalizado o listo para usar) y comunicaciones.[1]

Hoy en día, en casi cualquier ámbito que podamos imaginar, conviven sistemas informáticos, muchas veces de forma independiente y muchas otras en las que cooperan entre ellos para realizar tareas. En la mayoría de los casos, estos sistemas son diseñados por diferentes empresas y lo más común es que no estén preparados para comunicarse con otros sistemas, sobretodo si han sido desarrollados por corporaciones distintas. Es por este motivo por el cual, el ámbito de la integración de sistemas se ha vuelto cada vez más importante día tras día, muchas empresas necesitan conectar los sistemas con los que trabaja entre sí para continuar creciendo.

1.1 Motivación

El proyecto que se presenta viene motivado por la necesidad de la empresa Embalpack de autogestionar el proceso de producción de sus embalajes. Durante este proceso se usa cola y quieren que su temperatura sea siempre la óptima. La temperatura de dicha cola es muy importante para ellos porque repercute en la calidad del producto, llegando a invalidar unidades por falta de calidad provocando pérdidas.

Actualmente de esto se encargan los trabajadores, que están pendientes y encienden o apagan los calentadores en los depósitos. A colación de esto, se busca en un futuro acabar contando con un sistema que autogestione todos los parámetros involucrados en el proceso de fabricación.

1.2 Problemas

Dejando a un lado el problema principal, en el planteamiento de este proyecto se nos presentan otros obstáculos. Por ejemplo, hemos dicho que vamos a medir la temperatura de la cola, pero para ello debemos elegir un sensor que pueda medir

[1] Ángel Eullses Ortiz, ¿Qué es la integración TI o integración de sistemas? (2020)

dicha temperatura de tal forma que no estropee dicho sensor, ya que como sabemos la cola es pegajosa, y si entrase en contacto con el sensor lo dañaría.

Respecto al software, debemos tener en cuenta también algunas limitaciones. Hay que tener en cuenta que estas mediciones deben hacerse y actuar en consecuencia prácticamente en tiempo real, con lo que los tiempos de respuesta y comunicación se vuelven importantes.

Por otra parte, debido al entorno del problema, un entorno industrial y muy dinámico, la solución que se desarrolle deberá estar preparada para constantes cambios y deberá permitir adaptar el sistema de forma fácil a dichos cambios.

1.3 Objetivos

Los objetivos a cumplir en este proyecto son los siguientes:

El principal:

- Lograr la autogestión del proceso de producción, controlando la temperatura de la cola de forma autónoma por el ordenador.

Este es principal objetivo que queremos conseguir, pero para llegar a él, primero debemos y nos centramos en cumplir los siguientes:

- Monitorizar la temperatura en silos y puntos de aplicación (antes solo se medía en silos)
- Mostrar en un visualizador las temperaturas medidas en cada componente
- Permitir la futura implementación de componentes de forma sencilla
- Avisar mediante un indicador cuando se debe activar o desactivar los calentadores

1.4 Apartados

Los puntos que vamos a tratar a lo largo del documento son: las herramientas y estrategias que existen en el contexto tecnológico para desarrollar soluciones de integración, un planteamiento más extenso y detallado del problema, analizaremos el problema y extraeremos la información que necesitamos para abordarlo, diseñaremos una solución que se adapte y resuelva el problema que tenemos, mostraremos la implementación de dicha solución y finalmente haremos una conclusión sobre el trabajo realizado.

1.5 Metodología de trabajo

Para el Desarrollo de este proyecto hemos seguido una metodología de trabajo basada en el análisis de la situación que tenemos, un posterior diseño de la solución que desarrollaremos y finalmente su implementación. Vamos a ver con más detalle cómo se ha desarrollado paso a paso:

- Análisis: en esta etapa detectaremos cada uno de los elementos con los que deberemos trabajar, los casos de uso que deberemos cubrir y tendremos una visión más heterogénea del problema al que nos enfrentamos.
- Diseño: en la etapa de diseño iremos revisando uno a uno cada caso de uso para decidir qué estrategia y solución podemos desarrollar para cubrir todos los requisitos que nos exija. Al final de esta etapa tendremos una vista/diseño general, algo menos abstracto que el obtenido en la etapa de análisis.
- Implementación: en esta última etapa lo que realizaremos será, basándonos en el diseño obtenido en la etapa anterior, seleccionar las tecnologías que usaremos para desarrollar la solución propuesta y comenzar a implementarla. Una vez se tengan todos los componentes desarrollados y la infraestructura montada se realizan pruebas para validar el correcto funcionamiento.

2. Contexto tecnológico

Dado el hecho de que en este documento se va a ver cómo solucionar un problema de integración de sistemas, es conveniente introducir primero la estrategia de integración y los distintos tipos que existen.

Estas estrategias persiguen proporcionar un intercambio de datos y servicios entre múltiples aplicaciones empresariales de manera abierta, eficiente, fiable y segura. La integración de aplicaciones se ve impulsada por ciertos factores, que son:

- El auge de internet
- La demanda de soluciones basadas en COTS
- Integradores de soluciones
- Integración cadenas de suministro (supplychain)

- Integración de frontendsy backends(los sistemas legados no se pueden desestimar)
- Interés de los clientes
- Agilidad de negocio

La integración de aplicaciones ha sido un reto, han ido apareciendo tecnologías que permiten albergar infraestructuras, plataformas y software en diferentes localizaciones, los requisitos de negocio cambian constantemente y los métodos tradicionales de integración no cumplen los nuevos retos que han aparecido o no son lo suficientemente ágiles.

La integración de aplicaciones es compleja, pero necesaria, y correctamente usada puede proporcionar un valor estratégico y técnico muy alto.

2.1 Estrategias de integración

Dentro de las estrategias de integración, encontramos de distintos tipos, diferenciados por el nivel en el que se aplican o por la escala del problema a solucionar.

Si nos centramos primero en la diferenciación por niveles, encontramos estos tres tipos:

- **Integración de datos:** pueden ser mediante transferencia de ficheros, la forma más simple de integración, adecuada cuando las actualizaciones no son muy frecuentes y requiere consensos para el intercambio, o pueden ser bases de datos compartidas, que proporciona un intercambio rápido de datos, se imponen fuertes requisitos de cooperación, es un modelo más rígido y menos flexible y su extensibilidad técnica es viable, pero en la realidad es compleja.
- **Integración de procedimientos remotos:** en este caso se aplica el principio de encapsulación escondiendo los datos para mantener su integridad, requiere que se publiquen y se conozcan interfaces de procedimientos, pueden ser desarrollados como “capas” sobre software ya existente. Cabe mencionar que los servicios web proporcionan estándares abiertos, como SOAP.
- **Mensajería:** es una estrategia que proporciona alta velocidad, una comunicación asíncrona, es confiable, permite desarrollar arquitecturas desacopladas, se necesita un consenso para los mensajes y es fácilmente extensible.

Hemos visto los distintos tipos de estrategias que hay según el nivel al que se aplica. Ahora pasamos a comentar los tipos según la dimensión del problema a solucionar:

Integración a pequeña escala

- **Ad-hoc:** con esta estrategia se desarrollan componentes específicos y dependientes de la solución. Es adecuada para problemas sencillos de los que se tiene cierta garantía de que no escalarán. No requieren de mucha infraestructura para funcionar.
- **Colas de mensajería:** la comunicación se realiza mediante un intermediario, un middleware, encargado de dichas comunicaciones. Es de las soluciones más usada, se puede aplicar en pequeñas y en grandes soluciones, es escalable. Permite desacoplar productores y consumidores de mensajes y ofrece mecanismos para una comunicación asíncrona de manera natural. Por otra parte, tiene como pega la necesidad de consensos en los formatos de datos.

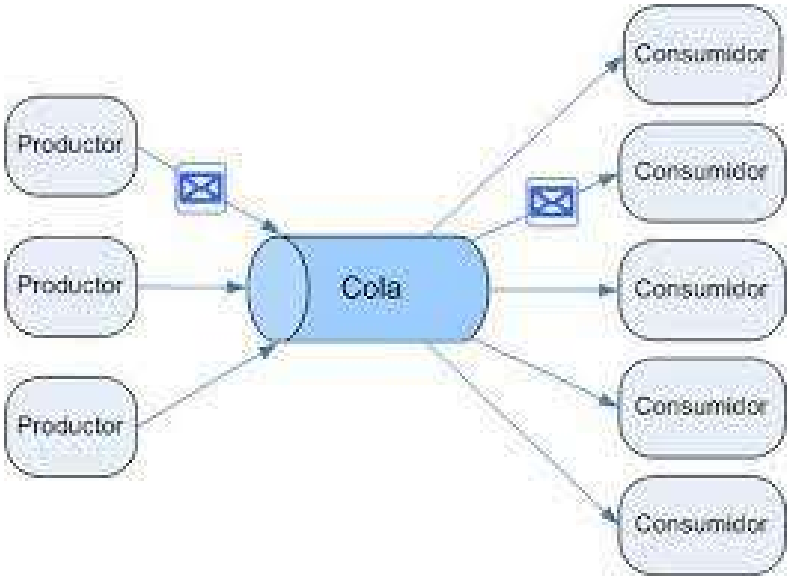

Ilustración 1: esquema funcionamiento cola de mensajería

Integración a gran escala

- **Enterprise Application Integration:** se basa en la aplicación de principios arquitectónicos para resolver problemas complejos de integración. Lo consta un conjunto de estrategias, planes, herramientas, prácticas y guías que permiten coordinar diferentes aplicaciones empresariales. En esta estrategia se hace uso de frameworks, middleware y componentes de integración.

Esta estrategia nos ofrece ciertos beneficios: permite compartir información entre aplicaciones y mantener los datos consistentes, reduce potencialmente el panorama tecnológico, ya que reduce la heterogeneidad y agiliza y facilita el despliegue de una aplicación nueva o actualizada.

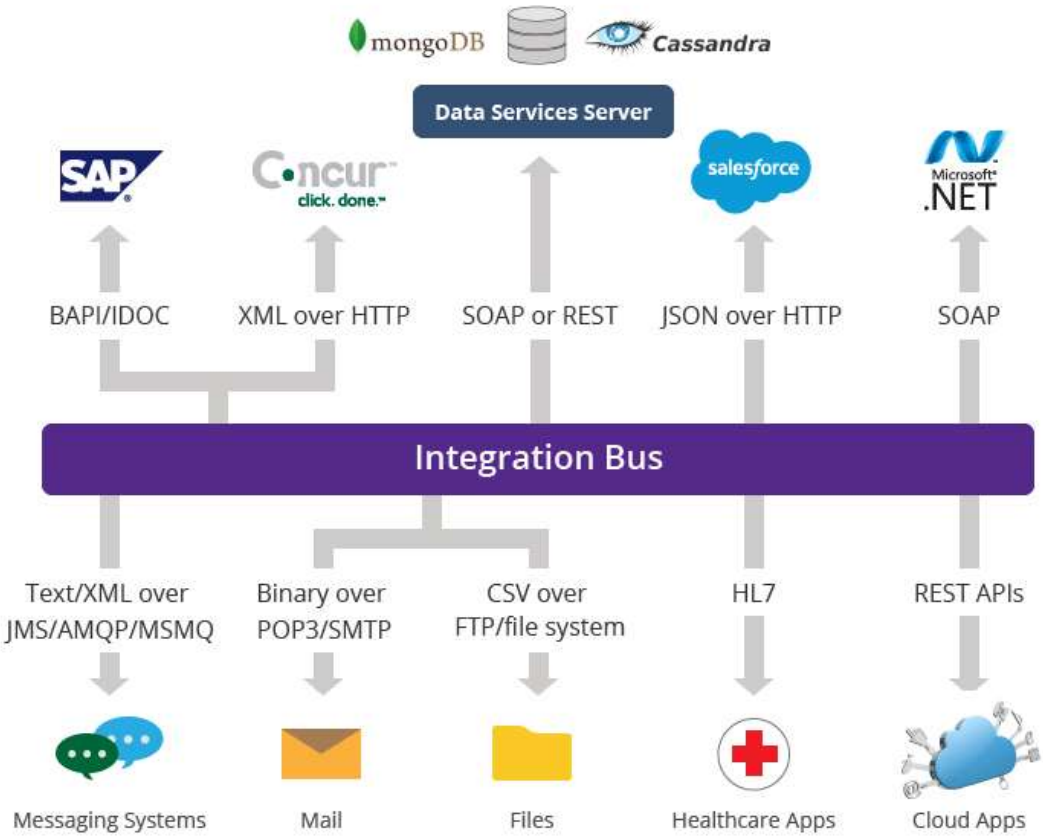

Ilustración 2: esquema arquitectura EAI usando ESB

2.2 Pros y contras de la integración

Como hemos comentado antes, la integración de sistemas es un ámbito muy importante hoy en día. Nos permite desarrollar soluciones de comunicación y ERPs de forma más escalable y organizada.

Antiguamente esto no se planteaba, ya que las aplicaciones que se usaban en las empresas eran menos, más simples y muchas veces desarrolladas internamente a medida. Ha sido por esta razón y por las ventajas que ofrece a las empresas que lo implementen, como por ejemplo aumentar su productividad, optimizar su gestión de datos, reducir gastos a largo plazo y mejorar su servicio al cliente.

Pero como todas las cosas, no todo es bueno. Por otro lado, hay que tener muy en cuenta la seguridad, ya que se está mandando información constantemente entre máquinas e implantar dicha integración conlleva un alto gasto inicial para su implementación[2].

