Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño de una plataforma software para definir cuadros de mandos flexibles y configurables para entornos productivos

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: López Pino, Carlos

Tutor: Fons Cors, Joan Josep

Pelechano Ferragud, Vicente

Curso: 2021-2022

Resumen

El presente proyecto tiene como objetivo el diseño de una arquitectura de software que sea capaz de integrarse en cualquier entorno productivo para definir cuadros de mandos flexibles y configurables, en los que sea el propio usuario el que tenga control de los datos que se pueden visualizar en él. De esta manera, el cuadro de mandos podrá adaptarse a las necesidades que tenga actualmente el entorno productivo, para lograrlo analizaremos el contexto tecnológico de los entornos productivos actualmente, centrándonos en los sistemas SCADA, los sistemas más ampliamente utilizados actualmente en ellos.

La metodología utilizada en este proyecto ha sido la metodología ágil Scrum, de esta manera de manera visual hemos sido capaces de gestionará través de hitos, realizando modificaciones y marcando objetivos de manera que se pudiera realizar el proyecto a tiempo.

Para la realización de este proyecto se ha utilizado un sistema de mensajería tipo publicador/suscriptor basado en topics. Utilizando el protocolo MQTT y siguiendo los principios de integración del internet de las cosas, se definirá este sistema de mensajería, creando un middleware de comunicaciones pueda ser desplegado en cualquier entorno productivo, permitiendo que en un futuro se añadan más dispositivos sin necesidad de realizar modificaciones, generando la posibilidad de definir cuados de mando web flexibles y configurables, y definiendo una arquitectura escalable.

Palabras clave: cuadro de mandos, MQTT, integración de sistemas, middleware, visualización de datos, internet de las cosas,

Abstract

The objective of this project is to design a software architecture that is capable of being integrated into any productive environment to define flexible and configurable dashboards, in which the user himself has control of the data that can be displayed in he. In this way, the dashboard can be adapted to the needs that the production environment currently has. To achieve this, we will analyze the technological context of production environments today, focusing on SCADA systems, the most widely used systems in them today.

The methodology used in this project has been the agile Scrum methodology, in this way in a visual way we have been able to manage it through milestones, making modifications and setting objectives so that the project could be carried out on time.

To carry out this project, a publisher/subscriber type messaging system based on topics has been used. Using the MQTT protocol and following the integration principles of the internet of things, this messaging system will be defined, creating a communications middleware that can be deployed in any production environment, allowing more devices to be added in the future without the need for modifications. generating the possibility of defining flexible and configurable web control panels, and defining a scalable architecture.

Keywords : dashboard, MQTT, system integration, middleware, data visualization, internet of things.

Resum

El present projecte té com a objectiu el disseny d'una arquitectura de programari que siga capaç d'integrar-se en qualsevol entorn productiu per a definir quadres de comandaments flexibles i configurables, en els quals siga el propi usuari el que tinga control de les dades que es poden visualitzar en ell. D'aquesta manera, el quadre de comandaments podrà adaptar-se a les necessitats que tinga actualment l'entorn productiu, per a aconseguir-lo analitzarem el context tecnològic dels entorns productius actualment, centrant-nos en els sistemes *SCADA, els sistemes més àmpliament utilitzats actualment en ells.

La metodologia utilitzada en aquest projecte ha sigut la metodologia àgil *Scrum, d'aquesta manera de manera visual hem sigut capaces de gestionarà través de fites, realitzant modificacions i marcant objectius de manera que es poguera realitzar el projecte a temps.

Per a la realització d'aquest projecte s'ha utilitzat un sistema de missatgeria tipus publicador/subscriptor basat en topics. Utilitzant el protocol MQTT i seguint els principis d'integració de la internet de les coses, es definirà aquest sistema de missatgeria, creant un *middleware de comunicacions puga ser desplegat en qualsevol entorn productiu, permetent que en un futur s'afigen més dispositius sense necessitat de realitzar modificacions, generant la possibilitat de definir cuados de comandament web flexibles i configurables, i definint una arquitectura escalable.

Keywords : quadre de comandaments, MQTT, integració de sistemes, middleware, visualització de dades, internet de les coses,

Tabla de contenidos

Tabla de figuras

1. Introducción

El presente proyecto tiene como objetivo el diseño de una plataforma software para definir cuadros de mandos flexibles y configurables en entornos productivos a través del IoT[1] “Internet Of Things”, de esta manera el usuario será capaz de gestionar los elementos que integran su entorno productivo de manera autónoma y eficiente adaptándose a las necesidades actuales del mercado y a las necesidades que irán surgiendo a lo largo del tiempo.

1.1. Motivación

Actualmente los sistemas industriales están basados en sistemas SCADA[2] indispensables en los entornos productivos ya que permiten la automatización y el control industrial de manera eficiente. No obstante, la utilización de estos sistemas conlleva una clara desventaja, y es su baja flexibilidad.

En la mayoría de las ocasiones los sistemas SCADA vienen con una información predefinida obtenida en un estudio inicial de las necesidades del mercado y la empresa, o simplemente por los estudios realizados por proveedores de software ajenos a la empresa, lo que dificultará su adaptación a las necesidades que vayan surgiendo en el entorno productivo y en el mercado. Esto supondrá un alto coste de recursos a la empresa que tendrá que adaptarlo a las necesidades que vayan surgiendo y/o pagar a terceros para que lo adapten.

La principal motivación de este proyecto es resolver este inconveniente, diseñando una plataforma de software que permita gestionar la información del entorno productivo, creando un cuadro de mandos que muestre la información que el usuario desee conocer en cada momento de manera rápida y eficiente, y sin suponer ningún sobrecoste a la empresa. El único requisito es que el elemento del cuál se quiera conseguir la información forme parte de una red de comunicación que proporcionará nuestra solución y “reporte” sus datos en la misma, en la cual serán tratados, filtrados y adaptados para ser accesibles de manera rápida y eficaz desde nuestro cuadro de mandos.

1.2. Objetivos

El principal objetivo de este proyecto es el diseño de una plataforma software que sea capaz de integrarse en cualquier entorno productivo logrando que el usuario sea capaz de gestionar la información de manera intuitiva, eficiente y flexible, a través de un cuadro de mandos configurable, que sea capaz de adaptarse a las necesidades actuales y futuras del entorno productivo en el cual se implemente y que sea accesible desde multitud de dispositivos y lugares.

Otro de los principales objetivos es el de integrar la información de los diferentes softwares (cada dispositivo o máquina tiene el suyo propio) del entorno productivo en el que nos encontremos. En el cuadro de mandos antes mentado será donde el usuario tendrá acceso a toda la información producida y podrá configurarla seleccionando que desea visualizar en cada momento. Esta información será suministrada mediante la propia red de comunicación, que se desplegará sobre el entorno productivo y que se visualizará en la interfaz de visualización final, que llamaremos cuadro de mandos.

También nos gustaría que además de mostrar los datos recibidos por los elementos que conforman el entorno productivo, se diseñe un sistema que también sea capaz de almacenar un histórico de ellos, que deberá ser accesible junto al resto de datos en tiempo real desde el cuadro de mandos en todo momento, como permitiría un sistema SCADA actual, pero ofreciendo un nivel de configuración con el mínimo esfuerzo, tiempo y recursos.

1.3. Estructura

La estructura en la que dividiremos el proyecto será la siguiente:

- Para comenzar, hemos realizado una breve introducción al tema sobre el que versará este proyecto, además de plantear la motivación por la cual va a ser llevado a cabo y sus principales objetivos.
- A continuación encontramos el presente punto en el cual comentamos brevemente la estructura del proyecto.
- En el segundo apartado indagaremos con que metodología pensamos llevarlo a cabo.
- Más tarde, en el tercer punto exploraremos el contexto tecnológico en el que nos encontramos hoy en día en el mundo de los entornos productivos. Además,

realizaremos una crítica de este y redactaremos una propuesta que trate de esclarecer el espacio tecnológico que busca llenar nuestro proyecto.

- En el cuarto apartado examinaremos detalladamente la problemática que hemos detectado, identificaremos las posibles soluciones, haremos un breve análisis de ellas y especificaremos por cual nos hemos decantado.
- En quinto lugar detallaremos el diseño final de la solución, también mostraremos e indagaremos en su arquitectura reflejando las tecnologías y sus características que nos servirán para llevar a cabo la solución.
- En el sexto realizaremos una pequeña implementación sobre un entorno ficticio probando si nuestra solución cumple con las expectativas.
- En séptimo lugar mostraremos la explotación de la solución, donde podremos visualizar sus posibles usos y como se utiliza la solución.
- En el octavo apartado extraeremos las conclusiones obtenidas durante la realización del proyecto, así como comentaremos brevemente los conocimientos que hemos adquirido para plantear el diseño.
- Para finalizar, presentaremos una serie de aspectos que deberían de ser ampliados y/o tenidos en cuenta para la futura implementación del proyecto, e introduciremos una breve explicación de la razón por la cual no ha sido posible incluirlos en la memoria.
- Finalmente encontraremos las referencias bibliográficas de las cuales nos hemos servido para la investigación, documentación y verificación del presente documento.

2. Metodología

Una vez detallada la introducción de nuestro proyecto y teniendo definidos una serie de objetivos, hemos de centrarnos en el proceso para lograr de la manera más eficiente y dinámica posible llevarlos a cabo, para eso necesitaremos un plan de trabajo. A lo largo de mi corta trayectoria profesional dentro del ámbito de la informática he utilizado la metodología ágil SCRUM[3], por lo que debido a mi familiaridad con su uso me siento cómodo con ella, por esta razón trataré de adaptarla a la realización de este proyecto teniendo en cuenta que en este caso no trabajo con un equipo formado por varias personas, así que usaré esta metodología para llevar un ritmo de trabajo adecuado.

La metodología SCRUM es como hemos dicho una metodología ágil y flexible ampliamente utilizada en el desarrollo de proyectos en el ámbito de la informática, esto es debido entre otras cosas a la capacidad que brinda de autogestión y su desarrollo y adaptación continua de cara a objetivos planteados. Con este tipo de metodología se consigue que el cliente como el equipo encargado se entusiasme y comprometa con el proyecto ya que a través de entregas regulares el proyecto va creciendo poco a poco pudiéndose observar sus cambios con facilidad.

La principal razón por la cual implementaremos esta metodología, a parte de la familiaridad previamente comentada, es que de una manera muy visual se puede gestionar un proyecto que en muchas ocasiones tendrá especificaciones dinámicas, es decir, durante la realización de este irán evolucionando para adaptarse a los objetivos planteados en un tiempo limitado. De esta manera poco a poco iremos creando una solución completa entrega a entrega, como hemos comentado previamente; Al no disponer de un equipo habrá partes de esta metodología que en la realidad no tengan sentido, por la que la adaptaremos a nosotros.

La herramienta que hemos elegido para trabajar con esta metodología es Trello[4] ya que permite la creación de tableros en los cuales tendremos total control, podremos crear diferentes listas e ir creando fichas entorno a los hitos que busquemos conseguir, marcándoles un tiempo e indexándoles archivos relativos a los mismos. Todas las fichas que creemos podrán ser distribuidas a conveniencia de manera que podamos tener una visión rápida y directa de los elementos en los cuales queremos trabajar, los que hemos terminado y aquellos en los cuales estamos trabajando actualmente. Además podremos modificar y eliminar hitos con total facilidad, por lo que en caso de necesitarlo, podremos añadirlos en función de las necesidades del proyecto.

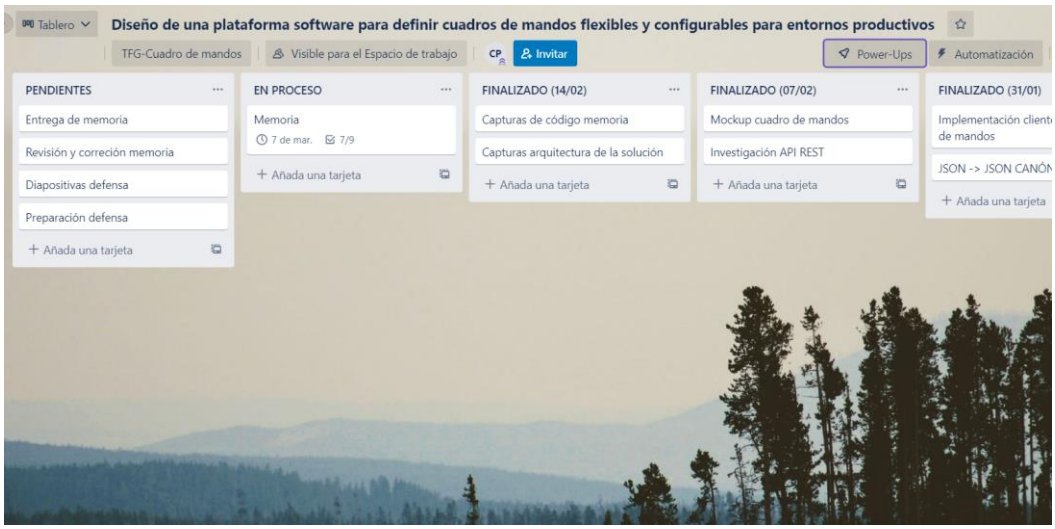


Figura 1: Tablero de trello con nuestra organización de las tareas

En nuestro tablero de trello que podemos observar en la figura 1, teníamos listas principales pendientes, en proceso y finalizadas (una para cada semana). A través de esta metodología podíamos tener una visión general de lo que nos quedaba por realizar así de lo que teníamos entre manos, en ocasiones me auto definía fechas para las que debía tener acabado algo para así lograr entregar el TFG a tiempo, intentando no dedicarle más tiempo del necesario a algunas tareas.

3. Contexto tecnológico

Actualmente los entornos productivos cuentan con una gran cantidad de dispositivos produciendo, generando y analizando una cantidad ingente de información, que utilizada correctamente puede suponer la diferencia entre una empresa, negocio o institución exitosos.

De esta manera surge una vertiente tecnológica muy interesante, el Internet de las Cosas o IoT, que es la ciencia que busca la interconexión de dispositivos en una red de información, donde todos ellos son visibles e incluso pueden interaccionar y compartir información entre ellos.

Hoy en día la industria es la principal interesada en lograr crear esta gran red de información, ya que cada día el volumen de datos con el que han de trabajar aumenta exponencialmente, no obstante a su vez también han de ser capaces de analizar y gestionar la gran cantidad de datos que genere esta red. Todo esto sumado a que los datos que actualmente necesitamos o en los cuales estamos interesados pueden cambiar de la noche a la mañana genera una problemática, pero también unas oportunidades muy interesantes.

3.1. Entornos productivos en la actualidad: Sistemas SCADA[2]

La principal tecnología utilizada por los entornos productivos para el control de dispositivos e infraestructura son los sistemas SCADA, cuyas siglas significan “Supervisory Control and Data Acquisition”, lo cual no quiere decir otra cosa que sistema encargado de la supervisión, el control y la adquisición de datos. Estos sistemas suelen ser una combinación de software y hardware, como por ejemplo controladores lógicos programables (PCL) y unidades terminales remotas (RTU).

