Universidad Zaragoza
1542

Trabajo Fin de Grado

Medición del Impacto en el trabajador de soluciones Industria 4.0 mediante el uso de logs.

Measurement of the Impact on the worker of Industry Solutions 4.0 using logs.

Autor/es

Beatriz Franco García

Director/es

Francisco José Lacueva Pérez

María de la Vega Rodrigálvarez Chamarro

Ponente

Sergio Ilarri Artigas

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2018

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. BEATRIZ FRANCO GARCÍA,

con nº de DNI 73016310 G en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) GRADO, (Título del Trabajo) MEDICIÓN DEL IMPACTO EN EL TRABAJADOR DE SOLUCIONES INDUSTRIA 4.0 MEDIANTE EL USO DE LOGS

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 30 ENERO 2018

Fdo: BEATRIZ FRANCO GARCÍA

MEDICIÓN DEL IMPACTO EN EL TRABAJADOR DE SOLUCIONES INDUSTRIA 4.0 MEDIANTE EL USO DE LOGS.

RESUMEN

La llegada de la Industria 4.0 ha supuesto un cambio de mentalidad en los trabajadores, debido a la incorporación masiva de las tecnologías digitales en toda la cadena de valor de la industria.

El trabajo se realiza en los entornos de Thermolympic, empresa participante en el proyecto europeo Facts4Workers. Dicha empresa ha comenzado el proceso de digitalización modificando la maquinaria de forma que sea esta la encargada de realizar los controles de calidad en las piezas producidas, y los trabajadores únicamente deban verificar y encargarse de aquellas piezas que la maquinaria haya considerado defectuosas. Para facilitar este proceso a los trabajadores y gestionar estas piezas de forma cómoda, como parte del proceso de digitalización de la empresa, se les han entregado unas tablets que se encargan de detectar la posición del fallo de la pieza y de avisar a los trabajadores mediante una notificación de que hay nuevas piezas defectuosas. Todas estas tareas que los trabajadores van realizando a través de la tablet se van registrando en unos ficheros de log que esta genera. Estos logs se encargan de recoger información de todas las peticiones realizadas por los trabajadores con la tablet, registrando tiempos de acceso y respuesta, así como el estado de finalización de las peticiones.

El Trabajo Fin de Grado que se presenta a continuación se centra en la extracción de información procedente de estos ficheros de log para su análisis con el fin de medir el impacto que introduce el uso de las nuevas tecnologías incorporadas en la Industria 4.0 y el grado de adaptación que tienen en los trabajadores.

Para la realización de dicho análisis se aplican técnicas de Big Data, en particular, se lleva a cabo un proceso de Data Science. En primer lugar, se establece un canal de mensajería que se encarga de extraer la información de los logs en tiempo cuasi-real y de transmitirla al sistema de recogida que se ha implementado. Este sistema se encarga de recoger la información procedente del canal de mensajería, procesarla y almacenarla en una base de datos para su posterior análisis. Tras ello, se implementa un script encargado de leer del sistema de almacenamiento la información procesada que se ha recibido de los logs y analiza estos datos para guardar sus resultados en un repositorio. Finalmente, el Trabajo Fin de Grado se completa mediante la implementación de un interfaz web que se encarga de recoger los resultados del análisis del repositorio y mostrarlos de manera gráfica al usuario que desea consultarlos.

Durante este proceso tanto el análisis de los datos como el procesamiento y almacenamiento de los que llegan del canal de mensajería se está efectuando de manera continua, en tiempo cuasi-real.

Finalmente, se considera que este proyecto es de gran interés debido a la actualidad del tema a tratar y a la importancia que tiene la medición del impacto de la Industria 4.0 en los trabajadores para que el proceso de digitalización se complete con éxito.

La realización de este Trabajo Fin de Grado ha permitido conocer y aplicar nuevas técnicas y tecnologías que se utilizan en muchos procesos de análisis de datos. Además, ha sido útil comprender la importancia del tratamiento y procesamiento de estos datos con cuidado para poder extraer conclusiones fiables y comprender como la visualización de estos resultados es tan importante como su análisis.

Contenidos

Índice de figuras

Índice de tablas

Índice de abreviaturas

- IoT –> *Internet of Things*
- Pwc –> *Price Waterhouse Coopers*
- F4W –> *Facts4Workers*
- BI –> *Business Intelligence*
- I+i –> Investigación e innovación
- TFG –> Trabajo Fin de Grado
- THO –> *Thermolympic*
- API –> *Application Programming Interface*
- SQL –> *Structured Query Language*
- REST –> *REpresentational State Transfer*
- ACK –> *ACKnowledgement*
- TCP –> *Transmission Control Protocol*
- TLS –> *Transport Layer Security*
- IDE –> *Integrated Development Environment*
- BSON –> *Binary JSON*
- JSON –> *JavaScript Object Notation*
- HTTP –> *HyperText Transfer Protocol*
- CRISP-DM –> *Cross Industry Standard Process for Data Mining*

Glosario de palabras

- ***Data Lake***: Repositorio donde se almacenan todos los datos, estructurados y no estructurados, sin ningún procesamiento y sin ningún esquema para que sean analizados posteriormente.
- ***Price Waterhouse Coopers***: Es la firma de servicios profesionales más grandes del mundo, prestando servicios de auditoría, consultoría y asesoramiento legal y fiscal a las principales compañías, instituciones y gobiernos a nivel global.
- **Industria 4.0** [1, 2, 3, 4] También se conoce como cuarta revolución industrial, industria inteligente o ciberindustria del futuro y consiste en la introducción de las tecnologías en las fábricas, es decir, hace referencia a la digitalización de los procesos productivos en las fábricas mediante sensores y sistemas de información para transformar los procesos productivos y hacerlos más eficientes.
- ***Facts4 Workers*** [5]: Proyecto europeo de investigación que trata de resolver los problemas planteados en la definición del concepto Industria 4.0, centrándose en el componente humano de la producción.
- ***Big Data***: Actualmente el volumen de los datos masivos crece constantemente y este término hace referencia a la cantidad de datos tal que supera la capacidad del software convencional para ser capturados, administrados y procesados en un tiempo razonable.
- ***Data Science***: También conocida como ciencia de datos, es un campo interdisciplinario que involucra métodos científicos, procesos y sistemas para extraer conocimiento de datos, estructurados o no. Esta ciencia involucra campos de análisis de datos como la estadística, la minería de datos, el aprendizaje automático y la analítica predictiva.
- **Bases de datos NoSQL**: Sistemas de almacenamiento que no cumplen con el esquema entidad-relación ni utilizan una estructura de datos en forma de tabla sino que hacen uso de formatos clave-valor, mapeo de columnas o grafos. Tienen mejor escalabilidad y rendimiento que las bases de datos tradicionales.
- **Replicación primario-copia**: Modo de replicación en el que el primario es el que se encarga de realizar las operaciones de lectura y escritura que van llegando y posteriormente los secundarios replican sus datos para usarlos como copia de seguridad o bien para atender peticiones de lectura, en caso de que el primario deje de responder son estos secundarios los que se encargan de elegir uno nuevo.
- ***Sharding***: También conocido como fragmentación es un método de almacenamiento que consiste en distribuir los distintos datos entre las diferentes máquinas.
- ***Topic/Tema***: Categoría en la que se publican los registros que llegan a Kafka y al que se subscriben los consumidores para obtener los mensajes que van llegando. En Kafka los topics pueden tener desde cero hasta múltiples consumidores.
- ***Canal***: Sistema distribuido destinado a la recolección, agregación y movilización de grandes cantidades de datos no estructurados y semi-estructurados desde múltiples fuentes a un almacén de datos central.

Capítulo 1

Introducción

En los últimos años la tecnología está evolucionando vertiginosamente, lo que ha dado lugar a una nueva revolución industrial, conocida como Industria 4.0 [1], en la que no sólo las Tecnologías de la Información juegan un papel importante sino que también la Comunicación, ya que dicha revolución pretende una incorporación masiva de tecnologías digitales en toda la cadena de valor de la industria. Este concepto supone una nueva forma de organizar los medios de producción, permitiendo que las fábricas inteligentes del futuro sean capaces de adaptarse mejor a las necesidades de los clientes.

La Industria 4.0 presenta un modelo que promete un incremento de la competitividad sin sacrificar el empleo, sino incorporando robots que colaboren con humanos, máquinas que hablen entre sí, cadenas de producción adaptadas a la demanda y sistemas capaces de prever fallos antes de que estos ocurran. Todo esto permitirá procesos más rápidos, más eficientes y más flexibles para la fabricación de productos de mayor calidad con un coste más reducido.

A través de la digitalización y el uso de plataformas conectadas, la Industria 4.0 ofrece una capacidad de adaptación constante a la demanda, con un servicio al cliente más personalizado. Además, logrará diseñar, producir y vender productos en menos tiempo, creando series de producción más cortas y rentables, siendo para ello un punto importante el aprovechamiento de la información para su análisis desde múltiples canales donde ser capaces de analizarla y explotarla en tiempo real.

Además, la Industria 4.0 supone un cambio en la mentalidad importante, ya que son muchos los puntos que pretende incorporar: *Big Data* y análisis de datos, *cloud computing*, ciberseguridad, robótica, IoT [6] y realidad aumentada entre otras. A pesar de todo, el reto no está en conseguir integrar todos ellos.

El verdadero reto estará una vez más en las personas, en cómo lideren este proceso de transformación digital dentro de la organización y en el cambio que supondrá para ellas adaptarse a trabajar en los nuevos entornos conectados de la Industria 4.0. La Industria 4.0 no supondrá una factoría auto-dirigida gestionada por robots a corto plazo, sino que muchos procesos serán gestionados mediante robótica o incluso drones pero dirigidos por humanos en la distancia. Es por tanto un factor importante el adaptar este ritmo de digitalización a las personas que trabajan en las plantas de producción, de manera que no suponga un cambio drástico para estas, ya que en ello residirá la clave del éxito.

Es de gran importancia tener en cuenta que en estos casos muchas veces el éxito no va ligado a implantar lo último en nuevas tecnologías, sino que es fruto de aplicar las soluciones que mejoren la eficiencia en cada proceso. De esta manera, es evidente que es de gran importancia hacer un seguimiento del impacto que tienen las nuevas tecnologías y comprender cómo afectan estas en las organizaciones y en sus trabajadores.

En este Trabajo Fin de Grado se realizará un seguimiento del impacto de estas nuevas tecnologías a través del análisis de la información que las herramientas tecnológicas generan siguiendo para ello un proceso de *Data Science*.

En este apartado se presenta una breve introducción al *Data Science*, una breve descripción del problema a resolver, el contexto en el que se enmarca el trabajo y la motivación para realizarlo, los retos tecnológicos que enfrenta y los objetivos que se pretenden alcanzar, detallando por último cómo se organiza el resto de la memoria.

1.1. Data Science

La ciencia de datos o *Data Science* es un campo interdisciplinario que involucra métodos científicos, procesos y sistemas para extraer conocimiento o un mejor entendimiento de datos en sus diferentes formas, ya sea estructurados o no estructurados. Hay quién considera este proceso como una continuación de algunos campos de análisis de datos como la estadística, la minería de datos, el aprendizaje automático y la analítica predictiva. Jim Gray [7], ganador del premio Turing, lo definió como un cuarto paradigma de la ciencia que surge debido al cambio que se está produciendo en la misma debido al impacto de la tecnología de información y al diluvio de datos. Al realizar un proceso de *Data Science* los investigadores se apoyan en sistemas y procesos muy diferentes a los utilizados en el pasado, como son los modelos, ecuaciones, algoritmos, así como en la evaluación e interpretación de resultados.

Etapas del *Data Science*

Antes de llevar a cabo cualquier análisis de datos es necesario recoger los datos que se desea analizar y convertirlos a un formato que resulte útil y manejable a la hora de realizar dicho análisis (Figura (1.1)). Esta recogida de datos puede realizarse desde diversas fuentes, siendo de especial interés almacenar los datos en crudo, sin ningún procesamiento, por si es necesario realizar nuevas hipótesis no tener que volver a obtener todos ellos. Otro problema secundario que surge en esta fase es el almacenamiento de datos, las cantidades de datos que se manejan en este tipo de problemas hace que el almacenaje en un único lugar sea inviable, por lo que estos deben almacenarse en servidores remotos.

Una vez se ha recogido la información necesaria es muy probable que estos datos necesiten ser limpiados o modificar su formato para que nos resulten útiles. Esto es debido a que normalmente los datos en crudo que se han recogido no se encuentran en formatos adecuados para el análisis ya que no se almacenaron con ese fin. Además, estos datos pueden contener errores semánticos o formatos inconsistentes, por lo que este proceso de limpieza de datos es necesario para simplificar el posterior proceso de análisis de los datos. Este proceso suele ser el más tedioso y el que más tiempo consume, pudiendo realizarse tanto manualmente como mediante el uso de *scripts* y siendo en ocasiones necesario integrar información procedente de diversas fuentes de información para que sean accesibles más cómodamente.

Tras ello, se lleva a cabo la principal fase del proceso de *Data Science*, es decir, el análisis de datos. Para ello se pueden diseñar diferentes programas, conocidos como "*data analysis scripts*", que se encarguen de obtener información útil de los datos recolectados. Esta fase consiste en un ciclo iterativo de crear *scripts*, ejecutarlos y obtener resultados, de manera que tras una serie de ejecuciones se han corregido todos los fallos y mejoras detectados y se tienen los resultados deseados. Para poder detectar errores y mejoras en los *scripts* que se están ejecutando es necesario alternar esta fase con una fase de interpretación de los datos.

La fase final de todo el proceso es la diseminación o visualización de los resultados que ayuda a la toma de futuras decisiones. En la mayoría de ocasiones estos resultados suelen presentarse como informes o presentaciones, siendo su principal reto consolidar todas las notas y resultados de salida obtenidos en la fase anterior.

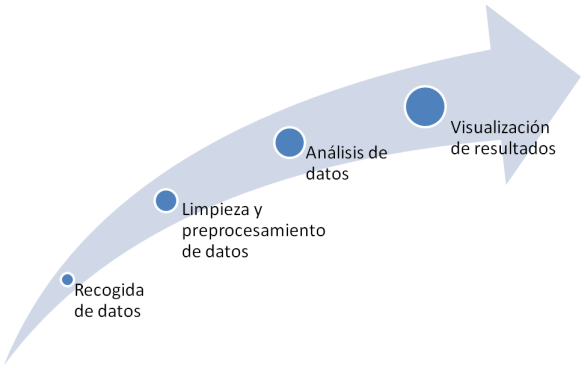

Figura 1.1: Etapas proceso data science

1.2. Problema a resolver

La Industria 4.0 presenta un modelo que promete un incremento en la competitividad de las empresas a la vez que requiere un cambio de mentalidad importante para los trabajadores, ya que son muchas las novedades tecnológicas introducidas. Por tanto, es necesario adaptar dichos cambios a las capacidades de los trabajadores para que este proceso de adaptación no sea tan drástico.

Un informe presentado por Pwc [8] señala que la velocidad en la digitalización en España es mucho menor que en el resto del mundo, estimando que para 2020, a nivel global, el 72 % de las industrias estarán digitalizadas, mientras que en España únicamente el 19 % de las mismas lo habrán conseguido (Figura (1.2)). Esto es debido, entre otros factores, a la falta de cultura digital y al bajo número de profesionales formados en nuestro país.

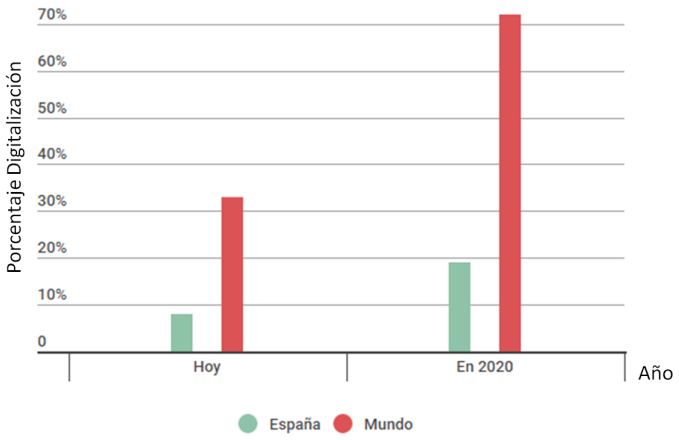

Figura 1.2: Ritmo de digitalización

Este Trabajo Fin de Grado se centrará en la extracción y análisis de logs y la información que estos contienen para medir el impacto que se produce al introducir estas nuevas tecnologías y el grado de aceptación de los trabajadores a las mismas.

