UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseñando una fábrica del Futuro con Internet de las Cosas. Una experiencia práctica.

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Miguel Tortosa Calabuig

Tutor: Joan Josep Fons i Cors

Curso 2018-2019

Resumen

Actualmente muchas empresas están considerando en automatizar al máximo las tareas que se realizan dentro de las fábricas, desde que entra la materia prima, hasta finalizar todos los procesos requeridos para convertir dicha materia en un producto final. Este concepto esta conocido como **Industria 4.0** (industria inteligente).

Esta revolución aporta a la empresa nuevas tecnologías como la analítica, la **inteligencia artificial**, **nanotecnología**, el **IOT** (internet de las cosas) y la robótica entre otras.

Uno de los problemas principales que nos encontramos en el IOT, es que todos los recursos que debemos utilizar para comunicar cada proceso virtual de la producción suelen hacerse con tecnologías completamente diferentes y con información cuya estructura es también diferente. Por lo tanto, es complicado que todos los sistemas tengan una comunicación e integración fácil de implementar.

En este documento mostraremos una experiencia práctica de cómo automatizar los procesos para hacer más eficiente la cadena de producción y mejorar la experiencia de trabajo para los empleados de la empresa EMBALPACK.

Palabras clave: Industria 4.0, IOT, inteligencia artificial, nanotecnología, robótica.

Abstract

Nowadays, a lot of companies are considering automate the processes that the company has. Starting from the input of the raw materials all the way up to transforming those inputs into final products. This concept is known as Industry 4.0 or smart industry.

This is a revolution that involves analytics, artificial intelligence, nanotechnology, IOT (Internet of Things), robotics, etc.,

One of the main problems that we have to face with the IOT is that all the communications that we have to use between each virtual production process, are normally done with completely different technologies and structure. Therefore, is highly complicated that all systems have an efficient communication and easy integration.

During the current document we will show a practical case of how to automate all the processes in order to make all the production process more efficient and to ease and improve work experience for employees to the company EMBALPACK.

Keywords: Industry 4.0, Artificial Intelligence, nanotechnology, robotics.

Resum

Actualment moltes empreses estan considerant a automatitzar al màxim les tasques que es fan dins l'empresa, des que entra la matèria primera fins que acaben tots els processos per a convertir aquesta matèria en el producte final. Aquest concepte és conegut amb el nom d'Indústria 4.0 (indústria intel·ligent).

Aquesta revolució aporta a l'empresa noves tecnologies com ara l'analítica, la intel·ligència artificial, la nanotecnologia, l'IOT (l'internet de les coses) i la robòtica, entre altres.

Un dels problemes principals que ens trobem en l'IOT és que totes les comunicacions que fem servir per a comunicar cada procés virtual de la producció solen fer-se amb tecnologies completament diferents, i l'estructura de la seua informació també és diferent. Per tant, és complicat que tots els sistemes tinguen una comunicació i una integració fàcils d'implementar.

En aquest document mostrarem una experiència pràctica de com automatitzar els processos per a fer més eficient la cadena de producció i millorar l'experiència de treball per als empleats de l'empresa EMBALPACK.

Paraules clau: Indústria 4.0, IOT, intel·ligència artificial, nanotecnologia, robòtica.

Tabla de Contenidos

1. Introducción 9
1.1 Motivación 11
1.2 Objetivos 11
1.3 Metodología 12
2. Contexto Tecnológico 13
2.1 ¿Qué es IOT? 13
2.2 Herramienta De Desarrollo: Intellij Pycharm 14
2.3 Arquitectura del proyecto: Basada en Microservicios. 16
2.3.1 ¿Qué son los Microservicios? 16
2.4 Hardware de simulación: Intel Galileo 17
2.5 Lenguaje de programación: Python 18
2.6 Tecnologías 19
2.6.1 Despliegue: DOCKER 19
2.7 Tipo de datos de los mensajes entre los servicios. 21
2.7.1 JSON 21
2.8 Protocolo de Comunicación 22
2.8.1 HTTP 22
2.8.2 MQTT 23
2.8.3 REST 23
2.9 Broker De Mensajería: mosquitto 24
3. Caso de Estudio 27
3.1. Introducción 27
3.2. Descripción 28
3.3. Análisis de la Solución. 29
4. Diseño de la Solución 31
4.1 Diseño de la base de datos 34
4.1.1 Trazabilidad 34
4.1.2 Identificación 34
5. Implementación de la Solución. 37
5.1 Implementación microservicio identificación 37
5.2. Implementación Microservicio Trazabilidad 41
5.3. Implementación máquina generadora de tortas. 42
6. Conclusión 45
7. Bibliografía 47
8. Anexos 49

Tabla de Figuras

Figura 1 Esquema industria 4.0 10
Figura 2 diagrama de gantt 12
Figura 3 IOT 13
Figura 4 Logo PyCharm 14
Figura 5 Esquema IoT 16
Figura 6 Imagen intel galileo 17
Figura 7 Logo Python 18
Figura 8 Logo Docker 19
Figura 9 Logo Json 21
Figura 10 Logo rest 23
Figura 11 Logo mosquitto 24
Figura 12 esquema virtual embalpack 28
Figura 13 esquema grafico implementación 30
Figura 14 muestra del contenido recibido 31
Figura 15 Muestra del contenido enviado 32
Figura 16 muestra código identificación 37
Figura 17 mensaje recibido petición http 37
Figura 18 pedir resources postman 38
Figura 19 pedir un recurso especifico POSTMAN 39
Figura 20 pedir todos los resources de tipo bob 39
Figura 21 ejemplo código trazabilidad 40
Figura 22 código maquina generadora de tortas 41
Figura 23 Ejemplo mensaje enviado al tópico de la maquina 42
Figura 24 kitematic 50
Figura 25 kitematic funcionando 51

1. Introducción

En la actualidad la expresión “Industria 4.0” es cada vez más conocida, poco a poco las empresas tienden a automatizar todos sus procesos y hacer las cadenas de producción más inteligentes y eficientes.

Las empresas que ya han dado el salto a la industria 4.0 están muy relacionadas con el Internet de las Cosas (IoT), gracias a este concepto se pueden desarrollar todos los mecanismos posibles para la revolución de la industria 4.0.

[1] Con esta revolución podemos decir que en las fábricas, el mundo virtual y real se fusionan, con lo cual la industria 4.0 es la digitalización de la fabricación y todos los servicios que conlleva, por lo tanto, las instalaciones se gestionan de forma mucho más autónoma, flexibilizando respuestas y atendiendo más rápidamente a las demandas del mercado

Entonces, ¿estamos ante la cuarta revolución industrial?

Pues bien, la primera revolución industrial apareció con el sistema mecanizado a través del vapor y la tracción hidráulica en 1784, la segunda apareció tras la primera cinta transportadora, posteriormente surgió la tercera revolución tras conseguir el primer controlador programable.

Así que, sin duda hoy en día nos encontramos ante la cuarta revolución industrial ya que la producción está totalmente digitalizada, automatizada y se comunica a través del IoT tal y como veremos en la Figura 1.

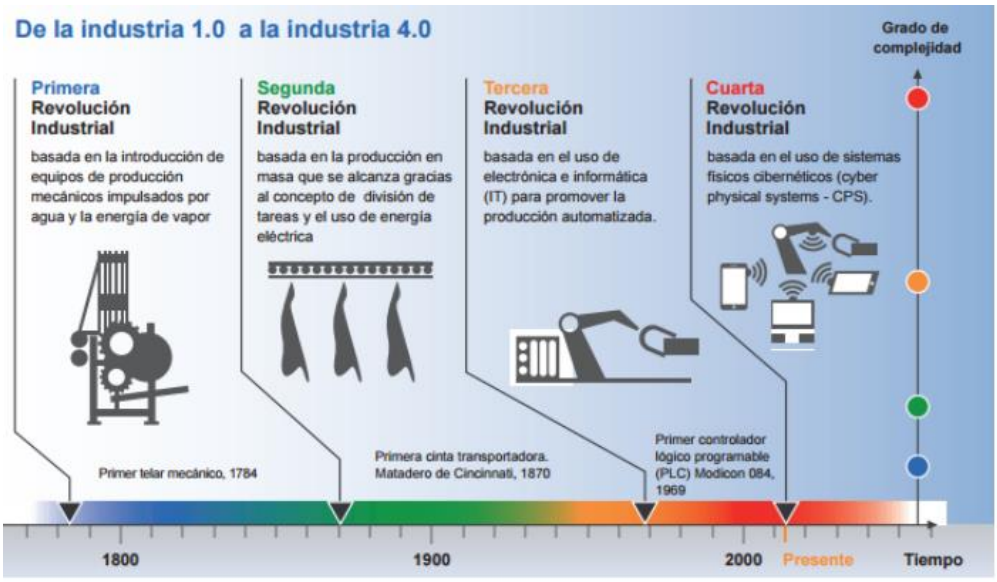


FIGURA 1 ESQUEMA INDUSTRIA 4.0

[3] Esta situación, por un lado, conlleva una serie de ventajas, ya que se obtienen procesos más depurados y sin errores ni alteraciones debido a que se reduce significativamente el error humano. Además, conseguimos una producción ininterrumpida y disponible las 24 horas del día. También se acortan drásticamente los tiempos de producción y otra ventaja destacable es que podemos monitorizar todos los procesos y sacar datos en tiempo real o realizar estadísticas al detalle en cada parte del proceso de producción, desde el aprovisionamiento de las materias primas, el almacenamiento, la transformación de materias primas en producto final y su distribución.

Sin embargo, la industria 4.0 también tiene algunas desventajas, como por ejemplo los avances industriales a excesiva velocidad que pueden derivar en que la caducidad de las innovaciones sea cada vez más corta con lo que supone para una empresa una inversión en I+D constante y onerosa.

Además, debido a la automatización del proceso productivo, la mano de obra poco cualificada estará en desventaja frente a las generaciones preparadas. Es probable que crezcan las desigualdades y que exista cierta fragmentación social a corto plazo. Finalmente, como es lógico, hay que tomar en cuenta los altos costes que conlleva automatizar toda la producción de una empresa.

1.1 Motivación

Mi motivación para inclinarme sobre el tema de la industria 4.0 para mi trabajo fin de grado, radica en ser pionero y dar a conocer, al menos en un entorno local, este nuevo paradigma que al parecer en poco tiempo va a revolucionar más nuestras empresas y por lo tanto nuestras vidas.

Lo que está claro, es que países como Holanda, Alemania, Suecia y muchos países asiáticos altamente industrializados como Japón y otros, están apostando fuertemente por la automatización de sus empresas y eso es debido a que la mano de obra poco cualificada es cada vez menor en estos países, lo cual nos indica que una industria 4.0 o industria automatizada no solo reduce los costes de producción como es obvio, sino que también crea empleos cualificados y fomenta altamente la formación profesional en las generaciones más jóvenes.

Es por eso por lo que ahora más que nunca, en España deberíamos decidir si queremos aprovechar esta gran oportunidad para a acercarnos a los países más desarrollados o queremos resignarnos una vez más.

1.2 Objetivos

El objetivo principal del proyecto es Identificar los retos que existen en la implantación de soluciones IoT en una fábrica/entorno industrial real. Vamos a introducir o extender las capacidades de comunicación entre los diferentes recursos dentro de una fábrica, comunicándolos entre si aplicando IoT e introduciendo cierta inteligencia. Para poder realizar este objetivo, tendremos que introducir una tecnología que permita establecer comunicaciones fluidas, enriquecer recursos de la fábrica para que puedan computar y comunicarse. Es decir, crear un contexto de producción digital, donde podamos ver los objetivos de producción esperados y los producidos realmente.

Para realizar el objetivo principal, destacaremos 2 subobjetivos:

1) La solución arquitectónica que abordará estos retos será crear la infraestructura necesaria para poder comunicar de manera digital todos los procesos de una cadena de producción (máquinas personas, materia prima), desde la entrada de materia prima hasta la creación del producto final y dotarlos de funcionalidades inteligentes.