En los sistemas SCADA el primer paso suele ser la adquisición de datos de la institución, la cual se obtiene a través de las PCL y las RTU, ya que son las encargadas de comunicarse con los sensores, maquinaría… etc. Una vez recopilados los datos se envían al siguiente nivel donde el usuario podrá consultar y gestionar la información recopilada a través de Interfaces Humano-Máquina (HMI), que no son otra cosa que pantallas que el usuario utiliza para comunicarse con el sistema.

Los entornos productivos utilizan estos sistemas para controlar los equipos de sus centros, monitorizarlos, realizar informes a partir de sus datos en tiempo real y además son capaces de archivarlos, las aplicaciones son infinitas. Sin embargo, los sistemas SCADA no están preparados para el procesamiento avanzado de grandes cantidades de datos y en muchas ocasiones son poco flexibles y adaptables al crecimiento o necesidades de la empresa, generando un alto coste además de una gran pérdida de información.

3.2. Crítica a los sistemas tecnológicos del panorama industrial actual .

La principal problemática reside en lo comentado anteriormente, por un lado la industria se ve beneficiada de la gestión de datos a un muy alto nivel ya sea para diferenciarse del resto, detectar oportunidades en el mercado, debilidades. Por otra parte la industria se apoya en gran medida en los sistema SCADA que sí, son capaces de supervisar, controlar y adquirir esta información, no obstante, son sistemas que en ocasiones pueden llegar a ser demasiado rígidos ya que son sistemas basados en el control centralizado de procesos para lo cual han de apoyarse en grandes bases de datos que aunque por un lado aseguran su fiabilidad y respuesta rápida a consultas, por otro lado conllevan inevitablemente un límite en cuanto a cantidad gestionable y mantenible de información se refiere.

Además los sistemas SCADA se basan en gran medida en sus HMI que en ocasiones vienen predefinidos y con escasas opciones de personalización conllevando un coste de recursos muy alto para su implementación y mantenimiento. Además teniendo en cuenta la época actual, podemos constatar que no basta tan solo con ofrecer un buen servicio, sino que también hay que hacerlo de una manera rápida y eficaz en un entorno cada vez más competitivo. Todo esto sumado a una dificultad técnica altísima para interconectar los dispositivos del entorno productivo actualmente, como de los dispositivos que formarán parte en un futuro crea una tendencia imposible de mantener en el tiempo.

En conclusión, los entornos productivos se encuentran en una encrucijada, por un lado necesitan tratar una gran cantidad de información y por otro lado los sistemas actuales son incapaces de mantener el ritmo gestionando esa información de manera eficiente. Sumado a una dificultad técnica muy alta a la hora de implementar mejoras en el sistema o mantenerlo provocan unos costes que el entorno productivo ni es capaz ni está dispuesto a asumir, provocando que los sistemas con los que se trabajan poco a poco se vayan quedando obsoletos impidiendo al entorno productivo aprovechar al máximo sus recursos.

3.3. Propuesta tecnológica

Como hemos podido observar la industria se enfrenta una problemática muy compleja ya que el sistema estándar es difícil de mantener y sobre todo muy difícil y costoso de configurar, esto provoca que en muchas ocasiones el sistema sea incapaz adaptarse a las necesidades de la empresa y tenga que ser la propia empresa la que a través de altos costes tenga que adaptarlo a la fuerza.

A través del IoT, una ciencia que cada vez cobra más y más importancia en el panorama actual, el proyecto actual tratará de solventar esta problemática apoyándose en las tecnologías que utilizan sus principios. Mediante ellos se tratará de ofrecer un sistema de comunicación que logre la interconexión de cada uno de los dispositivos que conformen un entorno productivo, así como permitir la implementación de nuevos dispositivos sin ser necesaria ninguna medida especial por parte de la empresa más allá de la adquisición y registro del dispositivo.

Si combinamos este sistema de comunicación flexible y escalable que nos brinda oportunidad de crear el IoT, a una interfaz de visualización eficiente, concretamente un cuadro de mandos altamente configurable, en el cuál podamos acceder a cada uno de los datos, de cada uno de los dispositivos contenidos en nuestra red. De esta manera seremos capaces de mejorar la gestión de la información, así como de brindar una experiencia que el propio usuario podrá adaptar a las necesidades que tenga independientemente de que estás vayan cambiando, el único requisito será que el dispositivo forme parte del sistema.

4. Análisis del problema

Una vez tenemos el problema contextualizado procederemos a analizarlo más detalladamente, con las conclusiones que se han extraído en el análisis del contexto tecnológico podemos observar una problemática clara pero no por ello simple.

Por un lado tenemos que lidiar con la ingente cantidad de datos, ya que los entornos productivos generan cada vez más y más información, en la mayoría de las ocasiones este crecimiento de datos acaba generando que inevitablemente los entornos productivos se vean resentidos, siendo incapaces de gestionar, analizar y utilizar la información que generan siendo incapaces de utilizarla en su propio beneficio, y ya no tan solo eso, sino provocando un gasto abrumador al tratar de conseguirlo intentando mantenerse en entornos cada vez más y más competitivos.

Por otro lado también buscamos que todos los dispositivos formen parte de la misma red de comunicación para que así sean capaces de compartir y recibir información entre ellos, principio en el cual se basa el IoT. Además también queremos que sea fácil implementar nuevos dispositivos a la red, para así no provocar un sobrecoste que se sume en gastos a la adquisición de los propios dispositivos, generando una gran flexibilidad y escalabilidad a nuestro sistema. Como ha quedado patente las industrias son cada vez más complejas y tratan siempre de buscar los últimos avances tecnológicos, para así mantenerse competitivos y refinar sus procesos en un intento de mantenerse en sectores muy competitivos.

Otro gran problema de los entornos productivos son los propios sistemas SCADA con sus HMI (Interfaces Humano-Maquina), como ya hemos comentado con anterioridad permiten la gestión de sus dispositivos, y debido a que estas interfaces son indispensables en todos los entornos productivos porque a través de ellos se les permite obtener continuamente la información necesaria de sus dispositivos. Aun así tienen una serie de inconvenientes insalvables a la larga, ya que se requiere de un personal cualificado para su puesta en funcionamiento que evidentemente hay que pagar. Además otra desventaja inherente a estas interfaces es que se ha de realizar un estudio de los usuarios que van a utilizar el sistema, para así establecer la estrategia más eficiente a la hora de crear los diferentes menús y opciones, para lo que también se necesitará un equipo de profesionales cualificados. Aunque hay que tener en cuenta el perfil de los usuarios tampoco basta solamente con eso ya que la información que necesitan esos usuarios cambia continuamente en el contexto actual.

A todo esto hemos de sumarle el crecimiento exponencial de dispositivos que se produce en los entornos productivos y la capacidad que han de tener las interfaces de permitir a sus usuarios, independientemente del tipo que sean, la capacidad de acceder a aquella información generada por estos dispositivos más relevante para ellos, así la como capacidad de descartar aquella información que no lo sea o a la que ni siquiera deberían tener acceso.

Finalmente todos estos problemas provocan que los entornos productivos sean incapaces de avanzar al mismo ritmo que su entorno, ya que para intentar tratar con todos los datos que se generen habrían ya no solo de realizar una HMI con innumerables menús, sino que los propios menús de estas quedarían obsoletos en poco tiempo, provocando la creación y modificación constante de estás interfaces, siendo imposible asumir el coste que produciría su constante adaptación a las necesidades de los usuarios, en ocasiones provocando que se funcione con sistemas obsoletos o desactualizados.

Aunque los sistemas SCADA no permitan una alta capacidad de configuración en cuanto a interfaces se refiere, o lo permitan pero a un coste en ocasiones inasumible, son ampliamente utilizados en la industria por varias razones. Una de estas razones es su alta capacidad de automatización y fiabilidad, antiguamente los entornos productivos únicamente se basaban en el control de sus dispositivos mediante los operarios, lo que provocaba una gran cantidad de errores humanos, cosa que se vio subsanada cuando se implementaron estos sistemas.

Además este tipo de sistemas también cuentan con unos sistemas de almacenamiento de información muy robustos basados en bases de datos que permiten almacenar una gran cantidad de datos y acceder a ellos mediante consultas de manera rápida, aunque poco a poco se vayan viendo sobrepasadas por la ingente cantidad de información que se genera.

Todo esto nos lleva a pensar que la solución no radica en eliminar de la industria los sistemas actuales, sino en tratar de que todos estos puntos clave sean capaces de coexistir en el mismo ecosistema, aprovechando los puntos fuertes de cada tecnología y eliminando o minimizando las debilidades de los sistemas actuales. Por lo que trataremos de buscar un enfoque que permita lograr este objetivo y que además sea flexible ante los requisitos de cada uno de los entornos productivos en los que se implementen. [5]

4.1. Requisitos de la solución

Para tener una idea clara de las posibles soluciones hay que tener claros los puntos clave en los cuales se ha de basar nuestra solución:

- Interconexión de dispositivos: capacidad de agrupar todos los dispositivos en una red en la cual todos ellos sean visibles y capaces de interaccionar.
- Escalabilidad: la capacidad de registrar nuevos dispositivos con facilidad sin suponer ningún inconveniente.
- Accesibilidad y adaptabilidad: interfaz única capaz de agrupar toda la información producida y adaptarse a las necesidades de los diferentes usuarios que formen parte del entorno, permitiendo además acceder en cualquier lugar y momento.
- Almacenamiento: capacidad de almacenar datos de interés eficientemente.
- Capacidad de gestión de la información: capacidad para gestionar una gran cantidad de información.

4.2. Identificación y análisis de posibles soluciones

Una vez tenemos claros los principales requisitos podemos observar dos núcleos bien diferenciados en el problema a resolver, por un lado nos enfrentamos a la necesidad de establecer una especie de red de comunicación entre los dispositivos del entorno productivo basado en la IoT, capaz de gestionar y almacenar gran cantidad de información, y por otro lado necesitamos crear una interfaz configurable y accesible que se adapte a las necesidades de usuarios tanto actuales como futuras. Para cada uno de estos núcleos habrá que buscar una solución que sea capaz de integrarse con la otra.

4.2.1. Red de comunicación entre dispositivos.

En primer lugar nos encontramos con la problemática que genera la búsqueda de una red de comunicación entre dispositivos, ya que para que esta tenga éxito en un entorno productivo ha de ser eficiente y escalable, la ciencia en la cual basaremos todas las posibles soluciones será el IoT, ya que está en gran parte basada en estos principios.

Dentro de la IoT existen varias metodologías y tecnologías con las cuales podríamos desarrollar nuestra solución. Nuestro objetivo principal es que los dispositivos de nuestra red sean capaces de enviar y/o recibir información, un patrón de comunicación existente dentro de la IoT es el patrón Publicador/Suscriptor. Esta metodología consiste en el envío de

información por parte de un publicador y la recepción de un suscriptor, es decir, un publicador genera un mensaje, que a través de un enrutador será recibido en un suscriptor. No obstante, esta tecnología por sí sola no sería capaz de lograr nada, ya que nuestro objetivo es integrar esta red de comunicaciones en entornos productivos con una gran cantidad de dispositivos con este fin existen varias infraestructuras de servicios en el IoT, para el presente proyecto las más interesantes son el modelo basado en una cola de mensajes y por otro lado el modelo basado en un sistema de mensajería a través de tópicos, ambas basadas en el patrón publicador/suscriptor.

Un sistema de colas de mensajes consiste en el envío por parte de un publicador a un bróker[6], el bróker generaría una cola de mensajes única para cada cliente que inicie la suscripción, aunque existen mecanismos para lograr distribuir a múltiples clientes. Una vez el publicador envía un mensaje indica a que cola de mensajes desea enviarlo, cuando es realizado el envío, el bróker reenviará los mensajes al suscriptor de esta cola. El bróker almacenará todos los mensajes recibidos hasta que son entregados al cliente.

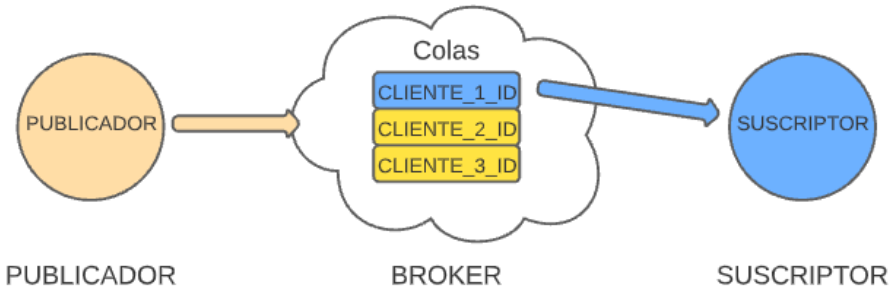


Figura 2: Esquema sistema de colas de mensajería

Un servicio de mensajería puro consiste en un bróker distribuyendo los mensajes entre los clientes conectados al mismo. En este caso se filtrarán los mensajes por algún criterio, ya sea haciendo que los suscriptores se suscriban a un tópico o por contenido del propio mensaje. El proceso es el mismo, el publicador envía un mensaje indicando el tópico relativo a este mensaje, pero posteriormente el bróker distribuirá este mensaje a todos los suscriptores de este tópico. La principal diferencia con las colas de mensajes es que los mensajes que se

entreguen mientras que el cliente se encuentre desconectado se pierden, no obstante existen maneras de asegurar una persistencia de datos también en este modelo.

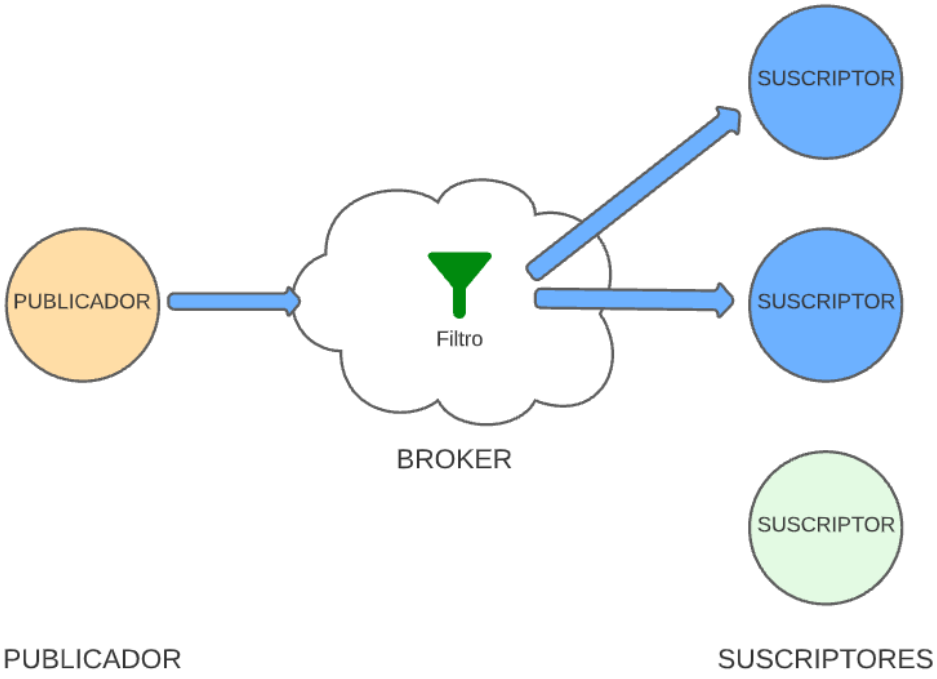


Figura 3: Esquema servicio de mensajería

Una vez tenemos claras las diferentes infraestructuras relevantes y los modelos en los cuales se basan se puede comprobar que existen múltiples protocolos[7] basados en sus principios, entre las que destacan:

- MQTT: Es un protocolo de mensajería muy estandarizado de OASIS para el IoT, está diseñado basándose en el protocolo publicación/suscripción pero haciéndolo extremadamente ligero y fácil de implementar, siendo ideal para conectar con dispositivos remotamente con un espacio de código pequeño y un ancho de banda de red mínimo.
- XMPP: Es un protocolo abierto basado en XML para aplicaciones de mensajería instantánea.
- AMQP: Es un protocolo Publicador/Suscriptor de Cola de mensajes, diseñado para asegurar su confiabilidad e interoperabilidad pensado para aplicaciones corporativas con mayor rendimiento y redes de baja latencia, por lo que no resultaría adecuado para aplicaciones de IoT con recursos bajos.