2.3 Tecnologías existentes

Hay varias formas de implementar dicha integración, pero las más comunes son mediante el uso de APIs o colas de mensajería. Veamos un poco más en detalle cada una.

API[3]

Una API es la interfaz de programación de aplicaciones, es decir, de la misma manera que existen interfaces de usuario que nos permiten interactuar con sistemas como móviles, ordenadores, etc… Existen interfaces para que programas/aplicaciones puedan interactuar con otras. Encontramos varios de tipos de APIs, APIs web, librerías, clases Java… Pero las que usamos para integrar sistemas son las APIs web, usadas cuando el enfoque es ofrecer servicios. A continuación, comentaremos los dos tipos más usados de APIs web.

- **REST**: sigue los principios RESTful, una arquitectura de transferencia de estado representacional. Usa una arquitectura cliente/servidor sin estado, basada en HTTP. Implementa cada operación CRUD (create, read, update, delete) mediante las operaciones ya existentes de HTTP (POST, GET, PUT, DELETE). Los recursos están representados mediante URIs, que son autodescriptivas, ayudan a saber qué operación realiza dicho recurso. Por ejemplo, un posible recurso GET /alumnos seguramente lo que hará será devolver una lista de alumnos. La misma URI, pero con distinto método, en este caso crearía un nuevo alumno, POST /alumnos. Se suele usar formato JSON para transferir los mensajes, aunque en ocasiones se usa XML.

[2] Chloe Henderson, 4 tipos de Integración de sistemas-Pros frente a contras de cada método (2020)
[3] Anónimo, Tipos de API: Web API, API de código, API heredada y de desarrollo

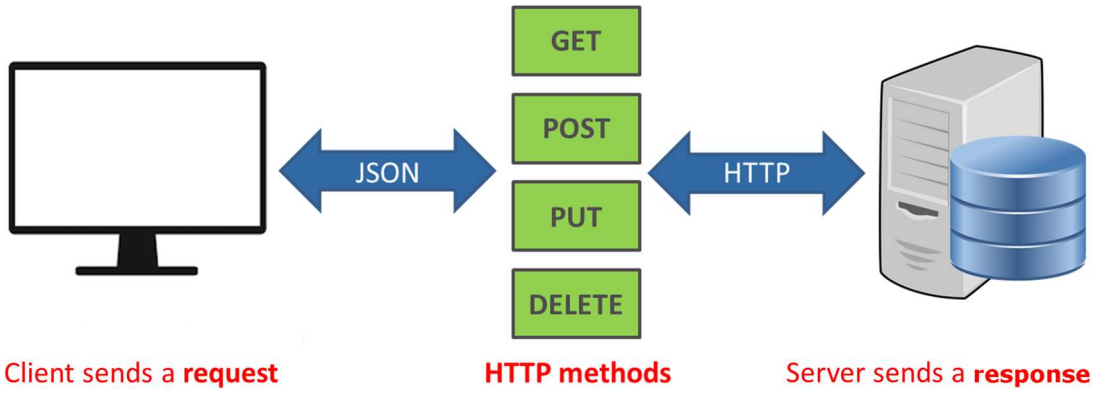

Ilustración 3: arquitectura API REST

- **SOAP**: son las siglas de Protocolo Simple de Acceso por Objetos. Es un protocolo liviano para intercambiar información. Va asociado al lenguaje WSDL, basado en XML, que se usa para describir que servicios ofrece dicha API SOAP. WSDL es leíble por la máquina y por las personas. De esta forma obtienes el WSDL, averiguas que servicios ofrece dicha API y mediante peticiones HTTP con mensajes XML (definidos también en el WSDL) a las direcciones que indica dicho documento consumes los servicios ofrecidos.

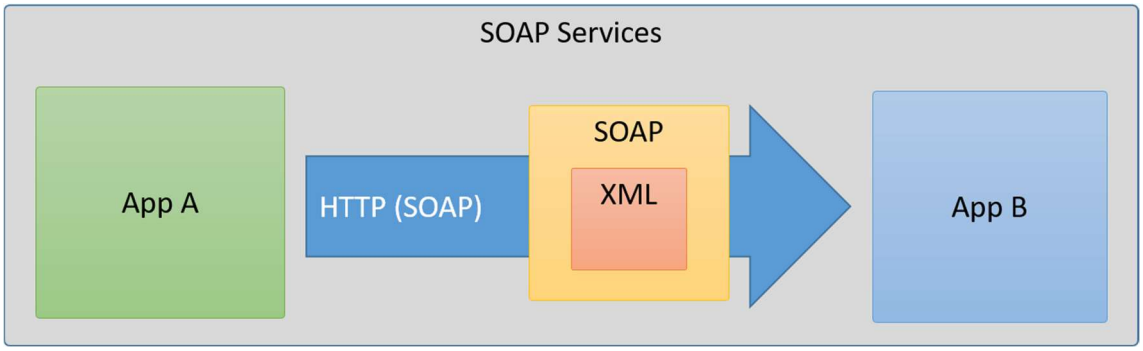

Ilustración 4: arquitectura API SOAP

Colas de mensajería[4]

El uso de colas de mensajería es una forma de comunicación asíncrona, a diferencia de la anterior tecnología que requiere que todos los elementos estén activos a la vez para funcionar.

[4] Amazon, Colas de mensajes: Mensajería asíncrona para ejecutar trabajos por lotes y desacoplar aplicaciones

No siguen la arquitectura cliente/servidor, existe un bróker que es el encargado de gestionar las distintas colas a las que llegan mensajes y existen los dispositivos productores o consumidores que se conectan a dichas colas. Los mensajes que se envían desde los productores se almacenan en la cola hasta que un consumidor lo procesa y se elimina. Cada mensaje se procesa una sola vez. Aquí se puede observar cómo trabaja de forma asíncrona.

Esta tecnología permite un fuerte desacoplamiento, en componentes pequeños más fáciles de mantener. Los mensajes se pueden enviar con el formato que se desee; JSON y XML son los más comunes.

3. Planteamiento del problema

En la introducción de este documento hemos descrito el problema que se nos presenta de forma breve. A continuación, vamos a ver todas las dificultades que se nos presentan de forma más detallada, como la dificultad de la instalación de dichos sensores de temperatura, el software que vamos a utilizar, los tiempos de respuesta...

Ilustración 5: cantoneras de cartón

Ilustración 6: bobina de papel usado en la fabricación

Ilustración 7: cola usada en la fabricación

Para poder profundizar en nuestro problema, es conveniente empezar desde la raíz de este. Partimos de Embalpack, empresa que se dedica a la fabricación de embalajes y cantoneras, donde hay varias líneas de producción. En este proceso de fabricación, se usan materiales como rollos de papel que se prensan entre ellos para formar dichas cantoneras, y cola, que se aplica entre capa y capa de este papel para que queden pegados y no se desmonten una vez terminados.

Como es obvio, al igual que en otras empresas, lo que se pretende siempre es poder producir lo máximo posible con el menor coste y la mejor calidad, y para ello es necesario tener controlado ciertos aspectos en el proceso de producción. En este caso, tras haber estado en la planta de producción y haber observado cómo funciona y haber hablado con el supervisor, sacamos como conclusión que el elemento más importante a la hora de fabricar la cantonera es la cola y la temperatura a la que se aplica.

La cola realiza un recorrido por la línea de producción: se tiene almacenada en un silo con unos calentadores para controlar la temperatura y desde el cual se rellena una balsa de rebose, que cuenta también con unas resistencias y está justo debajo de los difusores. Dichos difusores aplican la cola al papel que va pasando por la línea y la balsa recoge la cola sobrante para reaplicarla hasta que dicha balsa se va vaciando y se rellena con el silo.

Ilustración 8: línea de producción cantoneras y silo de cola

Actualmente, como hemos comentado, la temperatura se puede regular desde silo y balsa pero solamente tienen monitorizada la temperatura del silo. Con esta medición no les basta, ya que la temperatura de la cola en el silo no es la misma que en el momento de aplicación ni es la misma que en la balsa porque mientras ésta va viajando se va enfriando, y aquí es donde aparece nuestro problema: poder medir la temperatura de la cola en estos 3 puntos.

Sí es cierto que el objetivo final es poder tener la temperatura todo el rato en el rango óptimo, pero como objetivo en este proyecto vamos a diseñar e implementar la monitorización de la temperatura en estos 3 puntos y una señal de activación y desactivación de resistencias, dejando ya despejado el camino para la futura implementación de un sistema de autorregulación o más líneas de producción.

Una vez puestos en contexto, podemos pasar a ver cuáles son cada una de las dificultades que se nos presentan.

Respecto al hardware, solo nos tenemos que preocupar de cómo vamos a medir la temperatura, ya que estamos trabajando con un fluido viscoso y pegajoso que si entrase en contacto con un elemento electrónico lo acabaría dañando, por lo que debemos buscar una solución que permita medir este parámetro ya sea sin contacto directo o totalmente sin contacto.

Donde verdaderamente se nos presentan dificultades es en el software. La solución que debemos diseñar debe recopilar y mostrar los datos prácticamente a tiempo real, por qué una temperatura incorrecta puede estar provocando pérdidas en la producción por lo que se debe poder reaccionar de forma rápida para solucionarlo.

Sabiendo esto, debemos buscar una solución que permita estar a la espera de nuevos datos que se van generando constantemente y que a ser posible sea ligera, ya que habrá un considerable flujo de mensajes. Aparte de estos dos requisitos, también hemos comentado que debe dejar el “ecosistema” abierto a posibles nuevos elementos, ya que una empresa puede crecer e incorporar más dispositivos, por lo que nuestra solución debe ser fácilmente escalable.

Por ejemplo, el futuro sistema de autorregulación será un nuevo elemento que se comunicará con el resto de sensores para poder decidir cuándo activar o desactivar las resistencias del silo o la balsa, las cuales a su vez serán otros elementos. Todos estos dispositivos que he mencionado se deben poder conectar y comunicar con el resto sin mucha dificultad. Este se podría decir que es el objetivo con más importancia del proyecto y al que más atención debemos prestar.

Una vez hemos visto el problema con más profundidad habiendo partido de la raíz, tenemos una idea más clara de a lo que nos enfrentamos. El siguiente paso es diseñar una solución que se ajuste a todas las condiciones que hemos comentado. En el siguiente apartado veremos la solución que hemos adoptado y por qué la hemos elegido.

4. Análisis

En puntos anteriores del documento hemos visto de forma breve el problema al que nos enfrentamos y posteriormente lo hemos visto con algo más de detalle, incluso con algunas pinceladas de los elementos y operaciones que contiene. En el siguiente apartado vamos a analizar el problema en profundidad para descomponerlo y poder abordarlo de mejor forma.

Viendo a lo que nos enfrentamos se pretende desarrollar una solución que capture datos de diferentes dispositivos de soporte a la producción, en este caso concreto, se tratan de los relacionados con el sistema de cola/pegamento. Estamos trabajando en un entorno cambiante, por lo que la solución que diseñemos deberá estar preparada para poder adaptarse a los cambios y ser flexible respecto a:

- Nuevos dispositivos que puedan aparecer en el sistema en un futuro
- Diferentes tecnologías que se usen dentro de la empresa
- Y a la relación entre los sistemas informáticos de la empresa, ya que no queremos introducir dependencias fuertes entre los sistemas (la solución no puede estar diseñada pensando en un par de sistemas y su comunicación, esto desembocaría en acabar desarrollando soluciones para cada comunicación entre sistemas)

Dentro de la problemática identificamos diferentes necesidades a cubrir:

- Queremos conocer la temperatura de la cola en diferentes puntos del sistema para poder relacionarla con la calidad obtenida en cada orden de producción (los puntos son silo, balsa, difusor…) De momento solo se monitoriza en el silo pero no queda registrado en ningún sistema informático
- Auto-gestionar el control de la cola, ya sea la temperatura, el nivel de llenado en silos y balsas… Que se activen los procesos necesarios para ello.

- Visualizar en todo momento la situación de producción (temperatura, nivel, calidad…) Esto puede ser visualizado en distintos dispositivos, ya sea una ordenador mediante una web, una aplicación móvil, o visualizarse por niveles, por ejemplo visualizar solo datos sobre una línea específica, solo sobre los silos, visualizar la fábrica entera…
- Tomar decisiones en base a los datos que recopilemos: ajustar temperaturas y demás, todo relacionado con el control de calidad y gestión de recursos

Todos estos escenarios los veremos con más detalle junto a los elementos involucrados en cada uno y sus interacciones, pero primero hagamos un vistazo general a los elementos que podemos detectar.

Si recordamos lo que hemos ido comentando a lo largo del documento, podemos detectar ciertos componentes: contamos con varias líneas de producción donde cada una cuenta con un silo que almacena cola para ir rellenando la balsa, una balsa donde rebosa la cola aplicada al papel y de donde se va reaplicando al papel hasta que se vacía y cuatro difusores que aplican la cola al papel que pasa por la línea. Nos vamos a ocupar en este proyecto de solamente una línea, por lo que en total tenemos seis elementos de momento.

Cabe recordar también que se quería autorregular el sistema mediante un controlador que regulase las temperaturas de forma autónoma, el cuál desarrollaremos. Éste será otro dispositivo, con lo que ahora tendremos siete componentes. Por último, necesitamos también un sistema de visualización de temperaturas, a desarrollar también, para que los trabajadores sepan en todo momento con qué temperatura de cola están trabajando y poder establecer relaciones temperatura/calidad en el producto. En total nos encontramos con ocho dispositivos. Como cada elemento cuenta con su calentador para la cola aparte de sus sensores, los contamos por separado y subimos el número de elementos a catorce.