2) Realizaremos una pequeña validación de la propuesta a través del desarrollo de un caso de estudio en concreto y real, con dispositivos que podrían utilizarse a nivel industrial.

1.3 Metodología

Utilizaremos una metodología de desarrollo clásico, donde nos hemos organizado de la siguiente manera.

1) Investigación y aprendizaje (5 semanas): principalmente dedicamos el esfuerzo en entender como comunicar-se los diferentes servicios, vía colas de mensajería o servicios REST
2) Caso de estudio (5 semanas): Se plantea el caso de estudio, y se determina cual es la manera óptima para implementarlo
3) Diseño e implementación (4 semanas): propuesta, configuración e implementación necesaria para plantear el caso de estudio. Hemos utilizado para las comunicaciones 2 tipos de protocolos, servicios API-REST creado con Python y MQTT para comunicarse mediante colas de mensajería, este también implementándolo en Python. Para persistir nuestros datos hemos instalado y configurado una base de Datos MYSQL.

Podremos observarlo de una manera más gráfica en el diagrama de grantt que podremos ver en la Figura 2

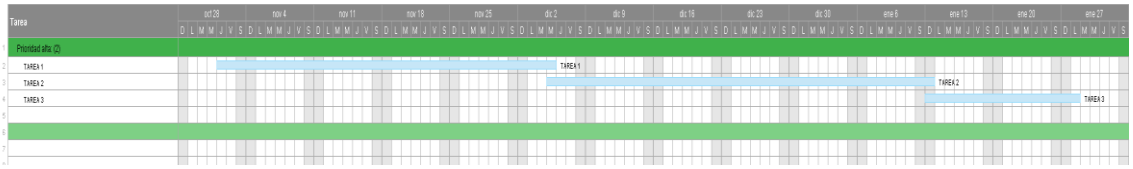


FIGURA 2 DIAGRAMA DE GANTT

2. Contexto Tecnológico

Una vez realizada la introducción, procedemos primero a comprender más a fondo qué significa IOT y posteriormente describiremos las herramientas, tecnologías y protocolos utilizados para desarrollar el proyecto. Estas no son las únicas que podríamos haber utilizado para la elaboración de nuestro proyecto, pero hemos realizado un pequeño estudio de cada alternativa y por motivos de eficiencia y facilidad para construir nuestro proyecto que describiremos durante este apartado.

2.1 ¿Qué es IOT?

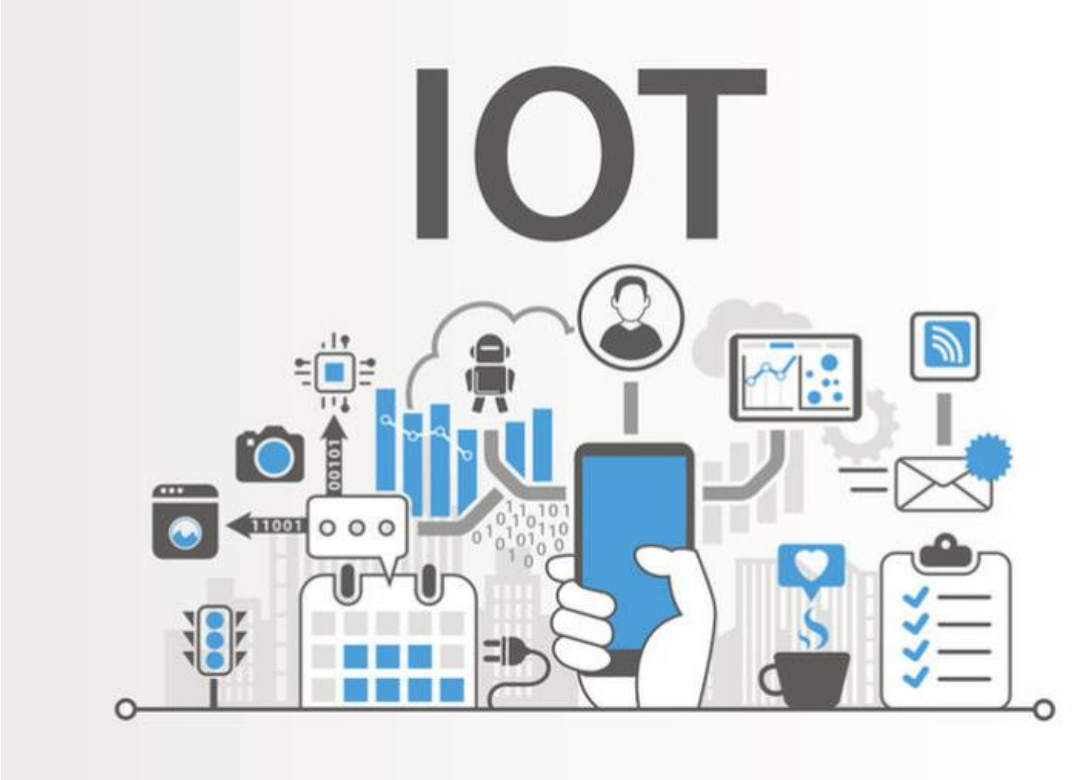


FIGURA 3 IOT

IoT viene del inglés "Internet of Things" y se refiere a la interconexión de dispositivos a través de una red, donde los objetos pueden comunicarse e interactuar sin necesidad de intervención humana.

Esta evolución de la tecnología se ha creado con la intención de mejorar y facilitar la vida de las personas, así como la vida empresarial debido a la infinidad de aplicaciones y utilidades que se le pueden dar, desde encender la calefacción con un móvil o automatizar una cadena de producción, hasta mejorar el control del tráfico implantando "smart cities", etc.

Sin el concepto de IOT no habrá ni cuarta revolución industrial, ni industria 4.0, ni fabricas inteligentes.

2.2 Herramienta De Desarrollo: Intellij Pycharm

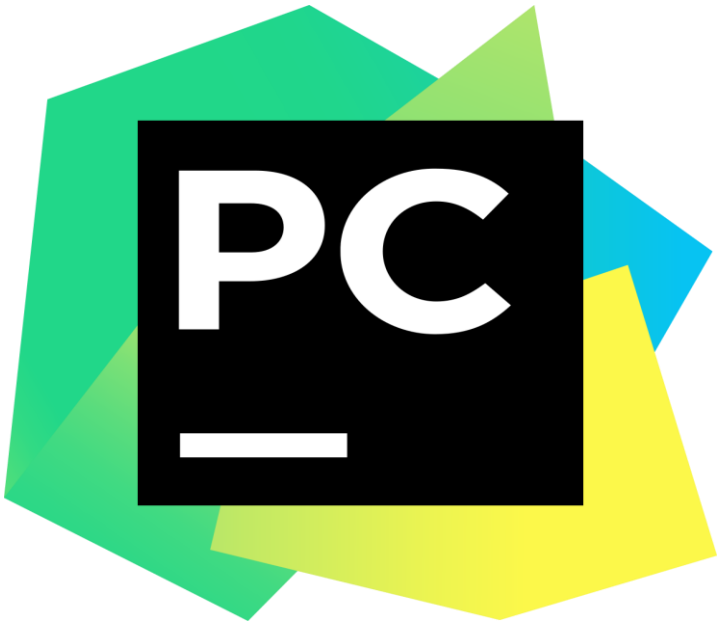


FIGURA 4 LOGO PYCHARM

PyCharm - *[6]* Python IDE es un conjunto completo de herramientas para el desarrollo productivo con el lenguaje de programación Python. Además, el IDE proporciona capacidades de alto nivel para el desarrollo web profesional con el marco Django y Google App Engine. Cuenta con una potente asistencia de codificación, navegación, una gran cantidad de funciones de refactorización, una estrecha integración con varios sistemas de control de versiones, pruebas de unidades, un potente depurador y una personalización muy completa. PyCharm es un IDE impulsado por desarrolladores. Fue desarrollado con el objetivo de brindarle casi todo lo que necesita para su desarrollo cómodo y productivo.

Después de esta descripción proporcionada por la empresa propietaria del software, destacaremos porque hemos utilizado esta herramienta:

1) Es un software propietario de pago, pero la empresa facilita una licencia de educación, gratuita, que hemos aprovechado para realizar el desarrollo.
2) Editor inteligente que nos facilita el autocompletado de código. Tiene atajos de teclado que nos permite navegar entre código o navegar entre clases y así ser más productivos en el desarrollo.

3) Una de las características más importantes de PyCharm es la posibilidad que tiene de refactorizar el código, esto significa modificar el código sin comprometer la ejecución de este.
4) Hay una gran cantidad de desarrolladores que trabajan con PyCharm y que genera una cantidad grande de complementos y temas, para poder ajustar aún más si cabe PyCharm a nuestro trabajo.

Alternativas a PyCharm:

1) Eclipse: Es un IDE muy completo y de código abierto, pero para nuestro desarrollo debíamos instalar una serie de complementos como, por ejemplo: Pydev, para que fuese compatible con Python, por lo tanto, se descartó.
2) Eric: Es otro IDE, pero es más incompleto que Eclipse o Pycharm, menos intuitivo y se detectó que podría penalizarnos en tiempo de desarrollo por lo tanto se decidió no utilizarlo.
3) Por último, podíamos haber utilizado cualquier editor de texto, esta opción se descartó, porque con un IDE tienes muchas facilidades para aumentar el rendimiento del desarrollo y su calidad.

2.3 Arquitectura del proyecto: Basada en Microservicios.

Durante esta sección explicaremos brevemente que son los microservicios y como los vamos a utilizar en nuestro proyecto.

2.3.1 ¿Qué son los Microservicios?

Una arquitectura de microservicios es una aplicación software compuesta por pequeños servicios, que son ejecutados de forma autónoma, desacoplada y comunicándose entre sí.

Hemos optado por este tipo de arquitectura, ya que encaja muy bien con el concepto de industria 4.0, debido a que 2 servicios pueden estar trabajando conjuntamente para llegar a un fin específico, pero a su vez alguno de estos puede están trabajando con un tercero para la consecución de un fin distinto. Veremos un ejemplo en la siguiente Figura 5

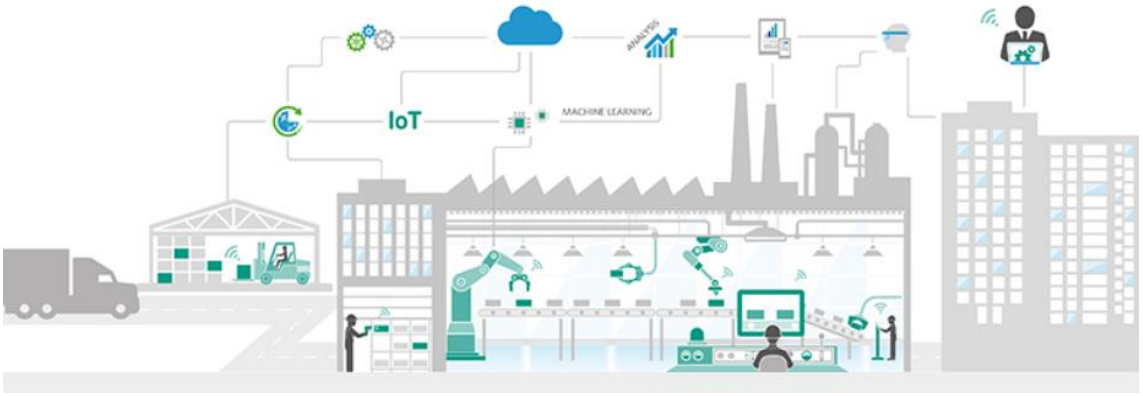


FIGURA 5 ESQUEMA IOT

2.4 Hardware de simulación: Intel Galileo

FIGURA 6 IMAGEN INTEL GALILEO

La placa de desarrollo Intel Galileo Gen 2 es la segunda de una familia de placas de desarrollo y para prototipos certificada por Arduino basada en la arquitectura Intel y diseñada específicamente para creadores, estudiantes, educadores y entusiastas de la electrónica.