4.2.2.Interfaz configurable.

En segundo lugar nos centraremos en la búsqueda de diferentes opciones para la creación de una interfaz configurable y única, que sea capaz de agrupar toda la información producida por los dispositivos que conformen el entorno productivo. También ha de ser capaz de adaptarse a las necesidades de los diferentes usuarios que formen parte del entorno, permitiéndoles acceder a aquella información que les resulte más relevante en cada momento, haciéndolo además desde cualquier lugar y en cualquier momento. Analizando los requisitos se puede observar que necesitamos un sistema muy versátil e intuitivo.

Enfocándonos en la adaptabilidad del sistema a los distintos usuarios que puedan usarlo existirían dos opciones, por un lado una interfaz con gran cantidad de menús y por otro una interfaz capaz de permitir que sea el propio usuario el que genere sus vistas y de además seleccionar el formato en el que los ve. En primera instancia la opción más atractiva sería la de generar múltiples menús siendo también la más convencional, por un lado es más fácil ofrecer al usuario una serie de menús, logrando de esta manera optimizarlo todo, ya que estarían adaptados especialmente para él. No obstante de esta manera nos veríamos abocados a un gran coste, ya que habría que realizar un estudio muy detallado tanto de todos los dispositivos que componen nuestro sistema como de las necesidades de cada usuario, además si en un futuro evolucionaran las necesidades del usuario habría que reprogramar esos menús.

Por otro lado tendríamos la opción de ofrecer al propio usuario la capacidad de configurar sus propios menús ya que de esta manera cuando necesitara un dato concreto sería capaz de seleccionarlo, indicar el formato en que desea verlo y distribuirlo como desee. No obstante esto conllevaría también que el usuario pudiera verse abrumado por la cantidad de datos, siendo incapaz de seleccionar los datos que busca, por esta razón habría que hacer un planteamiento respecto a la forma en la cual se muestran los datos al usuario para que tenga facilidad a la hora de realizar una elección.

Una vez tenemos claro las opciones disponibles para lograr la versatilidad que buscamos, hay que comprobar de qué manera se podría lograr la mejor accesibilidad posible, para ello la propia interfaz debería permitir acceso a los usuarios desde múltiples dispositivos. La primera opción que se nos viene a la mente sería instalar el software en varios dispositivos y repartirlos entre los distintos usuarios para que pudiesen acceder al sistema. Sin embargo en la realidad esto crearía diversos problemas, ya que al ser un software instalado sobre un sistema habría que asegurarse de instalarlo en todos los dispositivos, previamente habiéndolos adquirido o instalándolos en el entorno productivo, lo que generaría un alto coste en tiempo y dinero.

En segundo lugar existe la opción de crear una aplicación que fuese instalable en los distintos dispositivos, ya sean ordenadores, tablets, móviles …etc. No obstante habría que realizar demasiadas versiones para poder correr el programa en todos los dispositivos, por lo que el coste de desarrollo aumentaría, además de ser necesario contar con un equipo multidisciplinar.

Por último lugar se podría implementar una aplicación web, lo cual permitiría acceder a cualquier tipo de dispositivo, únicamente sería necesario que la aplicación fuese capaz de redimensionarse reaccionando a las dimensiones de la pantalla.

4.3. Solución propuesta

Como solución para la problemática expuesta nos decantaremos por un sistema de mensajería basado en MQTT, utilizando tópicos para encaminar los mensajes. Debido a la magnitud de los entornos productivos y con el fin de mostrar los datos de la manera más ordenada y eficiente posible, se hará un cribado de los datos enviados por los dispositivos ya que cada dispositivo corre su propio software por lo que creara mensajes en distintos formatos, cada mensaje será recibido por un módulo encargado de transformar los mensajes a un formato que el cuadro de mandos sea capaz de comprender y para filtrar los datos de un dispositivo ya que no tiene por qué enviar un único tipo de datos.

Respecto a la interfaz visual nos hemos decantado por una aplicación web, ya que de esta manera el cuadro de mandos podrá ser accedido desde cualquier punto por distintos usuarios y en infinidad dispositivos, el único requisito será tener acceso a internet. Además esta aplicación permitirá al usuario a través de un sistema de widgets crear su propio cuadro de mandos, donde podrán seleccionar la información que desean ver y el formato en el que desean hacerlo. Para simplificar la forma en que estos datos viajan a través del sistema de mensajería y la operatoria a la hora de seleccionar los datos desde el propio cuadro de mandos por parte del usuario, se ha optado que el tópico a través de cual viajen los datos sea el propio tipo de datos, es decir, por ejemplo que todo viaje a partir de un tópico llamado temperatura, así será fácil catalogar y filtrar la información tanto para el sistema como para el usuario ya que este vera la información en tópicos y podrá seleccionar de que dispositivos quiere ver la temperatura y con que formato quiere verlo. Respecto a la persistencia de datos para almacenar un histórico de datos se contará con una API REST a la cual se le harán peticiones para registrar datos y consultarlos en caso de ser necesario. Para crear un sistema más robusto y con una alta capacidad de almacenamiento de datos esta API REST se conectará a un

sistema de base de datos, en la cual se ira volcando periódicamente la información almacenando un histórico accesible desde la página web a través de la API REST.

5. Diseño de la solución

En el presente apartado abarcaremos el diseño completo de nuestra solución logrado a partir de los objetivos definidos al comienzo del proyecto y a través de la idea de solución propuesta en el apartado anterior, indagando en un entorno productivo ficticio en el cual integraremos todos los módulos y componentes necesarios para llevar a cabo la misma.

5.1. Arquitectura del sistema

Basándonos en los objetivos que buscamos lograr en nuestro proyecto y desplegando nuestra solución sobre un entorno productivo ficticio la arquitectura finalmente elegida para nuestra solución es la siguiente:

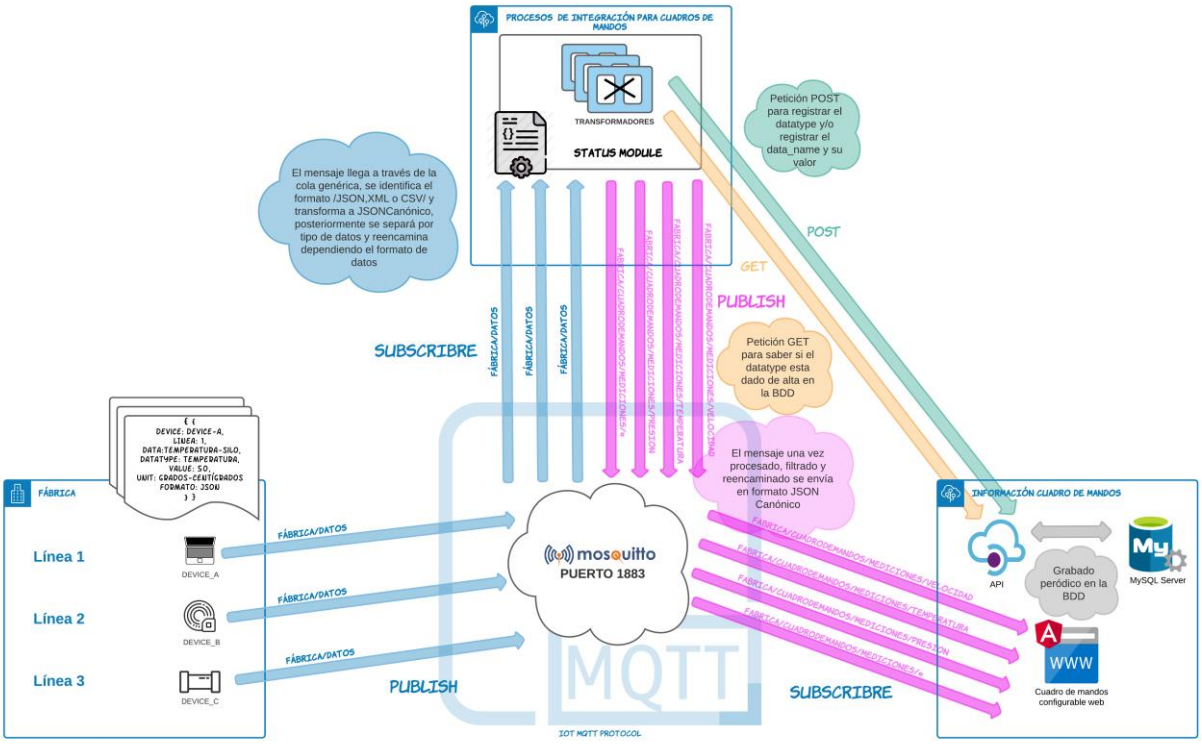


Figura 4: Arquitectura solución

Para comprender mejor está arquitectura podemos identificar una serie de módulos bien diferenciados en los que se pueden dividir nuestra solución, entre los que se encuentran:

- Módulo “Fábrica”

- Bróker Mosquitto
- Módulo “Procesos de integración para cuadros de mandos”
- Módulo “Información cuadro de mandos”

A continuación veremos más en detalle cada uno de estos módulos explicando cuál es su función y los elementos que contienen.

5.1.1. Módulo “Fábrica”

En este módulo de nuestra solución están todos los dispositivos que componen un entorno productivo ficticio al cual hemos bautizado como fábrica, en este entorno los dispositivos están funcionando, generando datos. Estos datos se generan en diferentes formatos dependiendo el software de cada dispositivo como en un entorno productivo al uso. Este módulo puede contener un numero n de dispositivos, la única consideración a tener en cuenta en caso de querer añadir un nuevo dispositivo será asegurarse de que reporte sus datos en el topic “fabrica/datos” indicando su formato.

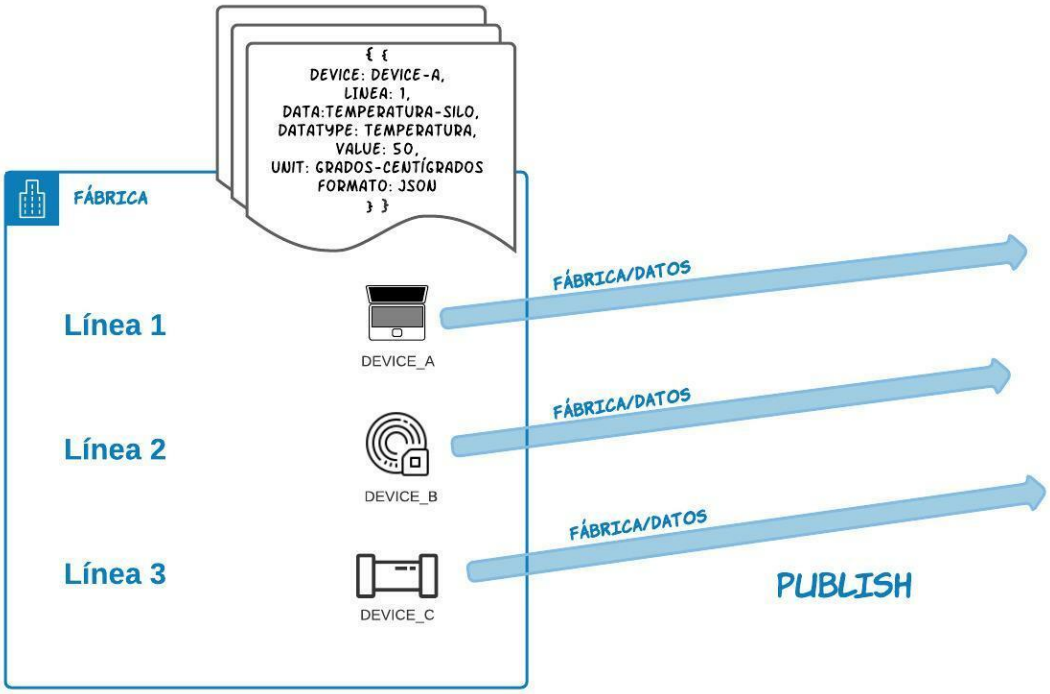


Figura 5: Módulo "fábrica"

5.1.2.Bróker Mosquitto

Es el bróker que se encargará de la recepción y envío de mensajes de los agentes que componen el protocolo MQTT, a través de él transitaran todos los mensajes de nuestra solución, todo esto será posible gracias a una serie de topic por los que circularan los mensajes de un agente a otro. El bróker de Mosquitto se despliega sobre el puerto 1883 manteniéndose siempre a la escucha de cualquier mensaje, serán los propios publicadores/suscriptores los que le indicarán para que topic envían o esperan recibir mensajes.

Figura 6: Bróker de MQTT Mosquitto

5.1.3.Módulo “procesos de integración para el cuadro de mandos”.

Este módulo está compuesto de dos piezas muy importantes el “status module” y los transformadores (contenidos en el propio status module), en estas piezas se llevará a cabo la recepción, procesado y reenvío de mensajes procedentes del entorno productivo reencaminándolos a través de una serie de topic exclusivos para el cuadro de mandos.

En esta primera pieza, el “status module” se procesan los mensajes que llegan a través del topic “fabrica/datos”, una vez recibidos analizará cada mensaje y se encargará de comprobar el formato de datos. Una vez hecho “status module” se comunicará con nuestro segundo componente, los transformadores, indicándole el tipo de datos que ha detectado. Los transformadores se encargarán de transformar ese mensaje en distintos mensajes, uno por cada tipo de datos contenido en el mensaje inicial, por ejemplo un dispositivo puede enviar varios datos a través de un mismo mensaje como podrían ser peso y tiempo. Cada uno de los

mensajes tendrá un formato canónico idéntico preparado para que resulten comprensibles posteriormente al cuadro de mandos, una vez procesados se reenviarán a través de distintos topic según el tipo datos.

