Monitorización de equipos BigleLegal

Gerard López Sánchez

Resumen—Este proyecto se ha llevado a cabo en el departamento de desarrollo de la empresa Bigle Iberia SL, una start-up que se dedica a la automatización de documentos legales. En los departamentos de Desarrollo y el de Legal de la empresa el uso de equipos informáticos es fundamental en el trabajo diario de los trabajadores. Este uso intensivo lleva a la necesidad de poder monitorizar en todo momento sus prestaciones. Para ello hemos desarrollado un entorno de monitorización que recoge información de los equipos, se manda a una base de datos en la nube y se puede recuperar a través de una aplicación web. De esta forma los responsables autorizados podrán conocer en todo momento el uso y prestaciones de cada equipo y preveer la necesidad de cambios o actualizaciones de los mismos.

Palabras clave—Agente, Angular, JAVA, KPI (Key Performance Indicator), Maven, monitorizar, thread, TypeScript.

Abstract—This Project has been carried out within the Bigle Iberia S.L.'s development department, a Legaltech start-up whose core business is the automation of legal documents. In the Developmet and Legal departments the informatic equipment usage is essential for employe's daily work. This itensive use makes necessary monitoring at all times its capabilities. So we have developed a monitoring environment which collects the equipment information. This information is sent to a cloud database and they can be recovered through a web application. This way the authorized people can know at all times the use and the capacities of each machine and they can foresee the necessity of changes or any update of these ones.

Index Terms— Agent, Angular, JAVA, KPI (Key Performance Indicator), Maven, monitoring, thread, TypeScript.

1 INTRODUCCIÓN

BIGLE IBERIA, SL (en adelante, Bigle Legal) es una start-up que se dedica a la automatización de documentos legales. Transforma la forma en la que las empresas crean y administran estos documentos. Crea plataformas web que capacitan a las empresas para que creen documentos legales en poco tiempo y sin errores humanos. Actualmente existen 15 personas dentro de la empresa repartidas en 3 departamentos: Marketing y Ventas, Legal y Desarrollo. Este proyecto se lleva a cabo dentro del departamento de Desarrollo. Me integré a la empresa hace 1 año y 8 meses como desarrollador en prácticas. Al finalizar las prácticas entré a formar parte de la empresa como desarrollador full stack.

Bigle Legal es una empresa del sector Legaltech. Este término se comenzó a utilizar hace unos años en Estados Unidos y el Reino Unido. Se trata de una revolución del sector legal que viene a resolver un problema constatado por Law Technology Today [1] «la experiencia del cliente en la mayoría de los bufetes de abogados es, desde hace 50 años, la misma». Según TechCrunch [2], desde diciembre de 2014, «la Legaltech está en pleno apogeo, con sociedades que intentan innovar dentro del mercado legal en todos sus niveles».

Cuando hablamos de Legaltech hacemos referencia a algo relativamente nuevo, que como concepto lleva poco más de 5 años con cierta fuerza. Consiste en aplicar la tecnología a la prestación o a la comercialización de servicios legales. Por ejemplo, si un abogado utiliza tecnología para mejorar sus servicios está usando, hasta cierto punto, herramientas que entran en el cajón de la Legaltech. Es ahí donde la empresa juega un papel importante, pues ofrece la tecnología para mejorar los servicios de todo aquel que dedique su vida profesional al mundo de la legislación.

La Legaltech ayuda al profesional, como tecnología aplicada a unos servicios y su comercialización, pero también ayuda al cliente para acceder al abogado, por ejemplo, a sus servicios o para calificarlo. Esta orientación al consumidor, a la experiencia, a la inmediatez es lo que la diferencia de los servicios tradicionales. Por ello está revolucionando todo el mercado, ya que esta forma de trabajar, con el transcurso de los años, se volverá cada vez más importante.

Por este motivo Bigle Legal, al querer ser de las mejores empresas en este sector tecológico ha optado por integrar un software *homemade* para saber si sus equipos informáticos pueden rendir correctamente y ejecutar todas las tareas que deben realizar sus trabajadores durante sus jornadas laborales.