La placa Intel Galileo Gen 2 proporciona a los usuarios un entorno de desarrollo de software y hardware de código totalmente abierto. Además, complementa y amplía la línea de productos Arduino para ofrecer funciones informáticas más avanzadas a los usuarios ya familiarizados con las herramientas de prototipo Arduino.

Por sus características de diseño, la placa de desarrollo Intel Galileo Gen 2 viene con software, hardware y pines compatibles con una amplia gama de protectores Arduino Uno R3. Veremos cómo son dichas placas en la Figura 6

Por otra parte, permite a los usuarios incorporar llamadas de firmware Linux en la programación para bocetos de Arduino.

Después de un amplio estudio de qué hardware se acoplaba más a nuestro proyecto hemos detectado, que, instalando Linux a las Intel galileo, conseguimos un hardware muy completo a un buen precio y con muchas posibilidades de desarrollar satisfactoriamente nuestro proyecto.

2.5 Lenguaje de programación: Python

Este proyecto ha sido desarrollado íntegramente con Python, tanto para crear el API rest, como para la simulación de todos los procesos virtualizados de una empresa.

FIGURA 7 LOGO PYTHON

[9] Python es un potente lenguaje de programación, que es muy fácil de utilizar. Tiene estructuras de datos de alto nivel muy eficientes y una simple pero efectiva programación con un enfoque orientado a objetos

Python tiene una elegante sintaxis y un tipado dinámico que lo hacen un lenguaje ideal para el script y para un rápido desarrollo en distintas áreas de muchas plataformas.

El intérprete de Python es fácilmente extendido a muchas funciones. Python es adecuada como una extensión de lenguaje para aplicaciones personalizables.

Después de una larga tarea de investigación elegimos Python ya que es un lenguaje rápido, robusto y que se complementa muy bien con los sistemas unix, además cumple correctamente con las necesidades del proyecto.

Otras alternativas a Python que podíamos haber utilizado:

[7] Java: también es un lenguaje orientado a objetos, compilado e interpretado, con esto queremos decir, que el código compilado no se ejecuta directamente en el procesador, si no que necesita de una máquina virtual (JAVA VIRTUAL MACHINE).

Hemos descartado este lenguaje ya que, con Python para hacer este tipo de desarrollo con mucho menos código y tiempo, se puede llegar a un nivel de desarrollo muy similar al conseguido en java.

Como gestor de módulos Python utilizaremos PIP.

PIP es un acrónimo que significa Programa de instalación preferida. Es una herramienta para la línea de comandos que permite instalar o desinstalar paquetes PyPI (módulos Python) con un comando.

2.6 Tecnologías

En este apartado vamos a definir qué tipo de tecnologías hemos utilizado para llevar a cabo varios procesos de nuestro proyecto.

Empezaremos describiendo que software necesitamos tener instalado y configurado, aparte de los ya citados anteriormente.

2.6.1 Despliegue: DOCKER

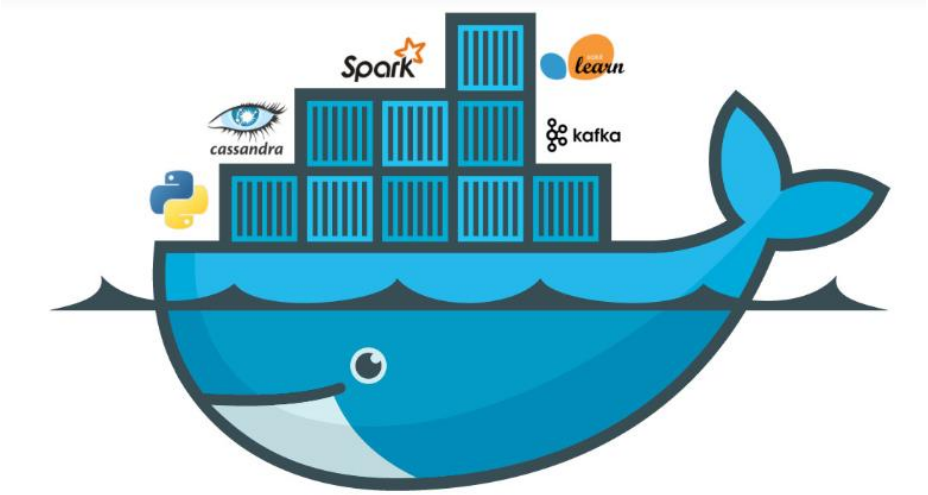


FIGURA 8 LOGO DOCKER

Docker es un proyecto de código libre que se ha convertido en uno de los términos de moda por las ventajas que proporciona, entre otros, a los profesionales del desarrollo web y de aplicaciones, o los administradores de sistemas, por la facilidad que supone el trabajar con el concepto de contenedores.

Docker está transformando la forma en que se desarrolla, distribuye y ejecuta el software. La ventaja es muy evidente, podemos encapsular todo el entorno de trabajo

de manera que los desarrolladores pueden estar trabajando en su servidor local, con la seguridad de que, al llegar el momento de ponerlo en producción, van a estar ejecutándose con la misma configuración sobre la que se han hecho todas las pruebas.

De esta forma, vamos a poder reducir los tiempos de testeo y adaptaciones al hardware del que se dispone en el entorno de producción.

[10] Hemos decidido utilizar Docker, ya que sin tener que instalar nada más que el gestor de contenedores, de una manera muy rápida y sencilla, hemos instalado y configurado distintos servidores de Base de datos (MySQL) específicos para cada microservicio, y así poder desacoplar totalmente cada microservicio, de esta manera, solo usando Docker y configurando contenedores podemos simular tener varios servidores diferentes.

También gracias a Docker hemos instalado y configurado un broker de mensajería (MQTT). En concreto nos hemos descargado el contenedor de eclipse-mosquitto.

También para las configuraciones más avanzadas utilizaremos Docker-compose.

Docker Compose es una herramienta que permite simplificar el uso de Docker, generando scripts que facilitan el diseño y la construcción de servicios.

2.7 Tipo de datos de los mensajes entre los servicios.

Se ha decidido, siempre y cuando sea necesario, comunicar servicios entre sí, enviando y recibiendo información mediante el tipo conocido como JSON.

2.7.1 JSON

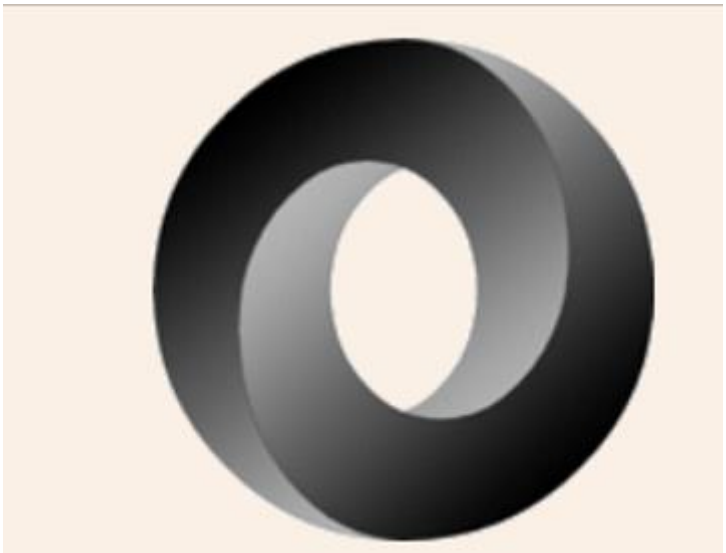


FIGURA 9 LOGO JSON

JSON significa JavaScript Object Notation, sin embargo, aunque debido al nombre parezca lo contrario, no es necesariamente parte de JavaScript, sino que es un estándar basado en texto plano para el intercambio de información, por lo que se usa en muchos sistemas que requieren mostrar o enviar información para ser interpretada por otros sistemas.

[5] Una de las más grandes ventajas que supone usar JSON, es que tiene un formato totalmente independiente de cualquier lenguaje de programación. Esto permite que los servicios puedan compartir información sin necesidad de hablar el mismo idioma (lenguaje de programación). Con lo que emisor y receptor pueden utilizar lenguajes distintos, pero son capaces de decodificar cadenas de JSON gracias a su librería propia.

Ahora bien, hemos utilizado JSON ya que se integra perfectamente con Python además de ser un objeto muy ligero y fácil de entender.

Cabe destacar que no es el único tipo de dato que se puede utilizar para llevar a cabo un proyecto de estas características, también hubiésemos podido utilizar XML, pero se ha descartado, ya que este suele llevar un tratamiento de datos más complejo, que para el alcance de nuestro proyecto no aportaba ningún valor.

2.8 Protocolo de Comunicación

Hemos utilizado dos de los muchos protocolos de comunicación que hay, ya que son los que más se acoplan la idea de industria 4.0.

2.8.1 HTTP

Es un protocolo que se utiliza para transmitir y solicitar contenido a través de una red informática o internet. HTTP está orientado a transacciones y funciona con un esquema cliente servido, petición / respuesta. El cliente ejecuta la petición y el servidor responde la respuesta.

Una característica de HTTP es la escritura y negociación de la representación de datos, lo que permite que los sistemas se construyan independientemente de los datos que se transfieren.

Se definen una serie de códigos de estado, para informar al cliente qué ha podido pasar en cada transacción.

[4] HTTP ha estado en uso por la iniciativa de información global World-Wide Web desde 1990. Esta especificación define el protocolo denominado "HTTP / 1.1", y es una actualización de RFC 2068

HTTP lo utilizaremos para comunicarnos con la API rest que utilizaremos para identificar recursos dentro de la empresa.

Para el resto de las comunicaciones entre los diferentes microservicios que tenemos en nuestro proyecto utilizaremos el protocolo MQTT.

2.8.2 MQTT

MQTT que significa MQ Telemetry Transport, es un protocolo de mensajería de publicación / suscripción, extremadamente simple y liviano, diseñado para dispositivos restringidos y redes de bajo ancho de banda, alta latencia o poco confiables. Los principios de diseño son minimizar el ancho de banda de la red y los requisitos de recursos del dispositivo, al mismo tiempo que intentan garantizar la confiabilidad y cierto grado de seguridad de la entrega. Estos principios también hacen que el protocolo sea ideal para el emergente "máquina a máquina" (M2M) o "Internet de las cosas" IoT del mundo de los dispositivos conectados y para las aplicaciones móviles donde el ancho de banda y la energía de la batería son importantes.

2.8.3 REST

FIGURA 10 LOGO REST

REST es una arquitectura de desarrollo web que puede ser utilizada en cualquier cliente HTTP. Es más sencilla que otras ya existentes, como por ejemplo *XML-RPC* o *SOAP*.

[2] REST es cualquier interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos.

Fue creada por Roy Fielding en el 2000, quien a su vez es uno de los principales creadores de la especificación del protocolo HTTP.

Lo que le da superioridad a esta arquitectura es que ha proporcionado a la web una mejor escalabilidad, lo cual permite dar soporte a un mayor número de componentes y a sus interacciones.

Las principales características de la arquitectura REST son que le proporcionan esta superioridad son:

- Es un protocolo sin estado, ya a que no se guarda la información en el servidor. Por lo que, toda la información será enviada por el cliente en cada mensaje HTTP, consiguiendo un ahorro en variables de sesión y almacenamiento interno del servidor.
- Presenta un conjunto de operaciones, siendo las más importantes *GET*, *POST*, *PUT* y *DELETE*, que se emplea en todos los recursos.

Se ha implementado en Python la API REST con un microframework que está muy extendido en la comunidad Python, este es conocido como FLASK.

Flask es un microframework muy robusto, con una curva de aprendizaje muy corta, por lo tanto, nos facilita su implementación, flask está completamente testeado por la comunidad Python y es un fremwork que se utiliza en muchísimos proyectos actuales ya que está bajo una licencia BSD.