Vemos que hay bastantes elementos y que se van a comunicar de diferentes formas entre ellos. Llegado el momento de plantearse como se puede “montar” una solución que cubra las necesidades mencionadas seguimos una estrategia IAP que nos va a permitir independizar los sistemas entre sí, hacer la comunicación entre ellos más fácil gracias al desacople que ofrece y definir una solución escalable y flexible que es lo que buscamos.

Para ello basamos nuestra solución en el intercambio de mensajes, lo que nos permite que cada sistema funcione por su cuenta internamente sin estar ligando

estrechamente con los demás. El único “acuerdo” que existirá entre los sistemas será el formato que va a tener la información intercambiada. Se puede llegar a proponer usar un formato canónico, como por ejemplo JSON, con el que todos los sistemas se comuniquen, aunque internamente trabajen con otros formatos totalmente distintos.

Usando una solución basada en el intercambio de mensajes siguiendo la estrategia IAP conseguimos de forma sencilla y sin esfuerzo, solamente por plantearlo de esta forma, una solución desacoplada, escalable y flexible, debido a lo comentado, cada sistema funciona de forma independiente de los demás, solamente se comunican usando un acuerdo establecido entre ellos. Además, haremos uso del sistema central que ya usa la empresa como ESB, punto único donde se conectarán todos los elementos para comunicarse, como si de una tubería central donde viajasen los mensajes se tratase.

Si nos propusiésemos desarrollar una solución siguiendo un estilo más convencional donde se desarrollase un middleware y soluciones para conectar cada sistema entre sí, acabaría convirtiéndose en algo caótico, con multitud de middlewares activos, haciendo que fuses inmantenible e inescalable. Imaginemos que contamos con cuatro sistemas; habría que desarrollar seis middlewares para que se pudiesen comunicar todos con todos. Se nos quedaría una estructura parecida a la mostrada en la siguiente imagen:

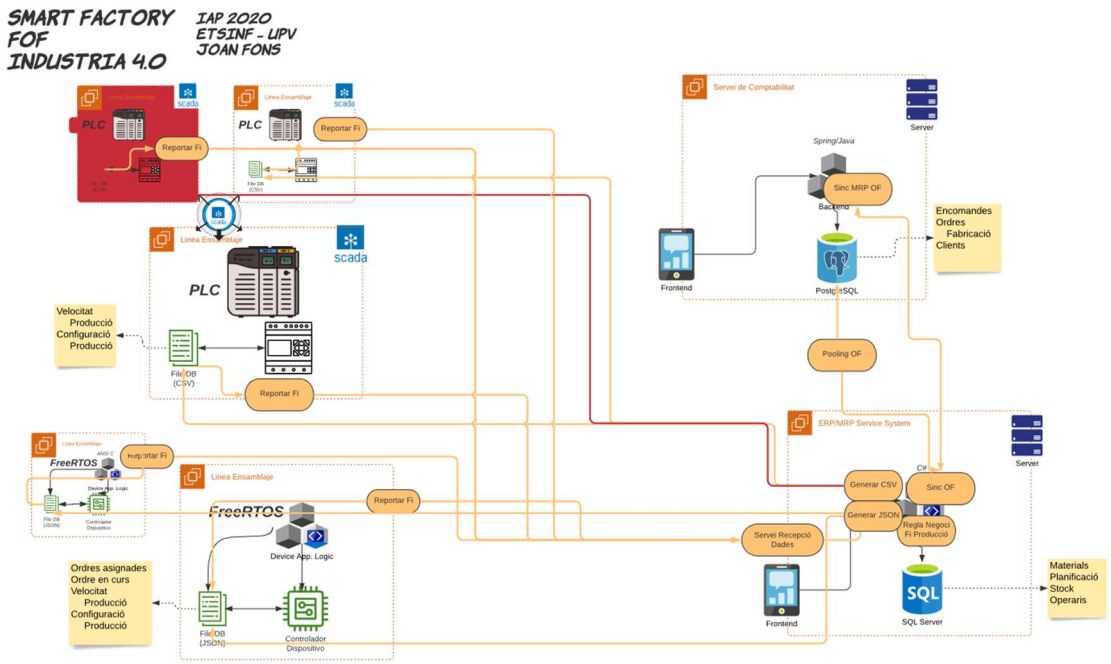

Ilustración 9: esquema ecosistema caótico

Esta imagen se ha obtenido de los recursos ofrecidos en la asignatura de IAP. Se nos presentaba y se presenta en este documento como un contra-ejemplo de lo que se pretende conseguir y lo que se persigue mediante la aplicación de estrategias de IAP para la resolución de problemas de integración. En la ilustración se observa un ecosistema caótico, algo totalmente opuesto a un ecosistema organizado y estructurado, justo lo que perseguimos desde el comienzo del documento y lo que se persigue desde el punto de vista de las estrategias IAP.

4.1 Conocer la temperatura

En este caso lo que se busca es conocer, como hemos dicho, la temperatura en los diferentes puntos del sistema. A cada momento se están fabricando cantoneras y mediante unos sensores instalados en puntos estratégicos queremos saber cuántos grados tiene el pegamento en todos los puntos (silo, balsa y difusores) y poder asociarla a cada cantonera fabricada.

De momento solo se conoce en el silo y dado que la temperatura del pegamento va cambiando según viaja entre contenedores o hasta los difusores no se puede conocer con exactitud el valor exacto en cada punto. Es por eso que se necesitan instalar estos sensores para que generen mediciones asociables a la producción.

En este escenario participan el silo, la balsa y los difusores, o mejor dicho los sensores de cada uno. Estos sensores envían mensajes, en cuya información consta la temperatura, al ERP de la fábrica y de ahí se distribuyen a los elementos que necesiten las mediciones.

Para llevar a cabo este proceso, los elementos que actúan como productores son el sensor del silo, la balsa y los difusores. Estos sensores medirán la temperatura de la cola y el sistema de la línea al que están conectados creará un mensaje con la temperatura. Dicho mensaje llegará al ERP de la fábrica donde será distribuido a los dispositivos que requieran de esta información.

En la siguiente imagen se ve el proceso de comunicación en este caso de uso:

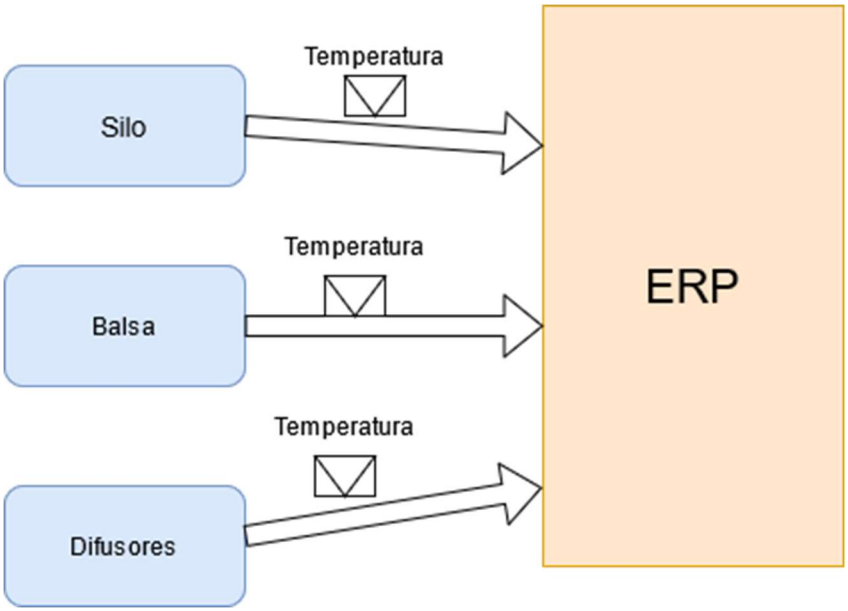

Ilustración 10: esquema escenario reporte temperaturas

4.2 Auto-gestionar el control de la cola

En este escenario lo que se persigue es conseguir que las acciones que se realizan manualmente ahora, relacionadas con el control de la temperatura del pegamento o el nivel de llenado en balsas y silos, los cuales son regulados mediante la activación de calentadores en los depósitos y bombas de succión usadas para trasladar pegamento de un contenedor a otro, se realicen de forma autónoma.

Esto se conseguiría mediante la instalación de unos activadores en cada elemento, los cuales se activasen, valga la redundancia, y llevasen a cabo la operación necesaria antes ejecutada a mano por los trabajadores. De esta forma se ahorra tiempo al personal y lo pueden dedicar a otras labores.

Como elementos presentes en este caso de uso tenemos, como hemos mencionado, a los actuadores encargados de activar y desactivar bombas o calentadores y al ERP de la fábrica como en el caso anterior. Dichos actuadores funcionarán en base a los mensajes que reciban del ERP con órdenes de activación o desactivación.

El proceso se lleva a cabo de la siguiente manera: en el ERP llegarán mensajes con órdenes de activación o desactivación. Estos mensajes se distribuirán a los elementos de regulación correspondientes que activarán los elementos de control.

A continuación, tenemos una imagen que ilustra este proceso:

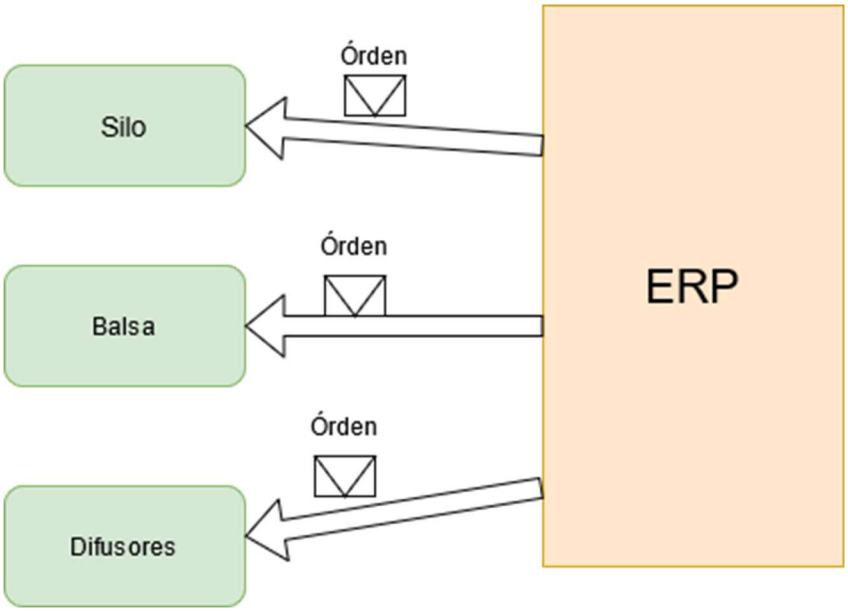

Ilustración 11: esquema escenario auto-gestión

4.3 Visualización

Otro punto que cubrir es la visualización del estado de producción en cualquier momento por los trabajadores. Se quiere conseguir que se pueda tener controlado y monitorizado el proceso de producción y todos sus parámetros, en este caso la temperatura de la cola, para de esta forma tener constancia de dichos parámetros en el momento de producción de cantoneras y de esta forma poder relacionar calidades con temperaturas.

Buscamos que pueda ser visible en distintos dispositivos, ya sea una web, en un móvil, en los propios sistemas de producción…

Se puede plantear también una monitorización por niveles: se puede tener monitorizado por líneas de producción, por elementos, por ejemplo, solo temperaturas en silos o difusores, o a nivel general de planta, para tener una vista de cómo está funcionando todo.

En dicho proceso encontramos como elementos el ERP de la fábrica, siempre presente, y los posibles sistemas de visualización (ordenadores, móviles, etc). Estos elementos de visualiación tendrán que ser desarrollados cuando se lleve a cabo la implementación de este escenario en la solución.

Este proceso funciona de manera que el sistema/s de visualización actúan como consumidores. Reciben los mensajes con las temperaturas del ERP y muestran por pantalla las lecturas realizadas junto al elemento al que corresponden.

En la siguiente ilustración tenemos el escenario de visualización:

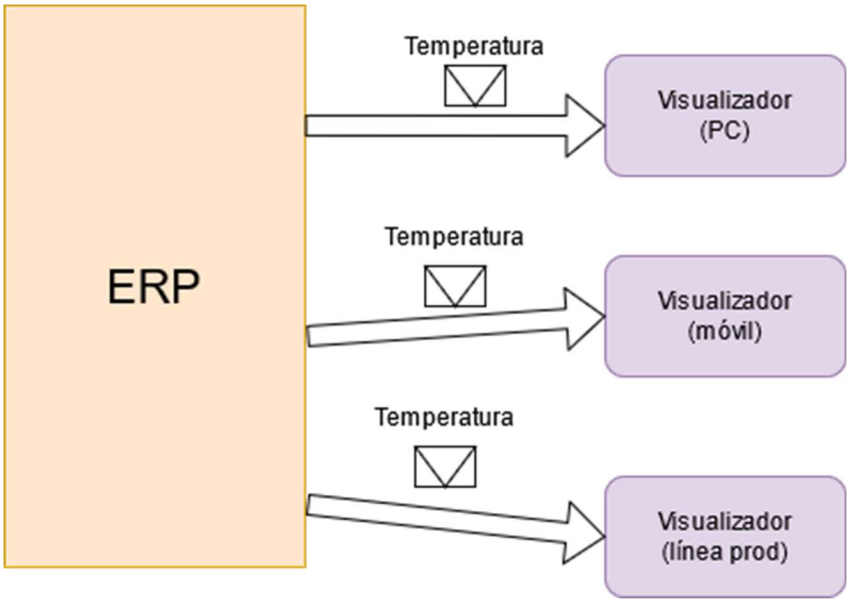

Ilustración 12: esquema escenario visualización

4.4 Toma de decisiones

Como último escenario tenemos el de toma de decisiones. En él lo que se quiere conseguir es, en base a unos parámetros establecidos por la empresa, ajustar la temperatura del pegamento en los depósitos existentes, balsas y silos, para tener siempre la temperatura en el rango óptimo de temperaturas establecido.