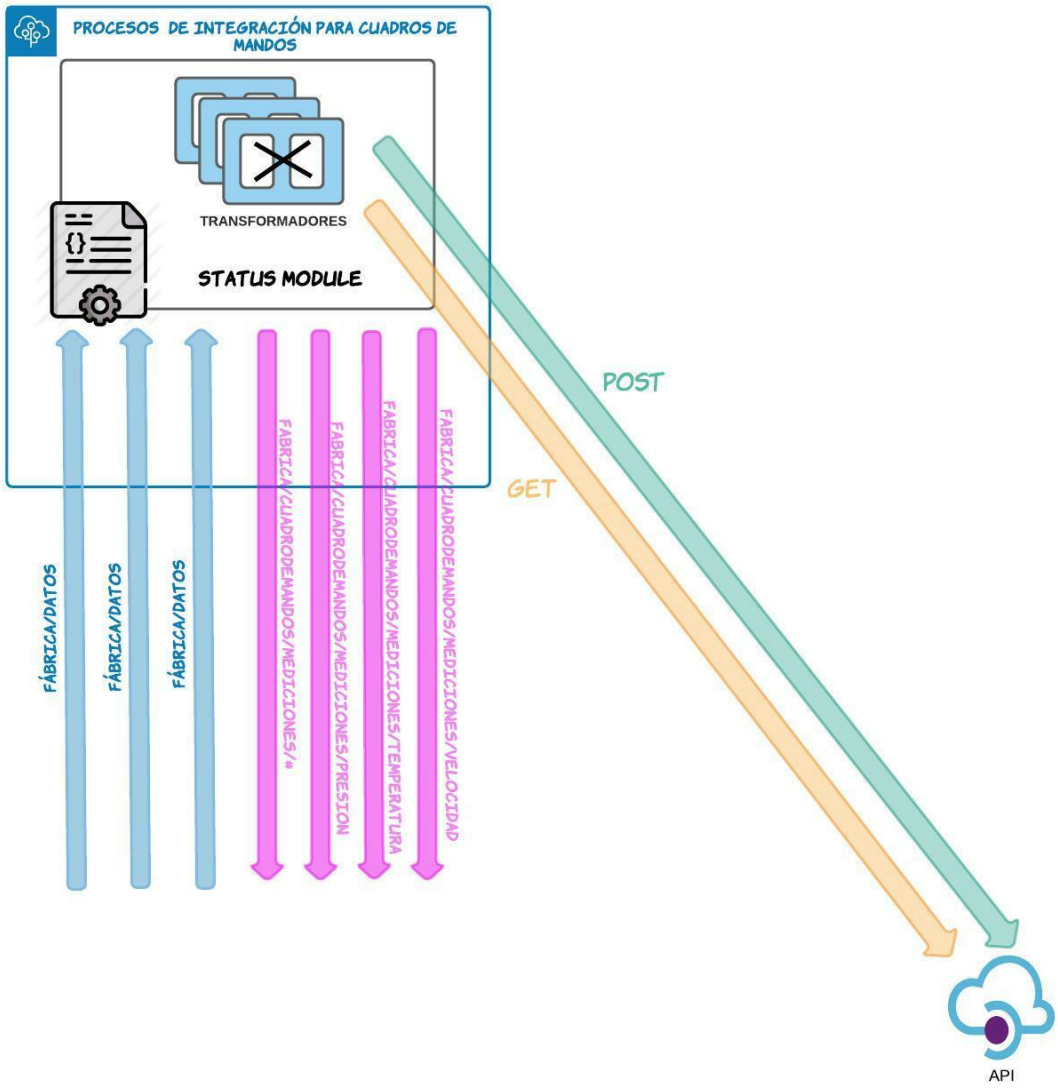


Figura 7: Módulo "Procesos de integración para cuadros de mandos"

Este módulo también es el encargado de comunicarse con la API para registrar los dispositivos o comprobar si ya están registrados por el sistema, así como almacenar los datos más relevantes de los mismos.

5.1.4. Módulo “Información cuadro de mandos”.

En este módulo están contenidas la API, la Base de datos y el cuadro de mandos web. Una vez los mensajes ya han sido procesados son recibidos por el cuadro de mandos que se mantendrá a la escucha de todos los mensajes procedentes del topic "fabrica/cuadrodemandos/mediciones" sabiendo en cada momento si han llegado por ejemplo por el topic "fabrica/cuadrodemandos/mediciones/velocidad" o por el contrario por "fabrica/cuadrodemandos/mediciones/temperatura", de esta manera nos aseguramos de que en caso de declararse más topic el cuadro de mandos también los contemple. El cuadro de mandos será una interfaz web configurable a través de la que el/los usuarios podrán acceder a los datos de las máquinas una vez procesados y reencaminados adecuadamente, seleccionando por ejemplo el dato de la máquina A la velocidad, de una lista de datos de velocidad de todos los dispositivos del entorno productivo.

Figura 8: Módulo "Información cuadro de mandos"

Además en este módulo también están contenidas la base de datos y la API, la API hará de capa intermedia entre todos los componentes y la base de datos, será la propia API la que se encargue del volcado de datos en la Base de datos, facilitando además la comunicación con los diversos componentes a través de peticiones HTTP.

5.2. Diseño detallado

Una vez conocemos cada uno de los módulos en los cuales hemos dividido nuestra solución analizaremos exactamente en que consiste la misma y como interactúan cada uno de los componentes entre ellos. Nuestra solución se basa en un sistema de mensajería de topics a través del protocolo MQTT.

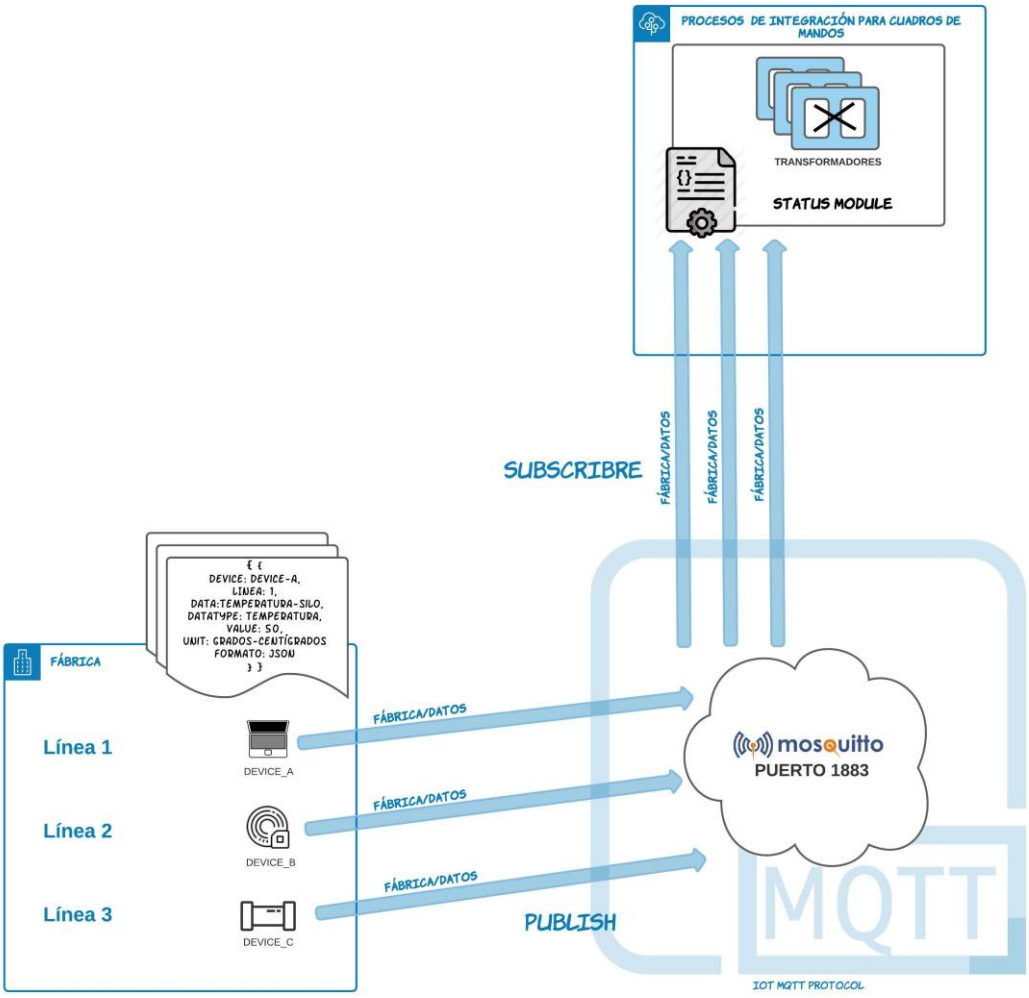


Figura 9: Viaje de un mensaje desde fabrica al módulo "procesos de integración para cuadros de mandos"

En primera instancia consideramos un entorno productivo en el cual funcionan una amplia variedad de dispositivos, cada uno de ellos enviará información en un formato de datos como podrían ser JSON[8], XML[9], CSV[10] …etc. Independientemente de cuál sea el dispositivo y el formato en el que generan sus datos, se enviarán utilizando el topic “fabrica/datos” a través del Bróker Mosquitto.

```
{
"Format": "JSON"
"device": "Maquina_A",
"productionLine": 1,
"data": [{
        "data_name": "temperatura-silo"
        "datatype": "temperatura",
        "value": 50,
        "unit": "grados-centígrados"
]}
"data": [
{
        "data_name": "presion-silo"
        "datatype": "presion",
        "value": 28,
        "unit": "psi"
}]
}
```


Figura 10: Ejemplo de mensaje enviado por un dispositivo del entorno productivo en este caso en formato JSON.

Una vez un mensaje como el que podemos ver en la figura 10, generado por un dispositivo del entorno productivo, se publica en el topic "fabrica/datos" del bróker, este lo reenviará a los suscriptores de este topic, en nuestro caso el suscriptor de este mensaje será el "status module" que desde el despliegue de la solución se mantendrá constantemente a la escucha de todos los mensajes publicados en este topic, una vez se reciba el mensaje se procederá con su procesado.

En primer lugar el módulo "status module" recibe el mensaje e identifica en que formato de datos se está enviando el mismo, cuando identifica el formato de datos llamará al correspondiente transformador de datos, cabe destacar que en nuestra solución será necesario que siempre que se integre un nuevo formato de datos se desarrolle un transformador específico para traducir los datos a un formato de datos canónico. De esta manera el cuadro de mandos siempre entenderá la información que le llegue independientemente del dispositivo que proceda.

En segundo lugar una vez se ha llamado al transformador adecuado, el mensaje se traducirá a nuestro formato canónico. A lo largo del proyecto hemos decidido que el formato más adecuado es JSON, así que hemos especificado un formato canónico figura 11 en el que recibirá siempre los mensajes nuestro cuadro de mandos. Una vez interpretado el mensaje por parte del transformador, el "status module" se encargará de separar el mensaje por tipo de datos, esto es debido a que nuestra solución se basa íntegramente en los tipos de datos recibidos por las máquinas. Los dispositivos en muchas ocasiones no generan tan solo un

dato, sino que por el contrario generan una gran variedad de ellos, nuestro sistema ha de estar preparado para recibir el máximo número de datos sin necesitar un envío especifico por cada uno de ellos, por lo que la mejor manera de lograr este objetivo es que el propio sistema sea capaz de separarlos. Como podemos ver en la figura 10, el dispositivo “Maquina_A” genera un único mensaje que contiene dos tipos de datos diferentes, una vez este mensaje ha sido procesado por el transformador correspondiente y procesado por el “status module” uno de los mensajes resultantes es el que podemos observar en la figura 11.


```
{
data: "presion-silo",
datatype: "presión",
productionLine: 1,
device: "Maquina_A",
value: 28,
unit: “psi”,
}
```


Figura 11: Mensaje en formato JSON Canónico

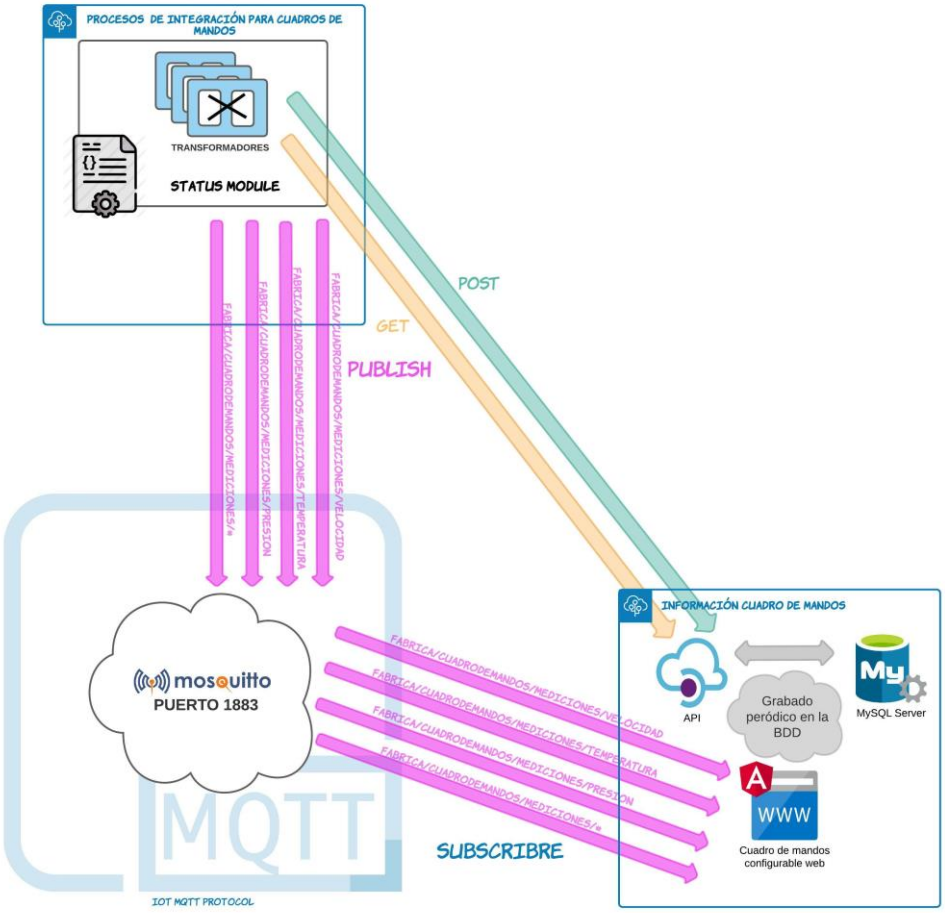


Figura 12: Viaje de mensaje hacia el módulo "información cuadro de mandos"

En tercer y por último lugar, una vez generados los mensajes transformados y procesados, se reencaminarán en diferentes topics como podemos ver en la figura 12, para facilitar la gestión de los datos por parte del cuadro de mandos. Hemos decidido que la manera más adecuada de hacerlo era por tipo de datos "datatype", por ejemplo, con el mensaje recibido en la figura 10 se generarían dos mensajes cada uno de ellos sería reencaminado por un topic diferente, uno por el topic "fabrica/cuadrodemandos/mediciones/temperatura" y otro por el topic "fabrica/cuadrodemandos/mediciones/presion" (el mensaje observado en la figura 11).

A parte de procesar los mensajes procedentes de "fabrica/datos" y reenviarlos por su topic pertinente, el "status module" también es el encargado de comunicarse con la API REST, esto será necesario para registrar los dispositivos que existen en nuestro entorno productivo por si en algún momento se quiere extraer un listado de los mismo y también para cada cierto tiempo ir almacenando datos de cada máquina. Esto lo logrará mediante peticiones POST y GET a la API para tanto introducir información como consultarla en caso de ser necesaria.

Una vez los mensajes han sido publicados a través de los topic específicos "fabrica/cuadrodemandos/mediciones" + "/" + "(magnitud correspondiente)" , entra en juego el siguiente modulo, el módulo "Información de cuadro de mandos" en este módulo se recibirán todos y cada uno de los topics "fabrica/cuadrodemandos/mediciones" independientemente de la magnitud, ya que en el cuadro de mandos web han de estar visibles todas ellas.