1.3. Contexto y motivación

El Trabajo de Fin de Grado surge de una propuesta de beca por parte del Instituto Tecnológico de Aragón (ITAINNOVA) dentro del grupo de Big Data y Sistemas Cognitivos.

ITAINNOVA es el referente en Aragón de la investigación e innovación tecnológica que analiza las necesidades del mercado y genera soluciones de I+i para la mejora competitiva de las empresas.

En concreto este Trabajo de Fin de Grado se realiza en el marco del proyecto Facts4Workers [5], proyecto de la Unión Europea cuyo objetivo es demostrar que las soluciones tecnológicas que se incluyen en las empresas sirven para incrementar tanto el rendimiento como el conocimiento de los trabajadores, en el que ITAINNOVA es uno de los 15 socios europeos que forman el consorcio.

En este caso el TFG se centra en realizar un estudio de cómo evoluciona el rendimiento de los trabajadores al utilizar las herramientas tecnológicas incorporadas en la Industria 4.0 analizando los ficheros de log que estas generan.

Más en detalle el proyecto se ha realizado en Thermolympic (THO) [9], empresa especialista en el moldeo por inyección de termoplásticos con sede en Zaragoza. Esta empresa se encuentra, como la mayoría en la actualidad, en un proceso de digitalización, ya que hasta ahora todos los procesos de gestión en planta se realizaban manualmente y no se poseía información. La llegada de la revolución Industria 4.0 ha hecho que esto cambie y muchos procesos se digitalicen incorporando herramientas tecnológicas para facilitar el uso de los trabajadores. THO ha optado por digitalizar los procesos mediante la incorporación de máquinas inteligentes que reducen el trabajo manual de los trabajadores. Los trabajadores no tienen necesidad de comprobar la calidad de los productos de forma manual y realizar todas las anotaciones a papel, sino que es la máquina la que lo hace por ellos. Además, en caso de que las piezas salgan defectuosas, la tablet que se ha incorporado en el proceso es la encargada de avisarles y de indicarles el lugar en el que se encuentra el defecto (Figura (1.3)).

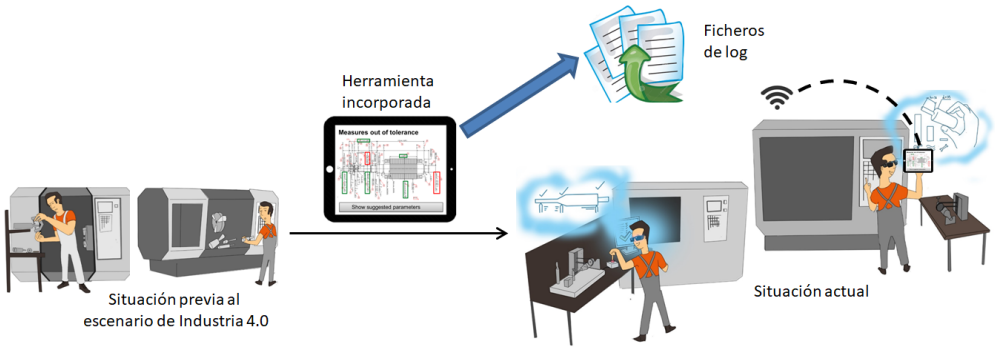

Figura 1.3: Esquema global del contexto. Figura adaptada de [10]

Todas las operaciones que los trabajadores van realizando con la tablet quedan almacenadas en unos ficheros de log que esta va generando y que serán los que se utilicen en este proyecto. En THO se ha aplicado esta solución de Industria 4.0 en tres casos de uso diferentes [10]. A continuación, se detalla uno de ellos y el resto pueden verse en el Anexo (H).

Antes del proceso de digitalización si los trabajadores eran rápidos revisando las medidas de las piezas y querían aprovechar el tiempo libre en mejorar y aprender nuevas cosas lo tenían difícil, ya que la única información de la que disponían era un papel en la pared. Esta información se encontraba en malas condiciones para ser consultada debido al paso de los años, y no disponía de la información de los últimos módulos (mejoras de equipamiento) introducidos en la empresa, haciendo que la motivación de los trabajadores por formarse en los procesos disminuya.

Con la implantación de soluciones Industria 4.0 los trabajadores no tienen que comprobar una a una las piezas ya que lo hace la máquina por ellos, y sólo en el caso de que esta detecte una pieza defectuosa deben actuar. Los trabajadores ahora disponen de una *tablet* en la que pueden consultar detalles de la máquina y de los procesos realizados, fomentando el aprendizaje. Además, en caso de que vean que algún proceso puede mejorarse pueden

poner un aviso y el jefe irá a hablarlo con ellos, de manera que los trabajadores mantienen su motivación ya que pueden aprender nuevas cosas y aportar su opinión. Esto no hace que el ritmo de producción de la empresa disminuya por no estar pendientes de las piezas, ya que en la propia tablet aparece una notificación cuando una pieza no cumple con las medidas que habían sido especificadas.

A pesar de ello, el uso de estas herramientas puede dar errores de utilización por parte de los trabajadores y es esto lo que se pretende analizar, con el fin de mejorar estos procesos de uso de las herramientas.

1.4. Retos tecnológicos

A continuación, se detallan los retos tecnológicos que llevan a la necesidad de utilizar una arquitectura de Big Data junto con las tecnologías y herramientas de esta en lugar de utilizar técnicas tradicionales de análisis de datos.

Uno de los factores más importantes es la gran cantidad de datos que se generan. La planta de producción de la empresa se encuentra abierta 24/7, realizando turnos entre los trabajadores, por lo que continuamente estos se encuentran generando información que se va almacenando en los ficheros de log. Toda esta información que se genera es necesario procesarla y poder almacenarla en tiempo real, por lo que las tecnologías tradicionales se quedarían cortas en este sentido. Las tecnologías tradicionales no están diseñadas pensando en almacenar datos de manera continua en los repositorios, sino que están pensadas para que estos datos se vayan recogiendo y se vuelquen en el repositorio de manera periódica.

Otra de las necesidades que lleva a la utilización de tecnologías propias de Big Data es la falta de datos con una estructura fija que se pueda definir de antemano. En el contexto presentado, las tablets generan información siguiendo una serie de patrones que varían en función de la invocación del trabajador y la respuesta que este recibe.

Además, se trata de un proyecto de futuro, ya que la Industria 4.0 es una revolución que se encuentra en sus inicios. En este proyecto en concreto, a pesar de que se ha comenzado digitalizando únicamente la información procedente de las tablets, una vez que los trabajadores se encuentren adaptados a este nueva forma de trabajar, se pretende incorporar otras fuentes de datos de las que también se deseará extraer información y correlarla con la ya disponible para realizar otros análisis. Considerando estos aspectos, se consideró como objetivo definir y desplegar una infraestructura adecuada que permitiera fácilmente futuras extensiones.

1.5. Objetivos

Este Trabajo fin de Grado pretende basarse en un análisis de ficheros de log que generan las herramientas introducidas en la Industria 4.0 para mostrar el impacto que tiene la incorporación de estas nuevas tecnologías. Cómo el uso de las tecnologías introducidas en la Industria 4.0 permite, no sólo mejorar las condiciones de los trabajadores, sino también su rendimiento y el de las organizaciones en las que estos trabajan. Para ello ha sido necesario acceder a los sistemas en los que se encontraban los ficheros de log y realizar una extracción de los datos de los mismos en tiempo real.

Seguidamente se detallan los objetivos específicos que se deben alcanzar a lo largo del proyecto:

- Conocer las diferentes herramientas disponibles para realizar extracción de datos en tiempo real y las principales ventajas de cada una de ellas.
- Comunicar/Conectar las diferentes herramientas elegidas para el proceso de extrac-

ción y almacenaje de datos.

- Entender los datos disponibles en los ficheros de log y a que hacen referencia cada uno de ellos.
- Observar qué tipos de análisis pueden realizarse a través de los datos disponibles en los diferentes ficheros de log.
- Implementar y programar aquellos análisis sobre los datos que se hayan considerado de interés.
- Visualizar los resultados del análisis a través de un interfaz web que nos permita extraer conclusiones sobre los datos.

Los objetivos desde el punto de análisis y explotación de la información que se pretenden lograr es realizar un análisis cuantitativo para buscar patrones de comportamiento con las herramientas con el fin de mejorar la experiencia de uso de los trabajadores con estas y optimizar los procesos que se llevan a cabo en la planta de producción.

1.6. Estructura de la memoria

Esta memoria expone de forma precisa y descriptiva el trabajo realizado a lo largo de todo el proceso, explicando de manera detallada los aspectos más relevantes de cada una de las etapas por las que se ha pasado. La memoria se divide en 5 capítulos y se complementa con varios anexos.

En el segundo capítulo se realiza un análisis del proyecto, en el que se analizan los requisitos, se valoran las necesidades y, de acuerdo a esto, se realiza un análisis y elección de las herramientas a utilizar para llevar a cabo el proceso de *Data Science*. Una vez tomada la decisión de las herramientas que se van a utilizar en el proyecto se detalla la arquitectura que se va a seguir para llevar a cabo la implementación, exponiendo inicialmente una versión general de la misma y, posteriormente, detallando para este proyecto qué elementos forman parte de cada nivel de la arquitectura. Además, se muestran aspectos de interés que es necesario tener en cuenta a la hora de realizar el proyecto.

En el tercer capítulo se explica en profundidad en qué consiste el proceso de *Data Science*, los aspectos que hay que tener en cuenta en cada una de las fases y más concretamente que es lo que se ha realizado para este proyecto en cada una de ellas.

El capítulo 4 expone las conclusiones obtenidas del proceso de *Data Science* junto con la experiencia y conocimientos adquiridos en la realización de este proyecto, las limitaciones encontradas a la hora de realizar el proyecto, futuras ampliaciones o aplicaciones que se le puedan dar al mismo y por último un apartado de gestión del proyecto.

Finalmente, en los anexos se puede encontrar información adicional relativa al Trabajo Fin de Grado. En el Anexo A se completa la información de las herramientas que se han utilizado a lo largo del proyecto. Se muestran ejemplos del código desarrollado en cada una de las etapas. En particular, el *script* que se ejecuta para obtener los datos del canal y preprocesarlos (Anexo D) y el que se encarga de analizar los datos (Anexo E). También podemos encontrar un manual de instalación en el entorno de desarrollo con los pasos a seguir en cada etapa (Anexo G). Se puede encontrar una sección en la que se detalla el formato original de los datos recibidos a través del canal (Anexo B), otra con las colecciones creadas tanto con los datos de entrada preprocesados como con los resultados que se van almacenando del análisis (Anexo C) y una con el mapa de navegación que se ofrece a través del interfaz web y con una explicación más detallada de cada una de las pantallas disponibles (Anexo F). Finalmente, se muestran con mayor detalle los diferentes casos de uso en los que se han introducido soluciones de Industria 4.0 en Thermolympic (Anexo H).

Capítulo 2

Análisis del proyecto

En este capítulo se describe un análisis previo a la realización del proyecto en el que se define la base del proyecto, estableciendo los requisitos que debe cumplir el proyecto, un análisis de las herramientas disponibles para poder llevar a cabo dichos requisitos y la decisión de las que se emplearán finalmente en el proceso de *Data Science*. Además, se detalla la arquitectura en la que se estructura el proyecto y las herramientas que se utilizarán en cada capa de la misma.

2.1. Análisis de herramientas

El principal punto de este TFG consiste en establecer el canal de mensajería, es decir, en establecer un canal que se encargue de extraer los datos de la fuente (en este caso de los logs) y de enviarlos al procesador de datos que será el que se encargue de almacenarlos en la base de datos para su posterior tratamiento y análisis. A la hora de tomar la decisión de qué herramienta cumplirá este papel es importante tener claro cuáles son las características de este y qué necesidades debe cumplir la herramienta. Por ello, en primer lugar, se van a establecer las necesidades del proyecto para ver luego qué herramienta se adapta mejor (Tabla (2.1)).

NECESIDAD	DESCRIPCIÓN
Escucha en tiempo real	Es interesante poder capturar la información que se añade a los logs a medida que esta se introduce, sin necesidad de tener que ejecutar comandos para conseguirla.
Grandes volúmenes de datos	Los distintos subsistemas están añadiendo información a los logs de manera continua, por lo que al unirla toda las cantidades de datos que se generan son enormes.
Distribución de carga balanceada	Evitando balanceo de carga estático evitamos perder rendimiento en caso de que ciertas zonas sean más costosas de procesar que otras, ya que los consumidores pedirán fragmentos a medida que los acaben y no los tendrán preasignados.
Tolerancia a fallos	Toda la información es importante, por lo que es importante poder recuperarla en caso de que se produzca una caída.
Filtrado de datos	Eliminar información que no nos va a resultar de interés para agilizar la fase de preprocesamiento de datos previa al almacenaje.

Tabla 2.1: Requisitos de la herramienta

Una vez definidas las características que debería poseer la herramienta seleccionada (Tabla 2.1), de entre las disponibles se ha hecho una preselección para determinar cuáles eran las que mejor se adaptaban a estas necesidades.

Entre las herramientas de las que se va a hablar están Flume, Kafka, LogStash y Syslog-ng.

Flume [11]

Flume es un servicio distribuido, disponible para recopilar, agregar y mover grandes cantidades de datos.

Flume tiene una arquitectura sencilla y flexible basada en flujos de data *streaming*. Se basa en un agente que consume datos que finalmente almacena en un repositorio o envía como entrada a otra herramienta. Se encuentra dividido en 3 elementos principales: la fuente, que es la encargada de consumir los eventos que le llegan y almacenarlos en uno o múltiples canales; los canales que son almacenes pasivos en los que se guardan los eventos hasta que son consumidos; y por último, el elemento destino que es el encargado de consumir eventos del canal y enviarlos a la salida.

Permite al cliente poder gestionar una gran variedad de datos, siempre que el formato del mensaje sea el definido en la fuente. La última versión disponible es de Octubre de 2016.

Kafka [12]

Kafka es un herramienta diseñada para el manejo de grandes cantidades de datos en tiempo real, en particular, ficheros de log.

Kafka se basa en un modelo de arquitectura productor-consumidor. Posee canales de mensajes de categorías particulares conocidos como temas o *topics*. Cada productor publica sus mensajes en un tema mientras que los consumidores se encuentran subscritos a estos y son los encargados de consumirlos.

Los servidores, conocidos como *brokers*, se agrupan en conjuntos formando el *cluster* de Kafka. Para coordinar estos *brokers* entre sí Kafka utiliza Zookeeper [13]. Zookeeper se encarga además de notificar tanto a productores como consumidores los cambios en la configuración del *cluster* (si falla algún *broker* o se añade alguno nuevo).

Kafka utiliza datos primitivos y ofrece integración con APIs para el filtrado de los datos.

Kafka ofrece una evolución incremental de versiones de manera que todas ellas son compatibles hacia atrás y son los clientes los que deciden qué versión usar, particularizando sus protocolos.

Kafka ofrece distribución con tolerancia a fallos para lo que utiliza replicación. La información se reparte entre los servidores disponibles que poseen distintos grupos de consumidores. Cada fragmento de información es enviado a un consumidor de cada grupo de manera balanceada.

LogStash [14]

LogStash es un servicio de código abierto, que se encarga de servir datos transportándolos desde multitud de fuentes simultáneamente, procesándolos, transformándolos y enviándolos al almacenamiento deseado.

Su arquitectura se basa principalmente en un módulo de entrada de diversos formatos de datos en el que se recogen datos de la fuente. Posteriormente estos se pasan por un módulo de filtrado potente en el que se les puede aplicar una serie de reglas para tener la información más disponible en el análisis y finalmente un módulo de salida que se encarga de devolver los datos al cliente de la manera deseada.

LogStash ofrece buena distribución y tolerancia a fallos, ya que cada entrada de la tubería se ejecuta en su propio hilo. Estas entradas se dirigen a un *worker*, bloqueándolas en caso de que todos se encuentren ocupados. El número de *workers* por tubería es configurable,

de manera que se le puede dar la escalabilidad deseada. Además mediante el uso de colas de persistencia logran ofrecer tolerancia a fallos. LogStash realiza el procesamiento en *batches*. Si un evento no concluye correctamente, no ha recibido un ACK, puede tratarse en más de una ocasión.

Syslog-ng [15]

Syslog-ng es una aplicación flexible y altamente escalable que permite enviar mensajes de log desde un host remoto a otros servidores o sistemas de almacenamiento. Este permite recoger y almacenar la información de logs centralizada en servidores dedicados.