Hemos visto ya el escenario de conocer la temperatura en los puntos clave del proceso de producción y el escenario de auto-regulación del pegamento. Bien, estos dos procesos van ligados a este, ya que para poder tomar decisiones se necesitan mediciones para analizar y actuar en consecuencia, en este caso enviando órdenes.

Como elementos tenemos, como en todos los anteriores, al ERP de la fábrica y al módulo de control que desarrollaremos en el momento de la implementación, el cuál se encargará de generar las órdenes de activación y desactivación según considere, en base a las temperaturas procesadas.

El proceso de toma de decisiones se lleva a cabo de la siguiente forma: del ERP recibe el módulo de control mensajes con mediciones de temperaturas, actuando de esta forma como consumidor. Leerá dichos mensajes y los procesará y en base a la decisión tomada creará un mensaje con una orden, para activar o desactivar un elemento de control. Dicho mensaje con la orden lo enviará al ERP, desde el cual será distribuido al dispositivo correspondiente, actuando ahora el módulo de control como productor.

En la siguiente imagen vemos el proceso:

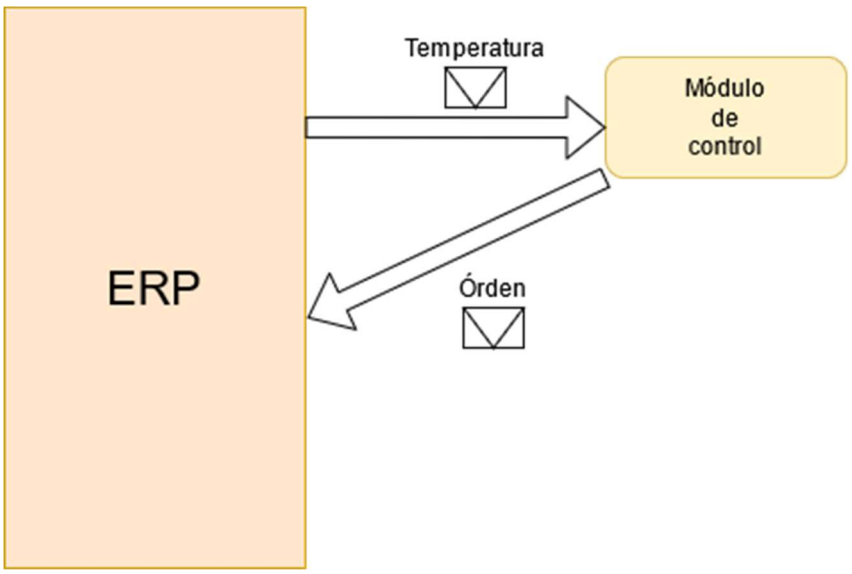

Ilustración 13: esquema escenario toma de decisiones

4.5 Diagrama de clases

Como hemos podemos ir vislumbrando, de todos estos componentes mencionados, algunos producirán mensajes y otros los recibirán para procesarlos, y en algunos casos harán las dos funciones; tendremos productores y consumidores, lo que nos lleva a ver cómo van a comunicarse e interactuar unos con otros.

¿Van a necesitar estar ambos conectados al mismo tiempo o si falta alguno, por ejemplo porque se ha tenido que reiniciar, no hay problema en que se pierda la información enviada? Y, por otra parte, ¿los mensajes que se envíen serán iguales para todos? ¿Serán distintos? ¿Cuál será su contenido y qué formato utilizaremos?

Muchas de estas cuestiones surgen a raíz de la planificación que hemos ido haciendo, relacionadas con el tipo de comunicación entre elementos y la información a enviar entre ellos. Son algunas de las varias decisiones que vamos a tener que ir tomando a lo largo del diseño de nuestra solución en el próximo apartado.

De momento, y para concluir este apartado de análisis, una vez hemos analizado el problema y hemos extraído todos los elementos con los que vamos a trabajar, los casos de uso que hay que cubrir y las comunicaciones que se producirán entre ellos, podemos realizar un diagrama de clases para ilustrar de forma clara y resumida todos los componentes que hay y cómo interactúan entre ellos.

A continuación encontramos el diagrama de clases del problema:

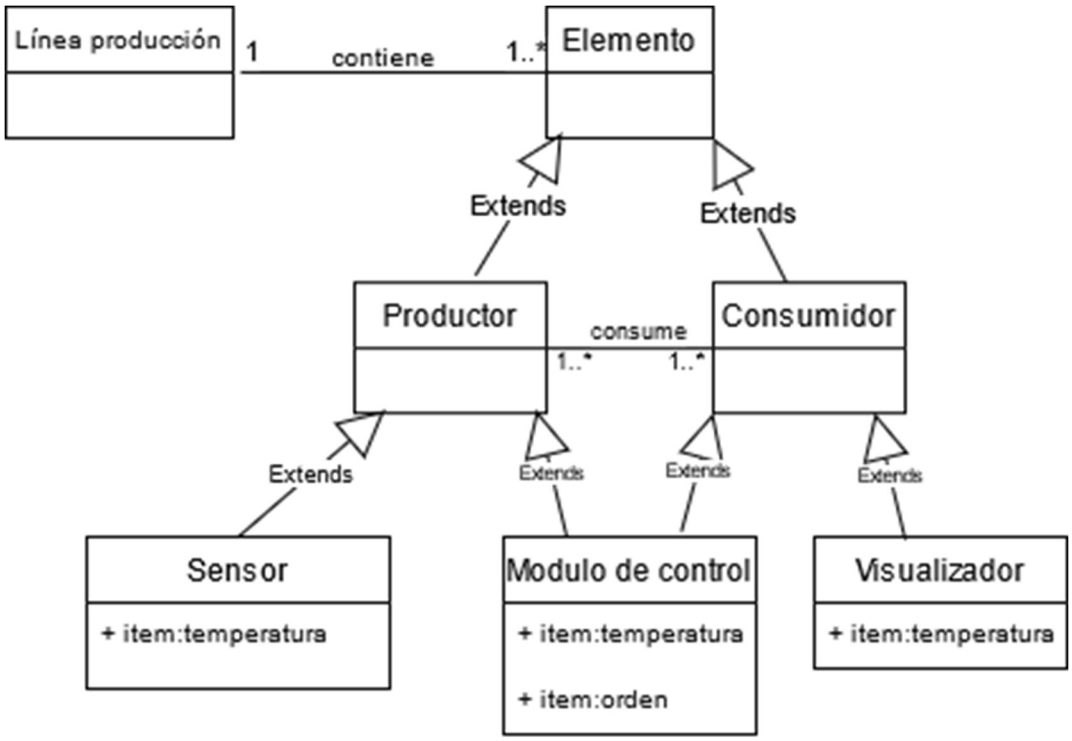

Ilustración 14: diagrama de clases del problema

Vemos según el diagrama de clases mostrado que contamos con líneas de producción, que a su vez contienen elementos. Dichos elementos, como hemos visto y como vemos reflejado, pueden ser de dos tipos: consumidor de mensajes o productor de mensajes. Entre ellos se comunicarán mediante dichos mensajes, pudiendo un consumidor obtener mensajes de uno o varios productores y un productor pudiendo suministrar mensajes a uno o varios consumidores.

Dentro de los elementos cuyo papel sea ser productor nos encontramos con los sensores de cada dispositivo en las líneas, como silo, balsa o difusores, y al módulo de control, en su fase de creación de órdenes.

Como elementos de tipo consumidor, observamos al visualizador y al módulo de control de nuevo, esta vez en la fase en la cual obtiene las mediciones para tomar decisiones de activación o desactivación.

5. Diseño

Hemos visto el problema que se nos presenta y en el apartado anterior lo hemos analizado, “desmenuzando” toda la problemática en pequeños escenarios, con los elementos que participan y sus comunicaciones. Hemos visto también cómo vamos a enfocar nuestra solución y ahora que contamos con esta información podemos diseñar dicha solución con las estrategias mencionadas para que cubra todas las necesidades y objetivos del proyecto.

Este apartado lo vamos a organizar de forma parecida al anterior: veremos desde una visión general cómo vamos a ir diseñando la solución y las decisiones que vamos a ir tomando respecto a las cuestiones que nos surgían en el apartado de análisis y después iremos escenario por escenario viendo como los resolveremos para finalmente ver una visión general de nuevo pero esta vez con todo resuelto.

5.1 Introducción general de la solución

La solución que proponemos es simple: contamos con elementos productores de mensajes, en este caso los sensores de temperatura y el módulo de control, los cuales envían datos a un servidor central ESB.

Estos mensajes podrán sufrir o no alguna transformación por el camino, no lo sabemos de momento, pero debemos estar preparados para ello y dejar la solución abierta a esto.

Este servidor central ESB es el encargado de comunicar todos los elementos entre sí, como habíamos mencionado en la fase de análisis cuando comentábamos que se iba a usar una estrategia basada en el intercambio de mensajes mediante una “tubería central”.

Por otro lado, contamos con componentes consumidores de mensajes, que recogen los datos del ESB, que pueden volver a sufrir alguna transformación. Estos dispositivos consumidores son el sistema de visualización, el módulo de control y los actuadores de las resistencias.

Como se fue comentando en la fase de análisis, es útil seguir una estrategia de IAP (integración de aplicaciones) ya que nos aporta una escalabilidad que da mucho valor a nuestra solución, porque como ya sabemos, estamos un producto para un ámbito industrial el cual cambia constantemente, ya sea en el número de máquinas que se usen o de sistemas que se instalen. Sin decir más, en un par de meses desde que hicimos el primer análisis en la visita a la fábrica ya ha habido algunos cambios, con lo que si en este corto periodo de tiempo ha habido cambios, de aquí a unos años ni nos lo podemos imaginar.

Es por eso, que, siguiendo esta estrategia, definiendo unos “contratos” de comunicación y unas pautas y estructuras, conseguiremos que cada aplicación funcionando en la empresa trabaje de forma independiente de las demás, y a la hora de comunicarse con otras sepa como hacerlo gracias a este ecosistema escalable que habremos diseñado. De esta forma, cuando otro sistema se quiera acoplar al ecosistema de la fábrica, solamente habrá que “enseñarle” la forma con la que se comunicará con los demás, y lo más importante, será la misma forma para todos.

Es por eso por lo que es tan importante enfocarlo de esta manera y no como soluciones personalizadas para cada máquina, ya que si fuese así, a la hora de instalar una nueva máquina se debería desarrollar tantas soluciones como sistemas a los que queramos conectarla, y no es difícil imaginar que se convertiría en un caos imposible de mantener en el tiempo.

Teniendo ya claro con que perspectiva lo vamos a diseñar, deberemos pensar en la forma en que se van a comunicar, el formato de datos que van a usar, si va a existir un formato de datos canónico… Todo esto vamos a ver como lo solucionamos en los siguientes párrafos.

Ya hemos resuelto una de las cuestiones que se nos presentaban en el anterior apartado: los elementos se van a comunicar a través de un intermediario usando una estrategia IAP, pero no de tipo “ad-hoc” donde se diseñan componentes hechos a medida para esta solución en concreto, usaremos una solución de gran escala, basada en EAI (Enterprise application integration), donde se usa un mediador central.

Esto va a ser así porque, como hemos mencionado ya antes en le párrafo anterior, hacer que se comunicasen directamente entre ellos de forma personalizada es totalmente inviable. Esa solución se puede aplicar a un ámbito muy pequeño y cerrado, donde no se tenga una visión de futuro con ampliaciones, pero no es nuestro caso. Necesitamos un elemento “central” que gestione todas las comunicaciones.

Como podemos ver, el concepto de la solución no es complejo. Es a la hora de pensar en su implementación en la industria y sus posibles futuros acoples de más elementos cuando se complica un poco más todo.

Si recordamos cuando introdujimos el problema al comienzo del documento, se mencionaron unos objetivos a cumplir y restricciones que solventar. Debemos enfrentarnos a cómo vamos a medir la temperatura con los sensores sin que éstos se dañen por el contacto con la cola.

Pensamos que un sensor por infrarrojos es una buena opción. Mediremos la temperatura sin contacto, por lo que la precisión es algo menor, asi que a la hora de adquirirlos debemos buscarlos de una calidad suficiente para no tener mucho error en las mediciones.

Estos sensores estarán conectados a unos miniordenadores que mandarán los mensajes al ESB. Estos dispositivos pueden ser implementados con tecnología Arduino-Raspberry, es bastante sencilla de usar y ya cuenta con algunos componentes preparados para funcionar sin problemas en ambientes industriales. Otra opción sería usar PLCs, menos sencillos de usar pero con más tiempo en el mundo de sistemas Scada. Aun así, pensamos que usar Arduino es suficiente y cubrirá sin ningún problema nuestras necesidades.

Una vez visto como solventamos los problemas de hardware, nos falta ver la parte software. Debemos definir cómo se van a enviar los datos al sistema central, cómo se van a comunicar los elementos entre sí. Aquí teníamos dos dificultades que solventar, tiempos de respuesta y fácil escalabilidad.