2.9 Broker De Mensajería: mosquitto

FIGURA 11 LOGO MOSQUITTO

[8] Eclipse Mosquitto es un intermediario de mensajes de código abierto (con licencia EPL / EDL) que implementa las versiones 5.0, 3.1.1 y 3.1 del protocolo MQTT.

Mosquitto es adecuado para su uso en cualquier dispositivo, desde computadoras de una sola placa de baja potencia, hasta servidores completos.

El protocolo MQTT proporciona un método ligero para realizar la mensajería utilizando un modelo de publicación / suscripción. Esto lo hace adecuado para la mensajería de IOT, como son sensores de baja potencia o dispositivos móviles como teléfonos, placas integradas o microcontroladores.

El proyecto Mosquitto también tiene disponible una biblioteca de C para implementar clientes MQTT, y los populares clientes de línea de comandos mosquitto_pub y mosquitto_sub de MQTT.

Mosquitto es parte de la Fundación Eclipse y es un proyecto iot.eclipse.org.

Se ha implementado esta solución como broker de mensajería, dada su facilidad de instalación y configuración en un contenedor Docker.

3. Caso de Estudio

3.1. Introducción

Durante este trabajo, hemos expuesto varias tecnologías diferentes pero que están muy relacionadas con el concepto IoT y fabrica 4.0. Ahora bien, lo que vamos a realizar durante las siguientes partes de este proyecto es la simulación de una fábrica 4.0.

Ahora bien, intentaremos incorporar la industria 4.0 a una empresa española real. Embalpack se dedica a la fabricación de cantoneras de cartón y están especializados en la fabricación, transformación y distribución de productos para todo tipo de embalaje con la máxima calidad y protección de su contenido.

Esta fábrica se ha dado cuenta de la necesidad de crecer tecnológicamente para mejorar su eficiencia debido a que no tienen control de toda la materia prima que almacenan en su fábrica y también por la necesidad de obtener la trazabilidad completa de todo su proceso productivo, desde el aprovisionamiento hasta el producto final. Esto les permitirá conocer dónde pueden ser más eficientes acortando los plazos en el proceso de producción.

Embalpack como materia prima utiliza papel cien por cien reciclado para la fabricación de todos sus productos.

Como recursos de la empresa entenderemos a todos los operarios de la empresa, todas la maquinas que trabajan en producción, y toda la materia prima utilizada.

Pues bien, para esta simulación hemos seleccionado tres procesos para componer una fábrica 4.0 y más concretamente, adaptarla en la empresa Embalpack. Estos han sido seleccionados por su simpleza y necesidad, siendo estos los siguientes.

- Identificación de recursos
- Trazabilidad de los recursos dentro de la fábrica
- Máquina generadora de tortas

3.2. Descripción

Embalpack, como muchísimas empresas de nuestra sociedad, requieren sacar el máximo provecho a sus instalaciones no solo con el personal existente, si no automatizando procesos para hacer la cadena de producción más eficiente. Es por eso por lo que han decidido modernizar su empresa, llevándola a una industria 4.0.

Con este cambio de generación, obviamente surgen muchos nuevos problemas que la industria común no tiene. Uno de los problemas de la industria 4.0, que está ligada completamente con la IoT radica en la manera de identificar qué es cada objeto con el que estamos tratando, es por eso que hemos decidido implantar un servicio para identificar y nombrar nuestros recursos dentro de la empresa, ya sean materias primas o maquinaria necesaria para transformar la materia prima, incluso identificar a los mismos operarios.

Ahora bien, uno de los recursos que queremos automatizar es la maquina encargada de transformar la materia prima en materia que posteriormente se podrá utilizar para hacer el producto final, esta materia se denomina torta, sin dicho procesamiento de la materia prima no se pueden hacer las tareas consecutivas, por lo tanto, se ha decidido que este proceso sería interesante simularlo y describirlo en este trabajo.

Por último, se ha pensado que se es importante tener una excelente trazabilidad de los procesos dentro de la compañía, ya que de esta forma podremos saber cosas como quién ha manipulado alguna materia o maquinaria de la empresa. Esta es una manera de obtener información para sacar estadísticas sobre el tiempo de acción y producción dentro de la fábrica y poder optimizar los recursos y hacer a la empresa más eficiente.

Como hemos descrito, creemos que es muy interesante que Embalpack pegue el salto a la nueva industria, ya que, automatizando los procesos, podrá generar producto con rendimientos más altos.

3.3. Análisis de la Solución.

El proceso de fabricación se divide en " tareas de transformación", como por ejemplo el rebobinado de papel o torta, que es la materia generada previa al producto final.

Lo que se busca en este proyecto, es que la información la genere la propia infraestructura. Hay un concepto principal que es el propio proceso de fabricación, que de alguna manera recoge todas las tareas individuales y secuenciales que transforman el papel de entrada en cantoneras preparadas y cada tarea de transformación se divide en diferentes pasos inicio, transformación y acabado.

Ahora bien, con las maquinas actuales de Embalpack no hay posibilidad de automatización completa, por lo tanto, lo que haremos es una simulación. Simularemos las maquinarias y virtualizaremos la materia prima. Así queda un escenario de la siguiente manera:

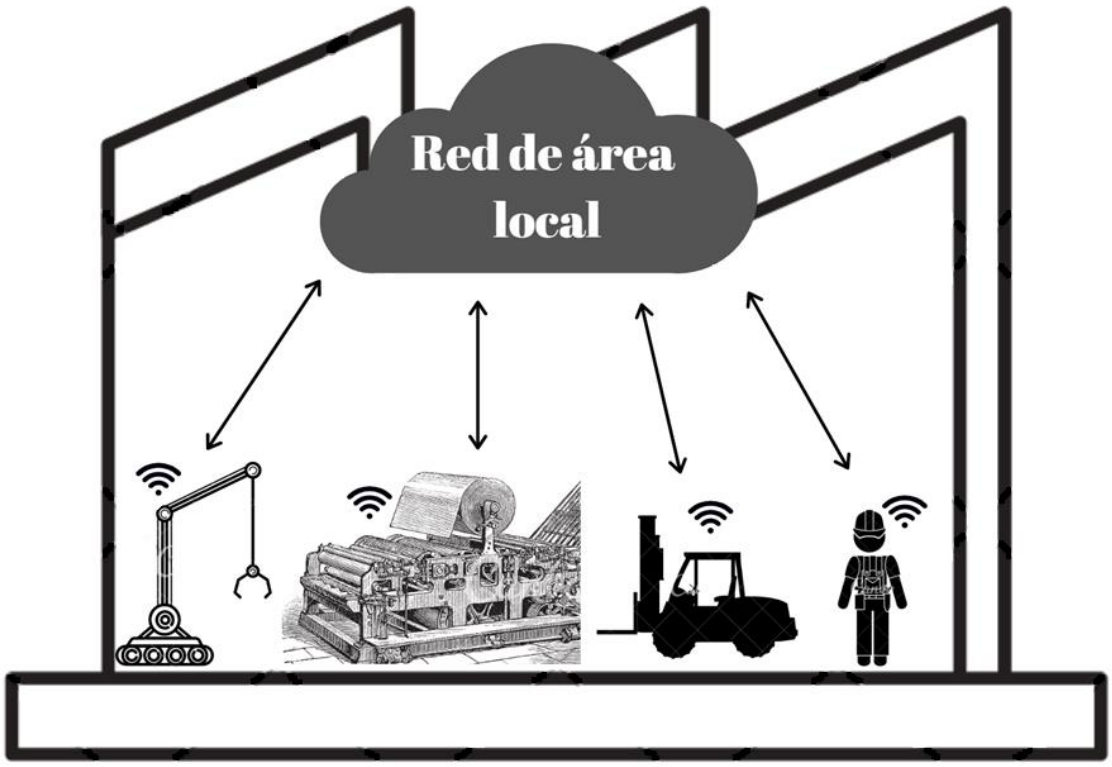


FIGURA 12 ESQUEMA VIRTUAL EMBALPACK

En cada tarea participará típicamente un operario, materia prima y una máquina de la fábrica.

Primero, tendremos una base de datos que almacenará sobre la trazabilidad, e identificación de los productos. Todos los datos que almacenaremos de trazabilidad

serán generados por la propia infraestructura de la empresa. Además tendremos una máquina simulada, que será la encargada de crear tortas y por último comentaremos que el toro de carga, será la encargada de la tarea que mueve la materia prima.

La nube que podemos ver en la imagen es el resto de los dispositivos interconectados en nuestra red, pero no serán necesarios para el desarrollo de nuestro proyecto.

Como resumen de la imagen y del proyecto podemos destacar que tenemos el siguiente funcionamiento:

El toro mecánico obtendrá de una estantería la materia prima, este informará a nuestro sistema de qué materia prima se trata, cuándo fue movida y quién la movió. Sin embargo, este último proceso no nos encargaremos de simularlo en este trabajo.

Mas tarde el toro mecánico deja la materia prima en la máquina que genera tortas, esta máquina lee de qué materia se trata y posteriormente empieza su trabajo. Se crearán tantas tortas como se puedan generar con una configuración específica, esta máquina es la encargada de preguntar a nuestro proceso de identificación qué UUID será el que le corresponde a cada torta cuando acabe de generarlas, informará a nuestra infraestructura que ya ha finalizado el proceso y enviará cuántas tortas se han generado, en cuánto tiempo se ha generado cada una y qué maquina las ha generado.

Una vez que la maquina informe su finalización, se almacenará toda la información recapitulada por la maquina generadora de tortas, así que con este proceso entran en juego nuestros tres servicios implementados.

4. Diseño de la Solución

Ya realizado el caso de uso, ahora plantearemos un diseño a utilizar para implementar la solución. Primero plantearemos una arquitectura general del proyecto.

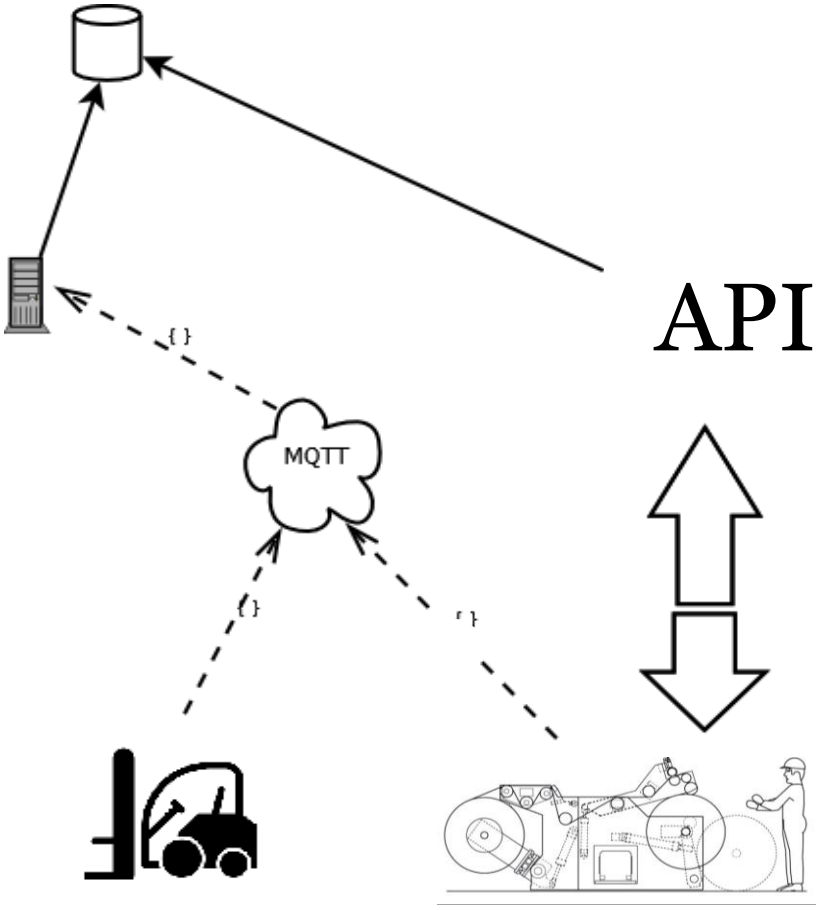


FIGURA 13 ESQUEMA GRAFICO IMPLEMENTACIÓN

Como podemos observar en la Figura 13, tenemos una representación de una parte de la producción de Embalpak esta pequeña representación consta de:

1) Maquina rebobinadora de papel: Es la encargada de leer la materia prima, se comunicará mediante una petición http con nuestra API (Servicio de identificación de la empresa) que se encargará de identificar cada torta generada.