El uso de protocolos TCP asegura que no se pierden mensajes. Además soporta TLS, de manera que ofrece seguridad, encriptando la comunicación para que esta no pueda ser accedida por terceros.

Muchos mensajes son difíciles de procesar, ya que son desestructurados. Syslog-ng ofrece la posibilidad de filtrarlos y clasificarlos en base a ciertos parámetros del mensaje, y utilizar expresiones regulares y operadores booleanos para detectar los mensajes de mayor importancia y llevarlos a los destinos deseados. Además, permite segmentar columnas para nombrar campos o modificar valores.

Conclusiones

Una vez conocido el funcionamiento de cada herramienta y definidas las necesidades ya se puede tomar una decisión de cuál es la que mejor se adapta.

En primer lugar se ha decidido optar por Syslog-ng para extraer los datos de los ficheros de log. El principal motivo es que F4W (Facts4Workers) (Página 1) había elegido Docker como herramienta de despliegue de los desarrollos y de esta manera era más fácil la integración. Además, los trabajos previos de evaluación han demostrado que esta herramienta funciona bien realizando este proceso.

Para alimentar siempre al analizador con nueva información, se necesita transferir datos en crudo mediante un sistema de colas que permita garantizar la utilización sólo de datos nuevos así como su procesamiento. LogStash no garantiza su tratamiento de manera única, por lo que la elección de la herramienta de transferencia queda reducida a Flume y Kakfa. La tabla (2.2) muestra una comparativa de sus características.

CARACTERÍSTICAS	**FLUME**	**KAFKA**
Balanceo de carga	Estático	Dinámico
Filtrado de datos	Mediante expresiones regulares	Mediante APIs externas
Tolerancia a fallos	Al sobrecargarse el canal puede perder datos	Buena
Volumen de datos	Regular	Muy bueno
Tiempo real	Buena	Buena

Tabla 2.2: Comparación de herramientas

Observando la tabla de comparativas (Tabla 2.2) se puede determinar que la herramienta que mejor se adapta a las necesidades del proyecto es Kafka ya que de esta manera se evita la pérdida de datos por sobrecarga del canal y a pesar de no poseer un filtrado propio se puede realizar el filtrado de datos en Syslog-ng y de esta manera evitar tener que añadir APIs externas.

2.2. Arquitectura del proyecto

Una vez tomada la decisión de qué herramientas se van a utilizar en el proyecto, se puede realizar un esquema de la arquitectura del sistema.

La arquitectura que se emplea en el proyecto adapta una arquitectura tradicional de *Big Data* (Figura (2.1)), ya que permite realizar los pasos tradicionales del proceso de *Data Science* de manera simple y sencilla.

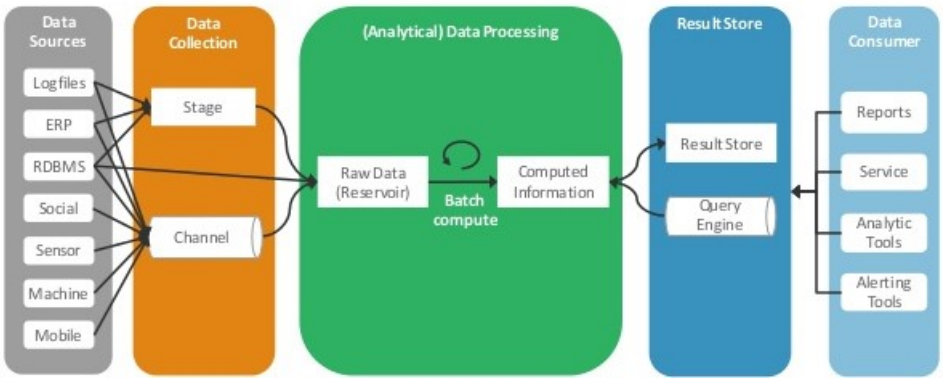

Figura 2.1: Arquitectura del proyecto. Extraída de [16]

Este tipo de arquitectura es común en proyectos de *Big Data* en los que es necesario recoger grandes cantidades de datos desde diferentes fuentes mediante canales para poder analizarlos. Una vez se tienen los datos recolectados, estos son procesados con el fin de obtener información que aporte algún tipo de valor. Esta información es analizada y estudiada cuidadosamente, además de ser almacenada en repositorios de datos con el fin de poder usarla para presentar informes visuales a los diferentes usuarios.

Como podemos ver en la figura (2.1), la arquitectura utilizada se compone principalmente de los siguientes componentes:

- Fuentes de datos (*Data sources*): son los puntos de abastecimiento de los datos con información potencialmente útil para el análisis. Al hablar de fuentes de datos se suele hacer referencia a información que ha sido digitalizada para su procesamiento.
- Recolector de datos (*Data collector*): es el elemento encargado de obtener los datos de las fuentes de datos seleccionadas para el análisis.
- Analizador/Procesador de datos (*Data processor*): se encarga de leer la información recogida por el recolector de datos, realizar el procesamiento necesario previo al almacenamiento y analizar los datos de interés.
- Repositorios y almacenes de datos (*Repositories*): son colecciones de datos orientadas a un determinado ámbito, integrados y variables con el tiempo que ayudan a la toma de decisiones.
- Consumidor (*Data consumer*): interfaz de usuario, sistema o herramienta encargada de usar los datos producidos.

A continuación se explica cómo se ha adoptado esta arquitectura en el proyecto, detallando las tecnologías que se ha decidido utilizar en cada una de las capas del modelo (Figura (2.2)).

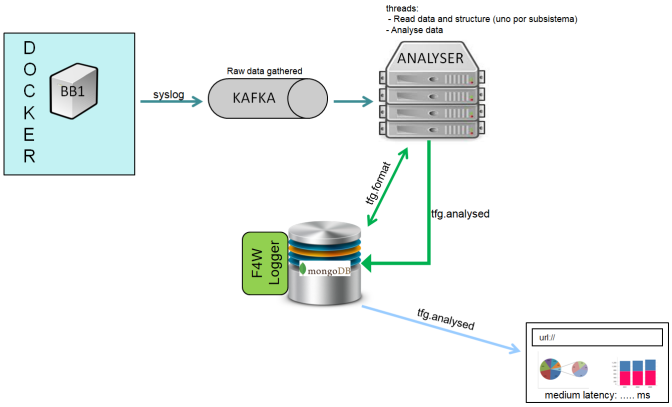

Figura 2.2: Arquitectura detallada

En este caso los ficheros de log, que serán los que actúen como fuentes de datos, se encuentran en *containers* de Docker (Anexo (A.1)), uno por cada subsistema a los que se ha aplica el proceso de análisis. Para realizar la extracción de la información de estos subsistemas se realiza a través de un canal tal y como se establece en la base de la arquitectura (Figura (2.1)), siendo en este caso Syslog-ng y combinándolo con Kafka, en el que se realizará el almacenaje de los datos en crudo. El analizador en este caso estará formado por *scripts* en Python y se encargará principalmente de dos tareas. Por un lado, se encargará de leer los datos que vayan llegando a Kafka procedentes de la extracción en tiempo real de los ficheros de log, filtrarlos y almacenarlos en los repositorios de datos con la estructura deseada en cada caso; y por otro se encargará de utilizar la información almacenada en los diferentes repositorios para realizar análisis que se consideren de interés. Como repositorio para los datos se utiliza MongoDB en el que se van a crear dos colecciones, una para los datos semiestructurados que se obtengan de las fuentes de datos y otra para los datos obtenidos en el análisis. Esta segunda colección será la que consulte el consumidor, que en este caso será un interfaz web, para mostrar de manera gráfica los resultados obtenidos en el análisis.

2.3. Definición de requisitos

En esta sección se realiza una especificación de los requisitos, es decir, una descripción completa del comportamiento del sistema que se va a desarrollar.

2.3.1. Tabla de requisitos

En la siguiente tabla (Tabla (2.3)) se muestra el catálogo de requisitos que debe cumplir el proyecto.

REQUISITO	DESCRIPCIÓN
F1	Los datos a analizar se obtienen de los ficheros de log de las herramientas.
F2	La información recogida es almacenada antes y después de ser procesada.
F3	La información sin procesar es almacenada en un sistema de colas.
F4	La información procesada es almacenada en bases de datos.
F5	La ingesta de datos se realiza mediante el uso de canales de mensajería.
F6	La ingesta de datos se realiza en tiempo real.
F7	Los datos que se pretenden mostrar son datos del rendimiento de los usuarios con las herramientas.
NF1	Los ficheros de log a analizar se encuentran almacenados en Docker.
NF2	La base de datos en la que se almacenará la información procesada y semiestructurada será MongoDB.
NF3	El sistema de colas en el que que se almacena la información es Kafka
NF4	El encargado de recoger los datos del canal de mensajería y procesarlos para su almacenamiento será un programa en Python.
NF5	La visualización de datos se realizará mediante interfaces web.
NF6	El canal de mensajería utilizado para la extracción de los datos es Syslog-ng junto con Kafka.
NF7	Para la implementación de interfaces web se utilizará Vaadin (Anexo (A.7)).

Tabla 2.3: Requisitos del proyecto

2.3.2. Diagrama de casos de uso

A continuación, se incluye un conjunto de casos de uso en el que se describen todas las interacciones que se realizarán (Figura (2.3)).

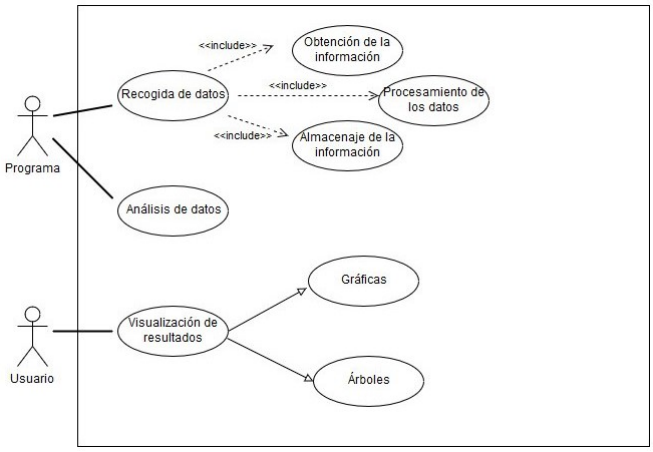

Figura 2.3: Diagrama de casos de uso

2.3.3. Descripción de escenarios

Escenario 1:

TITULO: Recogida de los datos.
OBJETIVO: Extraer los datos de los ficheros de log de las diferentes herramientas.
RECURSOS: Ficheros de log, *containers* desarrollados en Docker.
ACTORES: El sistema.
EPISODIO:

- Syslog-ng se escuentra escuchando de los ficheros disponibles en Docker.
- Cuando el fichero recibe nueva información se generan nuevos eventos que Syslog-ng registra e introduce la nueva entrada en el canal de mensajería.
- Syslog-ng envía estos datos de entrada en crudo, sin procesar, a Kafka.
- Kafka, utiliza su sistema de colas para almacenar la información en crudo.

Escenario 2:

TITULO: Procesamiento de los datos.
OBJETIVO: Procesar los datos almacenados en el sistema de colas de Kafka y almacenarlos en los repositorios.
RECURSOS: Información disponible en Kafka, MongoDB, patrones a encontrar.
ACTORES: El sistema, programa Python.
EPISODIO:

- Un programa en Python se encuentra ejecutándose y leyendo la información que va llegando a Kafka.
- Esta información es procesada de acuerdo a unos patrones y almacenada en MongoDB.

Escenario 3:

TITULO: Análisis de los datos.
OBJETIVO: Analizar la información almacenada previamente para extraer conclusiones.
RECURSOS: Información disponible en los repositorios de datos.
ACTORES: El sistema, programa Python.
EPISODIO:

- El sistema toma la información almacenada previamente en los repositorios de datos.
- Esta información es tratada y analizada para extraer conclusiones y resultados que permitan ver cómo se manejan los usuarios con las herramientas tecnológicas introducidas en los procesos.
- Una vez se han extraído los resultados son almacenados en otra colección de MongoDB.

Escenario 4:

TITULO: Visualización de los datos.
OBJETIVO: Mostrar los resultados obtenidos gráficamente mediante el uso de interfaces.
RECURSOS: Datos del análisis.
ACTORES: El usuario.
EPISODIO:

- El usuario a través de un navegador web pide ver los resultados obtenidos de los análisis de errores y descriptivos de la herramienta utilizada.
- El sistema utiliza los resultados almacenados en MongoDB como resultado del análisis para mostrárselos al usuario de manera gráfica.

2.4. Aspectos legales

A la hora de recoger la información, utilizarla y compartirla durante el proceso de investigación es importante conocer algunos aspectos relativos a la legalidad y estándares éticos que envuelven el proceso [17].

En este punto hay dos elementos importantes que hay que tener en cuenta, por un lado la seguridad de la información y por otro la protección de datos. La seguridad de la información tiene como objetivo proteger los datos para evitar su pérdida y la modificación de los mismos por personas no autorizadas (debe garantizar integridad, confidencialidad, disponibilidad y autenticidad en los datos).

Por otra parte, en la protección de datos el objetivo no es tanto los datos sino la información personal que estos pueden contener. Es importante garantizar el anonimato de estos al realizar el análisis para evitar que se pueda producir un abuso sobre estas personas. Para garantizar que esto se produzca en muchos países existen leyes que lo regulan, siendo en España la Ley Orgánica de Protección de Datos de Carácter Personal [18]. Además, es importante que previamente a la recogida de datos estos usuarios sean informados de modo expreso, preciso e inequívoco del fin con el que se va a recoger esa información y den su consentimiento.

En el caso de este Trabajo Fin de Grado, para tener todo ello en cuenta se ha informado de manera previa a los usuarios de la recogida de la información de los ficheros de log que generaban las herramientas que estaban utilizando. Además, debido a que estas máquinas utilizan la sesión del trabajador se han creado *tokens* de manera que no se pueda conocer quién está realizando las acciones. Estos *tokens* se generan de forma aleatoria y tienen un tiempo de validez, de manera que el mismo *token* en momentos distantes puede hacer referencia a distintos usuarios y durante la jornada de trabajador se han podido generar varios *tokens* para él mismo. De esta manera se consigue un anonimato total en los datos, resultando imposible asociar las acciones a un trabajador concreto. Finalmente, la parte de seguridad de la información se encuentra fuera del alcance del proyecto, ya que se supone que es garantizada por la empresa en la que se está realizando el proyecto, siendo la encargada de almacenar de forma segura los datos y facilitarnos el acceso a los mismos.

2.5. Uso de los datos

Uno de los problemas que puede surgir en este tipo de aplicaciones de *Big Data* es que estos procesos de recogida, procesamiento y análisis de datos se producen en tiempo real, por lo que puede haber momentos en los que la consistencia de los datos no se mantenga, dando lugar al problema de concurrencia conocido como productor-consumidor.

En el caso de este trabajo se está continuamente recibiendo información procedente de los ficheros de log, lo que hace que se esté continuamente actualizando la colección de datos de entrada. Esta colección, a su vez, está siendo consultada de manera continua por el *script* analizador de datos, lo que hace que las consultas que se realizan no tengan por qué corresponder todas con el mismo instante de tiempo.

Para solucionar este problema la solución que se ofrece por concurrencia sería establecer bloqueos a nivel de colección de base de datos pero esto haría que se perdiera mucho rendimiento en el proceso. Por este motivo, las soluciones de *Big Data* optan por asumir que este tipo de inconsistencias en aplicaciones de tiempo-real se producen y se establece el punto en el que ya se asume que los datos son consistentes. Un ejemplo de ello sería que si se están haciendo análisis diarios, los resultados que se vayan obteniendo del día actual no se consideran consistentes hasta el día siguiente, momento en el cual ya no se producirán actualizaciones en ese día.

Capítulo 3

Proceso de Data Science

La ciencia de datos o *Data Science* es una disciplina de reciente creación que involucra tanto métodos científicos como procesos y sistemas para extraer conocimiento de los datos. Durante estos procesos los datos pueden encontrarse tanto estructurados como no estructurados.

Es importante distinguir entre *Data Science* y *Business Intelligence* (BI) ya que esta última es una disciplina que utiliza datos completos y limpios e informa de lo que estos dicen, midiendo el rendimiento, mientras que el *Data Science* analiza grandes cantidades de datos incompletos y desordenados para impulsar decisiones sobre operaciones futuras.