Sabemos que si seguimos la estrategia IAP con mediador central, deberemos buscar una tecnología que nos ayude a implementarlo, que nos permita la conexión de todos los elementos a un mismo punto de forma igual para todos

Una opción sería una una API Rest. En esta situación existiría un servidor Rest en el ESB y se le realizaría peticiones HTTP con las operaciones correspondientes.

Por ejemplo, los productores de mensajes, en los casos de reporte de temperaturas y de generación de órdenes, realizarían peticiones POST al servidor con los mensjes.

Los consumidores por su parte realizarían peticiones GET para obtener dichos mensajes con la información.

A simple vista tenemos lo que queremos: un sistema donde todos los elementos saben cómo conectarse a un punto central con la información.

Pero surge un problema. Las conexiones HTTP son síncronas, es decir, cuando se manda un mensaje ambos componentes deben estar conectados o de no ser así el mensaje se pierde. Entonces, cuando se mandase información al servidor central, ésta no se estaría recuperando instantáneamente, debería quedar almacenada en algún lugar, por ejemplo una BBDD, para que cuando se realizasen las peticiones GET se pudiese recuperar esa información.

Esto ralentiza bastante el sistema y genera un gran flujo de peticiones HTTP. En resumen, podemos decir que es una solución lenta y "pesada", justo lo contrario a lo que estábamos buscando.

Hemos planteado un diseño con esta tecnología API Rest, pero no cumple algunos de los requisitos planteados al inicio del documento, así que otra opción que puede ser viable es el uso de colas de mensajería.

Si optamos por esta tecnología, contaríamos con un bróker en ESB. A dicho bróker, en este caso el bróker Mosquitto, con las colas de mensajería MQTT, le llegarían mensajes de los sistemas productores, sensores y módulo de control. Dichos mensajes quedan almacenados en colas, a las cuales están suscritos los consumidores, visualizadores, módulo de control y activadores.

Estos mensajes se irán consumiendo según vayan generándose y llegando a las colas, y si algún sistema falla y no está en el momento en el que se genera un mensaje, se puede configurar para que se almacene en la cola y se entregue una vez reiniciado. Esta característica se llama comunicación asíncrona, y nos es muy útil en nuestra solución, por la resistencia a errores que nos ofrece. Es importante porque en un entorno industrial donde la producción no puede parar, los errores deben tener el menor impacto posible.

Esta solución ya convence más: es "ligera", rápida y resistente a fallos. Cubre los objetivos que nos establecimos en un principio.

Ya tenemos decidido con qué tecnología vamos a desarrollar nuestra solución. Ahora para poder decidir con más detalle tipos de colas a usar, formatos de mensajes, etc vamos a ver por separado como vamos a solucionar cada uno de los escenarios vistos en la fase anterior de análisis.

5.2 Diseño del escenario “conocer la temperatura”

Comenzamos por el caso donde queremos conocer la temperatura en cada punto del proceso de producción. Si recordamos lo que discutimos en la fase de análisis, contamos con los sensores del silo, la balsa y los difusores como elementos participantes, junto al ERP de la planta. Se comentó que estos sensores envían mensajes al ERP, pero no vimos de que forma lo hacían y que tipo de mensajes eran, y es lo que vamos a ver en este subapartado.

En la introducción de este punto de diseño hemos decidido usar como método de comunicación las colas de mensajería MQTT, haciendo uso del bróker Mosquitto. Una vez decidido esto tenemos que pensar que tipos de colas vamos a usar, ya que existen de tipo FANOUT, TOPIC… En este caso, podemos enfocarlo de forma que usamos colas de tipo FANOUT, unas colas donde todos los elementos suscritos reciben los mensajes que llegan a dicha cola, para asociar una cola por cada línea de producción y así monitorizar todos los elementos internos, los cuales estarían publicando en dicha cola.

Si escogemos esta opción, dentro de los mensajes deberíamos indicar de qué elemento procede la medición, para poder asociar la temperatura con el punto donde se ha obtenido de alguna forma.

Es una opción que funcionaría bien y sería viable, pero vamos a ver cómo sería si usásemos colas de tipo TOPIC y de esta forma poder decidir mejor.

Antes de empezar la discusión, cabe comentar que el bróker Mosquitto no dispone de colas TOPIC como tal. En su lugar se usan colas de tipo FANOUT pero cuyo nombre sigue una estructura especial que actúa como TOPIC (elem1/elem2/elem3) y se va creando como un árbol. En este ejemplo elem1 sería la raíz y elem2 una rama de donde cuelga elem3 (serían subtopics de elem1). Te puedes suscribir a todas las ramas que cuelgan usando caracteres especiales, como #. Escribiendo elem1/# te estarías suscribiendo a todos los TOPICS que colgasen de elem1.

Tras este inciso, podemos continuar con la discusión. Si usamos colas de tipo TOPIC es cierto que existirían más colas que usando una de tipo FANOUT por cada línea, pero por otro lado, esto nos ayuda a tener más organizados y controlados los mensajes y su origen. Cada elemento interno de la línea tendría su propia cola TOPIC y elementos externos que quisiesen leer datos de ese elemento podrían hacerlo solamente estando suscritos a dicha cola, y directamente sabes que el mensaje lo ha generado ese elemento, sin necesidad de procesar el mensaje. Esto nos da la

posibilidad de poder filtrar si hiciese falta en algún momento los mensajes por procedencia sin tener que ir abriendo cada uno y comprobando de donde procede.

Una vez comparadas ambas opciones decidimos que vamos a usar colas de tipo TOPIC para este escenario.

Cabe mencionar que en todo momento se está hablando solamente de colas dentro de una línea de producción, ya que en este proyecto nos centraremos en el desarrollo en solo una línea, pero todo lo mencionado se aplicaría de la misma forma en el resto de líneas una vez llegado el momento de implementar la solución en el resto de ellas.

Como formato del mensaje podemos usar el canónico de la solución, que llamaremos **formato_temp**, un JSON con los pares **temp: 52, unidades: C**. Decidimos usar este formato así, separando valor de las unidades, para facilitar en un futuro si fuese necesario la conversión entre unidades de medida de la temperatura.

Ahora ya que tenemos claro como solventamos este escenario podemos hacer un resumen de cómo sería el proceso: los sensores se conectarían a colas de tipo TOPIC, donde van publicando las mediciones que van haciendo. Cada TOPIC permitirá identificar de qué elemento procede la medición, por lo que en los mensajes solo viaja la temperatura medida siguiendo el formato mencionado. Los elementos que quieran consumir las mediciones generadas solamente tendrán que suscribirse a las colas de los elementos de los que deseen obtener datos.

Esta solución adoptada consigue cubrir los objetivos que nos proponemos relacionados con la escalabilidad y flexibilidad. Usando estas colas de tipo TOPIC, conseguimos que en el caso de que aparezca un nuevo elemento en la planta de producción pueda fácilmente comenzar a reportar mediciones si así lo requiriese. Solamente debería crear su cola con su TOPIC asociado siguiendo el formato acordado para los nombres de los topics y ya estaría conectado al sistema, y otros dispositivos podrían comunicarse con el suscribiéndose a su cola. Además, contamos con la flexibilidad que nos aporta el uso del formato canónico en JSON antes mencionado. Esto significa que, aunque el nuevo dispositivo entienda un lenguaje distinto no tendría problema ya que a la hora de comunicarse todos usan el mismo formato, solamente sería necesario implementar un transformador de datos que convirtiese el formato con el que trabaja el dispositivo al formato con el que se comunican en la empresa y viceversa.

Para dejar más claro todo a continuación hay una imagen que lo ilustra:

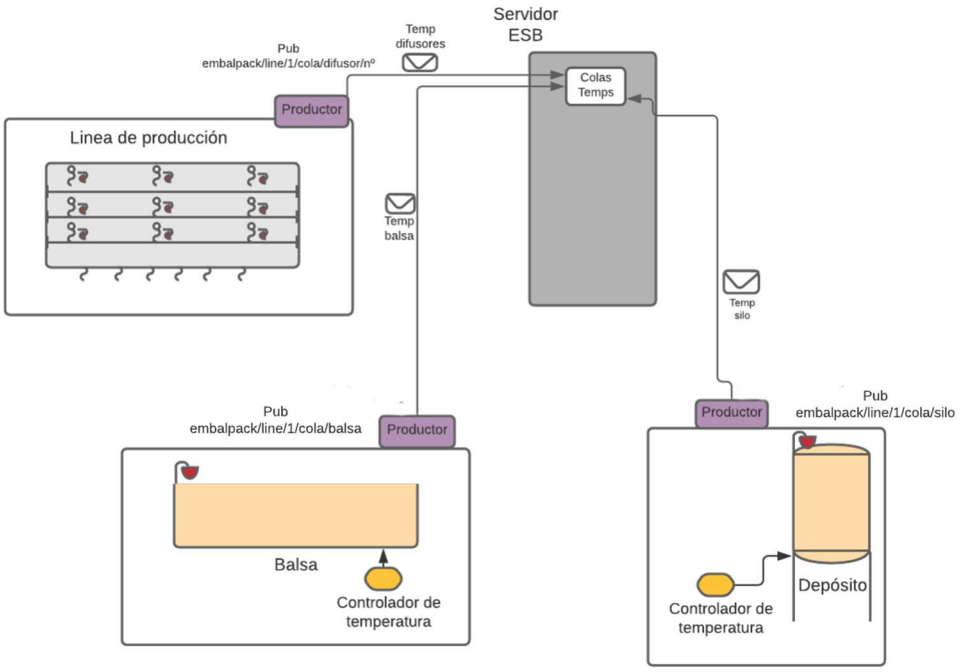

Ilustración 15: esquema diseño del escenario reporte de temperaturas

5.3 Diseño del escenario “auto-gestionar el control de la cola”

Continuamos viendo el diseño de la solución para cubrir el escenario de auto-gestión. Partiendo de lo hablado en el apartado de análisis, tenemos como elementos consumidores a los activadores de los calentadores y bombas de succión, que reciben mensajes con órdenes del ERP.

En este caso, la forma en la que se van a comunicar es muy sencilla. Estos elementos no tienen que crear ninguna cola ni generar mensajes, solamente se deben suscribir a la cola de tipo TOPIC asociada a él donde se publican las órdenes para sus activadores. Por ejemplo, el silo de una línea se suscribiría a su cola de control a la que llegan mensajes para activar o desactivar su calentador o su bomba.

Una vez visto que tipo de colas usamos y cómo se conectan los elementos participantes vamos a ver cómo se recibe la información, en este caso las órdenes.

Como hemos comentado, cada elemento cuenta con varios activadores, por lo que dentro del mensaje deberemos indicar en un campo el activador que debe actuar, y en otro campo deberemos indicar si debe encenderse o apagarse. Por tanto, podremos usar un formato que llamaremos **formato_orden** con los pares **elemento: calentador, acción: activar**. Este será también un formato canónico, en este caso el usado para las órdenes.

Como resumen del funcionamiento, los elementos de las líneas de producción se suscriben a colas de tipo TOPIC asociadas a ellos de donde reciben mensajes que les indican que actuador activar o desactivar.

Aplicando esta solución a este caso de uso estamos consiguiendo cubrir las restricciones que tenemos impuestas de escalabilidad y flexibilidad, las cuales hemos ido mencionando a lo largo del documento. Tal y como se ha diseñado este escenario, conseguimos un sistema escalable, ya que, de la misma forma que ocurre con el escenario anterior de reporte de mediciones, cada elemento cuenta con una cola asociada donde recibe órdenes, por tanto, si se quiere instalar un nuevo dispositivo en la planta y queremos que se auto-gestione, solamente deberá estar suscrito a la cola TOPIC asociada a él, con el topic formado siguiendo la regla establecida. Respecto a la restricción de flexibilidad ocurre lo mismo que en el caso anterior, contamos con un formato canónico para el envío de ordenes por lo que, si algún dispositivo nuevo trabaja con un lenguaje distinto al de los demás no hay problema, ya que a la hora de comunicarse todos usan el **formato_orden** definido en los párrafos enteriores. De la misma manera que en el escenario anterior solamente sería necesario desarrollar un transformador de mensajes para convertir los mensajes de un formato a otro.