Una vez que ya se han generado e identificado todas se enviará un mensaje al tópico de la cola de mensajería, representada con una nube en la imagen.

2) Máquina de trasporte mecánica: Es la encargada de mover o bien la materia prima o las tortas, cada vez que se mueva, enviará un mensaje al tópico informando qué transporta y qué máquina es la encargada de mover dicha materia.
3) Al optar por comunicaciones M2M (machin to machin) a través de colas de mensajería, hay procesos encargados de escuchar la información generada por los procesos hacia las colas, por ejemplo, la guardan en base de datos o se comunican con otros procesos. La tarea principal para leer la información generada por las colas de mensajería la denominaremos proceso de trazabilidad, será el encargado de leer dicha información y almacenarla en nuestro sistema. Posteriormente se explotarán estos datos en un sistema ERP.

Una vez descrito los procesos adjuntaremos un ejemplo de cada comunicación.

1) Maquina rebobinadora de papel: pedirá una identificación por cada torta que pueda crear, recibirá la información necesaria para identificar a dicha torta en una respuesta http.

FIGURA 14 MUESTRA DEL CONTENIDO RECIBIDO

Dentro de “content” tenemos la información que nos devolverá la API, todo lo demás son las cabeceras que mandan para que sea efectiva la comunicación mediante el protocolo http, en este caso es la información relacionada a la respuesta. Http.

2) La máquina rebobinadora una vez que ya tenga todas las tortas que se han podido crear enviará un mensaje al tópico correspondiente, advirtiendo que ya ha acabado, enviará un mensaje:

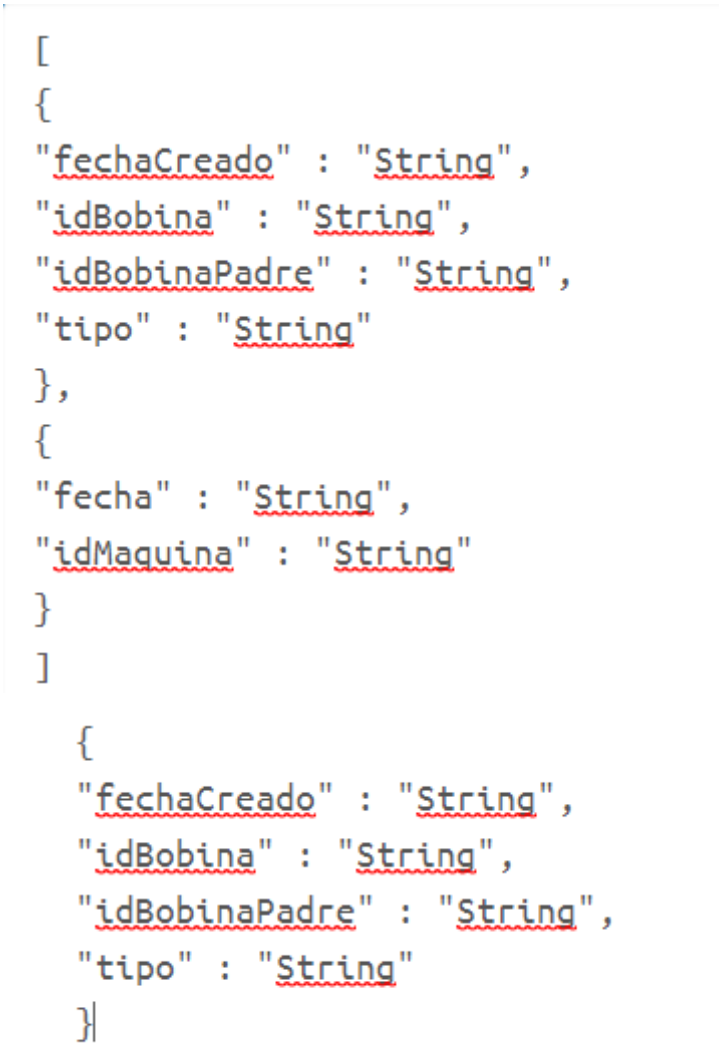


FIGURA 15 MUESTRA DEL CONTENIDO ENVIADO

Como podemos observar en la Figura 15, la maquina una vez que finaliza su trabajo, envía dos tipos de información diferentes, en la primera posición de la lista, enviará una lista de cuántas tortas ha creado e información relacionada con ellas. La información que envía por cada torta es:

1) Fecha de creación.
2) La identificación de la torta que se ha creado.
3) Identificación de la materia prima.
4) Qué tipo de recurso se ha creado.

En la siguiente posición de la lista, enviará la siguiente información acerca de la máquina que ha realizado dicha operación, con la siguiente información:

1) Fecha de finalización del trabajo.
2) Identificación de la máquina que ha realizado el trabajo.

4.1 Diseño de la base de datos

En este punto mostraremos todos los esquemas de bases de datos utilizados en el proyecto.

4.1.1 Trazabilidad

Para nuestro microservicio de trazabilidad hemos diseñado la siguiente base de datos:

```
create table if not exists EMBALPACK.Trazabilidad
(
  id                int auto_increment,
  id_recurso        varchar(255) null,
  estado            varchar(255) null,
  id_recurso_externo varchar(255) null,
  constraint Embalpck_id_uindex
    unique (id)
);

alter table EMBALPACK.Trazabilidad
  add primary key (id);
```

Podemos destacar que tenemos un id único y auto incremental, tenemos el recurso que estamos manipulando, el estado en el que se ha trabajado en dicho recurso (Creado por, Movido por, etc), “id_recurso_externo” sería qué recurso o maquinaria ha manipulado el recurso o materia prima.

4.1.2 Identificación

Para el microservicio de identificación se ha creado una base de datos con las siguientes columnas.

```
create table recursos
(
 id varchar(255) ,
 type varchar(255) not null,
 creation timestamp default current_timestamp,
  PRIMARY KEY (`id`)
);
```

Id se genera mediante código, el type lo definiremos como el tipo de materia o recurso que vamos a identificar y por último tenemos la fecha de cuándo hemos identificado dicho recurso o materia.

5. Implementación de la Solución.

Ya expuesto todo lo necesario del trabajo, procedemos a exponer su configuración e implementación de todas las tareas para poder llevar a cabo nuestro proyecto se necesitarán realizar una serie de pasos que podremos ver en el anexo 1 *Configuración de nuestra base de datos.*

5.1 Implementación microservicio identificación

Con este microservicio, lo más importante lo tenemos en la imagen que podemos ver a continuación.

```python
from app import app
from ConectorBD.conectorBD import mysql
from flask import jsonify
from flask import flash, request
import Servicio.srvApi as SrvAPi
@app.route('/add', methods=['POST'])
def add():
    try:
        _json = request.json
        _type = _json['type']
        if _type and request.method == 'POST':
            resp = jsonify( SrvAPi.SrvApi.insertarRecurso(_type))
            resp.status_code = 200
            return resp
        else:
            return not_found()
    except Exception as e:
        print(e)
@app.route('/resources')



def resources():
    try:
        rows = SrvAPi.SrvApi.getResources()
        resp = jsonify(rows)
        resp.status_code = 200
        return resp
    except Exception as e:
        print(e)
        resp = jsonify(SrvAPi.SrvApi.getResourcesID(id))
        resp.status_code = 200
        return resp
    except Exception as e:
        print(e)
@app.route('/resource/type/<type>')
def resourceType(type):
        print(e)









@app.errorhandler(404)
def not_found(error=None):
    message = {
        'status': 404,
        'message': 'Not Found: ' + request.url,
    }
    resp = jsonify(message)
    resp.status_code = 404
    return resp
if __name__ == "__main__":
    app.run(host='192.168.150.251')
```

resourceType() › except Exception as e

Event Log

FIGURA 16 MUESTRA CÓDIGO IDENTIFICACIÓN

En esta Figura 16,podemos destacar que en este fragmento de código introducimos los puntos de entradas a nuestra API. A continuación, detallaremos cada uno de los puntos:

Con el punto de entrada “/add” será una petición de tipo POST. En la petición a nuestro servidor se necesitará cuerpo en el mensaje que nos especificará qué tipo de recurso queremos añadir al sistema, por ejemplo: “TORTA”. Este controlador con el tipo de recurso que necesitamos crear llamará al servicio que será el encargado de darle una identificación y almacenarla en base de datos. En la Figura 17 podemos ver un ejemplo de mensaje enviado, dentro de content estará la identificación de la torta creada.

FIGURA 17 MENSAJE RECIBIDO PETICIÓN HTTP

Como podemos observar en el anexo *Código inserción en base de datos Identificación*, una vez que esté el recurso identificado y guardado en la base de datos, este mismo punto de entrada se encargará de responder el identificador de ese recurso.

Como segundo punto de entrada tenemos “/resources” de tipo GET. Este nos devolverá todos los resources que tenemos almacenados en nuestra base de datos y cómo podemos observar en la siguiente Figura 18

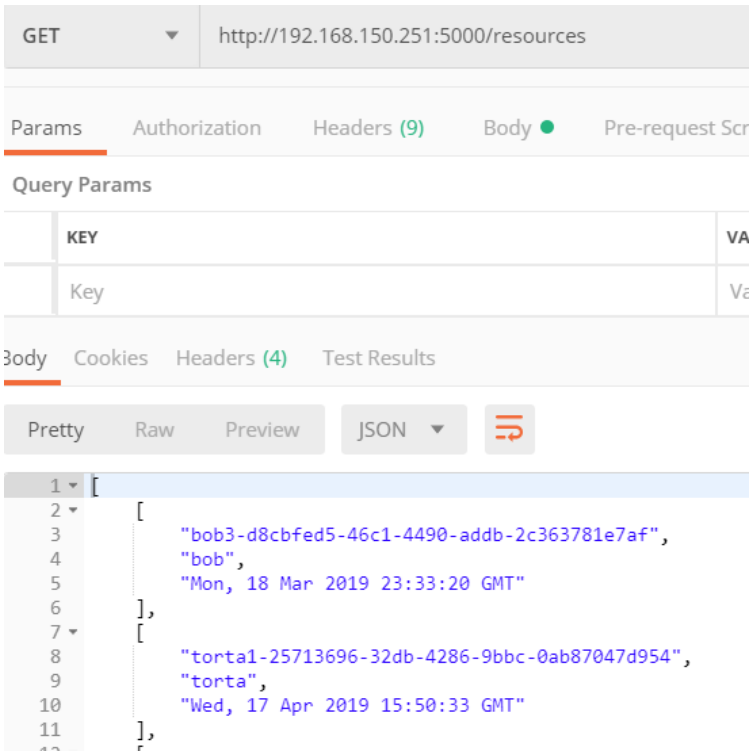


FIGURA 18 PEDIR RESOURCES POSTMAN

Podemos observar en el código mostrado en la Figura 16, que hemos puesto un try Except, por si nos da algún fallo al obtener los recursos, bien porque no se puede conectar a la base de datos o porque no tenemos datos en la base de datos. Este lanzara una excepción, y enviara un mensaje de error en la respuesta del mensaje.