Con la emergencia del *Big Data* y la apuesta de muchos organismos por los datos abiertos existe una disponibilidad de datos enorme. Además, tanto el gobierno como el resto de organismos están cada vez más interesados en la extracción de conocimiento de estos datos con el fin de mejorar la toma de decisiones y su operativa.

A continuación, se explica en detalle las diferentes fases del proceso y cómo se han aplicado a este proyecto para observar el impacto de soluciones Industria 4.0 en los usuarios.

3.1. Extracción de datos

Para poder llevar a cabo el proceso de extracción de información lo primero que se debe hacer es identificar qué información se encuentra disponible, donde está y como accederla.

Actualmente la información puede proceder de una gran cantidad de lugares (carpetas, redes sociales, Internet, bases de datos...) y puede ser accesible de múltiples maneras. Se puede extraer información de bases de datos mediante consultas SQL o mediante *query browsers* que ofrecen interfaces gráficos para poder realizar las consultas, modificarlas y optimizarlas. Por otro lado para obtener información de ficheros de texto se pueden crear *scripts* en diversos lenguajes de programación que nos vayan mostrando la información que contiene el fichero. De la misma manera se pueden usar servicios web para obtener la información disponible en la red y combinarlos con APIs para obtener información de bases de datos no SQL.

Esta información disponible suele clasificarse en tres grupos: información pasada que suele encontrarse en bases de datos, la información actual disponible mediante *web sockets* y la información en tiempo real disponible mediante tecnologías REST.

En el caso de este proyecto se va a trabajar con la información que generan las herramientas incorporadas en esta solución de Industria 4.0 y que van almacenando en ficheros de log. Para ello se ha implementado un sistema encargado de recoger esta información a medida que se va generando, es decir, se va a tratar con información en tiempo real.

En este caso los subsistemas se encuentran montados sobre Docker (Anexo (A.1)) en diferentes *containers*. Para poder extraer la información que se va generando, en primer lugar se ha detectado cuál es el fichero que interesa analizar. Una vez este se encuentra localizado se ha utilizado syslog-ng (Anexo (A.2)), instalado en cada uno de los diferentes subsistemas y se ha configurado este de manera que a medida que se va generando información este la vaya procesando. Una vez la información llega a syslog-ng este se encarga de pasarla al siguiente nivel, en el que se encuentra Kafka (Anexo (A.3)) que será el encargado de recogerla en distintos *topics* (uno por cada subsistema que se esta tratando de analizar).

Para poder realizar este paso de la información a través de syslog-ng ha sido necesario configurar dicha herramienta. La configuración e instalación de la misma se ha realizado en entornos Linux, ya que es el entorno para el que ofrece soporte. Para ello se ha modificado el fichero syslog-ng.conf (*Fichero* (G.1)) disponible en /etc/syslog-ng y se le ha indicado dónde se encuentra la fuente de datos, es decir, el fichero del cual debe leer la información y cuál es el destino de la misma, es decir, a dónde debe enviar todo lo que llegue a dicho fichero.

Para establecer el canal de comunicación entre syslog-ng y Kafka únicamente ha sido necesario desplegar Kafka en un *container* propio e indicar en el destino de syslog-ng la dirección en la que se encuentra dicho *container* junto con otras características de Kafka como el *topic* en el que queríamos almacenar los datos (Anexo (G)).

3.2. Limpieza y tratamiento de datos

Este proceso es el más importante a la hora de realizar un proceso de *Data Science*, ya que es el que va a permitir realizar mejores o peores análisis en función tanto de los datos que se han seleccionado para almacenar como de la calidad que posean los mismos.

Una de las cosas más importantes cuando se trata de analizar datos es ver qué información hay disponible, qué es lo que esta indica y cómo se puede almacenar para poder sacarle el mayor partido a estos datos. Normalmente los datos que llegan del mundo no son perfectos, sino que para poder manejarlos y operar con ellos es necesario tratarlos y así facilitar su análisis posterior. Los datos pueden llegar duplicados o con valores anómalos(*outlier*), o con valores ausentes, por lo que para garantizar la calidad de análisis posteriores es necesario generar funciones que reduzcan y corrijan el impacto de estos errores.

Para este proyecto, una vez se encuentra la información almacenada en los distintos *topics* de Kafka, mediante un proceso en Python (*Script* (D.2)) se va leyendo cada uno de los mensajes que van llegando al mismo. Este proceso es el encargado de recoger todos los datos que llegan al canal y procesarlos. Para ello, es necesario indicarle dónde se encuentra Kafka, para que una vez conectado con este vaya leyendo uno por uno los mensajes que van llegando, de manera que lee una a una las líneas que se van escribiendo en el fichero de log. Para cada una de estas líneas, el programa se encarga de identificar en base a unos patrones definidos previamente en un fichero auxiliar cada uno de los datos que resultan de interés para el análisis.

Una vez emparejada la línea de log con un patrón de los definidos previamente, estos datos son almacenados en la base de datos de MongoDB (Figura (C.1)). La instalación de MongoDB se ha realizado a través de un *container* de Docker usando una de las imágenes que este ofrece y en ella se ha creado una colección que será en la que guardemos toda la información preprocesada y semiestructurada para que posteriormente sea analizada.

Como se ha dicho previamente, los datos a reconocer en los mensajes que llegan del fichero de log se definen mediante patrones en un fichero auxiliar (D.1) que se le pasa por parámetro al *script* de procesamiento. Esta decisión de usar un fichero auxiliar para definir los patrones se ha tomado debido a los siguientes motivos:

- Generalizar el procesamiento de datos. De esta manera en caso de querer procesar otro fichero de log en el que la información se escriba de manera diferente no será necesario modificar el *script* de procesamiento sino que únicamente habrá que redefinir los patrones en otro fichero y seguirá funcionando de la misma manera. Es por este motivo también que se ha decidido pasar por parámetro la localización tanto de Kafka y Mongo así como el *topic* de Kafka en el que se encuentran los datos.
- Otro aspecto relacionado con la generalización del código es que mediante el mismo *script* se pueden procesar ficheros de log en los que se quiere reconocer un único patrón o un número elevado de ellos. Basta con añadirlos al fichero auxiliar en orden de prioridad (ya que una vez que reconozca uno ya no buscará más) y el *script* se encarga de realizar todo el proceso.
- El *script* es mucho más legible. En este caso, basta con anidar bucles que vayan leyendo patrones del fichero auxiliar y si no quedan más o ya ha reconocido uno que pase a la siguiente línea. Esto es muy beneficioso en caso de querer reconocer muchos tipos de patrones, ya que si se escribieran todos en el *script* quedaría ininteligible y repetitivo.

Para poder definir los patrones que se quiere reconocer es necesario conocer el formato del fichero de log (Anexo (B)), de manera que se conozcan todas las posibles formas en las que pueden llegar los datos y las empareje correctamente. En este fichero auxiliar (Fichero (D.1)) también se define el nombre de los campos de datos que se desea almacenar en la base de datos y que valor tomarán en cada caso, es decir, si será el valor que empareje el patrón o en algún caso puede ir definido en este fichero y será constante.

Es importante saber que al script no hay que pasarle como parámetro únicamente el fichero auxiliar con los patrones a reconocer, sino también dónde se encuentra Kafka (@IP:PUERTO), el *topic* de Kafka del cual se quieren analizar los datos y la dirección en la que se encuentra la base MongoDB en la que se almacenan los datos.

3.3. Análisis e interpretación de datos

Una vez se dispone de los datos limpios es momento de empezar a analizarlos. La dificultad en este punto radica en encontrar ideas de análisis que lleven a obtener resultados relevantes en aquello que se desea.

Para ello es importante observar bien los datos, saber de qué información se dispone y tratar de buscar patrones entre todos esos datos para ver el comportamiento de los mismos.

Una vez se han encontrado posibles patrones y se han observado los datos, es decir, se sabe qué es lo que se desea analizar, es momento de empezar con el análisis. En este punto es donde hay que utilizar los conocimientos estadísticos, matemáticos y tecnológicos que se conocen y utilizar las herramientas de análisis de datos que ayuden a sacar conclusiones. Además de todo ello, es importante conocer el problema que se está analizando ya que de esta manera será más fácil obtener conclusiones correctas y saber qué es lo que se desea buscar.

En el caso de este proyecto en el que se pretende medir la eficiencia de los trabajadores con las herramientas incorporadas en la Industria 4.0, tras estudiar los datos se ha decidido analizar los siguientes aspectos:

- Observar el número de usuarios que hay en un día de trabajo.
- Ver cuáles son las funcionalidades utilizadas por estos, es decir, los métodos invocados desde las diferentes herramientas incorporadas en este proceso de digitalización.
- Observar el número de errores producidos y qué funcionalidades se encuentran

implicadas en ellos.

- El rendimiento del sistema, es decir, el tiempo medio que tarda cada tipo de petición en ofrecer una respuesta.

A la hora de realizar el análisis en este proyecto se ha utilizado otro *script* en Python (Anexo (E)) donde se ha programado todos los análisis descritos anteriormente. Para ello el *script* requiere que se le pase como parámetro donde se encuentra el *container* de MongoDB en el que se han almacenado previamente los datos preprocesados. Posteriormente, este mismo *script* se encargará de ir guardando los resultados obtenidos del análisis en otra colección diferente para que estos puedan ser accedidos posteriormente para ser mostrados gráfica y visualmente a los usuarios.

3.4. Presentación de resultados

Esta parte de presentación de resultados es la menos técnica del proceso pero eso no implica que no sea importante. Su objetivo es soportar la toma de decisiones sobre mejoras en procesos y/o herramientas.

En este TFG se ha decidido utilizar Vaadin (A.7) para realizar una interfaz web que se muestra los resultados de análisis de manera gráfica a los usuarios. El interfaz web donde se muestra una navegación básica consta con cuatro pantallas (F) en las que se puede ver los resultados de los análisis elaborados mediante diferentes gráficos (barras, circulares...) junto con la descripción de lo que muestra cada uno de ellos. Además, de manera adicional se ha añadido información tanto del proyecto como del contexto de realización del mismo.

Capítulo 4

Conclusiones y posibles aplicaciones

Finalmente, en este último capítulo, se comentan las conclusiones obtenidas tras la realización de este trabajo y posibles cuestiones que podrían desarrollarse en un futuro junto con una pequeña explicación de la gestión del proyecto.

4.1. Conclusiones

Aprovechar todo el potencial de la Industria 4.0 requiere que cada empresa planifique cuidadosamente la implantación de sus soluciones. Para ello es necesario identificar los recursos tecnológicos que sean adecuados a todo el personal de la empresa, ya que son estos factores la clave del éxito del proceso de digitalización.

En este TFG se ha medido el grado de adaptación de los trabajadores de Thermolympic a las nuevas herramientas tecnológicas que se han introducido en los procesos. Para ello se ha realizado un proceso de *Data Science* basado en un análisis cuantitativo de los datos procedentes de los ficheros de log que generaban las herramientas tecnológicas introducidas.

Gracias a la realización de este TFG, se han podido aprender nuevas técnicas y herramientas que resultaban desconocidas hasta el momento. La elaboración del mismo ha permitido no sólo aprender cómo se utilizan estas nuevas herramientas y el potencial que tienen, sino que también conocer más de cerca el tratamiento de los datos, llevando a cabo un proceso completo de *Data Science* en el que el objetivo principal era establecer el canal de comunicación entre la fuente de datos, en este caso los ficheros de log, y la unidad de procesamiento, almacenamiento y análisis de los datos.

No sólo se han adquirido conocimientos técnicos durante la realización del TFG, sino que también se ha conocido el concepto de Industria 4.0. Es un hecho que puede apreciarse la incorporación de las tecnologías en las industrias y procesos que se llevan a cabo en la actualidad, pero este TFG ha permitido conocer el verdadero alcance de este proceso de digitalización y la importancia que está adquiriendo. Además, al pertenecer a un proyecto europeo se ha podido ver cómo se trabaja en estos proyectos y la importancia que tiene la comunicación de todas las partes para llevar a cabo el análisis deseado.

En cuanto al proyecto la principal finalidad de este era lograr la extracción de la información de los ficheros para poder almacenarla de manera centralizada y poder basar en ella estudios de calidad y uso de la herramienta. Se puede afirmar por tanto que esta fase se ha completado con éxito, logrando centralizar la información que se va almacenando en los logs en una colección de la base de datos. Otro de los objetivos propuestos tras extraer la información era analizar dichos datos con el fin de obtener unas primeras conclusiones sobre el uso de las herramientas. En este caso se ha optado por análisis de errores producidos por las herramientas, estudiando la procedencia de los mismos y el tipo, al

mismo tiempo que se ha estudiado el rendimiento de la herramienta, viendo aquellos procedimientos que tardan más en ofrecer una respuesta de manera que la empresa ha podido tomar medidas y tratar de simplificarlos y optimizarlos para ganar tiempo en el proceso de producción.

4.2. Limitaciones

La principal limitación del trabajo a la hora de extraer conclusiones ha sido el uso de tokens para mantener el anonimato de los trabajadores, cosa necesaria por ley. Esta necesidad legal de anonimato en los trabajadores, supone una limitación de análisis importante. Una de las cosas interesantes de analizar podía ser correlar las llamadas a servicios realizadas por los trabajadores a través de la tablet. De esta manera, se podía tratar de hallar un árbol de llamadas para ver si hay varios caminos para llegar a una solución y en caso de detectarse caminos muy largos alternativos a otros cortos, los cuales pueden implicar perder rendimiento en la producción tratar de eliminarlos.

4.3. Trabajos futuros

El proyecto de Facts4Workers engloba diferentes entornos, habiendo aplicado la solución a uno de ellos en concreto. Por tanto, dicho TFG sigue teniendo aplicación en los diferentes entornos de producción del proyecto F4W, volviendo a pasar por todas las etapas, ya que no en todos los entornos tienen por qué generarse los mismos ficheros de log. De esta manera, para cada uno de ellos habrá que adaptar mínimamente la extracción de los ficheros de log, el preprocesamiento de los datos, es decir, detectar los patrones que se desea detectar en cada uno de los diferentes entornos y posteriormente la fase de análisis personalizado a cada uno de ellos.

Además, el análisis de logs es una técnica muy empleada en la actualidad, ya que es una manera rápida de obtener grandes volúmenes de datos relativos a la actividad de los usuarios con las diferentes herramientas, donde se recoge mucha información relevante como puede ser los servicios utilizados por los usuarios, las respuestas y errores obtenidos de las mismas.

Todos estos datos aplicados a esta técnica de análisis pueden resultar de interés no sólo en el marco de este proyecto sino en otros muchos posteriores, ya que como se ha dicho previamente la Industria 4.0 está afectando a todos los sectores y el éxito de las empresas reside en adaptar este proceso de digitalización a sus trabajadores.

4.4. Gestión de proyecto

En la siguiente sección se detallan algunos aspectos que se han seguido al elaborar el proyecto. Principalmente se muestra la estimación realizada en la planificación del proyecto y el cronograma final en el que se muestran las fases que se han seguido en la realidad junto con las horas reales destinadas a cada una de ellas. Tras ello, se describe en mayor detalle a qué se han debido los desajustes entre la planificación y la realidad.

4.4.1. Cronograma inicial

En la siguiente figura (Figura (4.1)) se muestran las fases en las que se ha dividido el proyecto inicialmente con las tareas más importantes de cada una de ellas junto con las fechas estimadas de inicio y fin de las mismas.