Una vez se reciben los mensajes se inyectarán en el servidor del cuadro de mandos, una vez logueados a través de la web cada usuario podrá generar su propio cuadro de mandos entorno a sus necesidades actuales. Los datos recibidos a través de los diferentes topics serán accesibles ya que el cuadro de mandos creará un menú por cada tipo de dato, en el que se permitirá la selección de los datos recibidos de ese tipo, para cada dato que aparezca en los diferentes menús se indicará su nombre y la máquina de la que procede, el tipo de datos no será necesario indicarlo explícitamente debido a que se encontrará en un menú en el que únicamente habrán dispositivos para ese tipo de datos que además corresponderá con la cola en la que se publiquen.

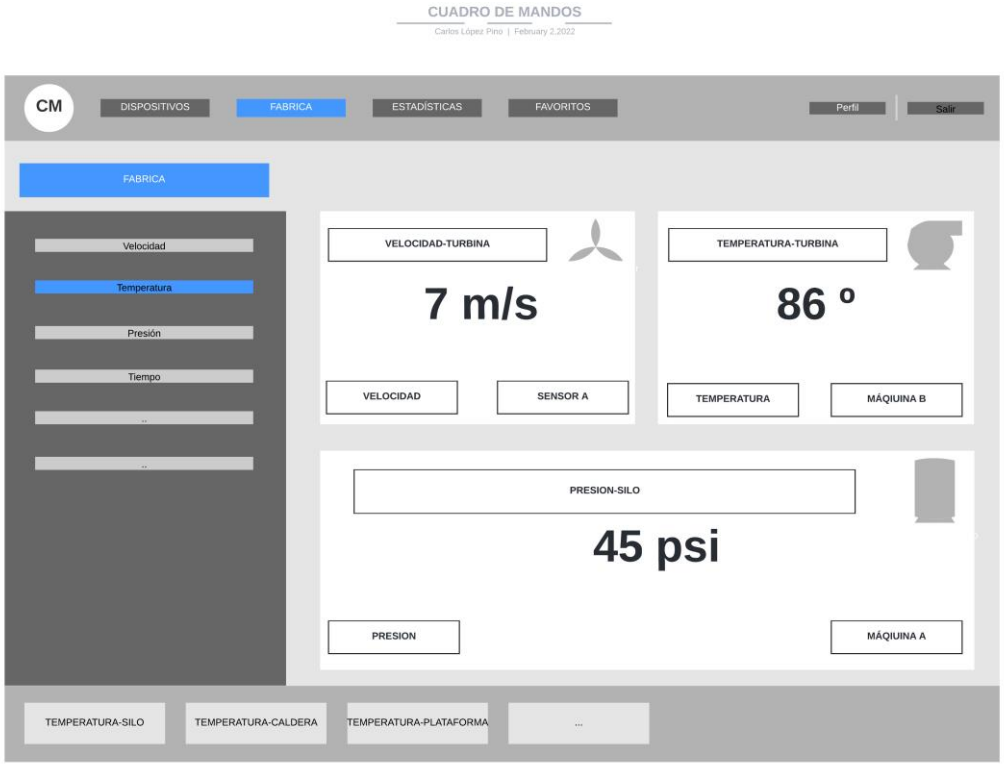


Figura 13: Mockup cuadro de mandos web

Como podemos observar en la figura 13, en el cuadro de mandos se mostrará los tipos de datos y se podrán seleccionar los datos por nombre que estén llegando a través de ese topic. Posteriormente en la parte derecha se podrá arrastrar el dato que queramos ver y distribuir a conveniencia entre el resto de los datos seleccionados.

El módulo información de cuadro de mandos también contendrá nuestra API y nuestra base de datos, la base de datos tan solo será accesible a través de la API de esta manera conseguiremos desacoplar la base de datos para que en caso de que sea necesario hacer cambios se pueda lograr de la manera menos engorrosa posible.

La API será accesible desde la propia web del cuadro de mandos, a través de peticiones se podrá obtener lista de los dispositivos, histórico de datos o incluso guardar información relevante para el usuario. Además como en la base de datos se contendrá las credenciales de los usuarios que tienen acceso al sistema para poder loguear habrá que hacer un GET con los datos del usuario que quiere loguear para saber que permisos tiene y cuáles no.

La base de datos se mantendrá actualizada debido al periódico volcado de datos por parte de la API, la API también será la encargada de volcar la información que se le brindará a través del procesamiento de mensajes en el status module, además también deberá responder a las peticiones procedentes del mismo.

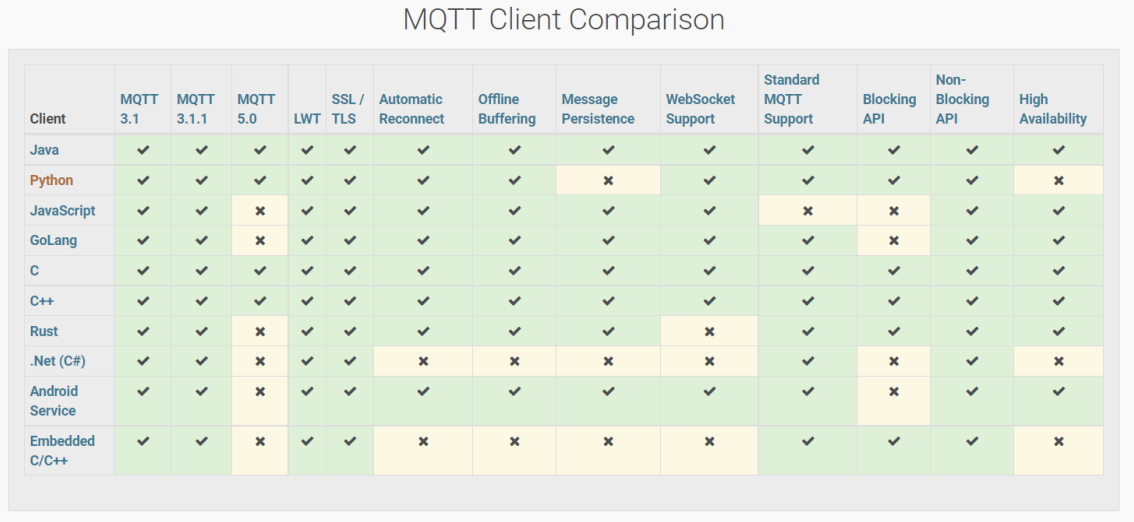
5.3. Tecnologías necesarias

Vamos a detallar los lenguajes, protocolos, frameworks y herramientas que hemos utilizado para realizar el diseño del proyecto, así como indicar en que partes del proyecto haríamos uso de ellos.

5.3.1. Protocolo MQTT[11]

MQTT es un protocolo de mensajería específico para el IoT, hoy en día es considerado un estándar dentro de esta ciencia, esto es debido a que es un protocolo de muy bajo peso basado en el patrón publicador/suscriptor. La principal ventaja de este protocolo es que permite la conexión remota de gran variedad de dispositivos, todo ello con la necesidad de generar muy poco código y de una manera que utiliza muy poco ancho de banda para ser llevado a cabo.

La forma en la cual hemos implementado en nuestro proyecto es mediante la fundación Eclipse Paho, un proyecto de código abierto que facilita su implementación en una gran multitud de lenguajes como podemos observar en la figura 14, mediante esta distribución se cubran gran parte de las necesidades de proyecto. Una de las principales razones de utilizar este proyecto es que facilita la definición de comunicaciones desde el lado del cliente pudiendo especificar las comunicaciones sin la necesidad de generar mucho código, haciendo que el mismo sea más comprensible y eficiente que implementar el propio protocolo MQTT. Además la utilización de este proyecto nos ofrece la posibilidad de un modelo sincrónico y asincrónico, siendo este último indispensable en nuestro proyecto, también ofrece un sistema integrado para asegurar la persistencia de datos entre clientes, cosa que el propio protocolo MQTT no garantiza por sí solo.



MQTT Client Comparison

Client	MQTT 3.1	MQTT 3.1.1	MQTT 5.0	LWT	SSL / TLS	Automatic Reconnect	Offline Buffering	Message Persistence	WebSocket Support	Standard MQTT Support	Blocking API	Non-Blocking API	High Availability
Java	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Python	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗
JavaScript	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓
GoLang	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
C	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
C++	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Rust	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
.Net (C#)	✓	✓	✗	✓	✓	✗	✗	✗	✗	✓	✗	✓	✗
Android Service	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
Embedded C/C++	✓	✓	✗	✓	✓	✗	✗	✗	✗	✓	✓	✓	✗

Figura 14: Tabla de los diferentes lenguajes disponibles en el Eclipse Paho Project y sus respectivas características.

5.3.2. Lenguaje de programación Java[12]

Java es un lenguaje de programación orientado a objetos muy utilizado mundialmente, permite facilidad a la hora de desarrollar programas informáticos, una amplia variedad de los proyectos actuales está basados en él. Utilizaremos este lenguaje principalmente para desarrollar el sistema de comunicación necesario para desplegar nuestra solución, ya que como hemos visto antes Java es el lenguaje sobre el que eclipse Paho nos ofrece más opciones de desarrollo para el protocolo MQTT, esta es la principal razón por la cual nos hemos decantado por utilizarlo, además de la familiaridad, ya que hemos utilizado este lenguaje de programación a lo largo de todo el grado.

5.3.3. Bróker Eclipse Mosquitto[13]

Mosquitto es un bróker MQTT Open Source muy utilizado en el contexto del internet de las cosas (IoT), destaca por su ligereza y facilidad de empleo y despliegue en multitud de escenarios, lo que lo hace ideal para desplegar en cualquier entorno productivo que deseemos independientemente de sus recursos, este es uno de los principios en los que se basa nuestro proyecto. Este bróker será indispensable ya que será el encargado de redirigir los mensajes dependiendo el topic en el que se publiquen, además a través de Eclipse Paho Project también podremos indicarle que comportamiento debe tener hacia los mensajes asegurando o no la persistencia de datos.

5.3.4.JSON[8]

JSON o “JavaScript Object Notation” es un formato de datos que sirve para guardar y enviar información de forma que cualquier persona pueda leerlos, lo hace de una manera estructurada y se usa principalmente para transferir datos entre servidor y cliente. JSON será el formato de datos más importante en nuestro proyecto ya que será el formato de datos sobre el cual se basará nuestros mensajes canónicos para que los cuadros de mandos puedan interpretar la información con facilidad.

La sintaxis de un objeto JSON está basada en dos elementos, por un lado las claves (subrayadas en la figura 13) son cadenas de caracteres (string) y los valores son los datos que pueden ser desde otros objetos JSON, valores numéricos, string … etc. Un objeto JSON comienza siempre por llaves “{” “}” , cada clave es seguida de “:” para distinguirla del valor.

Figura 15: Ejemplo objeto JSON

5.3.5.API Rest[14]

Una API, o interfaz de programación de aplicaciones, es un conjunto de reglas que define como las aplicaciones o dispositivos se han de comunicar entre sí aportando un mecanismo que permite a una aplicación o servicio acceder a un recurso dentro de otra aplicación o recurso, cuando hablamos de una API REST hablamos de una API que cumple los principios de la arquitectura REST. La ventaja de utilizar API REST es que se pueden desarrollar en casi cualquier lenguaje de programación dando soporte a una amplia variedad de datos, pero para que una API sea una API REST se ha de adaptar a seis principios: integridad uniforme, desacoplamiento entre cliente-servidor, sin estado, capacidad de almacenamiento en memoria cache y arquitectura del sistema en capas.

La razón por la que hemos decidido el uso de estas interfaces en nuestro proyecto es debido a la facilidad que otorgan a la hora de acceder a diferentes recursos, en nuestra aplicación utilizaremos esta interfaz para mediante peticiones HTTP guardar información en la BBDD y rescatar información de ella, en concreto los datos de los dispositivos, así como información sobre los mismos.

5.3.6.MySQL

MySQL[15] es un sistema de gestión de bases de datos relacional, es uno de los sistemas más extendidos en la actualidad, esto es en gran parte debido a que es código abierto. En nuestro proyecto será necesario un sistema de almacenamiento de datos para almacenar los datos que los dispositivos vayan generando, de esto se encargará la API que recibirá los datos a través de POST y los inyectara a través de MySQL. Una de las principales características por las que resulta interesantes es su arquitectura cliente servidor, además de su compatibilidad con el lenguaje SQL con el cual estamos familiarizados gracias al grado.

6. Implementación de un prototipo sobre un entorno productivo ficticio

En este apartado abarcaremos la implementación de un prototipo de nuestra aplicación sobre un entorno ficticio, detallando punto a punto las fases o procesos que recorre un mensaje desde el momento en el que se genera en el entorno productivo hasta que por fin llega al cuadro de mandos y se muestra por pantalla.

En primer lugar comenzamos en nuestro módulo fábrica, donde un dispositivo llamado dispositivo_A, en el desempeño de su labor dentro del entorno productivo generará un mensaje con los datos que ha obtenido. En este caso el dispositivo_A genera sus datos en formato JSON. Para generar este mensaje simularemos la obtención de datos por parte de la máquina, utilizaremos una función que genere datos aleatorios para que cada vez se envíe un mensaje diferente, en la figura 16 podemos observar este método llamado generarDatosJSON() , el mensaje generado será como el observado en la figura 17.

```
private static String generadorDatosJSON() {

    JSONObject main = new JSONObject();

    main.put("format", "JSON");
    main.put("device", clientID);
    main.put("production-line", 1);

    LocalDateTime dt= LocalDateTime.now();
    DateTimeFormatter formato = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");
    String tiempo = dt.format(formato);
    main.put("tiempo", tiempo);

    JSONArray array = new JSONArray();

    JSONObject data1 = new JSONObject();
    data1.put("data_name","temperatura-silo");
    data1.put("datatype", "temperatura");
    data1.put("value", (int)Math.floor(Math.random()*(100-(1+1))+(1)));
    data1.put("unit", "grados-centigrados");
    array.put(data1);

    JSONObject data2 = new JSONObject();
    data2.put("data_name","presion-silo");
    data2.put("datatype", "presion");
    data2.put("value", (int)Math.floor(Math.random()*(100-(1+1))+(1)));
    data2.put("unit", "psi");
    array.put(data2);

    main.put("data", array);



    return main.toString();
}
```

Figura 16: Método de generación de datos de un dispositivo en formato JSON

Figura 17: Mensaje JSON generado por dispositivo_A

En segundo lugar hemos definido un cliente MQTT en Java como podemos observar en la figura 18. Previamente, en la fase de diseño, hemos comentado que la aplicación utilizará un bróker, en concreto Mosquitto para MQTT, este bróker se mantendrá a la escucha en el puerto 1883, siendo necesario publicar los mensajes en este puerto. El cliente que hemos desarrollado será desplegado en el dispositivo y publicará en el topic "fabrica/datos", pasando el mensaje para que se trate en el bróker utilizando este topic.

```
public static final String HOST = "tcp://localhost:1883";
public static final String TOPIC= "/fabrica/datos";
private static final String clientID= "dispositivo_A";

public static MqttClient client;
public static MqttMessage message;

public static void main(String[] args) throws MqttException {
    boolean end = false;
    client= new MqttClient(HOST, clientID);
    MqttConnectOptions options = new MqttConnectOptions();
    options.setCleanSession(false);
    options.setConnectionTimeout(10);
    options.setKeepAliveInterval(20);
    client.connect(options);
    message= new MqttMessage();
    message.setQos(0);

    while(end == false) {
        String datos = generadorDatosJSON();
        message.setPayload((datos).getBytes());
        client.publish(TOPIC, message);
        esperar(5);
    }
}
```

Figura 18: Declaración de cliente MQTT en Java que publica cada 5 segundos en el topic "fabrica/datos"

Una vez el mensaje llegue al bróker, viajará a través de él, utilizando el topic "fabrica/datos" para que los suscriptores que el bróker detecte suscritos a este topic puedan recibir el mensaje en el otro extremo. Dependiendo de cómo hallamos definido nuestro cliente MQTT, el bróker estará preparado para tratar los mensajes de una forma distinta.