El tercer punto de entrada lo utilizaremos para buscar un resource en concreto, para saber si existe o no, y haremos la petición a la URL “/resource/id/<id>”. En esta URL, podemos observar que tenemos una de las partes dentro de <>, esto significará que buscaremos un resource por id. Lo que tenemos dentro de <> es un texto variable, podemos ver un ejemplo en la siguiente Figura 19

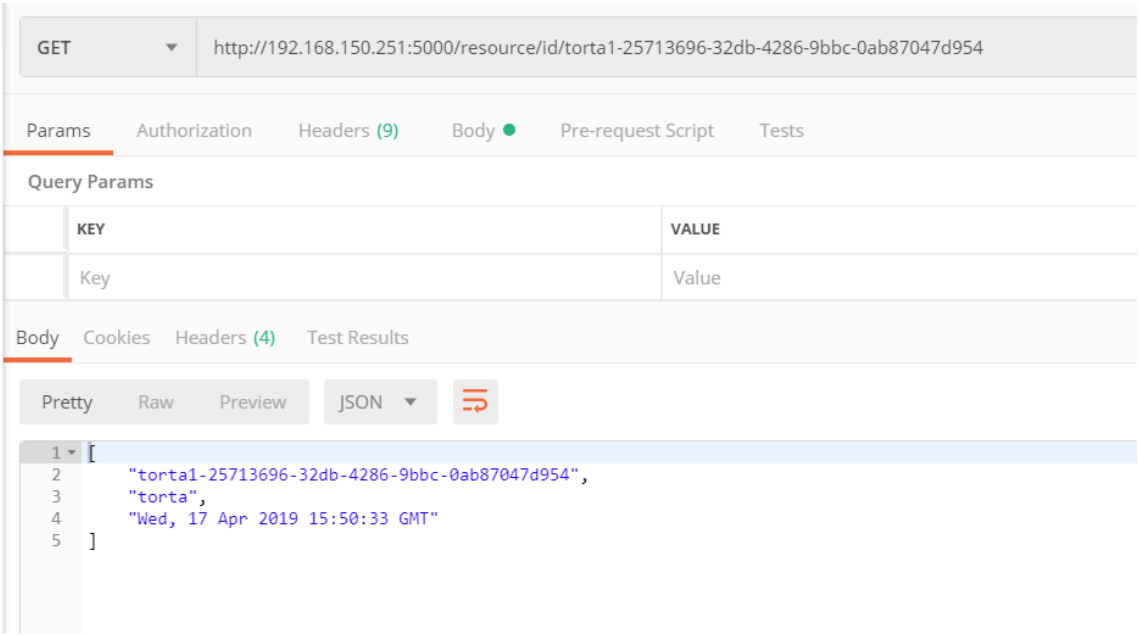


FIGURA 19 PEDIR UN RECURSO ESPECIFICO POSTMAN

El cuarto punto de entrada es "/resource/type/<type>" buscaremos únicamente todos los resources de un tipo.

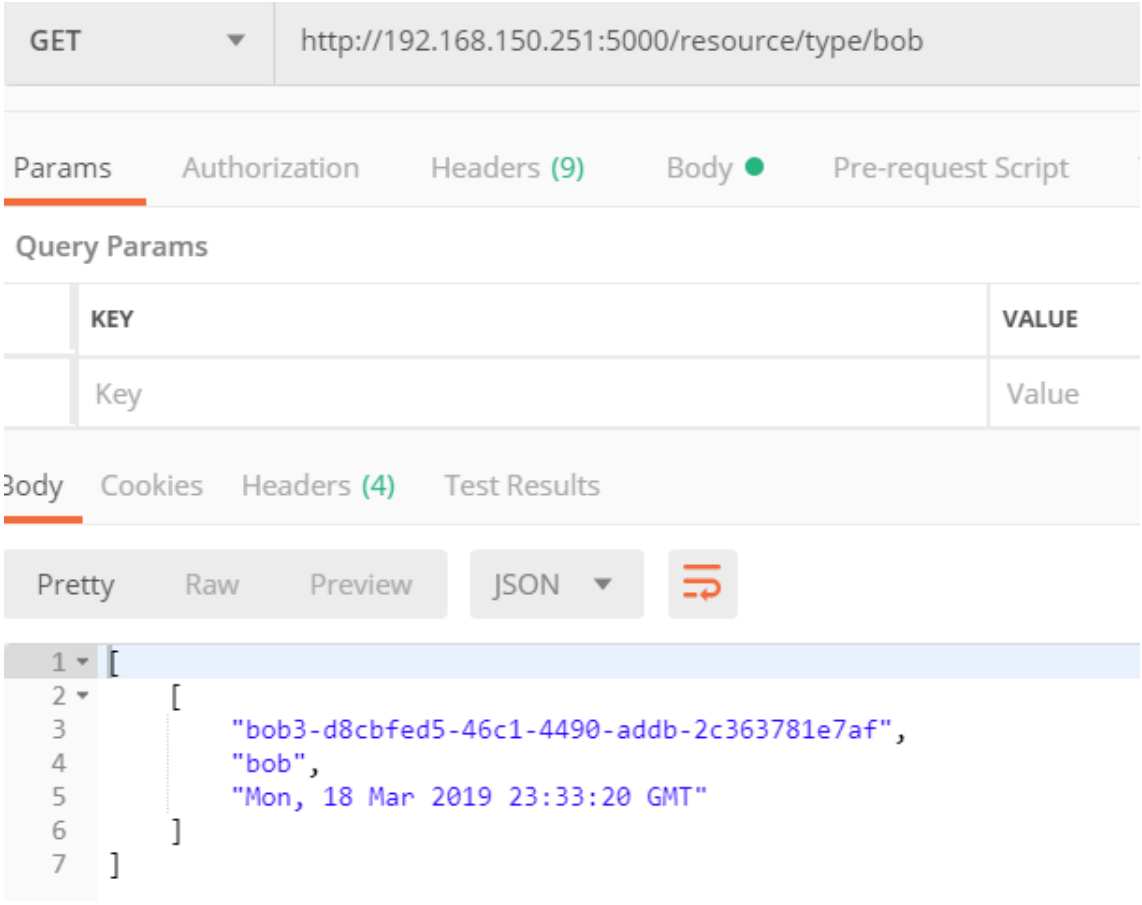


FIGURA 20 PEDIR TODOS LOS RESOURCES DE TIPO BOB

5.2. Implementación Microservicio Trazabilidad.

En este microservicio, detallaremos como conectarnos a un tópico mqtt.

```python
import paho.mqtt.client as mqtt
import ServicioTrazabilidad.srvTrazabilidad as srvTrazabilidad
import json

def on_connect(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))
    client.subscribe("Embalpack/Trazabilidad")
def on_message(client, userdata, msg):
    #id_recurso, estado, id_recurso_externo
    d = json.loads(msg.payload.decode())
    for i in d[0]:
        srvTrazabilidad.SrvTrazabilidad.insertarRecurso(i['idBobina'], "bobinaPadre  ", i['idBobinaPadre'])
        srvTrazabilidad.SrvTrazabilidad.insertarRecurso(i['idBobina'],"creador por ",d[1]['idMaquina'])
client = mqtt.Client()
client.connect("192.168.150.120", 1883, 60)
client.on_connect = on_connect
client.on_message = on_message
client.loop_forever()
```

FIGURA 21 EJEMPLO CÓDIGO TRAZABILIDAD

Como podemos ver en la Figura 21,anterior definiremos dos funciones, on_connect, y on_message.

Cuando se ejecute esta clase, ejecutará la función on_connect, que únicamente nos suscribe al tópico que hemos especificado, posteriormente la clase se queda en ejecución y espera recibir algún mensaje por dicho tópico, cuando lo recibe, se transforma el mensaje y se guardarán todos los datos que necesitamos en la base de datos.

Cuando la maquina rebobinadora de papel informe que ha terminado, enviará todas las tortas que ha creado, en formato JSON, y tendremos que transformarlo e insertarlas una por una, este proceso es el encargado de escuchar la información generada por los otros recursos de la empresa.

Este código de como almacenar en base de datos podemos verlo en el anexo *Código inserción en base de datos Trazabilidad*.

5.3. Implementación máquina generadora de tortas.

De este microservicio, detallaremos como hemos implementado el servicio, ya que es lo más complejo e importante de dicho microservicio.

```python
__author__ = 'Miguel Tortosa'

from Modelo.miniBobina import miniBobina
import time
import Modelo.miniBobina as minibobina
import requests
import json


class BobinasPequenyasServicio:
    def crearBobinas(msg, cuantas):
        bobinaPadreHttp = requests.get("http://192.168.150.251:5000/resource/id/"+msg.payload.decode())
        my_json = bobinaPadreHttp.content.decode('utf8').replace("'",'"')
        UUIDPadre= my_json.replace("[", "").split(",")[1]
        body ={
            "type": "torta"
        }
        url = ("http://192.168.150.251:5000/add")
        headers = {'Content-Type':'application/json'}
        listaBobinas = []
        for i in range(1, cuantas):
            response = requests.post(url, data=json.dumps(body), headers=headers)
            my_jsonTortosa = response.content.decode('utf8').replace("'", '"').replace("\n","")
            UUIDTorta = my_jsonTortosa.replace("[", "").split(",")[0]
            minibobina_ = miniBobina(UUIDTorta, 'torta', UUIDPadre, time.strftime("%c"))
            listaBobinas.append(minibobina.miniBobina.miniBobinaTOBD(minibobina_))
        return listaBobinas
```

FIGURA 22 CÓDIGO MAQUINA GENERADORA DE TORTAS

Como podemos observar en la imagen anterior, tenemos una función que lo primero que hace, es saber si la Bobina de materia prima esta identificada en nuestro sistema, ya que la maquina ha podido leer mal su identificador y por lo tanto no existir.

Para hacer esta consulta utilizaremos el microservicio de identificación, haremos una petición http pidiendo que nos devuelva la información relacionada a ese UUID, una vez que tenemos la información de la materia prima, la maquina empezara a crear tortas, tantas como se pueda con una configuración especifica en la máquina y ya finalizado dicho proceso se devolverá una lista que enviaremos al tópico de trazabilidad como podemos ver en el anexo *Envío mensaje JSON al tópico "Embalpack/trazabilidad".*

La información enviada al tópico de trazabilidad la podemos ver en un ejemplo en formato JSON en la siguiente Figura 23.

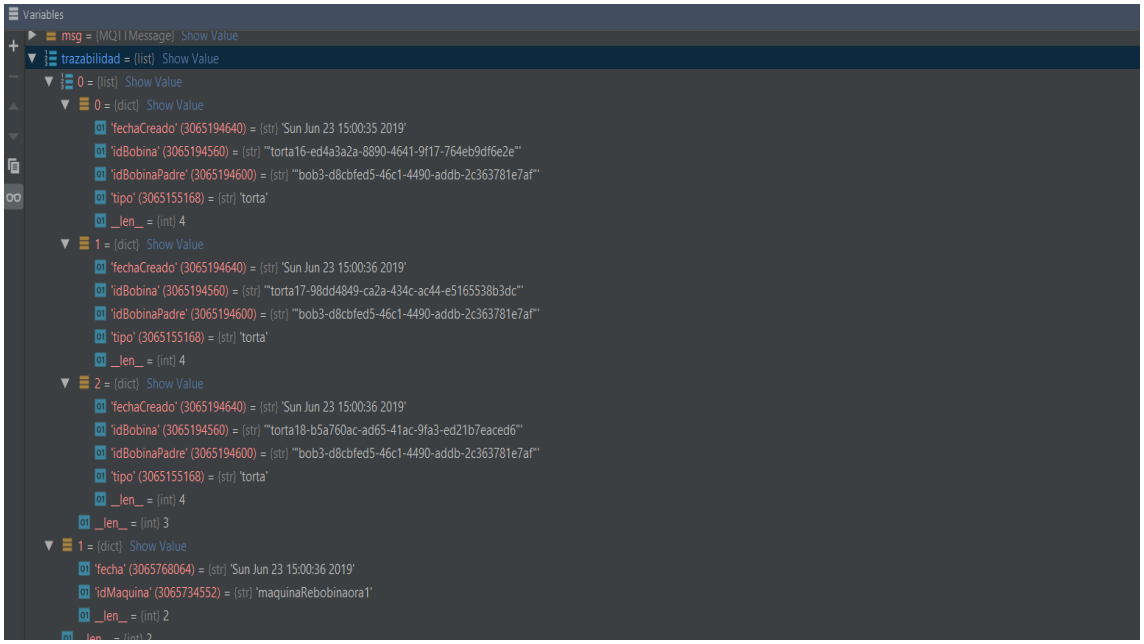


FIGURA 23 EJEMPLO MENSAJE ENVIADO AL TÓPICO DE LA MAQUINA

6. Conclusión

El día que mi tutor hablo conmigo para proponerme unos cuantos trabajos finales de carrera, vi en este proyecto un atractivo especial, muchísimos de ellos me parecieron viables, pero en este concretamente, se abarcaban temas muy interesantes que están ahora mismo en proceso de evolucionar y creo que es muy relevante para mi conocimiento, el poder investigar sobre todo lo relacionado con la industria 4.0. Creo que más empresas deberían de dar el salto de calidad tal y como lo está haciendo Embalpack, y así poder aumentar sus producciones y conseguir mejores productos en menor tiempo a menores costes.