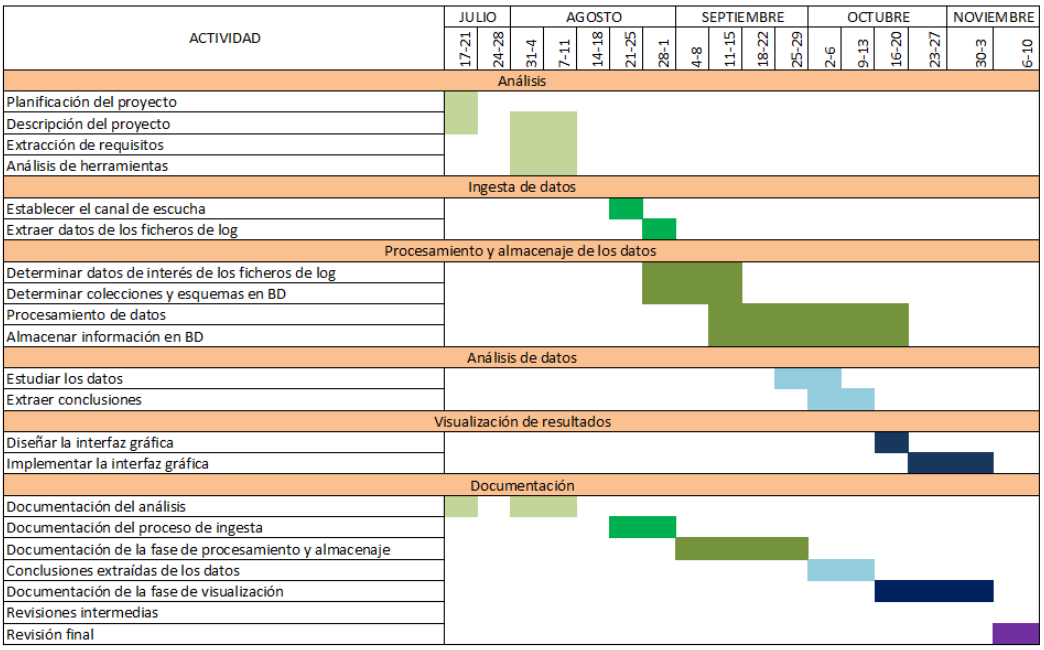

Figura 4.1: Cronograma inicial

Para dicha fase se han estimado aproximadamente 450 horas de trabajo, las cuáles se han contabilizado teniendo en cuenta 5 días de trabajo semanales, 7 horas diarias a excepción de días festivos y vacaciones ya que el proyecto se ha realizado a lo largo del verano.

4.4.2. Cronograma final

A continuación se muestra el cronograma final (Figura (4.2)) en el que se puede observar cuál ha sido la planificación final del proyecto, detallando también el coste aproximado en tiempo de cada una de las tareas.

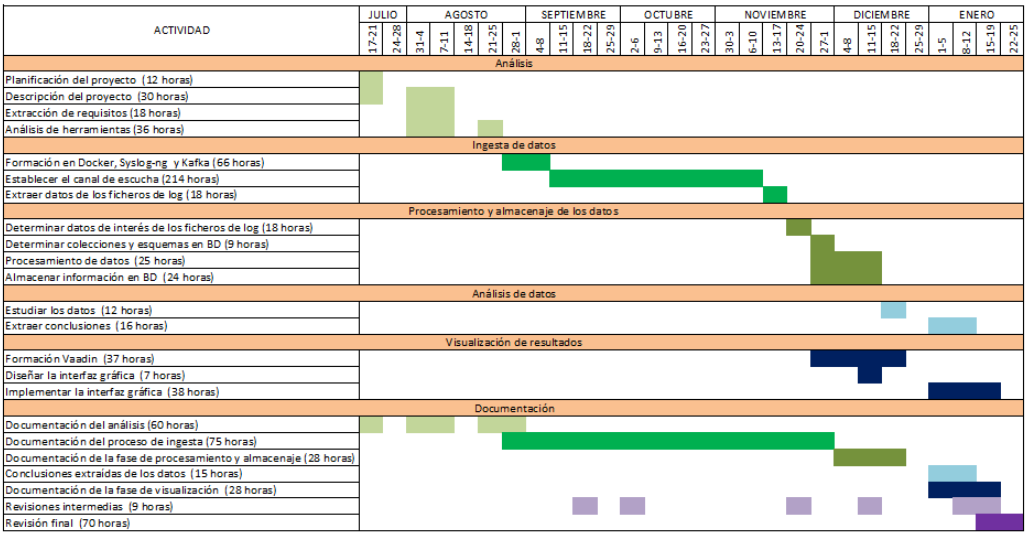

Figura 4.2: Cronograma final

Observando la planificación final vemos que el trabajo ha durado más de lo previsto inicialmente, quedando en unas 665 horas de trabajo real dedicado al proyecto y 105 horas de formación en las diferentes herramientas, lo que hace un total de 770 horas.

4.4.3. Diferencias entre esfuerzos reales del proyecto y los estimados

Observando los cronogramas que se muestran en los apartados anteriores y observando el tiempo dedicado y el estimado se aprecia claramente que ha habido algún desajuste respecto a la planificación inicial del proyecto. En esta sección se explica con mayor detalle a qué se han debido estos desajustes.

Uno de los problemas que no se habían tenido en cuenta era la formación en las diferentes herramientas que se han utilizado en el proyecto, las cuáles han resultado en su mayoría nuevas. Por ello, como se puede ver la mayor parte del desajuste del proyecto se explica en la formación de estas a medida que llegaba el momento de utilizarlas, se realizaba un aprendizaje básico de las mismas para posteriormente aplicarlo al proyecto.

Una vez tenido en cuenta el desajuste de formación en las herramientas citado anteriormente queda un pequeño desajuste todavía debido al proceso de ingesta de los datos. En esta fase, la empresa para la que se ha realizado el proyecto y que facilitaba los datos de las herramientas que habían añadido para digitalizar sus procesos, inicialmente pasó un formato de fichero del cuál era difícil obtener información. Por ello, fue necesario realizar modificaciones en la forma en la que los datos eran guardados y convertirlos a un formato más legible (Figura (B)) que fue el utilizado para extraer la información.

4.4.4. Metodología

El primer paso en la realización del TFG fue definir el proyecto que se iba a realizar completando la propuesta de este y estableciendo tanto los requisitos que debía cumplir el proyecto como los objetivos que se pretendía alcanzar con él.

Una vez definido el proyecto, el siguiente paso fue definir la arquitectura que se iba a seguir para llevarlo a cabo. En este punto, se definió el entorno de integración (*containers* de Docker), las herramientas propias que se iban a utilizar para centralizar los datos y recoger la información procedente de los ficheros de log.

Una vez definida la base del proyecto, se siguió la metodología CRIPS - DM [19] para la implementación del mismo. Esta metodología se divide en 10 etapas (Figura (4.3)) que representan un proceso iterativo que va desde el planteamiento del problema al desarrollo y mantenimiento del mismo.

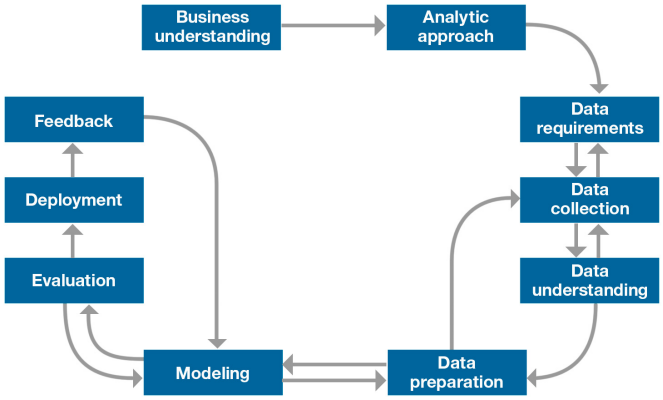

Figura 4.3: Fases metodología CRIPS- DM. Extraída de [20]

Durante todo el proyecto se hizo uso de herramientas colaborativas como Trello (Anexo (A.6)) para gestionar el proyecto y conocer de manera rápida el estado del mismo. Además de las reuniones y revisiones presenciales, se utilizó Skype para la realización de algunas reuniones con los directores.

ANEXOS

Apéndice A

Herramientas utilizadas

A continuación se describe con mayor detalle las herramientas que se han utilizado en la realización del proyecto, detallando al final de este anexo la versión utilizada en cada caso y si ha sido necesario instalar alguna dependencia para su funcionamiento.

A.1. Docker

La idea de Docker [21] es usar *containers* para que las aplicaciones software puedan ejecutarse en cualquier máquina que posea Docker instalado, independientemente del sistema operativo que posea la máquina y de las dependencias que esta necesite, facilitando así los despliegues.

Dichos *containers* son ligeros y portables. Muchas aplicaciones necesitan tener ciertos aspectos instalados a la hora de poder ser ejecutadas (versiones específicas de algún software, servidores de aplicaciones, librerías...) por lo que Docker mete todo esto en un *container* y ejecuta la aplicación desde este *container*, de manera que así se asegura que la aplicación se comportará de igual manera en todos los ordenadores en los que se encuentre.

Aunque se puede llegar a pensar que un container es lo mismo que una máquina virtual, existen diferencias entre ellos (Figura (A.1)). Docker es más ligero que una máquina virtual, además no es necesario instalar dentro de los containers el sistema operativo, ya que utilizan aquel que posee la máquina que está alojando Docker.

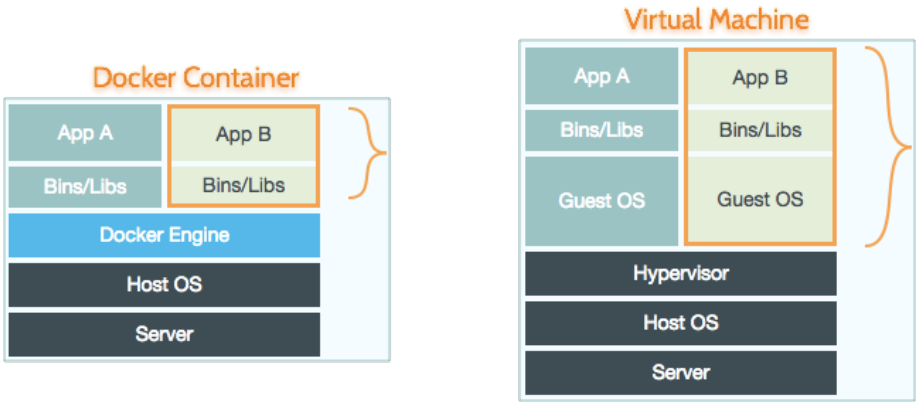

Figura A.1: Containers vs máquinas virtuales. Extraída de [22]

A.2. Syslog-ng

Syslog-ng [15] es una aplicación flexible y altamente escalable muy recomendable a la hora de centralizar los logs. Syslog-ng permite enviar mensajes de log de un host a servidores remotos, de manera que toda la información de los logs se almacena de manera centralizada en servidores dedicados. A la hora de transferir los mensajes se utiliza el protocolo TCP, que permite asegurar que no se pierden mensajes por el camino.

Uno de los aspectos destacables de syslog-ng es que además de actuar como canal entre los distintos servidores también permite filtrar los mensajes que pasan a través de él. Muchos de los mensajes que se envían no se encuentran adecuadamente estructurados de manera que dificultan su procesamiento. Es por eso que Syslog-ng permite filtrar y clasificar dichos mensajes, además de parsearlos y reescribirlos al formato deseado.

Syslog-ng permite enviar los mensajes que le llegan a multitud de destinos diferentes, bases de datos SQL, NoSQL, ficheros de texto o sistemas de monitorización.

Syslog-ng no es una aplicación de análisis. Syslog-ng permite filtrar mensajes y seleccionar únicamente aquellos que cumplen ciertos criterios, permite convertirlos y reestructurarlos o incluso dividirlos por campos, pero a pesar de ello no permite interpretar ni analizar el significado que estos poseen ni reconocer patrones de ocurrencia entre todos los mensajes que le llegan.

Debido a que en muchos sistemas mucha de la información que se almacenaba en los logs no era de importancia y dificultaba la monitorización de los eventos surgió Syslog-ng para poder filtrar los mensajes y separar aquellos que poseen interés de los que no lo tienen.

A.3. Kafka

Kafka [12] es una plataforma de transmisión distribuida, lo que significa lo siguiente:

- Permite publicar y suscribirse a transmisiones de datos (similar a una cola de mensajes).
- Permite almacenar transmisiones de datos ofreciendo tolerancia a fallos.
- Permiten procesar estas transmisiones de datos a medida que estas van ocurriendo, es decir, en tiempo real.

Kafka funciona como un cluster con uno o más servidores en el que se almacenan los registros de datos en categorías, conocidas como *topic*. En Kafka cada registro que se almacena consiste en una <clave, valor>junto con el timestamp correspondiente.

Un topic, también conocido como tema, es una categoría en la que se publican los registros de datos. Los topics en Kafka pueden no tener ningún consumidor o varios de ellos interesados en sus datos. Para cada topic el cluster de Kafka mantiene un sistema de particionado, siendo este un punto importante para el paralelismo que ofrece Kafka.

Cada una de estas particiones se distribuye entre los distintos servidores del cluster y se replica en varios servidores para ofrecer tolerancia a fallos.

Dos elementos importantes en Kafka, además del clúster de servidores, son los productores y los consumidores. Los productores son los encargados de publicar información en los topics de Kafka, mientras que los consumidores son los encargados de suscribirse a los topics que deseen para poder consultar su información.

Una de las utilidades de Kafka que también ha resultado de interés es la utilización de Kafka como sistema de almacenamiento. Todos los datos que se envían a Kafka se escriben en disco y se replican para soportar tolerancia a fallos. Kafka permite bloquear a los productores hasta que estos reciban la confirmación de que se ha replicado todo de manera correcta y que se ofrecen garantías de persistencia incluso si el servidor se cae.

A.4. MongoDB

MongoDB [23] es la base de datos NoSQL orientada a documentos líder. Se trata de una base de datos que permite a los esquemas cambiar rápidamente cuando las aplicaciones evolucionan, proporcionando además las funcionalidades de las bases de datos tradicionales (índices secundarios, lenguajes de búsquedas y consistencia estricta).

MongoDB en lugar de guardar sus datos en registros los guarda en documentos. Estos documentos los almacena en BSON, representación binaria de JSON.

La principal diferencia de MongoDB con las bases de datos relacionales es que no es necesario seguir un esquema, ya que los documentos de una misma colección (concepto equivalente a las tablas en bases de datos relacionales) pueden tener diferentes esquemas.

Aunque las colecciones de MongoDB no necesitan definir un esquema, es importante que diseñar uno en el proyecto que se está realizando.

A continuación se detalla de forma resumida las principales características que posee MongoDB:

- **Consultas Adhoc**: soporta búsqueda por campos, rangos y expresiones regulares.
- **Indexación**: al igual que se pueden hacer índices secundarios, cualquier documento puede ser indexado.
- **Replicación**: el formato de replicación que soporta MongoDB es el de replicación primario-copia.
- **Balanceo de carga**: MongoDB permite escalamiento horizontal usando el concepto de "sharding".
- **Almacenamiento de archivos**: aprovechando la capacidad de MongoDB para el balanceo de carga y la replicación de datos utilizando múltiples servidores, este puede ser utilizado también como sistema de archivos.
- **Agregación**: a través del framework de agregación que ofrece MongoDB pueden realizarse operaciones similares a las del Group By.
- **Ejecución de JavaScript del lado del servidor**: tiene la capacidad de hacer consultas mediante JavaScript que son enviadas directamente a la base de datos para que sean ejecutadas.

MongoDB se creó para ofrecer escalabilidad, rendimiento y gran disponibilidad escalando de una implantación de servidor único a grandes arquitecturas complejas de centros multidatos. De esta manera MongoDB logra ofrecer un elevado rendimiento tanto para lectura como para escritura, al mismo tiempo que su replicación nativa y la tolerancia a fallos automática ofrecen tanto fiabilidad como flexibilidad a la hora de operar.

A.5. LaTeX

Para la presentación de la documentación final se ha decidido utilizar LaTeX [24].

LaTeX es un sistema de preparación de documentos con el que se puede preparar documentos de manera muy cómoda (artículos, tesis, manuscritos, etc.), ofreciendo en el resultado final una calidad más profesional que la que se puede lograr con otras herramientas más conocidas como Word.

Como se ha dicho, LaTeX facilita la realización de tareas que en otros entornos pueden resultar tediosas. LaTeX se encarga de numerar páginas, capítulos, figuras y tablas de manera automática, además de organizar la bibliografía de manera adecuada y mantener índices y referencias cruzadas.

LaTeX se encuentra disponible para cualquier sistema operativo y para su manejo hace uso de tres componentes:

- Editor de texto: existen diversas aplicaciones interactivas que nos permiten escribir documentos en .tex. Muchos de ellos son especializados y ofrecen un rápido acceso a comandos usados habitualmente por los usuarios para que el proceso sea menos tedioso y no requiera tanto manejo de la herramienta para poder usarla. El editor de texto utilizado en este proyecto ha sido Texmaker, el cual es muy cómodo porque posee numerosos atajos para elaborar el documento y además permite compilar el fichero de manera cómoda y rápida y ver una vista previa de cómo queda el fichero portable.
- Distribución de LaTeX: esta distribución es el motor que se encarga de convertir los archivos fuente de LaTeX (.tex) en documentos portables (.pdf). En este caso se ha usado MiKTeX.
- Visor de documentos: aplicación que permita ver los documentos generados por LaTeX. En este caso se ha usado Adobe Acrobat Reader.