Cuando el bróker halla reencaminado al otro extremo el mensaje que le ha llegado a través del topic "fabrica/datos", el suscriptor que lo recibirá será un cliente MQTT definido en el módulo de "Procesos de integración para cuadros de mandos" que es el módulo encargado de integrar los datos de los distintos dispositivos del entorno productivo, ya que cada dispositivo publicará sus datos en distinto formato de datos distinto.

El módulo "Procesos de integración para cuadros de mandos" recibirá entonces los mensajes que provengan del topic "fabrica/datos", utilizando el cliente MQTT que podemos observar en la figura 19, en este caso recibirá los mensajes del dispositivo que hemos programado en el módulo de fábrica, el dispositivo_A que enviará mensajes como el observado en la figura 17, no obstante este módulo estará preparado para recibir un mensaje de cualquier dispositivo del entorno productivo.

```
public static final String HOST = "tcp://localhost:1883";
public static final String TOPIC= "/fabrica/datos";
public static final String TOPIC_CUADRO= "/fabrica/cuadrodemandos/mediciones/";
private static final String clientID= "statusModule";

//MQTT UTILS
public static MqttClient client;

public static void main(String[] args) throws MqttException{

    client = new MqttClient(HOST, clientID, null);
    client.connect();
    System.out.println("Cliente conectado");
```

Figura 19: Cliente MQTT del módulo "Procesos de integración para cuadro de mandos"

Para recibir el mensaje que proveniente desde el topic "fabrica/datos" lo primero tendrá que hacer el cliente MQTT es suscribirse al topic y declarar un Listener asincrónico que este a la espera de la llegada de mensajes. Una vez se detecta un mensaje se ejecutará un proceso que llamará al método messageArrived(), todo esto se puede observar en la figura 20.

```
client.subscribe(TOPIC, new IMqttMessageListener() {
    @SuppressWarnings("unchecked")
    public void messageArrived(String topic, MqttMessage message) throws Exception {
        final String payload = new String(message.getPayload());
        switch (formatDetector(payload)) {

        case "JSON": {
            JSONObject jo = tranformadores.msgJsonJsonCanonico(payload);
            System.out.println(jo.get("device"));
            JSONArray ja = (JSONArray) jo.get("data");
            System.out.println("Mostramos el Array-->");
            System.out.println(ja.toString());
            System.out.println("");
            //TRATAMIENTO DEL JSONOBJECT PARA SEPARAR CADA DATATYPE/DATANAME
            for (int i = 0; i < ja.size(); i++) {
                JSONObject ob = (JSONObject) ja.get(i);
                //CONSTRUCCIÓN MENSAJE NUEVO
                JSONObject tmp = new JSONObject();
                tmp.put("data", ob.get("data_name"));
                tmp.put("datatype", ob.get("datatype"));
                tmp.put("productionLine", jo.get("production-line"));
                tmp.put("device",jo.get("device"));
                tmp.put("value", ob.get("value"));
                tmp.put("unit", ob.get("unit"));
                Executors.newSingleThreadScheduledExecutor().execute(new Runnable() {
                    @Override
                    public void run() {
                        try {
                        String tmpTopic = TOPIC_CUADRO;
                        tmpTopic += ob.get("datatype");
                        client.publish(tmpTopic, new MqttMessage((tmp.toString()).getBytes()));
                        }catch (MqttException e){
                            e.printStackTrace();
                        }
                    }
                });
            }
```

Figura 20: Proceso de integración desde que el cliente MQTT recibe el mensaje hasta que lo publica por el respectivo Topic camino al cuadro de mandos

El método messageArrived() lo primero que hará será detectar el formato del mensaje recibido, una vez detectado lo redirigirá a través de un switch, lo hará teniendo en cuenta su formato de datos. Para lograr detectar el formato de datos se hace uso de un método que hemos creado llamado formatDetector() que podemos observar en la figura 21.

```
public static String formatDetector(String e) {
    String s ="";
    boolean salir = false;
    String aux[] = {"JSON", "XML", "CSV"};
    for(int i = 0; i < aux.length || salir; i++) {
        if(e.contains(aux[i])) {
            s=aux[i];
            salir= true;
        }
    }
    return s;
}
```

Figura 21: Método formatDetector() encargado de identificar el formato de datos independientemente del dispositivo

Por lo tanto tras recibirse el mensaje del dispositivo_A que hemos programado, se recibe en el cliente MQTT que está suscrito al topic “fabrica/datos”, luego el detector de formato nos dirá que viene en formato JSON y por último lo redirige al caso “JSON” del switch. Una vez estamos dentro del case JSON, se llevará la integración del mensaje para que el cuadro de mandos sea capaz de entenderlo, para ello enviaremos el mensaje al transformador correspondiente, para transformar mensajes de JSON a JSON canónico, formato que puede procesar el cuadro de mandos, haremos uso del método de la clase transformadores llamado msgJsonJsonCanonico() que podemos ver en la figura X. En este caso lo único que hará nuestro transformador será parsear la información del mensaje de texto a JSON, en casos como que el mensaje sea CSV o XML los transformadores de datos serán mas complejos.

```
public static JSONObject msgJsonJsonCanonico(String payload) throws ParseException {
    Object obj = new JSONParser().parse(payload);
    JSONObject jo = (JSONObject) obj;
    return jo;
}
```

Figura 22: Transformador de mensaje JSON a JSON Canónico

Tras convertir el mensaje a un formato de datos comprensible en siguiente paso de la integración es dividir la información, nuestra solución se basa en los diferentes datos que envían los dispositivos y está preparado para que en caso de que el módulo de integración reciba varias tuplas de datos datatype/dataname y aun así sea capaz de procesarlas. “Status module” creará para cada tupla de datos su mensaje correspondiente, añadiendo los datos que sean relevantes para el cuadro de mandos como pueden ser dispositivo al cual pertenece la información, tipo de datos, nombre del dato, unidad …etc. Teniendo en cuenta el mensaje que proviene del dispositivo_A que hemos podido observar en la figura 17, se generarán dos mensajes que podemos observar en la figura 23.

```
'[
  {
    "id": "dispositivo_A",
    "tiempo": "09/03/2022 19:50:39",
    "productionLine": 1,
    "unit": "grados-centigrados",
    "data_name": "temperatura-silo",
    "datatype": "temperatura",
    "value": 34
  }
]'
```

```
'[
  {
    "id": "dispositivo_A",
    "tiempo": "09/03/2022 19:50:39",
    "productionLine": 1,
    "unit": "psi",
    "data_name": "presion-silo",
    "datatype": "presion",
    "value": 80
  }
]'
```

Figura 23: Mensajes generados tras el procesado de statusModule de un mensaje del dispositivo_A

Cuando se ha realizado este proceso el siguiente paso es reencaminar el mensaje, por su topic correspondiente, que será igual a "fabrica/cuadrodemandos/mediciones/" más el formato de datos del mensaje.

Una vez es reencaminado por el topic correspondiente, pasaremos a el siguiente módulo de la solución llamado "información del cuadro de mandos", el cuadro de mandos que tendrá un cliente MQTT exactamente igual que los previamente vistos, estará a la escucha de todos los mensajes procedentes de "fabrica/cuadrodemandos/mediciones/#". El # es una herramienta de MQTT que facilita la suscripción a varios niveles del topic, gracias a esto seremos capaces de recibir todos los mensajes que vayan destinado al cuadro de mandos, de esta manera además no tendremos que modificar el cuadro de mandos ante la posible llegada de nuevas magnitudes de datos.

Los mensajes se recibirán en el formato de datos predefinido JSON Canónico, una vez recibidos el usuario tan solo habrá de loguearse con un perfil valido y podrá acceder a los mensajes que haya recibido en la pestaña "Dispositivos" donde aparecerá toda la información relativa a los mensajes procedentes de los distintos topic, pudiendo seleccionar por tipo de datos los datos más relevantes en cada momento y colocarlo en la pantalla de la manera que más se ajuste a sus necesidades. Esta pantalla será como un lienzo en blanco en el cual se podrá arrastrar y ajustar los datos que se estén recibiendo a través del topic "fabrica/cuadrodemandos/mediciones/#". En el caso por ejemplo de los datos enviados por el dispositivo_A, que comenzamos enviando desde el módulo fabrica y que han viajado a través de toda la solución, generando los dos mensajes que se pueden observar en la figura 23 que finalmente han llegado hasta nuestro cuadro de mandos. La manera en la cual se integrarían estos datos sería:

1. Se reciben los mensajes a través del cliente MQTT y se inyectarán en nuestra web, que será capaz de generar un cuadro de mandos por usuario registrado.
2. Una vez el usuario del entorno productivo logue en el cuadro de mandos web, se le proporcionará su lienzo en blanco (con sus correspondientes menús).
3. Los mensajes que se han recibido provenientes del dispositivo_A serán seleccionables y visualizables en el cuadro de mandos.

Los mensajes recibidos los podemos observar en el cuadro de mandos en las figuras 24 y 25.

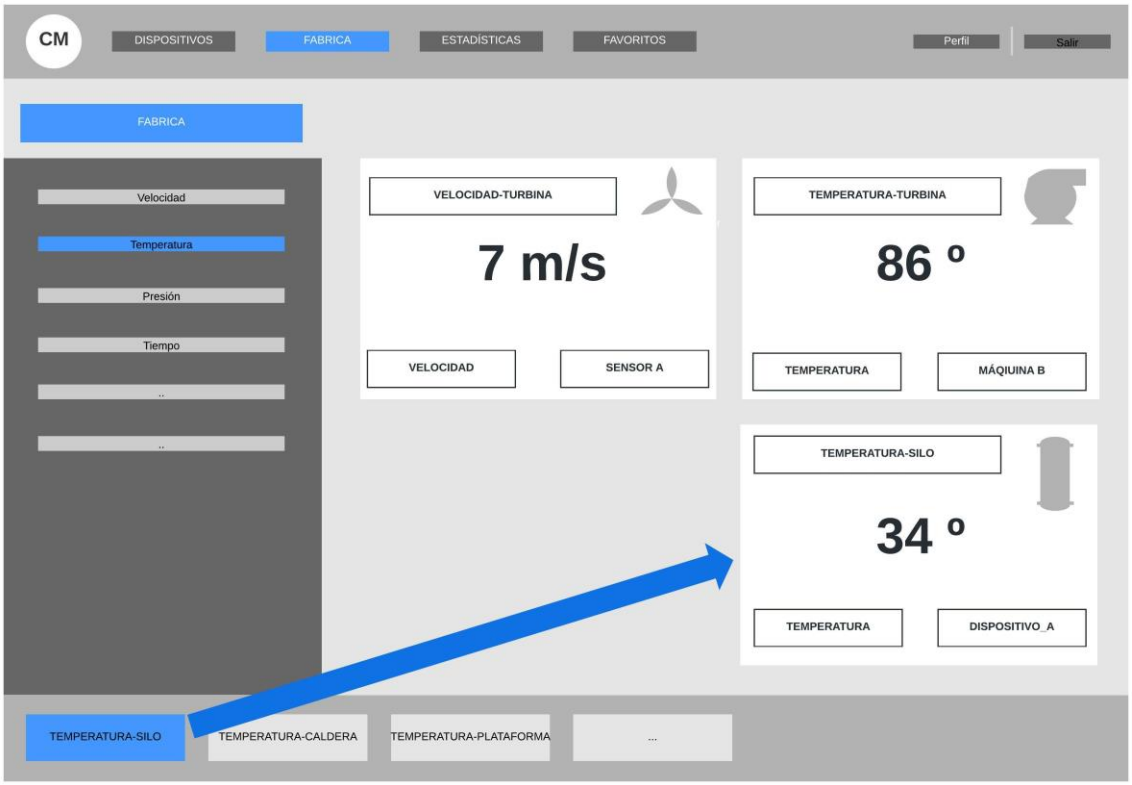


Figura 24: Temperatura-silo del dispositivo_A integrado en el cuadro de mandos

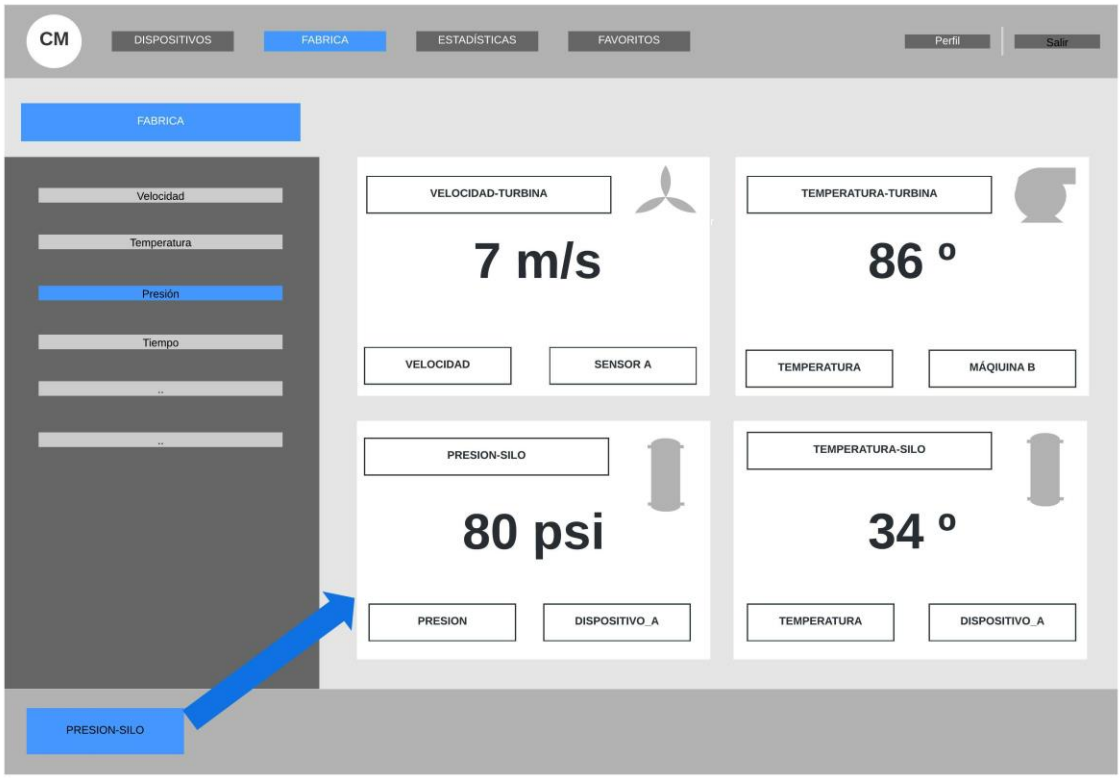


Figura 25: Presión-silo del dispositivo_A integrado en el cuadro de mandos

7. Explotación del prototipo

A lo largo del presente apartado realizaremos una explicación de las posibilidades de nuestra solución, además de explicar las funcionalidades de esta desde el punto de vista del cuadro de mandos web. Primero de todo hemos de remarcar que al ser un cuadro de mandos web el usuario, siempre que tenga autorización, podrá acceder desde cualquier punto y más importante aún desde cualquier dispositivo con acceso a internet, aunque se limitaran las posibilidades debido a las pantallas de estos. Nuestra aplicación será por tanto redimensionable y se adaptará a la superficie disponible en cada pantalla, por lo que no aconsejamos su uso en la pantalla de un móvil ya que no tiene casi superficie aprovechable.