En la siguiente imagen se ilustra el proceso:

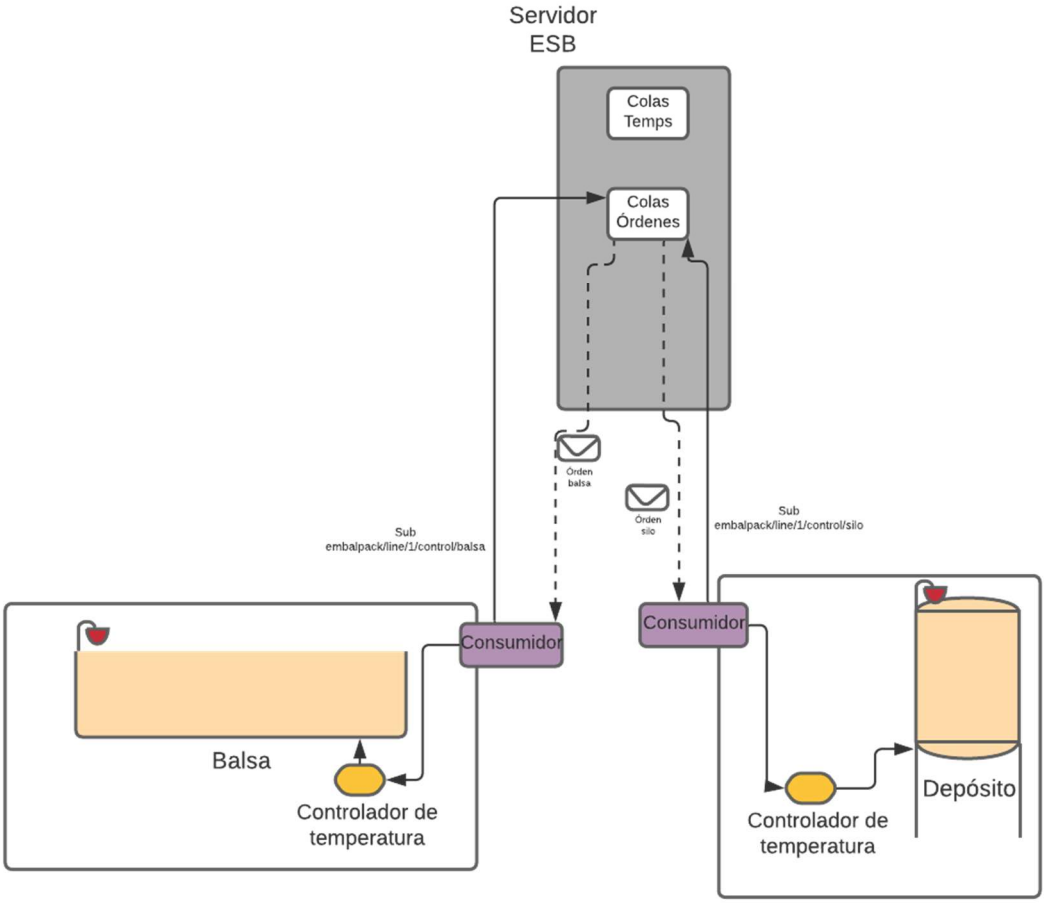

Ilustración 16: esquema diseño del escenario auto-gestión

5.4 Diseño del escenario “visualización”

Pasamos a diseñar la solución para el escenario de visualización de las temperaturas.

En este caso, recordando la etapa de análisis, sabemos que tenemos como elementos involucrados al ERP y los sistemas de visualización que se instalen por la fábrica y que reciben mensajes con lecturas de temperaturas del ERP.

El proceso es muy sencillo, muy similar al anterior de auto-gestión. En este caso, los elementos son consumidores y no tendrán que crear ninguna cola ni generar mensajes. Solamente se suscribirán a las colas creadas por los sensores, de los que hemos hablado en el punto 5.2, que son de tipo TOPIC e identifican el elemento generador de mensajes.

Además, si recordamos lo que también comentamos en el apartado de análisis, mencionamos que los visualizadores podrían filtrar por niveles, ya sea por líneas, por elementos o a nivel general de planta.

Esto lo hacemos usando un carácter especial de MQTT (#), que nos permite suscribirnos a los TOPICS que cuelguen de uno más general. De esta forma si queremos monitorizar una línea entera deberemos indicar el nombre del topic hasta el nivel que abarque todos los elementos de una línea, es decir, si tenemos elem1/elem2/elem3 y tomamos elem2 como identificador de la línea y elem3 como identificador del elemento productor, si nos suscribimos a elem1/elem2/# estaríamos suscritos a todos los elementos que cuelguen de la línea elem2.

Respecto al formato de los mensajes que reciben con la temperatura usaremos el mismo que para el proceso de reporte, el canónico JSON con la estructura **temp: 52, unidades: C**.

En conclusión, cada sistema visualizador se suscribe a las colas de los elementos a los cuales debe monitorizar, pudiendo elegir conjuntos concretos y recibe del ESB los mensajes con las temperaturas con el formato canónico definido.

En este escenario es donde más nos vemos beneficiados por el diseño que hemos adoptado, ya que con el uso de colas de tipo TOPIC conseguimos una flexibilidad a la hora de monitorizar enorme. A parte de las ventajas que conseguimos como la escalabilidad, ya que cada elemento visualizador que se implemente podrá conectarse al sistema tan fácil como suscribirse a las colas de elementos que se desee monitorizar, y flexibilidad gracias al uso de formatos canónicos, como hemos comentado en los escenarios previos, esta solución nos ofrece la posibilidad de tener una monitorización totalmente flexible. Esto es debido a que hemos definido que cada elemento cuenta con su respectiva cola TOPIC donde publica mensajes. Un visualizador tiene la libertad de poder monitorizar los elementos que desee por separado, estén o no en la misma línea, puede monitorizar una línea entera, puede monitorizar solamente una clase de dispositivos, por ejemplo silos, o puede monitorizar la planta entera. El uso de estas colas de tipo TOPIC nos ofrece la posibilidad de prácticamente tener “infinitas” posibilidades y variaciones de monitorizar los elementos con los que cuenta la planta de producción.

En la siguiente imagen se aprecia el proceso:

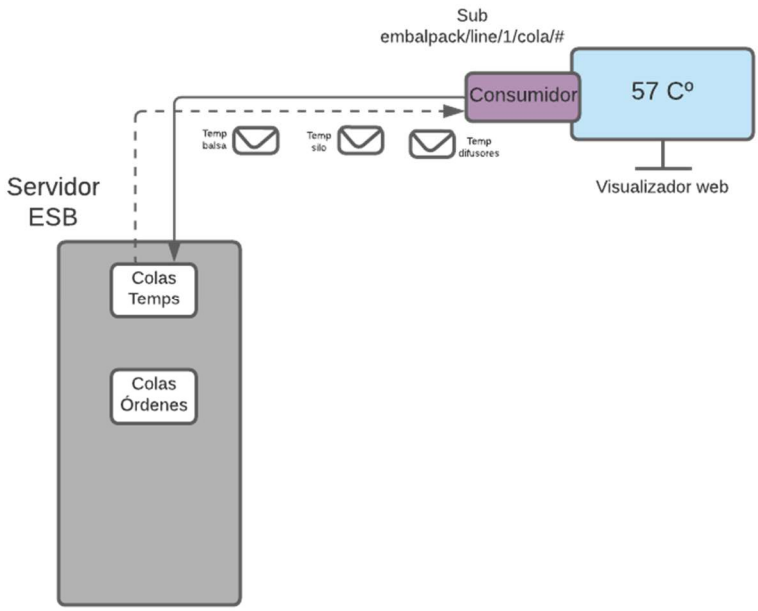

Ilustración 17: esquema diseño del escenario visualización

5.5 Diseño del escenario “toma de decisiones”

Por último, vamos a ver el diseño de la solución que adoptamos para cubrir el escenario de toma de decisiones.

En este caso de uso contamos con el ERP, presente en todos los casos, y el módulo de control, encargado de la toma de decisiones para controlar la calidad de la producción.

En esta ocasión, se publican mensajes y se consumen. El módulo de control recibirá del ERP mensajes con la temperatura, porque estará suscrito a todas las colas TOPIC de una línea creadas por los sensores en el proceso de reporte de temperaturas, de manera que tendremos un sistema de control por cada línea de producción. Para imaginarnos mejor el funcionamiento pensemos en un visualizador que monitorizase una línea; ambos sistemas se suscribirían de la misma forma a las colas para recibir los mensajes de todos los elementos que cuelguen de esa línea de producción.

Una vez recibe el mensaje lo procesa, y tomando como base la medición recibida y comparándola con los parámetros de calidad que tengan establecidos la empresa, si

se requiere tomar alguna acción sobre algún elemento dicho módulo de control generará un mensaje de orden el cuál publicará en una cola TOPIC asociada a un elemento de una línea.

Estas colas seguirán un patrón similar al que usan las colas donde se publican temperaturas: tendrán el nombre definido de forma que se pueda asociar fácilmente al elemento al que corresponda.

Respecto al formato de los mensajes usados, seguiremos usando los formatos canónicos definidos, tanto para temperaturas como para órdenes. Para recordarlos, el formato usado para enviar temperaturas es un JSON con los pares **temp: 52, unidades: C** y el formato usado para enviar órdenes es un JSON con los pares **elemento: calentador, acción: activar**.

Para resumir y dejar clara la solución adoptada, el módulo de control "monitorizará" los mensajes recibidos desde el ERP de las colas generadas por los elementos de una línea de producción, procesará dichos mensajes para leer las temperaturas y decidir si tomar acciones de control, y si decide tomar alguna acción, generará un mensaje de orden el cual enviará al ERP a una cola asociada al elemento sobre el que hay que llevar una acción a cabo.

Con este diseño, conseguimos una solución escalable y flexible, que es lo que buscamos. Escalable porque gracias al uso de colas TOPIC contamos con la misma ventaja que en el escenario anterior: a la hora de implementar nuevos módulos de control, contamos con un amplio abanico de posibilidades para controlar elementos, pudiendo controlar elementos de una línea en concreto, pudiendo controlar elementos concretos, etc. Flexible debido al uso de formatos canónicos ya comentados, tanto a la hora de recibir mediciones como de enviar órdenes. Aunque dicho módulo de control trabajase internamente con otro lenguaje no habría problema, desarrollando un transformador que cambiase de un formato a otro, del formato interno al definido para las comunicaciones, tendríamos el problema resuelto y el módulo se podría comunicar y trabajar sin problema.

A continuación tenemos una imagen para ilustrar de forma más clara el proceso y su solución:

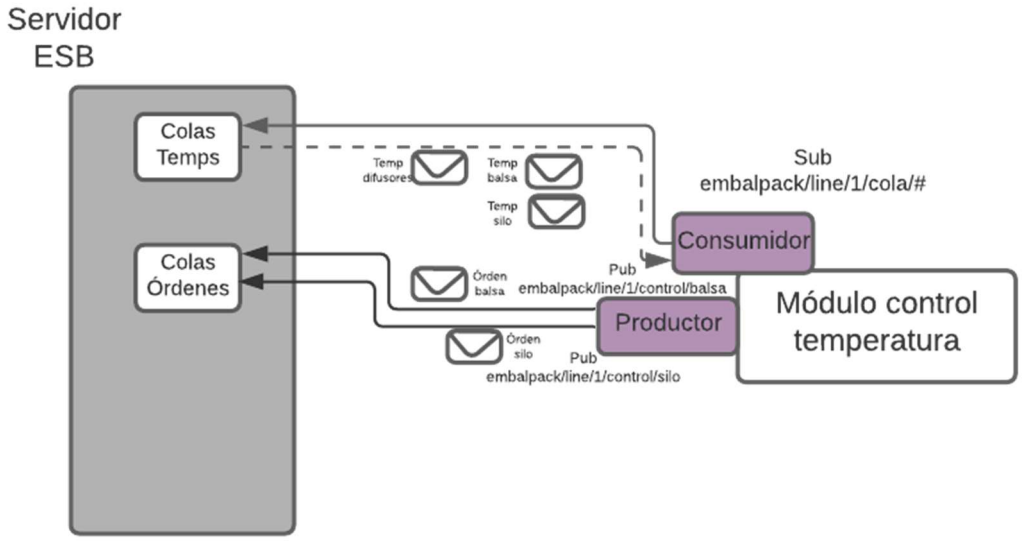

Ilustración 18: esquema diseño del escenario toma de decisiones

5.6 Diseño final adoptado

Una vez hemos visto en detalle cómo solucionar cada uno de los escenarios que habíamos detectado en la fase de análisis, podemos fusionarlos y obtener una imagen final de la solución que adoptaremos y pasaremos a implementar en el siguiente apartado.

Aquí se muestra un esquema del conjunto de todas las soluciones anteriores que conforman la solución final:

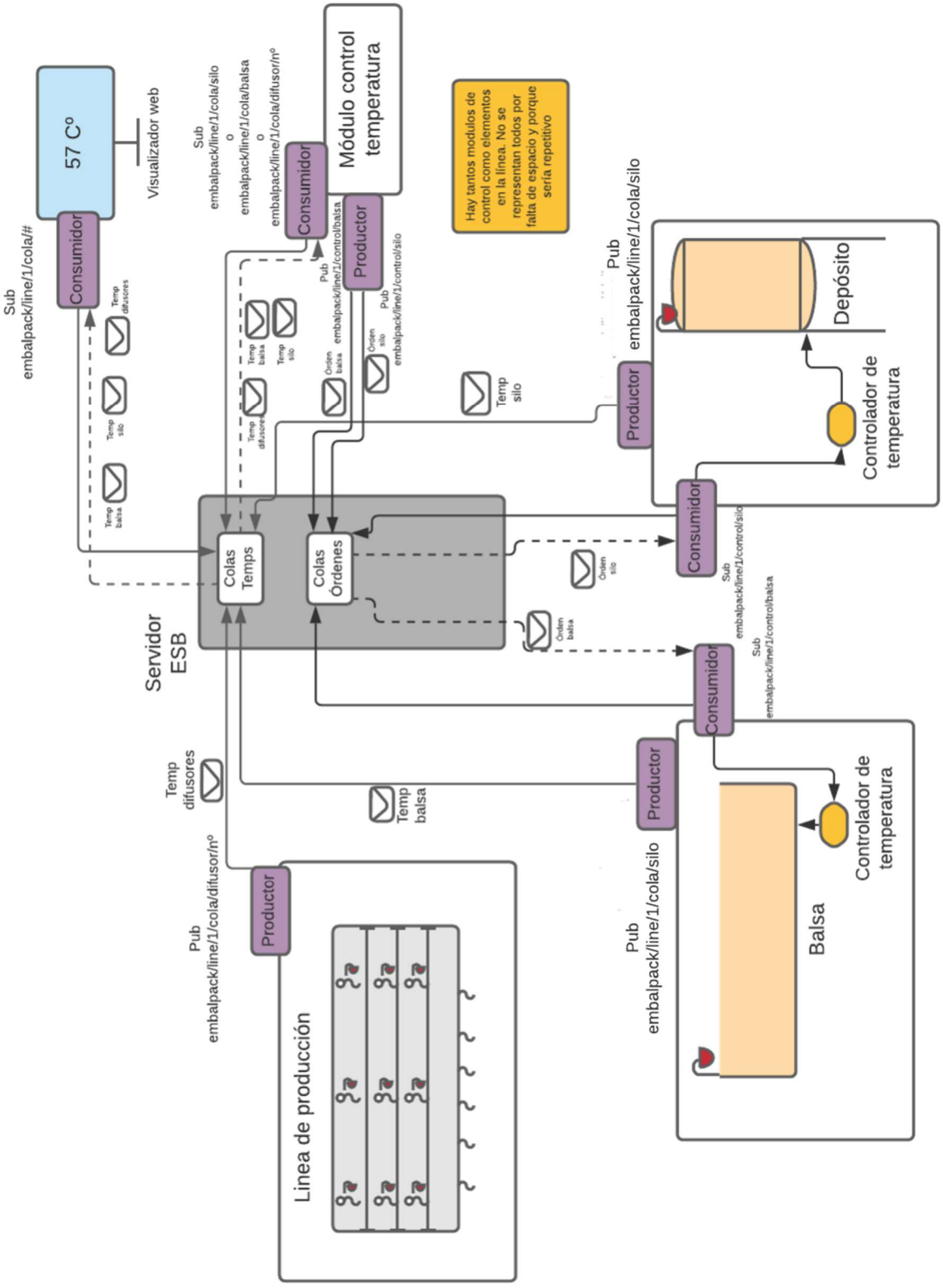

Ilustración 19: esquema de la solución propuesta

6. Implementación

En el apartado anterior hemos visto la solución que hemos adoptado, o mejor dicho hemos visto su diseño. Ahora el siguiente paso es su implementación y es el tema que vamos a abordar en este punto.