A.6. Trello

Durante el desarrollo del proyecto se ha utilizado Trello [25] para gestionar las tareas.

Trello es una herramienta colaborativa que permite organizar proyectos mediante el uso de tablones que nos permiten ver el estado global del proyecto de un solo vistazo. Trello se basa en la metodología Kanban [26].

A la hora de usar Trello se ha hecho de la forma más común, es decir, se ha creado un tablero que representase el proyecto y en él se han definido cuatro listas principales (tareas a realizar, tareas en proceso, terminadas y desechadas) de manera que se pudiera ver en qué estado se encontraban las diferentes tareas de un solo vistazo y saber en todo momento qué se había hecho y que quedaba por hacer (Figura (A.2)).

Además, el uso de esta herramienta ha permitido facilitar la comunicación con los directores del proyecto, ya que al permitir añadir miembros al tablero todos conocían el estado del proyecto en cada momento.

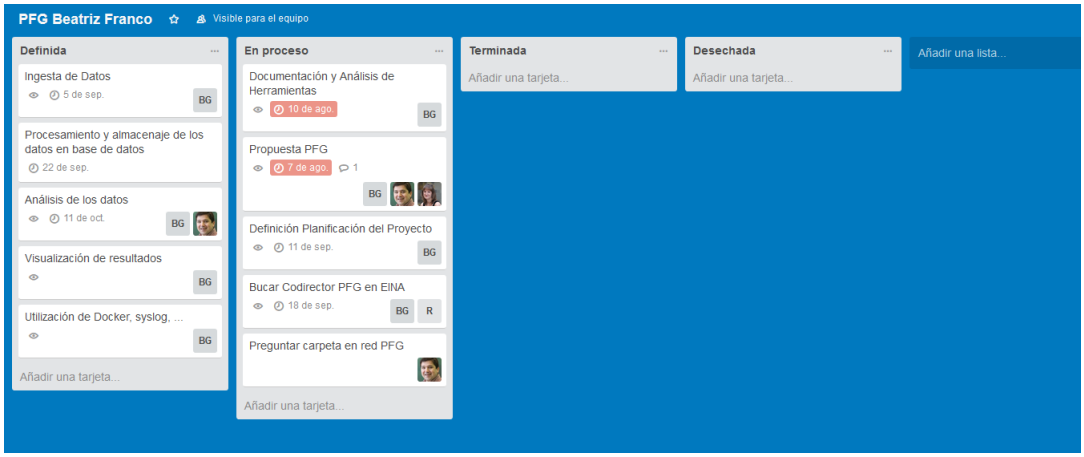

Figura A.2: Ejemplo de tablero de Trello

A.7. Vaadin

Vaadin [27] es una plataforma de desarrollo que ofrece las herramientas necesarias para el desarrollo de forma sencilla de aplicaciones web.

Vaadin ofrece componentes y herramientas profesionales que hacen más simple la vida de los desarrolladores. Gracias a ello permite asegurar un desarrollo rápido, de mayor calidad y con una mejor experiencia de uso para los usuarios.

Entre las herramientas que ofrece Vaadin encontramos las siguientes:

- **Diseño Drag and Drop**: su integración con Eclipse e IntelliJ permite a los usuarios diseñar interfaces de forma visual mediante drag and drop.
- **Bancos de pruebas**: Vaadin ofrece bancos de prueba que permiten asegurar la calidad de las aplicaciones mediante tests automatizados.
- **Gráficos**: haciendo uso de la librería interactiva permite visualizar los datos mediante gráficos para que sea más visual.
- **Boards**: el uso de layouts flexibles ayuda a crear dashboards de manera mucho más sencilla.

A.8. Eclipse

Eclipse [28] se ha utilizado como IDE de desarrollo a la hora de implementar la interfaz web. En concreto, se ha utilizado Eclipse Oxygen, al que se le han añadido los plugins necesarios para poder desarrollar con Vaadin.

Utilizando los plugins de Vaadin en Eclipse hace que el diseño de las interfaces sea mucho más cómodo y rápido. Esto es debido a que permite crear diseños mediante la selección y arrastre de componentes, los cuáles basta con configurarlos de la manera deseada en la pestaña de propiedades. Esto autogenera las clases de diseño necesarias con el código necesario para mostrar por pantalla.

A pesar de esto, hay clases cuya configuración es más compleja, por ejemplo, al insertar gráficos, ya que hay que realizarlo manualmente.

A.9. Python

Python [29] es un lenguaje de programación interpretado que busca crear una sintaxis que favorezca el código legible.

Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, programación funcional. Es un lenguaje interpretado que usa tipado dinámico y es multiplataforma.

Una característica importante de Python es la resolución dinámica de nombre, es decir, enlaza un método y un nombre de variable durante la ejecución del programa.

A la hora de programar Python, se ha realizado a través de línea de comandos y ficheros sin utilizar ningún IDE de programación para ello. A pesar de esto, existen IDEs disponibles para facilitar la programación de Python como son: Spyder, PyCharm o Atom entre otros.

A.10. Versión empleada en cada herramienta

A continuación, en la siguiente tabla (Tabla (A.1)), se muestra un resumen de la versión empleada en cada una de las herramientas utilizadas a lo largo de todo el proyecto.

HERRAMIENTA	VERSION	DEPENDENCIAS
Docker	17.05	docker-compose 1.18.0
Syslog-ng	3.13	nose=1.3.6, ply=3.6, pep8=1.6.2, pylint=1.5.0, astroid=1.4.0, logilab-common<=0.63.0
Kafka	2.12-0.0.2.1	Zookeeper = 3.4.10
MongoDB	3.4.10	
Trello	En la nube	
Vaadin	Vaadin 8	JDK8
Eclipse	Oxygen 4.7.0	Java
Python	Python3	
Skype	12.13.271.0	

Tabla A.1: Versiones de herramientas

Apéndice B

Ejemplo de fichero de log

La metodología que seguían en la empresa donde se realiza el estudio a la hora de guardar los ficheros de log era utilizar Kong [30] , que define una forma sencilla de almacenar la información de log.

De esta manera, los mensajes que llegaban tenían el formato que se muestra en el siguiente ejemplo:


```
{
   "started_at": 1508334933089,
      "response": {
         "status": 200,
         "size":"1080",
         "headers": {
            "server": "WEBrick/1.3.1 (Ruby/2.2.5/2016-04-26)",
            "access-control-allow-credentials": "false",
            "x-prev-page": "",
            "x-kong-proxy-latency": "112",
            "x-page": "1",
            "x-runtime": "0.559404",
            "x-total": "2",
            "x-request-id": "51b5f076-3814-4a46-afd7-7fea5db1f5ca",
            "access-control-expose-headers": "X-Next-Page,X-Offset,X-Page,X
                -Per-Page,X-Prev-Page,",
            "content-type": "application/json",
            "connection": "close",
            "content-length": "363",
            "x-total-pages": "1",
            "x-offset": "0",
            "x-next-page": "",
            "via": "kong/0.10.1",
            "access-control-allow-origin": "*",
            "x-kong-upstream-latency": "570",
            "vary": "Origin",
            "cache-control": "max-age=0,private,must-revalidate",
            "x-per-page": "10",
            "etag": "W/\"b03aaf591c87c28dda9b7eb015301128\""
         }
      },
      "request": {
         "method": "GET",
         "uri": "/tho/defsol/api/defects?access_token=
             a59263ca7d83eda4b12cc4d638a547b5b16b604417",
         "size": "230",
         "request_uri": "http://localhost:8000/tho/defsol/api/defects?
             access_token=a59263ca7d83",
         "querystring": {
            "access_token":"a59263ca7d83eda4b12cc4d638a547b5b16b604417"
         },
         "headers": {
            "host": "localhost",
            "connection": "keep-alive",
            "accept": "*/*",
```

```
            "accept-encoding": "gzip,deflate",
            "user-agent": "HTTPie/0.9.9"
         }
      },
      "client_ip": "172.19.0.1",
      "api": {
         "uris": [
            "tho/def/sol"
         ],
         "id": "aebdaf43-c78e-4aaf-8747-bc9e60b7e9f4",
         "upstream_read_timeout": 60000,
         "preserve_host": false,
         "created_at": 1504702657000,
         "upstream_connect_timeout": 60000,
         "upstream_url": "http://defsol:3000",
         "strip_uri": true,
         "https_only": false,
         "name": "defsol",
         "http_if_terminated": true,
         "upstream_send_timeout": 60000,
         "retries": 5
      },
      "latencies": {
         "request": 682,
         "kong": 112,
         "proxy": 570
      }
   }
}
```


De toda esta información que se almacena, la que interesa para realizar el análisis y la que se ha obtenido mediante patrones es la siguiente:

- ***started_at***: que hace referencia al UNIX timestamp del momento en el que se inició la petición.
- ***response.status***: posee el código HTTP que indica si la petición ha sido éxito o fracaso e incluso se puede obtener el tipo de error detallado.
- ***request.method***: indica si el usuario lee, escribe o borra del servicio.
- ***request.uri***: puede parsearse para ver a qué recurso se está accediendo.
- ***request.querystring.access_token***: hace referencia una sesión de usuario.
- ***latencies.{request,kong,proxy}***: indica lo que tarda el proceso en contestar según varios subsistemas.

Apéndice C

Datos almacenados

A la hora de almacenar los datos, estos se han almacenado en tres puntos del proceso. Por un lado se ha utilizado el sistema de colas de Kafka, de manera que en ese punto se almacenaban los datos en crudo, es decir, los mensajes completos tal y como llegaban de syslog-ng y como se almacenaban en el fichero de log. Seguidamente estos datos eran leídos por el programa en Python que se encargaba del preprocesamiento y una vez extraía aquellos datos que resultaban de interés los almacenaba en una colección de MongoDB dedicada a los datos de entrada.

A pesar de que MongoDB es una base de datos sin esquema en este caso, como todos los mensajes siguen el mismo patrón, se ha almacenado la misma información, por lo que la estructura que se podría obtener en dicha colección de datos de entrada es la siguiente:

Figura C.1: Datos de entrada preprocesados

```
{
   "_id" : ObjectID,
   "hora" : String
   "status" : Integer
   "UnixTimeStamp": Timestamp
   "funcionality" : String
   "access_token": String
   "ano": Integer
   "dia": Integer
   "latencies": Array
   "mes": String
   "date": String
   "method": String
}
```

Una vez se tenían los datos de entrada almacenados, estos se analizaban y posteriormente los resultados se guardaban en otra colección con toda la información. En este caso únicamente se ha creado una colección para todos los resultados del análisis por lo que aquí sí que se está aprovechando la capacidad que ofrece MongoDB para almacenar en una misma colección documentos que no sigan el mismo esquema.

A pesar de que todos los documentos almacenados no siguen el mismo esquema, a continuación se muestra un pequeño esquema de lo que se almacena en la colección de resultados para cada uno de los análisis realizados, campos que se encontraran completos o vacíos en función de los requerimientos del análisis que se esté realizando.

Figura C.2: Resultados análisis

```
{
   "_id" : ObjectID,
   "funcionalidades" : Array
   "numAnalisis" : Integer
   "UnixTimeStamp": Timestamp
   "repeticiones" : Integer
   "codeError": String
   "porcentaje": String
   "funcionalidad": String
   "rendimiento": String
   "operacion": String
}
```

Para ver qué se analiza en cada caso y qué elementos se cumplimentan en cada caso se puede ver el Anexo (E).

Apéndice D

Procesamiento de los datos

Previamente a que los datos de entrada fueran almacenados en la colección correspondiente en MongoDB, estos datos han sido preprocesados para facilitar la labor de análisis que se llevará a cabo más adelante.

El encargado de preprocesar estos datos y almacenarlos en la base de datos es un script en Python (Figura (D.2)). Este script es el encargado de leer uno a uno los mensajes que van llegando al *topic* de Kafka y de procesarlos, eliminando la información redundante que no se va a analizar y quedándose únicamente con aquellos datos que van a resultar de interés para medir el rendimiento de los usuarios.

Para ello, al script se le pasan como parámetros los siguientes datos:

- **KAFKA_IP:PUERTO**: la dirección y puertos en los que se encuentra el container de Kafka donde se almacenan los datos procedentes del fichero de log.
- **KAFKA_TOPIC**: el *topic* de Kafka en el que se encuentran almacenados los mensajes que han sido leídos del fichero.
- **MONGO_IP:PUERTO**: la dirección y el puerto en el que se encuentra el container de Mongo. En él se creará una colección de datos de entrada que se encargará de guardar estos datos que llegan desde el fichero de log y ya han sido preprocesados.
- **FICHERO**: localización del fichero en el que se encuentran los patrones que se deben reconocer para extraer los datos que resultan de interés.

Para poder realizar este procesamiento de datos, inicialmente el script inicia una conexión con el canal de Kafka que se ha pasado por parámetro y se subscribe al *topic* indicado, de manera que cada vez que un nuevo mensaje llegue a dicho *topic* este podrá ser leído por el script.

Una vez se ha subscrito a Kafka, el script se encarga de establecer la conexión con la base de datos de MongoDB que será utilizada posteriormente para ir almacenando la información que se obtenga del procesado de los datos.

Tras ello abre el fichero en el que se encuentran los patrones que se deben reconocer y almacena el contenido en una variable.

En este momento ya se está en condiciones de leer los mensajes que llegan al canal y procesarlos para obtener los datos necesarios, iniciando para ello un bucle infinito que va leyendo cada uno de estos mensajes.

Para cada mensaje debe buscar un patrón que se adapte a la estructura del mensaje y extraer de este los datos adecuados. Por ello inicialmente comprueba si empareja el primer patrón. En caso de que empareje va creando un array de claves-valor en el que asigna como clave lo que se indica en el fichero de patrones y el valor que ha emparejado en dicha posición.

Un hecho importante a destacar es que el usuario puede decidir añadir un campo con contenido preestablecido, lo cual debe indicarlo también en el fichero de patrones. Esto lo hace de la siguiente manera: asumiendo que la estructura del fichero de patrones es la que se muestra en la figura D.1, si en el array de *elements*, en la posición de dato a añadir viene un 0 se asume que debe almacenarse lo que se ha emparejado con el patrón, mientras que si por el contrario viene otra cosa será esto otro directamente lo que haya que almacenar.

Figura D.1: Fichero de patrones

```
{
     "patron": [array de patrones],
     "matching": [array de arrays de lo que se desea almacenar en la base
         de datos],
     "elements": [array de arrays que tiene 0 si almacena lo que empareja
         en el patron o un  valor en caso de que sea algo predefinido]
}
```

El proceso finaliza cuando no quedan más patrones por reconocer o cuando ya ha sido emparejado con uno de ellos. Por ello, es importante el orden en el que se introducen los patrones en el fichero, ya que si se pone lo general primero ya no llegará a leer el patrón específico sino que se habrá quedado con la información del inicio y puede que esté perdiendo información importante para la realización del análisis de los datos.

A continuación se muestra el código del *script* encargado de realizar todo este proceso:

Figura D.2: Fichero de patrones

```
#/usr/bin/env python
# -*- coding:utf-8 -*-

#dependencies
import pymongo
import re
from pymongo import MongoClient
from kafka import KafkaConsumer
import json
import sys
from termcolor import colored
import time

#ensure number parameters
if len(sys.argv)!=5:
        print(colored('El numero de parametros no coincide $PROGRAMA
            $KAFKA_IP $KAFKA_TOPIC $MONGO_IP $FICHERO_JSON','red'))
        exit(1)

print(colored(sys.argv[1],'green'))
print(colored(sys.argv[2],'green'))
print(colored(sys.argv[3],'green'))
print(colored(sys.argv[4],'green'))


#Kafka connection
consumer = KafkaConsumer(bootstrap_servers=sys.argv[1])
consumer.subscribe([sys.argv[2]])

#MongoDB connection
client=MongoClient('mongodb://'+sys.argv[3])
db=client.tfg.pruebas


#JSON file open
with open (sys.argv[4]) as json_data:
  d=json.load(json_data)
```

```
#reading messages and processing
while True:
  for message in consumer:
    k=0
    match=False
    while k < len(d['patron']) and match==False:
      #Proccess message, clean and gather
      elements = []
      m=re.match(r'%s' %d['patron'][k] ,message.value, re.I | re.M)

      if m:
        #Gather information in tfg.format collection
        match=True
        i=0
        j=1
        #info = d['matching']
        while i < len(d['matching'][k]):
          if d['elements'][k][i]=="0":
            if d['matching'][k][i]=='status':
              elem={d['matching'][k][i]:int(m.group(j))}
            else:
              if d['matching'][k][i]=='UnixTimeStamp':
                fecha = time.ctime(int(m.group(j))/1000)
                separate =  fecha.split()
                dia = separate[2]
                mes = separate[1]
                ano = separate[4]
                hora = separate[3]
                date = dia + mes + ano
                elem={d['matching'][k][i]:m.group(j), "dia":dia , "mes":mes
                    ,"ano":ano,"hora":hora,"date":date}
              else:
                elem= {d['matching'][k][i]:m.group(j)}
            elements = elem.items() + elements
            j=j+1
          else:
            if d['matching'][k][i]=='status':
              elem={d['matching'][k][i]:int(m.group(j))}
            else:
              if d['matching'][k][i]=='UnixTimeStamp':
                fecha = time.ctime(int(m.group(j))/1000)
                separate =  fecha.split()
                dia = separate[2]
                mes = separate[1]
                ano = separate[4]
                hora = separate[3]
                date = dia + mes + ano
                elem={d['matching'][k][i]:m.group(j), "dia":dia , "mes":mes
                    ,"ano":ano,"hora":hora,"date":date}
              else:
                elem= {d['matching'][k][i]:d['elements'][1][i]}
            elements = elem.items() + elements
          i=i+1
        print(elements)
        db.insert(dict(elements))
      k=k+1
```

Apéndice E

Análisis de los datos

El análisis de datos es la ciencia que examina datos en bruto con el fin de sacar conclusiones sobre la información. Se trata de una técnica que se encuentra en auge debido a la gran cantidad de datos existente hoy en día y es usado por varias industrias para que las organizaciones tomen mejores decisiones empresariales y en las ciencias para verificar modelos o teorías existentes.