Al realizar el presente proyecto, he aprendido lo mucho que pueden evolucionar las cadenas de producción de nuestras empresas.

Planteando el problema con el tutor, hemos visto la cantidad de materia que se desperdicia, ya que no hay un control exhaustivo de todo el material que se tiene en una empresa o el tiempo que se pierde con una maquina parada por alguna avería, ya que el responsable de la empresa puede estar realizando algún otro trabajo y no darse cuenta de que la maquina está detenida. También me he dado cuenta de la cantidad de información que se pierde al no tener nuestra industria más automatizada y parecida al concepto industria 4.0. Con este tipo de concepto se pueden explotar los datos que producen todos los recursos de nuestra empresa, y así saber en dónde se pierde tiempo, y por lo tanto estamos dejando de ganar dinero, así que podremos afirmar que si invertimos un poco de dinero en donde se pierde más tiempo, podremos obtener más beneficios ya que mejoraríamos el tiempo de producción.

Este es el resultado de mi esfuerzo depositado en este trabajo que, como se puede apreciar es solo un asomo a lo que significaría llevar a nuestras empresas a la vanguardia tecnológica y con ello impactar positivamente la economía nacional que más tarde o más temprano derramaría beneficios significativos a nuestra sociedad en el campo laboral pero también educativo basado en la especialización y con ello la superación generalizada de las condiciones de vida en el futuro próximo de nuestro país.

Me ha costado mucho, sobre todo al principio, familiarizarme con Python, ya que era un lenguaje completamente desconocido para mí, pero que está muy integrado en mundo de la informática, sin embargo, ha valido la pena descubrir así, que aún hay mucho que aprender y que innovar en este apasionante y revolucionario mundo de la informática.

En el trabajo, he utilizado muchas cosas aprendidas en el grado, pero sobre todo he utilizado conceptos, tecnologías y conocimientos de la rama que estoy cursando, Tecnologías de la información, más concretamente he podido utilizar muchas de las cosas que he aprendido en la asignatura de Integración de aplicaciones, ya que está muy relacionado en la industria 4.0, con las comunicaciones, y todos los subprocesos que hay en el concepto.

Aunque solo he podido representar 3 procesos que se han implementado en la empresa Embalpack, este proyecto tiene muchas metas que alcanzar todavía, por ejemplo, la automatización y aviso de los niveles de tinta de otra máquina, el rellenado de los pallets etc.

En este proyecto he intentado utilizar todo lo aprendido en el grado, no solamente la parte referente a la tecnología, pero también de disciplina, de compromiso con uno mismo y los demás, de trabajo y responsabilidad lo cual agradezco a mis maestros y amigos que me apoyaron en este proceso.

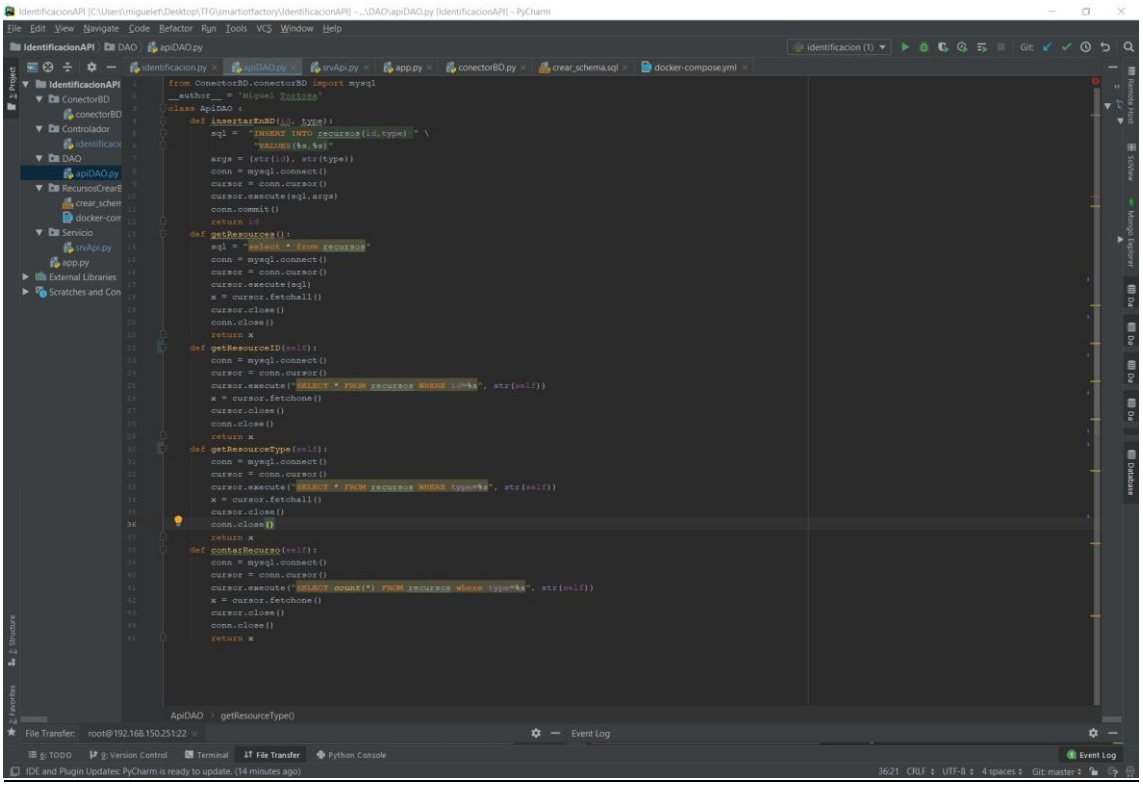
Por último, quiero agradecer muy especialmente a, Joan quién me apoyó y me dio confianza en el trayecto y apoyo en los días más difíciles con la redacción del trabajo. También agradecerle que me aceptara como alumno para realizar mi trabajo final de grado.

7. Bibliografía

1. AlDALKIN. (2017). *Industria 4.0. Qué es, ventajas e inconvenientes*. Obtenido de http://www.aldakin.com/industria-4-0-que-es-ventajas-e-inconvenientes/
2. BBVA. (23 de Marzo de 2016). *API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos*. Obtenido de https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos
3. CIC Consulting Informático. (20 de Febrero de 2019). *SCADA y la Industria 4.0*. Obtenido de https://www.cic.es/scada-y-la-industria-4-0/
4. Connolly, D. (01 de Septiembre de 2004). *Protocolo de transferencia de hipertexto - HTTP / 1.1*. Obtenido de https://www.w3.org/Protocols/rfc2616/rfc2616.html
5. Garibay, V. (17 de Julio de 2016). Obtenido de JSON marcando tendencias: https://medium.com/@victor.garibayy/qu%C3%A9-es-y-para-qu%C3%A9-sirve-json-be05fe02e67d
6. JetBrains. (s.f.). Obtenido de Powerful Python and Django IDE: https://confluence.jetbrains.com/display/PYH/PyCharm+IDE+and+Python+Plugin+for+IntelliJ+IDEA
7. Martinez, R. (s.f.). Obtenido de Principales Características de JAVA.: http://personales.upv.es/rmartin/cursoJava/Java/Introduccion/PrincipalesCaracteristicas.htm
8. Mosquitto. (s.f.). *Eclipse Mosquitto ™ Un broker MQTT de código abierto*. Obtenido de https://mosquitto.org/
9. Python Software Foundation. (28 de Junio de 2019). *El tutorial de Python*. Obtenido de https://docs.python.org/3/tutorial/
10. Villacampa, Ó. (11 de Julio de 2017). *Ondho*. Obtenido de Qué es Docker y para qué sirve: https://www.ondho.com/que-es-docker-para-que-sirve/

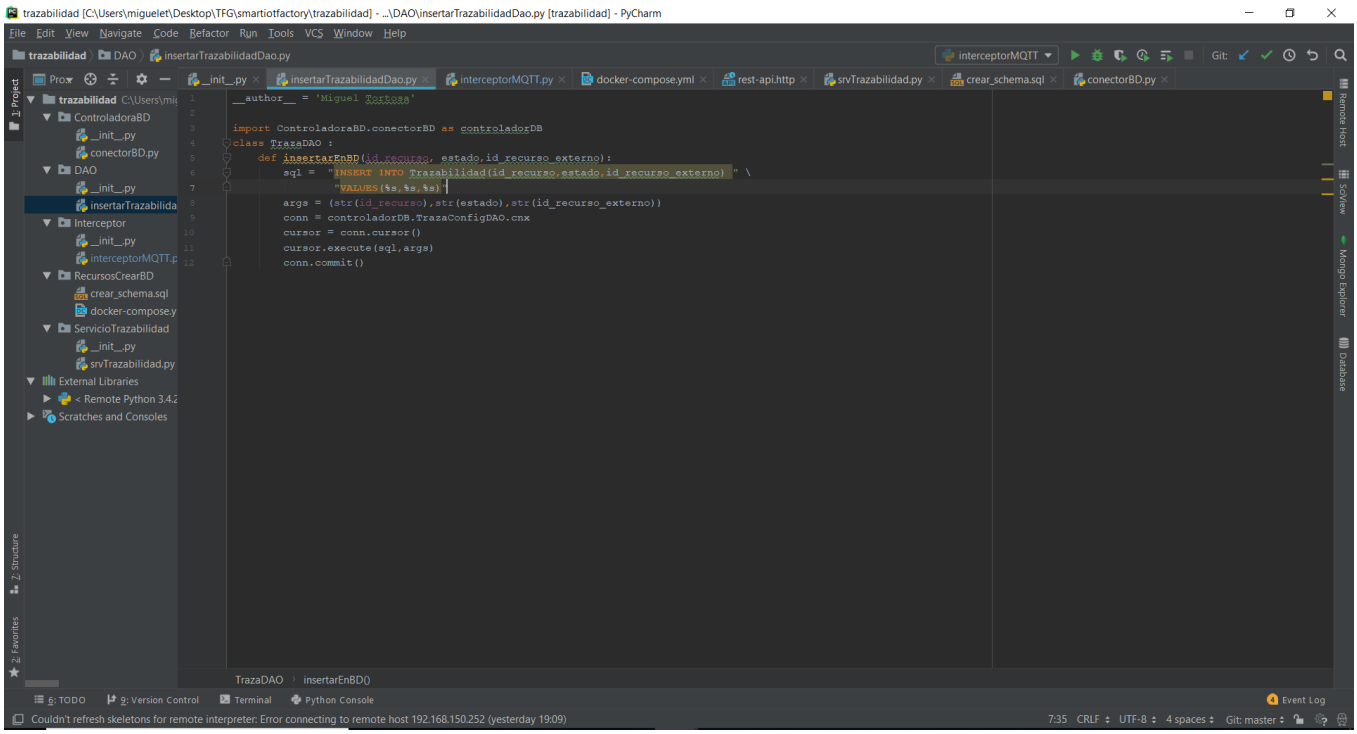
8. Anexos

1 Código inserción en base de datos Identificación



Como podemos ver en la anterior imagen, es el código necesario para interactuar con nuestra base de datos de identificación, ya sea para hacer una inserción o para extraer resultados de nuestra base de datos

2 Código inserción en base de datos Trazabilidad



En la anterior imagen podremos ver el código necesario para hacer una inserción en nuestra base de datos de trazabilidad

3 Envío mensaje JSON al tópico "Embalpack/trazabilidad"

```python
#!/usr/bin/env python3
__author__ = 'Miguel Tortosa'

import ...


# This is the Subscriber
def on_connect(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))
    client.subscribe("Embalpack/entradaBob")


def on_message(client, userdata, msg):
    listaBobinas = srvBobPeque.BobinasPequenyasServicio.crearBobinas(msg, randint(3, 7))
    trazabilidad = []
    trazabilidad.append(listaBobinas)
    maquina = {
        'idMaquina': 'maquinaRebobinaora1',
        'fecha': time.strftime("%c")
    }
    trazabilidad.append(maquina)
    publish.single('Embalpack/Trazabilidad', json.dumps(trazabilidad), hostname='192.168.150.120')
    if msg.payload.decode() == "Desconectar":
        print("Yes!")
        client.disconnect()


client = mqtt.Client()
client.connect("192.168.150.120", 1883, 60)
client.on_connect = on_connect
client.on_message = on_message
client.loop_forever()
```

En la anterior imagen veremos el código necesario para conectarse y enviar mensajes al tópico.