Como dispositivos hardware se usarán Arduinos, aunque aparte de su montaje se han desarrollado unos simuladores en Java, los cuales nos permiten ver cómo funciona el sistema entero sin montar ningún elemento hardware. Veremos por separado la implementación del simulador y el sistema hardware más adelante.

Se ha usado como tecnología de mensajería MQTT y para su implementación en Java se ha usado la librería Eclipse Paho/ Paho MQTT. Como hemos mencionado en el diseño de la solución, en esta tecnología no existen colas de tipo TOPIC, se usan del tipo FANOUT y el nombre de la cola son los "TOPICS". En nuestro caso hemos usado como formato de TOPIC el siguiente, siguiendo el patrón usado en la empresa: para las mediciones **embalpack/line/"nº línea"/cola/"elemento"** y para las órdenes **embalpack/line/"nº línea"/control/"elemento"**, donde **elemento** es silo o balsa y en caso de los difusores **difusor/"nº difusor"**.

Como formato de mensajes usamos JSON, un formato bastante sencillo de usar. Para el transporte de temperaturas seguimos el siguiente formato:

```
{
    "temp": "35",
    "unidades": "C"
}
```

Ilustración 20: formato mensaje temperatura JSON

Hemos decidido usar este formato separando dígito y unidades por si en un futuro es necesario realizar transformaciones entre unidades de medición. Respecto al formato de los mensajes de orden se ha decidido usar el siguiente:

```
{
    "elemento": "resistencias/bomba",
    "accion": "activar/desactivar"
}
```

Ilustración 21: formato mensaje orden JSON

De esta forma, separando acción y actuador, podemos enviar órdenes a los distintos actuadores de un elemento para futuros sistemas de autocontrol, como pueden ser sistemas de rellenado de la balsa activando la bomba del silo.

Como hemos venido comentando a lo largo de loa anteriores puntos, contamos con varios tipos de elementos, sensores, calentadores, módulo de control y visualizador. Para organizar mejor el desarrollo de cómo se ha implementado nuestra solución vamos a dividir el apartado en 3 subapartados, uno por cada tipo de elemento.

Es cierto que hemos tratado a los calentadores y sensores por separado, pero al encontrarse dentro del mismo sistema integrado en el elemento de la línea de producción, a efectos de implementación formarán un solo elemento.

Una vez hecha esta aclaración y habiendo introducido los siguientes puntos a tratar pasamos a ver primero la implementación del visualizador de temperaturas.

6.1 Visualizador

Se ha decidido que dicho sistema visualizador sea tipo web. Se ha desarrollado una web que se conecta al bróker MQTT mediante websockets, usando la librería Paho Javascript[5].

En la siguiente imagen se puede ver cómo se usa la librería y el método uusado para procesar los mensajes recibidos, **onMessageArrived**. Lo que hacemos cuando llega un mensaje es leerlo del JSON y obtenemos de donde proviene con la propiedad **destinationName** del objeto **message**. Una vez lo sabemos, podemos colocar la medición en la celda correspondiente de la tabla y comprobar mediante la función creada llamada **cambio** si hay que indicar con color una temperatura muy caliente o muy fría. También hay una imagen de la implementación de dicha función.

[5] Eclipse Foundation, Eclipse Paho JavaScript client

```
//Create a new Client object with your broker's hostname, port and your own clientId
var client = new Paho.MQTT.Client("tambori.dsic.upv.es", 9001, 'mqttjs_' + Math.random().toString(16).substr(2, 8));

var options = {

    //connection attempt timeout in seconds
    //timeout: 3,

    //Gets Called if the connection has successfully been established
    onSuccess: function () {
        alert("Connected");
        client.subscribe("embalpack/line/1/cola/#", {qos: 2});
    },

    //Gets Called if the connection could not be established
    onFailure: function (message) {
        alert("Connection failed: " + message.errorMessage);
    }

};

client.onMessageArrived = function (message) {
    //Do something with the push message you received
    var json = JSON.parse(message.payloadString);
    console.log(json.temp + " " + json.unidades);
    var elem = message.destinationName.split("/");
    if(elem.length == 6) {
        var topic = elem[4] + elem[5];
    } else if(elem.length == 5) {
        var topic = elem[4];
        //console.log(topic);
        //console.log(message.payloadString);
    }
    $("#" + topic).html(json.temp + " " + json.unidades);
    cambio(json.temp, "#celda" + topic);
};
```

Ilustración 22: fragmento de código donde se configura la conexión al bróker MQTT

```
function cambio(temp, celda) {
    if(temp > 60) {
        $(celda).css("background-color", "red")
    }
    else if(temp < 50) {
        $(celda).css("background-color", "cyan")
    }
    else {$(celda).css("background-color", "")}
}
```

Ilustración 23: implementación de la función cambio

Toda la lógica de la web se ha implementado con Javascript y se ha desarrollado una interfaz sencilla que consiste en una pequeña tabla con el nombre de cada uno de los elementos que generan temperaturas y sus correspondientes temperaturas. Además, mediante el uso de jQuery y CSS se ha implementado un indicador para que los

trabajadores vean fácilmente cuando la temperatura se encuentra por encima de 60 ºC o por debajo de 50 ºC.

Cuando se supera la temperatura se colorea de rojo el campo de la temperatura que lo supera y cuando desciende del mínimo se colorea de azul.

Ilustración 24: visualizador web en funcionamiento

6.2 Módulo de control

El sistema que recoge temperaturas y se encarga de enviar mensajes con órdenes a los calentadores está implementado en Java.

Se conecta a las colas creadas por los elementos de una línea en concreto, son controladores exclusivos de cada línea, y leemos las temperaturas con la ayuda de la librería JSON para Java[6].

En la siguiente imagen se aprecia como se crea el cliente que usa el módulo para conectarse a las colas y como se le asigna la clase **Callback** para el procesamiento de los mensajes recibidos.

[6] Eclipse Foundation, Eclipse Paho Java client

```
//Creamos cliente
MqttClient client = new MqttClient(broker, clientId);
//Asociamos el callback
Callback callback = new Callback(clientId, tempMax, tempMin, args[0].toString());
client.setCallback(callback);
//Creamos opciones de conexion y conectamos
MqttConnectOptions conOptions = new MqttConnectOptions();
conOptions.setCleanSession(true);
client.connect(conOptions);
//Nos suscribimos a las colas
client.subscribe(topic, qos);
System.out.println(clientId + " conectado");
```

Ilustración 25: creación de cliente MQTT y conexión a las colas para recibir temperaturas

En la imagen previa se ve como se procesan los mensajes que recibe el módulo de control. Vemos que lo primero transforma los mensajes JSON en algo con lo que poder trabajar haciendo uso de la librería para JSON mencionada. A continuación, averigua de donde proviene y modifica el TOPIC para enviar el posible mensaje de acción al mismo elemento pero a la cola de órdenes.

Obtenemos la temperatura y comprobamos si está dentro de los límites: si se encuentra entre el rango establecido entre 50 y 60 grados (rango seleccionado por la empresa) no ocurre nada pero si detecta que el valor no está dentro se encarga de crear un mensaje JSON con la orden adecuada y lo envía a la cola de control de dicho elemento con la temperatura equivocada para que realice la acción correspondiente sobre el calentador.

Cabe comentar que el módulo de control realmente usa dos clientes MQTT, uno para recibir y otro para enviar, uno se inicializa en el momento de arranque, momento en el cuál también se le indica la línea que monitoriza, y el otro cliente se inicializa dentro de la clase Callback, dejándolo listo para enviar mensajes.

Como último detalle a comentar, si nos fijamos se ha configurado el controlador para que pueda enviar mensajes de activación y desactivación de calentadores a los difusores, aunque estos no cuenten con calentadores. A pesar de ello se ha desarrollado así pensando en una futura implementación de dichos calentadores en este punto.

```
@Override
public void messageArrived(String arg0, MqttMessage arg1) throws Exception {
    // TODO Auto-generated method stub
    try {
          JSONObject json = new JSONObject(arg1.toString());
          double temp = json.getDouble("temp");
          System.out.println("Mensaje recibido del topic \"" + arg0.toString() + "\".");
          System.out.println("Temperatura: " + json.getString("temp") + " " + json.get("unidades"));
          topic = arg0.toString().replace("cola", "control");
          if(temp > tempMax) {
              System.out.println("Ahora desactivaria el calentador, temperatura muy alta.");
              JSONObject order = new JSONObject();
              order.put("elem", "calentador");
              order.put("accion", "desactivar");
              MqttMessage message = new MqttMessage(order.toString().getBytes());
              message.setQos(0);
              client.publish(topic, message);
          }
          if(temp < tempMin) {
              System.out.println("Ahora activaria el calentador, temperatura muy baja.");
              JSONObject order = new JSONObject();
              order.put("elem", "calentador");
              order.put("accion", "activar");
              MqttMessage message = new MqttMessage(order.toString().getBytes());
              message.setQos(0);
              client.publish(topic, message);
          }
        } catch (Exception e) {
          e.printStackTrace();
        }
}
```

Ilustración 26: implementación método messageArrived de la clase Callback

6.3 Productor

Como se ha comentado antes, vamos a ver por un lado la implementación del simulador y del sistema real.

Simulador

Nuestro simulador de temperaturas ha sido implementado en Java. Se han usado las mismas librerías que en el módulo de control, Eclipse Paho para la conexión al bróker MQTT y la librería JSON de Java para el tratamiento de los mensajes.

De lo que se encarga este simulador es de imitar el comportamiento de un sensor que va produciendo lecturas de temperaturas y las envía al bróker. Se ha implementado de forma que genera un número aleatorio entre 45 y 65 cada 5 segundos, crea el mensaje JSON a enviar y lo envía a la cola del elemento correspondiente.

En la siguiente imagen se muestra la generación y envío de los mensajes. La creación del cliente no se muestra porque sería repetitivo, ya que se realiza de la misma forma que en el módulo de control.

```
while (true) {
    temp = Double.toString(Math.floor(Math.random()*(65-45)+45));
    json.put("temp", temp);
    message = new MqttMessage(json.toString().getBytes());
    message.setQos(qos);
    client.publish(topic, message);
    Thread.sleep(5000);
}
```

Ilustración 27: implementación bucle que genera y envía temperaturas simuladas

Se ha implementado de forma que en el momento del arranque se pasa por parámetros el elemento al que imita. En la siguiente imagen se puede ver el momento de arranque de la aplicación:

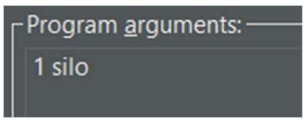

Ilustración 28: argumentos Productor

Sistema real

El sistema de medición se quería implementar en la fábrica y que fuese totalmente funcional, pero han surgido contratiempos que no nos lo han permitido. Lo que ha ocurrido es que los sensores adquiridos requieren una calibración un tanto precisa y se deben ajustar en el lugar donde se van a utilizar, si fuese de otra manera tendrían un error de medición bastante grande.

Debido a las limitaciones de la pandemia en las que se ha desarrollado este proyecto no se ha podido llegar a esta implementación, pero hemos conseguido otros puntos: la empresa con la que realizábamos el proyecto está contenta porque ha aprendido cosas sobre el sistema que al principio no tenían claras, han conseguido entender requisitos y saber mejor que es lo que quieren, dicha empresa ha trabajado de forma conjunta en el diseño, tanto en los inputs/outputs, la regulación, por lo que este prototipo inicial se ajusta a sus necesidades y por último la empresa ya tiene claro que va a seguir con este desarrollo, o bien a través de otro TFG que lo continúe o bien internamente, para implementar la solución completa.