Entre los análisis existentes en función del tipo de dato que se está tratando existen dos tipos:

- **Análisis cualitativo**: es aquel en el que se recolectan datos sin información numérica con el fin de construir teorías, describir sucesos, hechos o patrones y explicarlos para que ayuden a la toma de decisiones.
- **Análisis cuantitativo**: este tipo de análisis pretende probar hipótesis y teorías o establecer patrones de comportamiento a partir de datos del mismo tipo, basándose en la medición numérica y en el análisis estadístico.

En este caso se ha realizado un análisis cuantitativo, en el que se pretende buscar patrones de comportamiento en los datos con el fin de guiar a la empresa en el proceso de digitalización teniendo en cuenta la adaptación de los trabajadores a las herramientas de Industria 4.0.

De manera previa al análisis es importante haber estudiado los datos disponibles para ver cómo se pueden tratar para llegar al objetivo.

En este caso tras haber observado los datos disponibles se ha decidido realizar los siguientes análisis de los datos:

Teniendo en cuenta que uno de los factores importantes es medir el grado de adaptación de los trabajadores a las herramientas que se han introducido en los procesos industriales de la empresa para poder guiar el proceso de digitalización, el primer objetivo, es estudiar los errores que se producen con el uso de las herramientas. Para ello se va a tratar de buscar patrones en base a lo siguiente:

- Clasificar el porcentaje de error por tipo en función del código HTTP de error que devuelva la respuesta.
- Clasificar el porcentaje de error por la invocación que lo provoca con el fin de detectar los puntos problemáticos en los que se producen para tratar de corregirlo y reducir el número de errores en ese punto.

Estos análisis se han considerado de interés ya que permiten detectar qué puntos son confusos para los trabajadores y de esa manera buscar una solución para minimizar esos errores.

Otro de los aspectos que se consideran de interés es saber qué funcionalidades de las que ofrecen las herramientas son más utilizadas por los trabajadores. Esto puede ser de interés ya que si una funcionalidad es muy utilizada se puede tratar de buscar formas de invocarla de manera más rápida, por ejemplo, añadiendo algún acceso rápido a la misma para mejorar el rendimiento de los trabajadores.

Finalmente, se busca usar los datos extraídos de los ficheros de log para medir el rendimiento que posee la herramienta a la hora de trabajar. Para ello se han medido los tiempos de respuesta en función de la funcionalidad que invoque el trabajador a través de la herramienta. Estos datos, correlados con los de las funcionalidades que son más invocadas por los trabajadores pueden llevar a ver si es necesario mejorar algún proceso para que sea más rápido en el caso de que sea muy utilizado y su tiempo de respuesta sea elevado, de forma que lleve en un aumento del rendimiento de la empresa.

Para la realización de todos estos análisis que se han descrito anteriormente se ha realizado un *script* en Python, que se encarga de recoger los datos de la colección de datos de entrada de MongoDB y de realizar sobre ellos cálculos, operaciones y procesamientos con el fin de medir los estudios citados anteriormente.

La implementación de dicho script puede verse en la Figura (E.1)

Figura E.1: Script análisis de datos

```
#/usr/bin/env python
# -*- coding:utf-8 -*-

#dependencies
from __future__ import division
import pymongo
from array import array
from pymongo import MongoClient
import json
import sys
import bson
from termcolor import colored
import time
import collections

#ensure number parameters
if len(sys.argv)!=2:
    print(colored(El numero de parametros no coincide $PROGRAMA $MONGO_IP,
        red))
    exit(1)

#MongoDB connection
client = MongoClient(mongodb://+sys.argv[1])
# Raw data collection
db=client.tfg.pruebas
# Result collection
result = client.tfg.resultados

#Errors by HTTP code error
errores = db.count({"status":{"$gt":399}})
tiposError = db.distinct("status",{"status":{"$gt":399}})

i=0
repeticiones = [0]*len(tiposError)
while i< len(tiposError):
    repeticiones[i]=db.count({"status":tiposError[i]})
    result.insert({"numAnalisis":1,"codeError":tiposError[i],"porcentaje":
        str(round(repeticiones[i]/errores*100,2)),"repeticiones":str(
        repeticiones[i])})
    i=i+1
```

```
#Errors by funcionality
funcionalidades = db.distinct("funcionality",{"status":{"$gt":399}})

i=0
repeticiones = [0]*len(funcionalidades)
while i< len(funcionalidades):
    repeticiones[i]=db.count({"funcionality":funcionalidades[i],"status":{"
        $gt":399}})
    result.insert({"numAnalisis":2,"funcionalidad":str(funcionalidades[i],"
        porcentaje":str(round(repeticiones[i]/errores*100,2)),"repeticiones"
        :str(repeticiones[i])})
    i=i+1

# Funcioalities
total = db.count()
funcionalidades = db.distinct("funcionality")

i=0
repeticiones = [0]*len(funcionalidades)
while i< len(funcionalidades):
    repeticiones[i] = db.count({"funcionality":funcionalidades[i]})
    result.insert({"numAnalisis":3,"funcionalidad":funcionalidades[i],"
        porcentaje":str(round(repeticiones[i]/total*100,2)),"repeticiones":
        str(repeticiones[i])})
    i=i+1

#Time request by funcionality
i=0
repeticiones = [0]*len(funcionalidades)
while i<len(funcionalidades):
    repeticiones[i]=db.count({"funcionality":funcionalidades[i]})
    tiempo = db.find({"funcionality":funcionalidades[i]},{"latencies":1})
    time = 0
    for record in tiempo:
        words = record ['latencies'].split(",")
        request = words[0].split(":")
        time = time + int(request[1])
    result.insert({"numAnalisis":4,"funcionality":funcionalidades[i],"
        rendimiento":str(round(time/repeticiones[i],2))})
    i=i+1
```

Apéndice F

Pantallas de la aplicación y mapa de navegación

En este anexo se muestra de manera detallada la interfaz web diseñada para mostrar resultados. Para ello se ha realizado una pequeña aplicación web, la cual se describe con mayor detalle en las siguientes secciones.

F.1. Mapa de la aplicación

A continuación se muestra de manera gráfica cuál sería la navegación básica que puede realizarse en la aplicación para ver los diferentes aspectos que esta ofrece (Figura (F.1)).

Figura F.1: Mapa de navegación

El esquema anterior representa que la aplicación dispone de 4 interfaces las cuales permiten la navegación entre todas ellas en todo momento. Para realizar esto se ha dividido el interfaz web en dos zonas de manera horizontal (Figura (F.2)). A la izquierda se encuentra un menú que es el que permite la navegación entre las diferentes pantallas, y la zona derecha es la zona dinámica, es decir, es la que va cambiando en función de la pantalla que se esté visualizando.

Figura F.2: Descripción zonas interfaz

F.2. Pantallas de la aplicación

En esta sección se describe la información que aparece en cada una de las secciones del menú a las que se puede acceder desde la interfaz web.

Proyecto

Esta pantalla muestra una pequeña imagen del conjunto del proyecto (Figura (F.3)). En él se describe qué es lo que se pretende con el proyecto, los pasos que se han realizado y algunas de las tecnologías que se han empleado en el mismo.

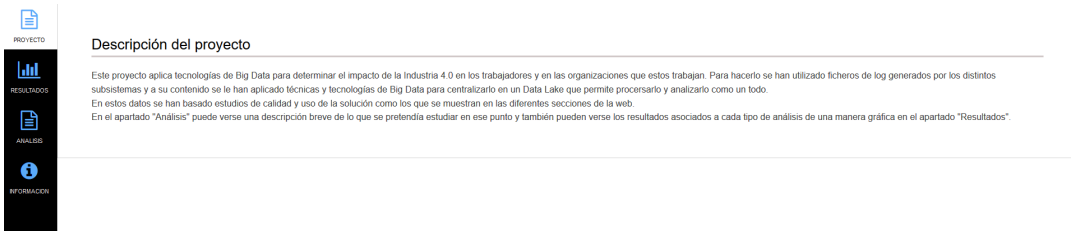

Figura F.3: Pantalla descripción proyecto

Resultados

En esta pantalla se muestran los resultados de los diferentes análisis que se han realizado (Figura (F.4)). Para cada uno de los análisis realizados se muestra un gráfico en el que se muestran de manera gráfica los resultados obtenidos y a su lado una tabla que actúa como leyenda del propio gráfico. A pesar de ello en el uso de la interfaz dichos gráficos son interactivos y al pasar por encima el cursor aparece un mensaje con la información de la sección indicada.

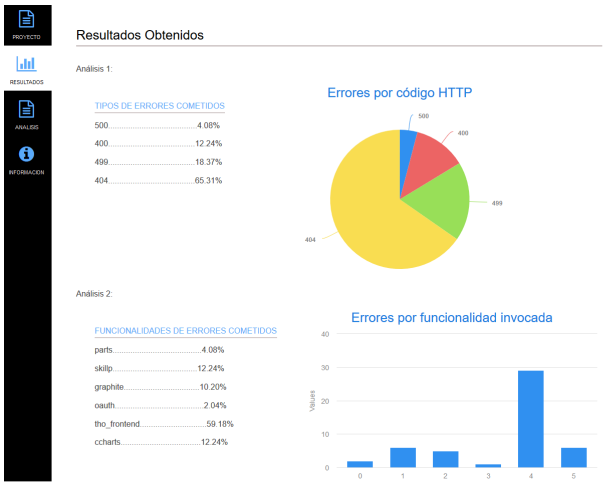

Figura F.4: Pantalla resultados del análisis

Análisis

En la pantalla de análisis se describe con profundidad los análisis que se han realizado (Figura (F.5)) y lo que se pretende mostrar en cada uno de ellos, de forma que luego al acceder a los resultados se puede comprender de manera más simple a qué hace referencia cada uno de los gráficos que se muestran.

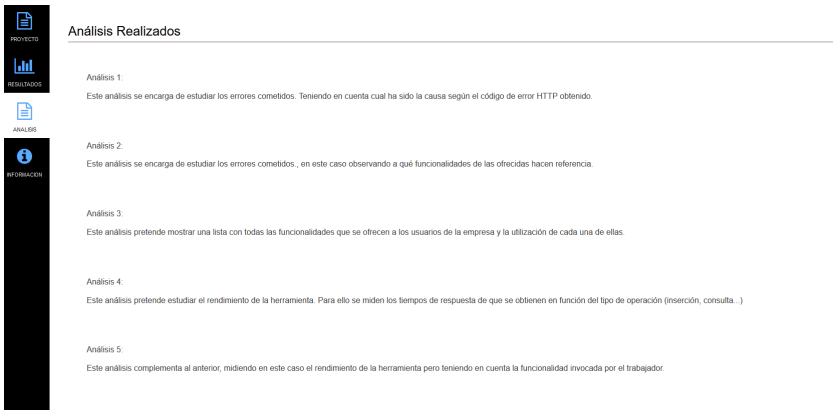

Figura F.5: Pantalla descripción de análisis realizados

Información

La pantalla muestra información adicional al proyecto (Figura (F.6)). En este caso se muestra información del contexto sobre el contexto sobre el que se ha realizado el proyecto. Una breve descripción del marco del proyecto en el que se encuentra el TFG.

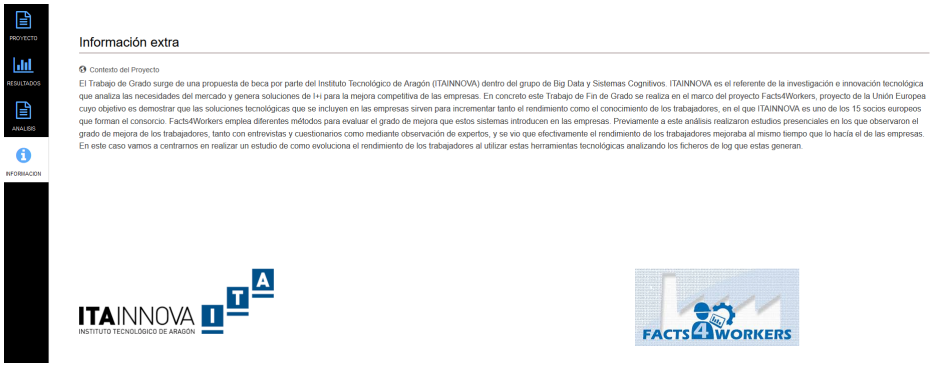

Figura F.6: Pantalla con información extra del proyecto

Apéndice G

Manual de instalación

A continuación, se detallan los pasos a seguir para poder en marcha el proyecto en los entornos de desarrollo:

Uno de los primeros pasos es instalar Syslog-ng de manera que a través de él se pueda leer del fichero de log y enviarlo a Kafka. Para ello, una vez instalado en el container de Docker correspondiente será necesario seguir los siguientes pasos:

- Para poder proceder al paso de mensajes de un sistema a otro es necesario haber copiado en dicho container las librerias de Kafka necesarias para que Syslog-ng pueda conectarse a este. Para ello, tras haber descargado Kafka se copiaran únicamente las librerias (disponibles en la carpeta /libs del directorio descargado) bajo el directorio /opt/kafka/lib en el mismo container de Docker en el que se haya instalado Syslog-ng.
- Tras ello se configurará Syslog-ng para que pueda procesar los mensajes que lea del fichero de texto y que lo envíe al *topic* de Kafka creado para dicho propósito. Para ello la configuración se establecerá en el fichero syslog-ng.conf de la siguiente forma:

Figura G.1: Configuración syslog-ng.conf

```
source s_kafka { file("/var/log/prueba");};
destination d_kafka {
        kafka(
                class_path("/install/syslog-ng/lib/syslog-ng/java-
                    modules/*.jar:/opt/kafka/lib/*.jar")
                kafka_bootstrap_servers("172.17.0.3:9092")
                topic("test")
        );
};
log {
        source(s_kafka); destination(d_kafka);
};
```

En la opción *kafka_bootstrap_servers* es necesario configurar la dirección en la que se encuentra el servidor Kafka.

En la fuente (*source*) es necesario indicar la ruta en la que se encuentra el fichero de log del cual se desean extraer datos.

Tras ello, una vez que se reinicie syslog-ng, cada vez que se almacene nueva información en el fichero de log indicado esta será enviada a Kafka.

Una vez configurado syslog-ng para que envíe la información que le llega del fichero de log a Kafka es necesario desplegar Kafka para que le lleguen estos datos.

En este caso para el despliegue de Kafka se ha optado por una configuración sencilla, utilizando uno de los containers disponibles de Docker que ya lo lleva todo instalado.