Figura 26: Login cuadro de mandos

Una vez entremos dentro del cuadro de mandos, por ejemplo con un usuario ficticio "cuadro_root", accederemos al menú principal que nos ofrecerá distintas opciones en función de lo que queramos visualizar, este menú lo podemos observar en la figura 27.

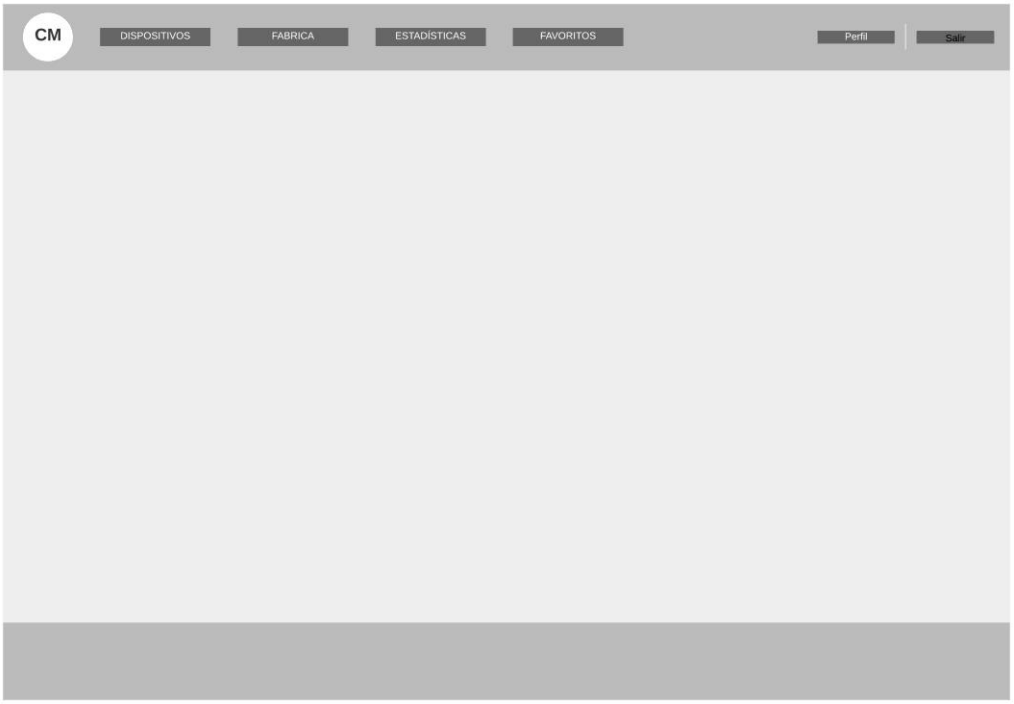


Figura 27: Menú principal del cuadro de mandos

En este menú principal se permitirá utilizar las distintas funcionalidades de nuestro sistema, en primer lugar podemos ver que está compuesto de una barra superior con seis botones:

- Botón “Dispositivos”: A través de este botón se mostrarán en pantalla una lista de los dispositivos que actualmente se encuentran en activo en el entorno productivo, pudiendo acceder clicando en ellos a los datos almacenados en la BDD.
- Botón “Fábrica”: Este botón será el encargado de mostrar un menú con todos los datos que se reciben a través del middleware de comunicaciones que brinda la información al cuadro de mandos como hemos podido ver con mas detalle en el apartado precio.
- Botón “Estadísticas”: Nos ofrecerá estadísticas de nuestro entorno productivo, así como de los dispositivos que lo conforman pudiendo acceder a los históricos registrados en la BDD.
- Botón “Favoritos”: Podremos preconfigurar vistas, de manera que no tengamos que definir vistas iguales cada vez que iniciemos sesión (Solo estarán disponibles si los dispositivos están online, sino mostrará históricos)
- Botón “Perfil”: Se mostrará la información relativa al usuario que ha realizado login en el cuadro de mandos.
- Botón “Salir”: Se podrá volver a la ventana de login.

Vamos a analizar cada uno de los menús resultantes tras clicar en cada uno de los botones disponibles en la pantalla inicial, menos tras clicar el botón salir que tan solo volverá al login que se puede observar en la Figura 26. En primer lugar tras clicar en el botón de "Dispositivos", se nos mostrará una lista de los distintos dispositivos que pertenecen a nuestro entorno productivo y que en algún momento han enviado datos a nuestro cuadro de mandos, como podemos ver en la figura 28. De cada uno de los dispositivos que aparezcan aparecerán los datos relativos a la misma como pueden ser: nombre del dispositivo, tipo de datos, línea de producción …etc.

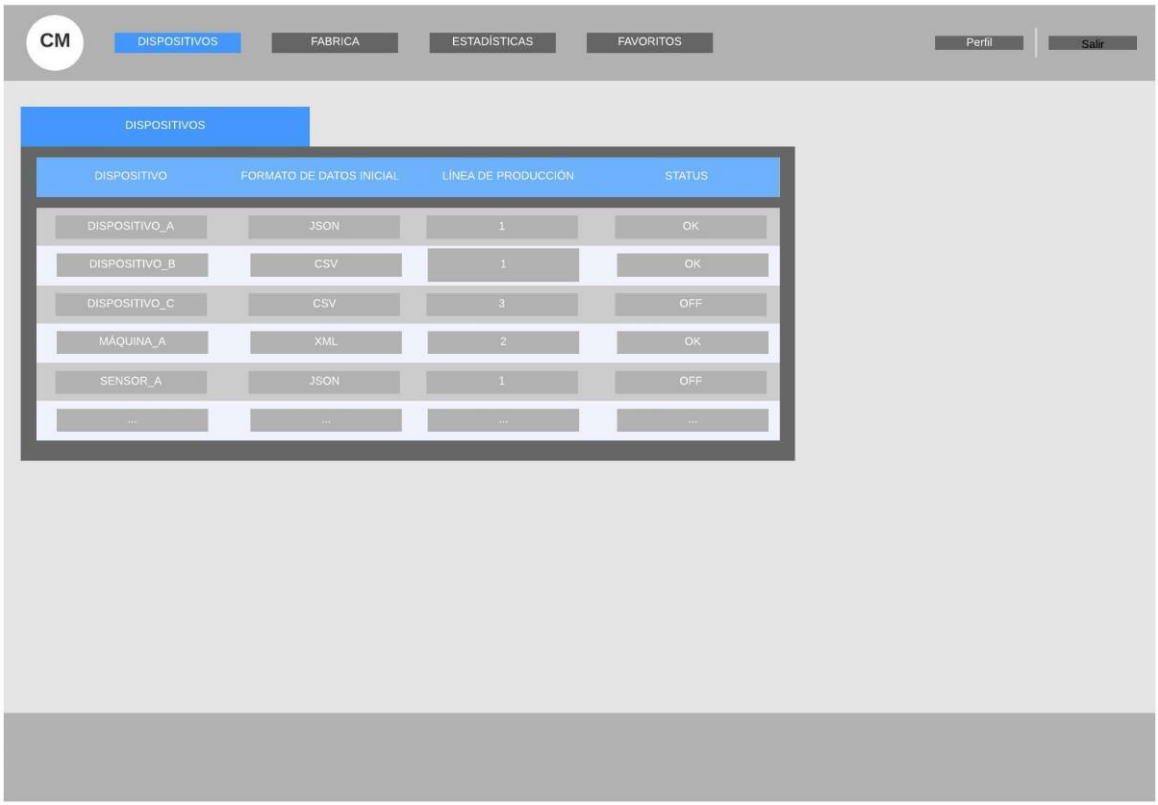


Figura 28: Menú dispositivos en nuestro cuadro de mandos

En segundo lugar tras clicar el botón "Fabrica" accederemos al apartado donde nos aparecerá toda la información que se reciba en tiempo real a través del cuadro de mandos, de aquí podremos extraer la información que llega de los dispositivos permitiendo al usuario final configurarse su propio cuadro de mandos personal. En este menú tendremos la posibilidad de ir seleccionando datos y distribuyéndolos por la pantalla como se puede observar en la Figura 29.

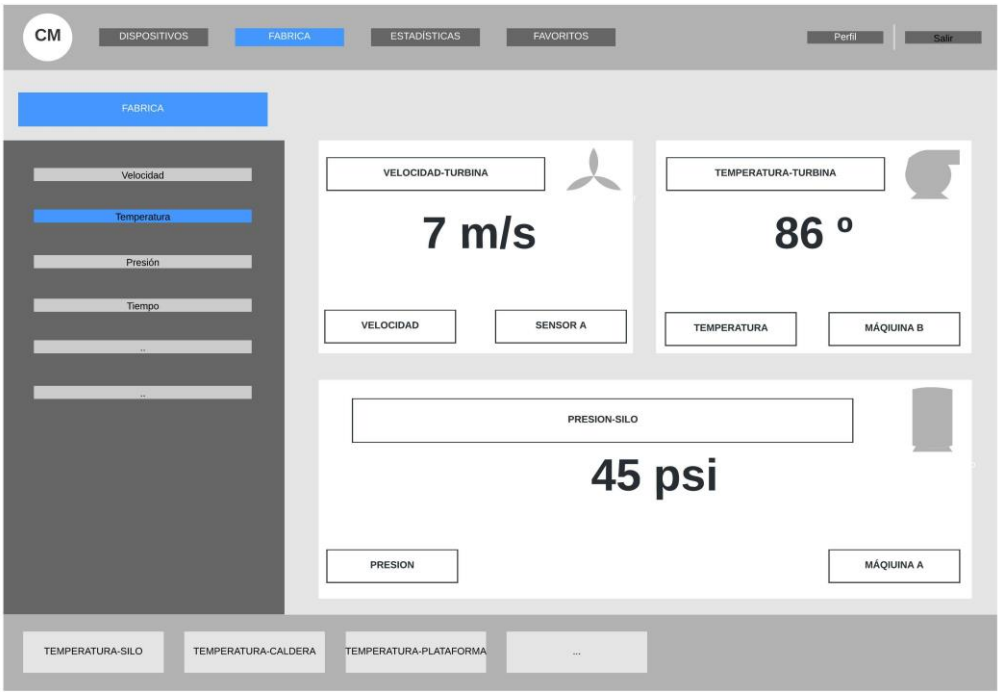


Figura 29: Menú fabrica tras distribuir datos en el lienzo disponible de la pantalla

En tercer encontramos el botón estadísticas, tras clicar como podemos visualizar en la Figura 30 , en esta figura se mostrarán las estadísticas que se construirán mediante consultas a la BBD, desde la propia interfaz del cuadro de mandos se podrá indicar los datos que se quieren visualizar (entre los disponibles en la misma). En la figura podemos ver una serie de datos y el proceso que se ha de seguir para añadir nuevas estadísticas de los datos que nos interesen.

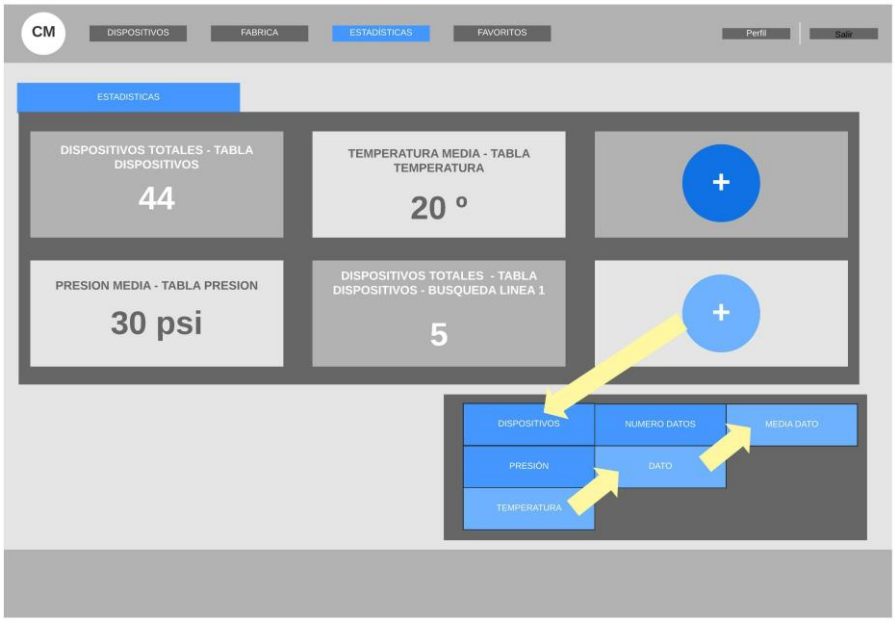


Figura 30: Menú estadísticas, proceso de añadir una estadística nueva

En cuarto lugar encontramos el botón de favoritos, a este menú podremos añadir todas las vistas que vayamos realizando en el cuadro de mandos de esta manera cuando seleccionemos una borrará todo el lienzo y pondrá los datos almacenados en la propia vista. Por ejemplo en la Figura 31 podemos observar como tras clicar en la vista definida como “Temperaturas fabrica” se muestran los datos relativos a la vista.

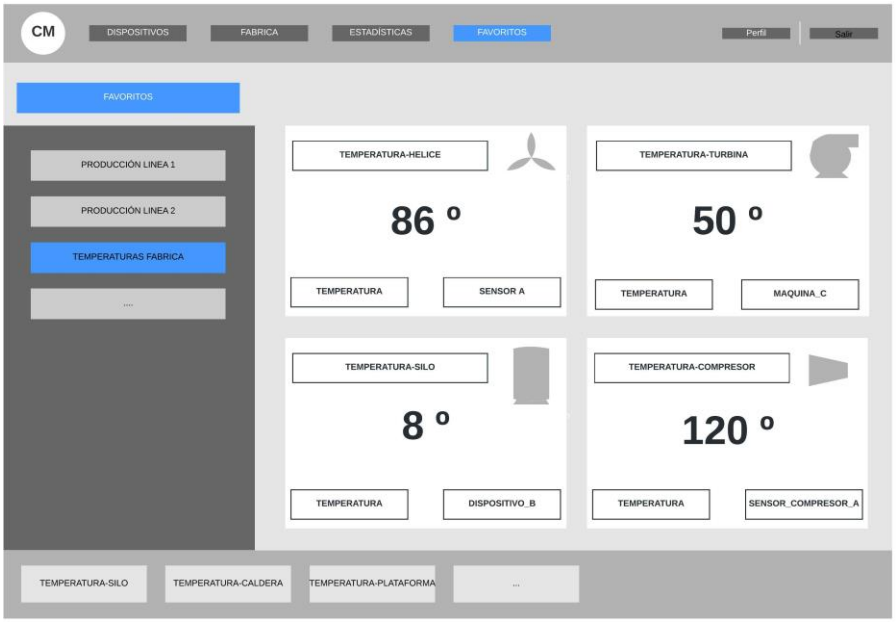


Figura 31: Menú Favoritos, vista predefinida "Temperaturas Fabrica"

Por último, encontramos el botón de Perfil que nos permite ver los datos del usuario que ha accedido a la web del cuadro de mandos, se mostrarían de la manera que se puede observar en la figura 32.