4 Instalación de Python en nuestras Intel Galileo.

Accederemos a nuestras intel galileo mediante ssh y ejecutaremos el comando "apt-get install python3.4". Cuando finalice este comando, ya tendremos Python instalado. Seguidamente instalaremos PIP con el siguiente comando "apt-get install python3-pip" y una vez ejecutados en nuestra terminal los dos comandos, ya tendremos instalado Python y PIP.

4.1 Instalación necesaria para el microservicio de identificación.

Para poder empezar a desarrollar nuestro microservicio de identificación, tendremos que instalar el módulo de flask, para ello, accedemos a nuestra Intel galileo y ejecutaremos el siguiente comando en nuestra línea de comandos "pip3 install flask".

También tendremos acceso a base de datos e instalaremos el módulo encargado de ello, por lo tanto, ejecutaremos el siguiente comando “pip3 install flask-mysql”.

4.2 Instalación necesaria para el microservicio de trazabilidad.

Para desarrollar este microservicio, necesitaremos tener el módulo para poder utilizar MQTT por lo tanto ejecutaremos en nuestra terminar el siguiente comando “pip3 install paho-mqtt”, este microservicio también tiene acceso a datos, pero como es un microservicio completamente desacoplado al de Identificación tendremos que ejecutar el siguiente comando “pip3 install PyMySQL”

4.3 Instalación necesaria para la simulación de a máquina rebobinadora.

Para la realización de esta simulación, necesitaremos tener el siguiente módulo, ejecutaremos el siguiente comando “pip3 install paho-mqtt”

Una vez realizados estos pasos, ya tenemos la puesta en marcha de todo lo relacionado con las intell galileo, por lo tanto, ya podríamos empezar a trabajar con ellas, pero primero vamos a realizar todas las tareas de configuración y posteriormente empezaremos con el desarrollo.

5 Configuraciones Docker.

En este apartado vamos a realizar las diferentes configuraciones de Docker para poder llevar a cabo el proyecto.

5.1 Configuración Broker De Mensajería Eclipse-Mosquitto.

Para obtener la imagen del contenedor de eclipse-mosquitto utilizaremos la herramienta Kitematic.

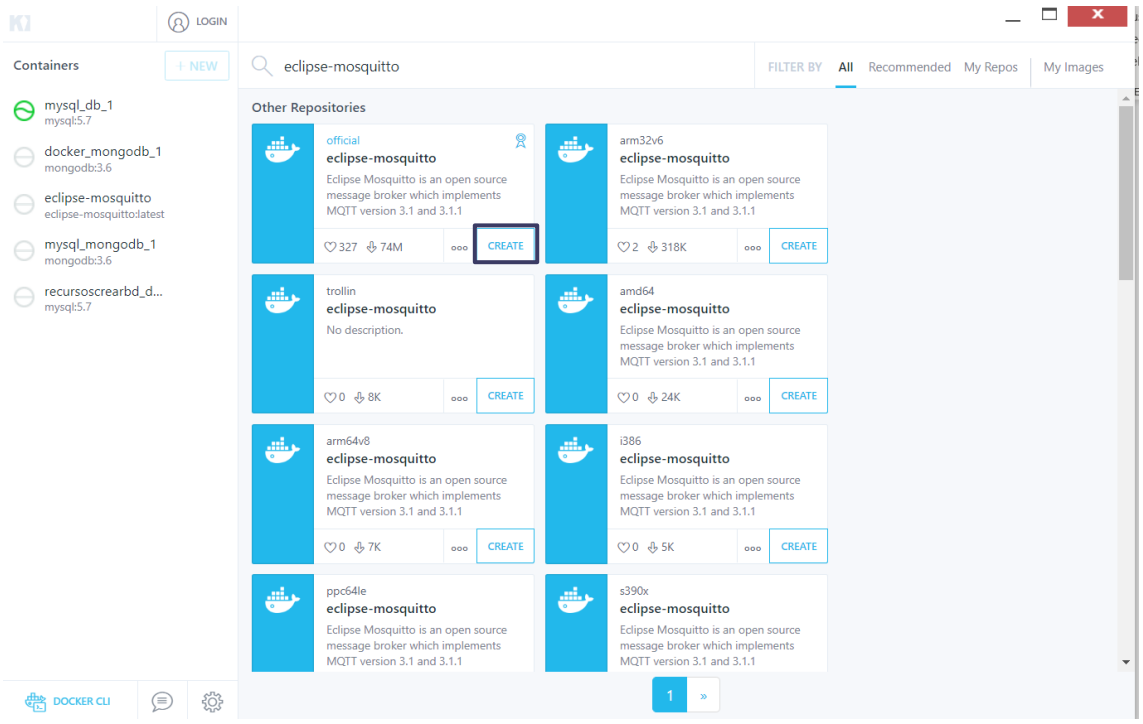


FIGURA 24 KITEMATIC

En la herramienta, como podemos observar en la Figura 24,hay un cuadro de búsqueda, en este cuadro buscaremos por el nombre, en este caso eclipse-mosquitto.

Una vez buscado podremos instalar varias imágenes, nosotros nos descargaremos la imagen oficial, hacemos clic en créate y nos esperamos que se descargue, una vez descargada dejaremos la configuración por defecto, ya que con esta configuración nos es suficiente para lo que necesitamos.

Una vez realizados estos pasos podemos comprobar que está funcionando si hacemos clic sobre el nombre del contenedor nos saldrá una ventana que nos advertirá si algo no está funcionando correctamente, o por lo contrario nos pondrá running si todo está funcionando correctamente.

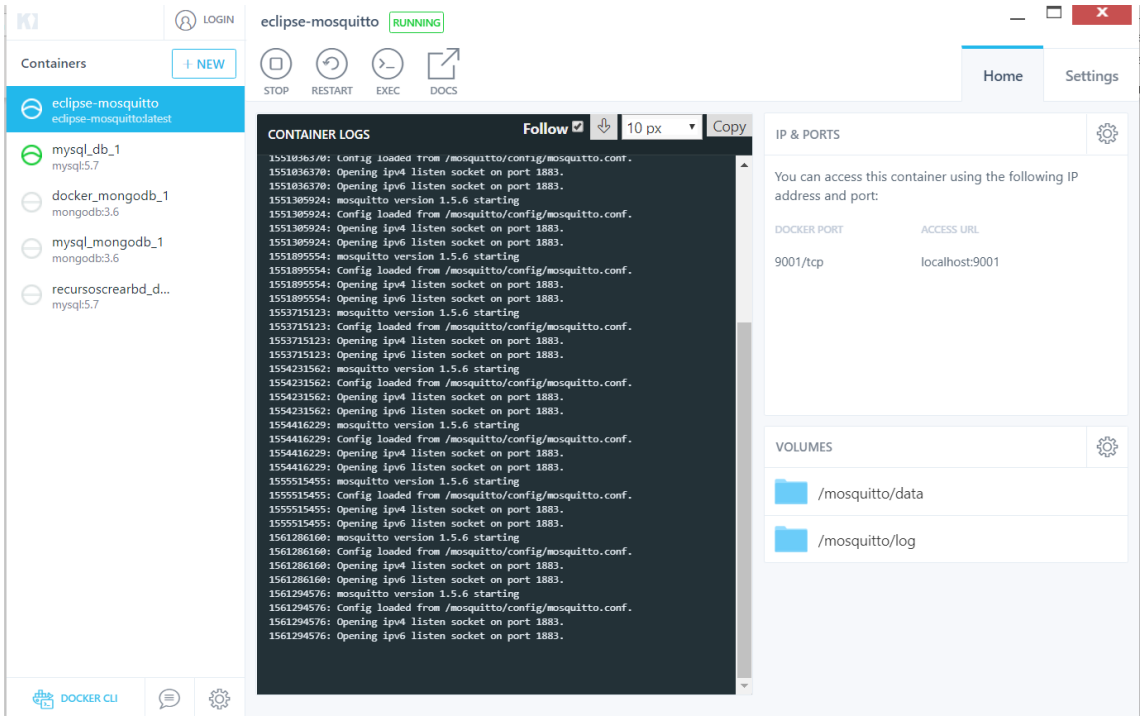


FIGURA 25 KITEMATIC FUNCIONANDO

5.2 Configuración de nuestra base de datos.

En este caso, no utilizaremos kitematic, ya que desde un inicio necesitaremos una configuración más avanzada y automatizada, así que utilizaremos Docker-compose.

```
version: '3.3'
services:
  db:
    image: mysql:5.7
    restart: always
    environment:
      MYSQL_DATABASE: 'TFG'
      # So you don't have to use root, but you can if you like
      MYSQL_USER: 'user'
      # You can use whatever password you like
      MYSQL_PASSWORD: 'password'
      # Password for root access
      MYSQL_ROOT_PASSWORD: 'password'
    ports:
      # <Port exposed> : < MySQL Port running inside container>
      - '3306:3306'
    expose:
      # Opens port 3306 on the container
```

```
    - '3306'
    # Where our data will be persisted
   volumes:
    - my-db:/var/lib/mysql
# Names our volume
volumes:
  my-db:
```

Este es el script que necesitaríamos para tener listo nuestro contenedor con una base de datos lista. Para generar este script basta con utilizar cualquier editor de texto, como por ejemplo el bloc de notas. Tenemos que nombrarlo como Docker-compose.yml.

Vamos a detallar los aspectos más importantes de este fichero, primero destacar que dentro de services irán todos los servicios que queramos arrancar de una vez dentro de nuestro script. Actualmente solo necesitamos el servidor mysql, por lo tanto, solo tendremos dentro de services, un único elemento, en “images” pondremos qué imagen es la que queremos instalar dentro de Docker. En nuestro caso será mysql-5.7, dentro de” envoirments” donde pondremos las configuraciones básicas de nuestra base de datos.

En ports, especificaremos qué puerto externo e interno de Docker utilizaremos, y en export, lo exportaremos a la maquina anfitriona.

Por último, en “volumes”, es la carpeta que tendremos que tener expuesta a la maquina anfitriona, ya que serán los datos almacenados; Si no lo exponemos a la maquina anfitriona nunca persistirán los datos, porque que al reiniciar Docker o pararlo perderíamos todos los datos.

Una vez guardado este fichero, abriremos un terminal en la ruta donde lo tenemos guardado y ejecutaremos el siguiente comando “docker-compose up”.

Una vez realizados estos pasos, ya tendremos lista nuestra base de datos.