Para el despliegue de Kafka se seguirán los siguientes pasos:

- Previamente a desplegar Kafka, para que este funcione es necesario tener en funcionamiento Zookeeper, por lo que lo se hará funcionar en un container de Docker mediante el siguiente comando:

Figura G.2: Lanzar zookeeper

```
> docker run  -d -p 2181:2181 --name zookeeper jplock/zookeeper
```

- Una vez se encuentra Zookeeper en funcionamiento ya se puede iniciar Kafka, para lo que se utiliza el siguiente comando:

Figura G.3: Lanzar Kafka

```
> docker run  -d --name kafka --link zookeeper ches/kafka
```

- Una vez se dispone de todo en funcionamiento es momento de iniciar MongoDB para poder almacenar allí una colección con los datos de entrada y otra con los resultados tras realizar el análisis. Para desplegar el container que ofrece Docker con la imagen de Mongo se ejecuta el siguiente comando:

Figura G.4: Lanzar MongoDB

```
> docker run  -d mongo
```

En este punto se dispone de las condiciones necesarias para poder poder poner en marcha los scripts que se encargarán tanto de procesar como de analizar la información que llega a Kafka a través de Syslog-ng.

Para ello únicamente basta con ejecutar los siguientes comandos:

- Para poner en marcha el script que se encarga de preprocesar los datos antes de almacenarlos en la base de datos de forma semiestructurada se utiliza el siguiente comando:

Figura G.5: Script de preprocesamiento

```
> python processing.py $KAFKA_IP $KAFKA_TOPIC $MONGO_IP fichero
```

Este fichero será el encargado de preprocesar los datos que llegan a Kafka y almacenarlos en la colección correspondiente de MongoDB para su posterior análisis.

- Para poner en marcha el script que analiza los datos para poder mostrarlos posteriormente gráficamente a través del interfaz web se ejecuta el siguiente comando:

Figura G.6: Script de preprocesamiento

```
> python analysing.py $MONGO_IP
```

Apéndice H

Detalle casos de uso Thermolympic

Con la llegada de la Industria 4.0 muchos procesos industriales están cambiando para adaptarse a la nueva revolución digital que se está viviendo actualmente. En Thermolympic este proceso de digitalización se ha aplicado en tres casos de uso principalmente. A continuación, se explica para cada uno de ellos que cambios se han llevado a cabo.

H.1. Defectos y soluciones

Este caso trata de aquellas situaciones en las que se producen fallos, bien en las piezas o bien en las máquinas y el proceso de reparación era tedioso. Al introducir soluciones de Industria 4.0 esto ha cambiado, siendo un proceso mucho más simple.

Caso 1:

Los trabajadores deben medir la calidad de las máquinas comprobando que el tamaño de las piezas que se están produciendo coincide con el de las especificaciones. Para ello emplean aparatos de medida apropiados para cada parte de la pieza y van anotando manualmente los resultados (Figura (H.1)).

Este proceso resulta tedioso para los trabajadores, además de que se pueden producir fallos por falta de concentración o al cambiar de unos productos a otros si no se hace con el cuidado necesario. Además puede ser un proceso lento en caso de que la máquina produzca las piezas a una velocidad mucho mayor a la que el trabajador comprueba las medidas.

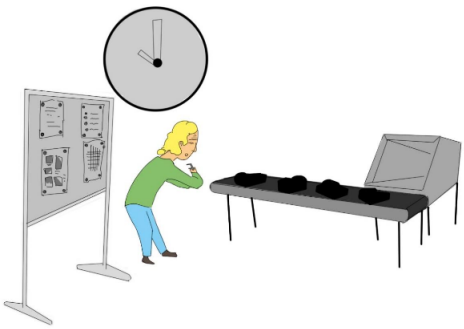

Figura H.1: Defectos y soluciones problema caso 1. Extraída de [10]

La solución de Industria 4.0 aplicada es la introducción de tablets para los trabajadores que les faciliten el trabajo (Figura (H.2)). En este caso los trabajadores pueden medir las piezas y su calidad mediante el uso de la tablet, introduciendo sus credenciales de acceso a la misma. En este caso además, se ha aplicado un nuevo módulo a la máquina que se encarga de comprobar las medidas de las piezas, por lo que los trabajadores únicamente tienen que comprobar aquellas piezas que la máquina detecta como defectuosas. En este caso, en lugar de tener que tomar todas las medidas y anotarlas manualmente, es la propia tablet la que toma las medidas e indica donde se está produciendo el error. Una vez localizado el defecto los trabajadores documentan en la tablet la decisión tomada, de manera que ahora toda la información se encuentra almacenada en tiempo real en almacenes de la empresa para poder medir la calidad de los procesos.

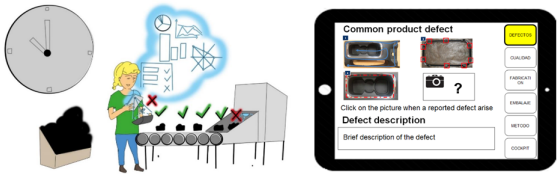

Figura H.2: Defectos y soluciones solución caso 1. Extraída de [10]

Caso 2:

Puede haber situaciones en las que los trabajadores detecten que la máquina está fabricando muchas piezas erróneas. En este caso los trabajadores deben parar la máquina y llamar al encargado para que evalúe el error que se está produciendo (Figura (H.3)). El encargado debe evaluar las medidas tomadas y observar la configuración de la máquina para localizar el fallo. Durante todo este tiempo la máquina se encuentra parada, de manera que se está perdiendo tiempo de producción.

Figura H.3: Defectos y soluciones problema caso 2. Extraída de [10]

La solución ya comentada de la tablet hace que la resolución de estos problemas sea mucho más cómoda y se reduzca menos el ritmo de la producción. Al detectar que se están produciendo fallos en los medidas de las piezas de manera continuada, los trabajadores pueden consultar la tablet, que les indica el error que se está produciendo y como previamente se ha registrado la solución adoptada en estos casos, el trabajador es capaz de solucionar el problema el solo siguiendo los pasos de la tablet sin necesidad de parar la máquina durante tanto tiempo (Figura (H.4)).

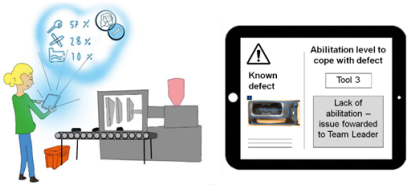

Figura H.4: Defectos y soluciones solución caso 2. Extraída de [10]

H.2. Gestión automatizada

En este caso de lo que se trata es de digitalizar los procesos de gestión de documentos.

Caso 1:

Cuando se realizan reuniones semanales para presentar los resultados de la semana, los datos están desactualizados, ya que para ello es necesario pasar toda la información que han ido anotando los trabajadores manualmente a hojas Excel, lo que hace que muchas personas se tengan que encargar del traspaso de la información (Figura (H.5)). El hecho de no tener los datos 100 % actualizados hace que al planificar la próxima línea de productos sean conservadores y baje mucho el número de producciones, lo que puede impactar en los ingresos anuales.

Figura H.5: Gestión automatizada problema caso 1. Extraída de [10]

El hecho de tener todo informatizado gracias al uso de las tablets hace que cuando se producen estas reuniones se disponga de la información en tiempo real, pudiendo mostrar resultados reales y haciendo estimaciones y tomando decisiones mucho más fiables (Figura (H.6)).

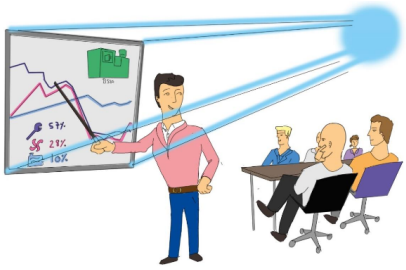

Figura H.6: Gestión automatizada solución caso 1. Extraída de [10]

Caso 2:

Cuando en la empresa se empieza a producir un producto los trabajadores deben configurar las máquinas de producción con las nuevas medidas de las piezas. Para ello, primero deben consultar la guía del producto en la que se encuentran las especificaciones del mismo y tras ello hacer algunos ajustes manuales que no vienen descritos en la guía pero que son necesarios de realizar (Figura (H.7)). Esto hace que la máquina tenga que estar parada mucho tiempo hasta que se configura toda de la manera correcta. Además, este proceso no puede documentarse de manera correcta, ya que hacerlo de manera manual es muy tedioso y lo dificulta.

Figura H.7: Gestión automatizada problema caso 2. Extraída de [10]

Gracias a la solución de Industria 4.0 cuando se empieza a producir un nuevo producto el trabajador únicamente debe seguir los pasos que le aparecen en la tablet para configurar la máquina para que produzca las nuevas piezas (Figura (H.8)). Además, en la tablet pueden aparecer comentarios de otros compañeros que simplifican el proceso de puesta en marcha de la máquina, lo que hace que esta configuración sea mucho más rápida y por tanto se pierda menos tiempo de producción.

Figura H.8: Gestión automatizada solución caso 2. Extraída de [10]

H.3. Aprendizaje en el puesto de trabajo

Antes del proceso de digitalización si los trabajadores eran rápidos revisando las medidas de las piezas y querían aprovechar el tiempo libre en mejorar y aprender nuevas cosas lo tenían difícil ya que la única información de la que disponían era un papel en la pared (Figura(H.9)). Esta información se encontraba en malas condiciones para ser consultada debido al paso de los años, y no disponía de los últimos módulos introducidos en la empresa, haciendo que la motivación de los trabajadores por formarse en los procesos disminuya.

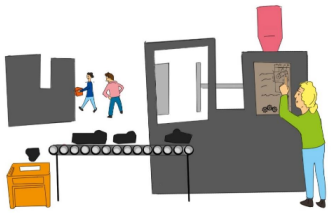

Figura H.9: Aprendizaje en el puesto de trabajo problema. Extraída de [10]

Con la implantación de soluciones Industria 4.0 los trabajadores no tienen que comprobar una a una las piezas ya que lo hace la máquina por ellos, y sólo en el caso de que esta detecte una pieza defectuosa deben actuar. Los trabajadores ahora disponen de una *tablet* en la que pueden consultar detalles de la máquina y de los procesos realizados, fomentando el aprendizaje (Figura (H.10)). Además, en caso de que vean que algún proceso puede mejorarse pueden poner un aviso y el jefe irá a hablarlo con ellos, de manera que los trabajadores mantienen su motivación ya que pueden aprender nuevas cosas y aportar su opinión. Esto no hace que el ritmo de producción de la empresa disminuya por no estar pendientes de las piezas, ya que en la propia tablet aparece una notificación cuando una pieza no cumple con las medidas que había sido especificado.

Figura H.10: Aprendizaje en el puesto de trabajo solución. Extraída de [10]

Bibliografía

[1] Ramón Archanco, *Qué es industria 4.0 y porqué debería importarte.* Fecha consulta: Enero 2018 http://papelesdeinteligencia.com/que-es-industria-4-0/

[2] Fundación CTIC, *Los habilitadores digitales de la Industria 4.0.* Fecha consulta: Enero 2018 http://www.fundacionctic.org/sat/articulo-los-habilitadores-digitales-de-la-industria-40

[3] Henning Kagermann, Reiner Anderl, Jürgen Gausemeier, Günther Schuh, Wolfgang Wahlster (Eds.), *Industrie 4.0 in a Global Context* Fecha consulta: Enero 2018 https://www.plattform-i40.de/I40/Redaktion/EN/Downloads/Publikation/industrie-40-in-a-global-context.pdf?__blob=publicationFile&v=1

[4] Plattform Industrie 4.0, *What is Industrie 4.0?.* Fecha consulta: Enero 2018 http://www.plattform-i40.de/I40/Navigation/EN/Industrie40/WhatIsIndustrie40/what-is-industrie40.html

[5] Facts4Workers, *Worker-Centric Workplaces for Smart Factories - FACTS4WORKERS.* Fecha consulta: Enero 2018 http://facts4workers.eu/

[6] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, *Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications.* Fecha consulta: Enero 2018 http://journals.sagepub.com/doi/pdf/10.1155/2015/431047

[7] Paul McJones, *James ("Jim") Nicholas Gray.* Fecha consulta: Enero 2018 https://amturing.acm.org/award_winners/gray_3649936.cfm

[8] Pwc, *Industry 4.0: Building the digital enterprise.* Fecha consulta: Enero 2018 https://www.pwc.com/gx/en/industries/industries-4.0/landing-page/industry-4.0-building-your-digital-enterprise-april-2016.pdf

[9] Thermolympic, Fecha consulta: Enero 2018 http://www.thermolympic.es/

[10] Denner et al., *"First version of requirements of workers and organisations". Project Report - FACTS4WORKERS: Worker - Centric Workplaces in Smart Factories.* Fecha consulta: Enero 2018 http://facts4workers.eu/wp-content/uploads/2016/12/FACTS4WORKERS_D1-2.pdf

[11] Flume, *Apache Flume.* Fecha consulta: Enero 2018 https://flume.apache.org/

[12] Kafka, *Apache Kafka.* Fecha consulta: Enero 2018 https://kafka.apache.org/

[13] Zookeeper, *Apache Zookeeper.* Fecha consulta: Enero 2018 https://zookeeper.apache.org/

[14] LogStash, *Elastic.co Logstash.* Fecha consulta: Enero 2018 https://www.elastic.co/products/logstash

[15] Syslog-ng, *Open Source log management solution with over a million users worldwide.* Fecha consulta: Enero 2018 https://syslog-ng.org/

[16] Guido Schmutz, *Big Data Architecture.* Fecha consulta: Enero 2018 https://www.slideshare.net/gschmutz/big-data-architecture-53231252

[17] Lacueva,F.J.; Mayo, S; Heinrich,P.;Moertl,P.;Hannola,L., *"Evaluation Framework". Project Report - FACTS4WORKERS: Worker - Centric Workplaces in Smart Factories.* Fecha consulta: Enero 2018 http://facts4workers.eu/wp-content/uploads/2016/12/FACTS4WORKERS_D6-1.pdf

[18] BOE núm. 298, *LEY ORGÁNICA 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal.* Fecha consulta: Enero 2018 https://www.boe.es/boe/dias/1999/12/14/pdfs/A43088-43099.pdf

[19] Pete Chapman (NCR), Julian Clinton (SPSS), Randy Kerber (NCR),Thomas Khabaza (SPSS), Thomas Reinartz (DaimlerChrysler), Colin Shearer (SPSS) and Rüdiger Wirth (DaimlerChrysler), *CRISP-DM 1.0.* Fecha consulta: Enero 2018 https://www.the-modeling-agency.com/crisp-dm.pdf

[20] John Rollins, Data Scientist, IBM Analytics, IBM, *Why we need a methodology for data science.* Fecha consulta: Enero 2018 http://www.ibmbigdatahub.com/blog/why-we-need-methodology-data-science

[21] Docker, *What is docker.* Fecha consulta: Enero 2018 https://www.docker.com/what-docker

[22] Nestify Team, *Why you should start using Docker Today?.* Fecha consulta: Enero 2018 https://nestify.io/blog/start-using-docker-today/

[23] MongoDB, *Move at the Speed of Your Data.* Fecha consulta: Enero 2018 https://www.mongodb.com/

[24] Luis Rández, *Introducción a Latex.* Fecha consulta: Enero 2018 http://pcmap.unizar.es/~pilar/latex.pdf

[25] Trello, *PFG Beatriz Franco.* Fecha consulta: Enero 2018 https://trello.com/b/XxswzhhC/pfg-beatriz-franco

[26] Laia Gilibets, *Qué es la metodología Kanban y cómo utilizarla.* Fecha consulta: Enero 2018 https://www.iebschool.com/blog/metodologia-kanban-agile-scrum/

[27] Vaadin, *Vaadin Fight for Simplicity.* Fecha consulta: Enero 2018 https://vaadin.com/

[28] Eclipse, Fecha consulta: Enero 2018 https://www.eclipse.org/org/

[29] Python Software Foundation, Fecha consulta: Enero 2018 https://www.python.org/

[30] Kong Inc, *Kong File Logs.* Fecha consulta: Enero 2018 https://getkong.org/plugins/file-log/

[31] Heinrich, P.; Richter, A., *"Captured and structured practices of workers and contexts of organizations". Project Report - FACTS4WORKERS: Worker - Centric Workplaces in Smart Factories.* Fecha consulta: Enero 2018 http://facts4workers.eu/wp-content/uploads/2016/12/FACTS4WORKERS_D1-1.pdf

Manuales

[32] Syslog-ng, *The syslog-ng Open Source Edition 3.13 Administrator Guide.* Fecha consulta: Enero 2018 https://www.balabit.com/documents/syslog-ng-ose-latest-guides/en/syslog-ng-ose-guide-admin/html/configuring-destinations-kafka.html

[33] Bruno Cascio, *Tutorial basado en el libro Docker Cookbook de O'reilly.* Fecha consulta: Enero 2018 https://github.com/brunocascio/docker-espanol