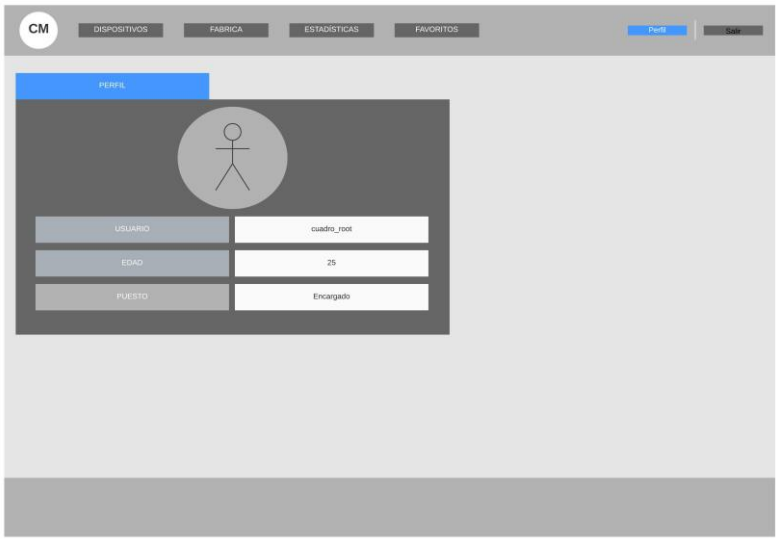


Figura 32: Menú Perfil

8. Conclusiones

Como conclusión a este proyecto, podemos remarcar que se han logrado la mayoría de los objetivos propuestos al comienzo del proyecto con excepción del diseño de una interfaz capaz de mostrar un histórico de los datos que van proporcionando los dispositivos. Aunque por un lado se ha mentado sin profundizar la intención de implementar esta funcionalidad a través de una API REST conectada con la base de datos, lo cierto es que por motivos de logística y tiempo no hemos sido capaces de definir una estructura para la propia base de datos y un protocolo para la comunicación con la API tanto desde el cuadro de mandos como desde el módulo de integración de datos.

Como estrategia a la hora de desarrollar el presente proyecto me he centrado en primera instancia en la definición de una plataforma software que fuera capaz de desplegarse e integrarse en cualquier entorno productivo de una manera sencilla que era el principal objetivo de este, posteriormente procedimos con la definición de un protocolo integración de software de distintos los dispositivos que componen un entorno productivo adaptándolos a un mismo cuadro de mandos.

Para la realización de este proyecto he tenido que profundizar muy a fondo en la ciencia del Internet de las Cosas o IoT, con el objetivo de comprender sus principios básicos y utilizarla como piedra angular en la realización del proyecto de manera que pudiéramos definir comunicaciones de una manera sencilla, haciendo participes a todos los dispositivos del entorno.

A nivel profesional este proyecto me ha ayudado a comprender lo que supone enfrentarse a un proyecto de gran envergadura intentando solucionar los problemas que van surgiendo. Esto ha sido posible gracias a los conocimiento adquiridos durante la carrera, pero también a la investigación de una gran cantidad de ellos que no se han podido profundizar tanto debido a que el mundo de la informática es muy amplio y existen infinidad de tecnologías que se pueden utilizar a cabo para llevar a cabo soluciones a los problemas que se van planteando, por lo que es importante investigar lo suficiente para lograr elegir la solución que mejor se adapte a tus conocimientos y a los requisitos del problema, tratando siempre de no reinventar la rueda y construir la mejor solución posible utilizando trabajo previamente desarrollado.

A la hora de elegir las tecnologías que mejor se adaptaban a la realización de este proyecto me base principalmente en la asignatura IAP “Integración de aplicaciones”, no obstante no

utilizamos las mismas sino los principios en los cuales se basaba esta asignatura, ya que en el ámbito de diferentes entornos productivos encontraríamos en ocasiones algunos de ellos con pocos dispositivos y otros que por el contrario se compondrían de multitud de dispositivos por lo que la tecnología de colas de mensajería que utilizábamos en la asignatura se quedaba pequeña. En su lugar optamos por investigar y profundizar en el protocolo MQTT que se basaba en el modelo publicador/suscriptor pero que aparte de ser una tecnología muy utilizada en el mundo de la IoT aportaba una serie de beneficios indispensables para nuestro proyecto.

Para tratar de definir un sistema de almacenamiento de datos profundizamos en el mundo de las API, en concreto aquellas que siguen la arquitectura REST ya que creímos que serían las que mejor se adaptarían a nuestra solución. La idea de relacionar esta interfaz con una base de datos es con el objetivo de abstraerla, permitiendo que ante futuras modificaciones cualquiera de estos dos elementos se vean en afectados mínimamente por estos cambios, incrementando la escalabilidad y desacoplando la comunicación con los clientes del almacenamiento de datos estrictamente hablando.

En el ámbito personal este proyecto me ha ayudado a potenciar las habilidades que he adquirido durante la carrera, también llamadas “Competencias transversales”, creo que muchas de ellas han sido indispensables para la realización de este proyecto como lo han sido la aplicación y pensamiento práctico, el análisis y resolución de problemas, la innovación, creatividad y emprendimiento, el aprendizaje permanente, y la planificación y gestión del tiempo. También he aprendido a lidiar mejor con la frustración y la incertidumbre, ya que durante la realización del proyecto hubo momentos en los cuales no encontraba la solución a los problemas que se iban presentando, no obstante gracias a la perseverancia poco a poco iba avanzando.

Uno de los errores que más factura han pasado a este proyecto es la gestión del tiempo, ya que en muchas ocasiones había tantas posibilidades y tecnologías que el simple hecho de contemplar cada una de ellas me hizo perder mucho tiempo, por lo que hubiera sido más importante ceñirse a los plazos de entrega. Además al ser un proyecto ambicioso también he tenido que aprender muchas tecnologías, procedimientos y protocolos, entre los que se encuentran MQTT, las API REST (que aunque las había visto anteriormente durante la carrera nunca había intentado desarrollar una por lo que me forme en nodeJS con su librería expressJS) y conocimientos en desarrollo web ya que nunca había afrontado la realización de un proyecto web de tanta envergadura.

9. Trabajos Futuros

Al ser un proyecto que trataba de abarcar varias áreas de estudio como lo son la integración de aplicaciones, el desarrollo web, bases de datos… ha habido temas en los cuales no se han podido profundizar de la manera adecuada teniendo en cuenta los tiempos definidos en el TFG. En un futuro me gustaría profundizar y desarrollar los siguientes aspectos: plataforma web, almacenamiento de datos y tratamiento de más formatos de datos, creemos que serán indispensables para el buen funcionamiento en cualquier entorno productivo en el futuro.

Al intentar permitir al usuario crear su propio cuadro de mandos configurables se requiere un amplio conocimiento en el desarrollo web por lo que hay que investigar en profundidad este tema para elegir las mejores tecnologías para lograrlo en el proyecto actual hemos hecho uso tan solo de JavaScript, CSS y HTML, por lo que creemos que seria interesante en un futuro implementar el uso de un Framework actual que pueda aportarle la funcionalidad que requiere nuestros cuadros de mandos.

Por otro lado y uno de los aspectos que más verdes han quedado en nuestro proyecto sería el almacenamiento y gestión de datos, como se concluyó en el contexto tecnológico para los entornos productivos era de vital importancia la gestión de múltiples datos para poder aprovecharlos en su propio beneficio, debido a las necesidades cambiantes del entorno. Me gustaría indagar mas a fondo en el uso de las API REST así configurar una que sea capaz de cumplir con todas las expectativas que se plantearon en el proyecto. También seria importante definir una estructura para la base de datos que estaría conectada a la API y crear un protocolo para el volcado de datos de la API en la BDD y la consulta de datos de la API a la BDD ante las peticiones del cuadro de mandos web.

Otro aspecto importante que desarrollar es la implementación de más transformadores de datos capaces de soportar los distintos formatos de datos, cosa que será indispensable en un entorno productivo real porque existen multitud de dispositivos que lo componen.

Además mejorar estos aspectos también creemos que es importante implementar este modelo en un entorno productivo real para así detectar y analizar requisitos que no hallamos contemplado, así como fallos en nuestra propuesta, de manera que podamos mejorarlas, ya que al fin y al cabo en un entorno productivo real siempre surgen distintas necesidades que nuestro proyecto debe ser capaz de cubrir.

Por último, es importante mejorar y optimizar los componentes de nuestra aplicación tras la puesta en escena en un entorno productivo, ya que aunque ha sido diseñado para funcionar de la manera más optima es importante hacer un ejercicio de autocrítica.

10. Referencias

[1] A. Ali Laghari, K. Wu, R. Ali Laghari, M. Ali, y A. Ayub Khan, «A Review and State of Art of Internet of Things (IoT)», *Arch. Comput. Methods Eng.*, vol. 1, p. 3, doi: 10.1007/s11831-021-09622-6.

[2] «Sistemas SCADA - Aquilino Rodríguez Penin - Google Libros». https://books.google.es/books?hl=es&lr=&id=cNQfjbBcUq8C&oi=fnd&pg=PA1&dq=scada&ots=4HOTuEHVVv&sig=oHNGJCwhyC74nBYfvQsu7BooxQM#v=onepage&q&f=false.

[3] M. Morandini, T. A. Coleti, E. Oliveira, y P. L. P. Corrêa, «Considerations about the efficiency and sufficiency of the utilization of the Scrum methodology: A survey for analyzing results for development teams», *Comput. Sci. Rev.*, vol. 39, p. 100314, feb. 2021, doi: 10.1016/J.COSREV.2020.100314.

[4] «Trello». https://trello.com/.

[5] R. Hunzinger, «SCADA FUNDAMENTALS AND APPLICATIONS IN THE IoT», 2017.

[6] S. Sen y A. Balasubramanian, *A Highly Resilient and Scalable Broker architecture for IoT applications*. 2018.

[7] M. O. Al Enany, H. M. Harb, y G. Attiya, «A Comparative analysis of MQTT and IoT application protocols; A Comparative analysis of MQTT and IoT application protocols», vol. 2021, pp. 3-4, 2021, doi: 10.1109/ICEEM52022.2021.9480384.

[8] «JSON». https://www.json.org/json-en.html.

[9] «Introducción a XML - XML: Extensible Markup Language | MDN». https://developer.mozilla.org/es/docs/Web/XML/XML_introduction.

[10] «Formato de archivo CSV y ejemplos - Documentación de IBM». https://www.ibm.com/docs/es/elm/6.0.3?topic=files-csv-file-format-examples.

[11] «MQTT - The Standard for IoT Messaging». https://mqtt.org/.

[12] «¿Qué es Java y para qué es necesario?» https://www.java.com/es/download/help/whatis_java.html.

[13] «Eclipse Mosquitto». https://mosquitto.org/.

[14] R. T. Fielding *et al.*, «Reflections on the REST Architectural Style and ``Principled Design of the Modern Web Architecture'' (Impact Paper Award)», doi: 10.1145/3106237.3121282.

[15] «MySQL». https://www.mysql.com/.

Anexo 1 – Objetivos de desarrollo sostenible

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**				X
ODS 4. **Educación de calidad.**				X
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**				X
ODS 9. **Industria, innovación e infraestructuras.**	X			
ODS 10. **Reducción de las desigualdades**.				X
ODS 11. **Ciudades y comunidades sostenibles**.			X	
ODS 12. **Producción y consumo responsables.**		X		
ODS 13. **Acción por el clima.**				X
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**				X
ODS 17. **Alianzas para lograr objetivos.**				X

Reflexión sobre la relación del TFG con los ODS más relacionados

El 25 de septiembre de 2015, los líderes mundiales adoptaron un conjunto de objetivos globales para erradicar la pobreza, proteger el planeta y asegurar la prosperidad para todos como parte de una nueva agenda de desarrollo sostenible. Cada objetivo tiene metas específicas que deben alcanzarse en los próximos 15 años.

1. Fin de la pobreza
2. Hambre Cero
3. Salud y Bienestar
4. Educación de Calidad
5. Igualdad de género
6. Agua limpia y saneamiento
7. Energía asequible y no contaminante
8. Trabajo decente y crecimiento económico
9. Industria innovación e infraestructura
10. Reducción de las desigualdades
11. Ciudades y comunidades sostenibles
12. Producción y consumos responsables
13. Acción por el clima
14. Vida submarina
15. Vida de ecosistemas terrestres
16. Paz, justicia e instituciones
17. Alianzas para lograr objetivos.

De los objetivos de desarrollo sostenibles listados previamente, nuestro proyecto está relacionado con:

- **Industria innovación e infraestructura**, personalmente creo que nuestro proyecto ayuda a la industria a renovarse ya que ofrece un modelo diferente al existente actualmente brindando la posibilidad a los entornos productivos de modificar su infraestructura para conseguir optimizar sus procesos. Nuestro proyecto se centra en la posibilidad de optimizar estos procesos, ofreciendo una interfaz eficiente y adaptable para los usuarios del entorno productivo. Todo esto hace que el propio entorno productivo gestione sus datos de manera que sea capaz de analizarlos y mantenerse en un entorno cada vez más competitivo. Además mediante su integración conseguiremos una industria mucho más productiva, ya que optimizará sus sistemas, lo que logrará que poco a poco que sean menos contaminantes.

- **Ciudades y comunidades sostenibles**, nuestro proyecto está pensado para implementarse en cualquier entorno productivo, por lo que implementándose

correctamente en cualquier central eléctrica o hidráulica podría suponer una administración y monitorización correcta de la energía que se genera en ellas. Por ejemplo permitiendo a saber en qué partes del proceso de fabricación de la energía se pierde más tiempo o recursos, de qué manera se podría optimizar el proceso de fabricación… etc. Todo esto podría traducirse en un ahorro de energía debido a la redirección de recursos a las áreas de producción que más lo necesiten, así como la producción de manera mas sostenible de la propia energía que indirectamente contribuyen al correcto desarrollo de las ciudades ya que cada día más la gestión y producción de energía es un problema más acentuado en la sociedad actual.

- **Producción y consumos responsables**, nuestro proyecto de una manera indirecta propone una solución que ofrece la posibilidad a los entornos productivos de desarrollar su cadena de producción de una manera más responsable, esto es debido a que a través de la monitorización continua y eficiente serán capaces de detectar aquellas áreas en las cuales se podrían optimizar los procesos de fabricación. Además, al optimizar estos procesos se produciría inevitablemente un ahorro de energía y materiales, ahorrándose costes y aún más importante haciendo un uso inteligente de los mismos, ya que una mala gestión de estos en su producción crearía sistemas poco sostenibles y que no respetaran el entorno.