Además, se ha desarrollado un plan B: se ha hecho una pequeña maqueta y se han calibrado los sensores de dicha maqueta para simular el trabajo real que realizaría.

Dicha maqueta se conectará al bróker y el resto de componentes funcionarán de la misma manera pero con datos generados con mediciones reales en lugar de simuladas, aunque no sean de la línea de producción.

En la siguiente imagen se puede apreciar en consola las mediciones realizadas por cada uno de los sensores antes de enviarlas a las colas. Si nos fijamos, se aprecia también si algún mensaje de control se ha recibido.

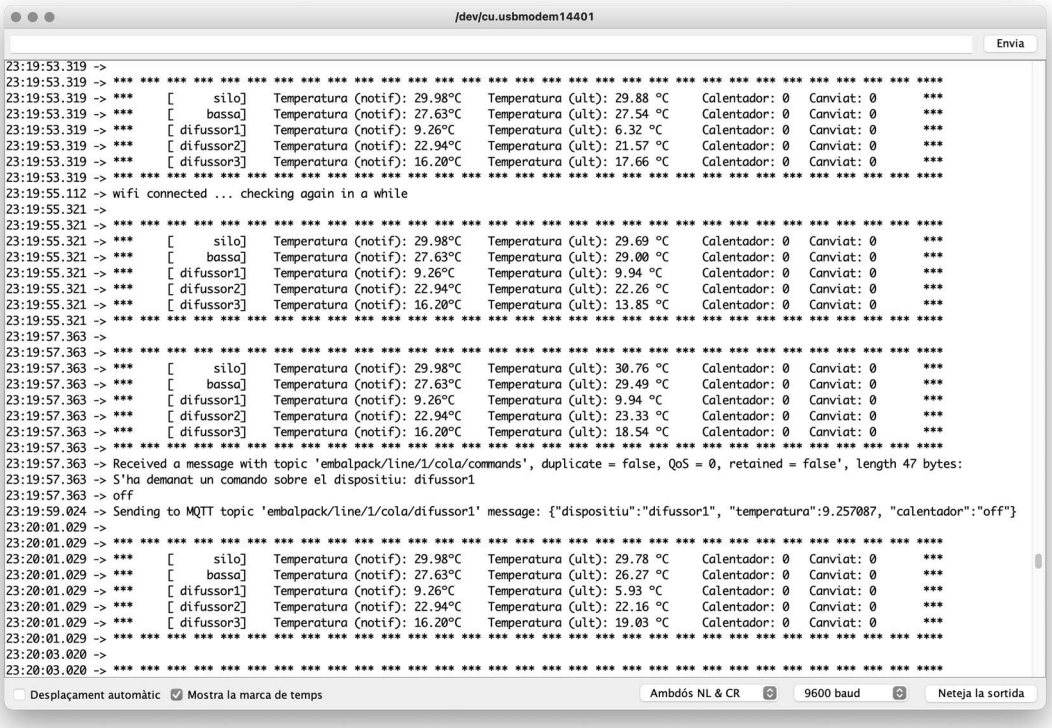

Ilustración 29: consola Arduino con las mediciones tomadas por los sensores

En las siguientes imágenes se puede apreciar el fragmento de código en el cual se realizan las configuraciones de los sensores, la configuración del cliente MQTT encargado de publicar dichas mediciones y la publicación de dichas mediciones:

```
// Configuració dispositius
controlador.begin(LINEA_PRODUCCIO);
silo.begin("silo", PIN_SILO_TEMPERATURA, PIN_SILO_CALENTADOR);
silo.setThresholdVariacioTemperatura(THRESHOLD_VARIACIO_TEMPERATURA_SILO);
controlador.addDispositiu(&silo);

bassa.begin("bassa", PIN_BASSA_TEMPERATURA, PIN_BASSA_CALENTADOR);
bassa.setThresholdVariacioTemperatura(THRESHOLD_VARIACIO_TEMPERATURA_BASSA);
controlador.addDispositiu(&bassa);

difussor1.begin("difussor1", PIN_DIFUSSOR1_TEMPERATURA, PIN_DIFUSSOR1_CALENTADOR);
difussor1.setThresholdVariacioTemperatura(THRESHOLD_VARIACIO_TEMPERATURA_DIFUSSOR1);
controlador.addDispositiu(&difussor1);

difussor2.begin("difussor2", PIN_DIFUSSOR2_TEMPERATURA, PIN_DIFUSSOR2_CALENTADOR);
difussor2.setThresholdVariacioTemperatura(THRESHOLD_VARIACIO_TEMPERATURA_DIFUSSOR2);
controlador.addDispositiu(&difussor2);

difussor3.begin("difussor3", PIN_DIFUSSOR3_TEMPERATURA, PIN_DIFUSSOR3_CALENTADOR);
difussor3.setThresholdVariacioTemperatura(THRESHOLD_VARIACIO_TEMPERATURA_DIFUSSOR3);
controlador.addDispositiu(&difussor3);
```

Ilustración 30: configuración sensores temperatura

```
void connectMQTTClientIfNeeded() {
  if (!mqttClient.connected()) {
    Serial.print("connecting mqtt client to host: "); Serial.print(mqtt_host); Serial.print("    port: "); Serial.println(mqtt_port);
    mqttClient.connect(mqtt_host,mqtt_port);

    // set the message receive callback
    mqttClient.onMessage(onMqttMessage);
  } else {
    Serial.println("mqtt client already connected ...");
  }
  //
  tasks.after(MQTT_CHECK_TIMEOUT, connectMQTTClientIfNeeded); // after MQTT_CHECK_TIMEOUT seconds call connectMQTTClientIfNeeded() again
}
```

Ilustración 31: configuración cliente MQTT Arduino

```
void publicar_estat_dispositiu(DispositiuCalefactable *d) {

   strcpy(topic, baseTopic);
   strcat(topic, "/");
   strcat(topic, d->getId());

    // send message, the Print interface can be used to set the message contents
    // int MqttClient::beginMessage(const char* topic, bool retain, uint8_t qos, bool dup)
//    mqttClient.beginMessage(topic, true, 1, false);
    mqttClient.beginMessage(topic);
    mqttClient.print(d->getJSONInfo(msg));
    mqttClient.endMessage();

  //
  Serial.print("Sending to MQTT topic '"); Serial.print(topic); Serial.print("' message: ");
  Serial.println(d->getJSONInfo(msg));

}
```

Ilustración 32: envío de mensaje con la medición de la temperatura

7. Conclusiones

7.1 ¿Qué hemos conseguido desarrollar?

Nos hayamos al final del documento y una vez aquí podemos permitirnos recapitular y observar hasta dónde hemos llegado.

Si recordamos, partíamos de un problema el cual nos requería autogestionar la producción de cantoneras, de forma que se monitorizaran ciertos puntos clave en la producción de éstas y se tomaran acciones según dichas mediciones.

Se estableció como objetivo de este proyecto la monitorización de las temperaturas de aplicación del pegamento usado en su fabricación en todos los puntos clave, ayudando a los trabajadores a saber cuál es en cada momento y en cada lugar y de esta forma poder actuar en consecuencia.

El objetivo se ha cumplido: se ha implementado un sistema que mide mediante sensores las temperaturas en los puntos clave que detectamos en nuestra visita inicial a la empresa y que nos indicaron los supervisores de la planta y que muestra dichas mediciones en una web.

Además, se ha desarrollado también una primera versión de un módulo de control para el siguiente paso del proyecto de autorregulación. Dicho controlador monitoriza las temperaturas medidas e indica que acciones llevaría a cabo, si activaría o desactivaría calentadores. El controlador es totalmente funcional, pero se ha decidido no implementar todavía al 100%, prefieren hacer pruebas antes en la empresa. En lugar de enviar las órdenes a los elementos y estos actuar en consecuencia, son unos LEDs que imitan lo que serían unas resistencias encendidas o apagadas quienes reciben los mensajes con las órdenes.

7.2 Futuro del proyecto

Si enfocamos el trabajo desde la visión del futuro, nuestro proyecto ha sido el primer paso de uno más grande. Como se comentó al inicio del documento, el objetivo a largo plazo es la implementación de un sistema de autorregulación del proceso de

producción. Es por ello por lo que un requisito de nuestro trabajo era crear una solución fácilmente escalable.

En un futuro, se seguirán desarrollando componentes como más módulos de control, se añadirán más sensores, o más visualizadores… Una empresa puede crecer y ampliar su planta con más líneas de producción, y dicha línea deberá estar monitorizada como las demás o quizá se quiera monitorizar otros parámetros de la producción, como el nivel de cola en el silo o la balsa para autorrellenarse, por ejemplo.

Con la solución que hemos desarrollado no habrá problema. Todo elemento nuevo que llegue y quiera conectarse al sistema lo tendrá muy fácil: siguiendo prácticamente los mismos pasos y usando las mismas herramientas se acoplará sin problemas al ecosistema existente.

En conclusión, se ha dejado la “puerta abierta” a nuevas incorporaciones.

7.3 Valoración personal

Como broche final al documento, vamos a ver una valoración personal de lo que ha supuesto el solventar este problema. Ha sido un proyecto muy enriquecedor, se ha podido ver como son los problemas en la vida real y como hay que enfocarlos realmente si se quieren solucionar. Se ha aprendido a diseñar una solución partiendo de una análisis por partes del problema, seguido de una lluvia de ideas de posibles formas de afrontarlo, eligiendo una idea que nos gustase y en la que viésemos potencial, a continuación darle forma poco a poco iterando sobre esa misma idea para ir perfeccionando la solución a todos los problemas e incógnitas que iban surgiendo y finalmente implementándola con las tecnologías que más se acoplasen al problema en cuestión.

Ha resultado bastante gratificante el haber visto al final funcionar todo el sistema y recordar cuando era solamente un concepto por desarrollar.

8. Referencias

[1] Jcabelloc, (06/08/2019). Porque una solución de Integración de Aplicaciones Empresariales Disponible en: https://jcabelloc.github.io/arquitectura%20de%20software/2019/08/06/porque-solucion-integracion-aplicaciones.html

[2] Gregor Hohpe Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions

[3] Sam Newman, (02/2015). Building microservices

[4] Rafael Corchuelo, Jesús González Martí, Rafael Z. Frantz, (09/2008). Una Comparación de ESBs desde la Perspectiva de la Integración de Aplicaciones Disponible en: https://www.researchgate.net/publication/230819015_Una_Comparacion_de_ESBs_desde_la_Perspectiva_de_la_Integracion_de_Aplicaciones

[5] Rafael Corchuelo, Rafael Z. Frantz, Carlos Molina-Jiménez. Una arquitectura para el diseño de soluciones de integración de aplicaciones empresariales con soporte para tolerancia a fallos Disponible en: http://www.gca.unijui.edu.br/members/rzfrantz/publications/jsweb-2010.pdf

[6] G. Holpe and B. Woolf, (2003). Enterprise integration patterns – Designing, building, and deploying messaging solutions

[7] Gustavo Álvarez. Integración de procesos de negocio de alto nivel con soluciones EAI Disponible en: http://paradigma.uniandes.edu.co/images/sampledata/PARADIGMA/ediciones/Edicion2/Numero2/Articulo6/ga-alvarez-3.pdf

[8] Alvaro Aguilar, Oscar, (2010). Integración de aplicaciones empresariales mediante el enterprise service bus, caso de uso: Programa Nacional de Apoyo Directo a los Más Pobres-PCM Disponible en: https://hdl.handle.net/20.500.12672/15380

[9] Vilca Oquendo, Dagne Helen, (2017). Optimización Del Proceso De Desarrollo De Proyectos, Bajo El Enfoque De Business Process Management (Bpm) En El Área De Integración De Aplicaciones Empresariales (Eai) En Una Empresa De

Telecomunicaciones Disponible en: http://repositorio.untels.edu.pe/jspui/handle/123456789/329

[10] Rafael Corchuelo, Rafael Z. Frantz, Jesús González. Un Marco de Referencia para Comparar ESBsdesde la Perspectiva de la Integración de Aplicaciones Disponible en: https://www.researchgate.net/profile/Rafael-Z-Frantz/publication/220268715_Un_Marco_de_Referencia_para_Comparar_ESBs_desde_la_Perspectiva_de_la_Integracion_de_Aplicaciones/links/02e7e526ec87f20464000000/Un-Marco-de-Referencia-para-Comparar-ESBs-desde-la-Perspectiva-de-la-Integracion-de-Aplicaciones.pdf

[11] Jhon Francined Herrera C. (2011) Estrategias para integrar aplicaciones Disponible en: https://revistas.uniminuto.edu/index.php/Inventum/article/view/5/5

[12] Caridad Anías Calderón, Julio Cesar Jerez Camps,Yinett Mazón Fernández, Joan Manuel Granadillo (2012) APROXIMACIONES A LA INTEGRACIÓN DE INFORMACIÓN Y APLICACIONES Disponible en: https://revistatelematica.cujae.edu.cu/index.php/tele/article/view/72/72

[13] Bolo, M. (2006). Arquitectura de integración orientada a servicios. Interfases, (001), 19-46. Disponible en: https://doi.org/10.26439/interfases2006.n001.169

[14] Rafael Z. Frantz, Rafael Corchuelo (2007) Integración de aplicaciones Disponible en: http://gca.unijui.edu.br/publication/Frantz-2007-65-75.pdf

[15] Clemente Ibarra, Alejandro Fabián (2019) INTEGRACIÓN DE APLICACIONES PARA UNA ARQUITECTURA DE SISTEMAS IAAS Disponible en: https://hdl.handle.net/11673/46993