- *E-mail de contacto: gerard.lopezs@e-campus.uab.cat*
- *Mención realitzada: Tecnologías de la Información.*
- *Trabajo tutoritzado por: Jordi Pons Aróztegui (DEIC)*
- *Curso 2018/19*

En este informe se expone la necesidad de crear un software para monitoritzar los equipos de la empresa y la forma como se ha realizado. Se explica con detalle todo el desarrollo del proyecto, empezando los objetivos, la planificació, los requisitos a tener en cuenta y la metodología empleada. A continuación se describen los pasos seguidos para construir el entorno de monitorización y resolver todas las necesidades así como el resultado final explicando como debe utilizarse el producto.

2 FINES Y OBJETIVOS

De la misma forma que en la mayoría de empresas que se dedican a la creación de plataformas, aplicaciones y servicios web, es básico que los equipos de trabajo funcionen de forma correcta y tengan un buen rendimiento. Esta necesidad también la encontramos en BigleLegal. Para estar seguros de ello se plantea la necesidad de crear un sistema de monitorización de los distintos equipos utilizados. Por este motivo el objetivo principal del trabajo es crear un servicio instalable en todos los equipos que monitorice el estado y el rendimiento frente a las operaciones que se llevan a cabo, como pueden ser: peticiones constantes al cloud y a los servidores, operaciones dentro del terminal, deploys de aplicaciones, etc, y así permitir a cualquier responsable de la empresa con autorización poder visualizarlo. Se pretende monitorizar el rendimiento y estado de los equipos de la empresa para conocer su potencia y sus carencias (deploys de aplicaciones o servicios muy lentos, velocidad de procesamiento baja a la hora de testear, etc). De esta forma se puede conocer si un equipo es o no lo suficientemente potente para llevar a cabo las tareas necesarias del empleado de la empresa que lo utiliza. A su vez se puede elaborar un historial de rendimiento para así conocer picos de uso o carencias en algún momento dentro del período establecido. Cabe mencionar que todos los equipos utilizados por los empleados tienen un sistema operativo Windows, lo que puyede facilitar la creación de una plataforma uniforme y de uso simple para los usuarios.

Para poder conseguir este objetivo principal se plantean los siguientes subobjetivos:

- Recoger datos de rendimiento de cada máquina.
- Transformarlos a un formato estándar para poder desplegarlos en la BDD.
- Transmitir los datos cargados en BDD a una plataforma web para que los usuarios autorizados puedan visualizarlos.

Podemos ver en el diagrama de la Figura 1 los elementos básicos a desarrollar que permitirán alcanzar estos objetivos. Los tres subobjetivos son de la misma prioridad, la máxima, ya que si uno de ellos no se consigue, no podemos obtener una herramienta final funcional.

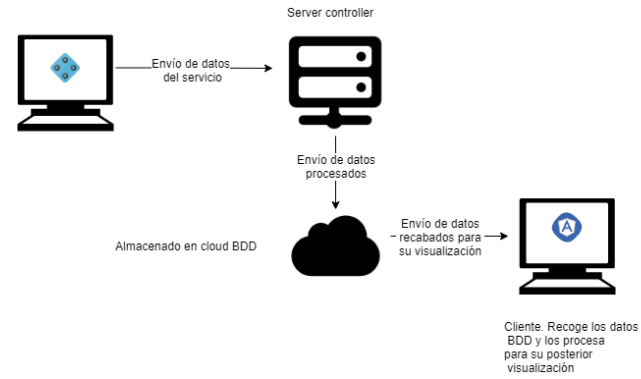


Fig. 1: Diagrama básico de los elementos del sistema

3 ESTADO DEL ARTE

Actualmente existe en el mercado software que monitoriza equipos y ayuda a visualizar su rendimiento y estado. Incluso los propios Sistemas Operativos proporcionan información del equipo desde el panel de control. Sin embargo, para este proyecto se ha considerado más adecuado desarrollar un agente propio que monitorice los equipos en vez de utilizar alguna herramienta ya existente.

Existen diversas razones por las que se ha optado por crear un agente de monitorización propio, tanto a nivel de desarrollo como a nivel económico.

En primer lugar, es necesario destacar que la mayoría de los trabajadores de la empresa no tiene conocimientos avanzados de informática. El conocimiento acerca de las herramientas que proporcionan los sistemas operativos sobre la información de un equipo informático, sobre todo para los que no se dedican a la informática es muy limitado por lo que el diseño y desarrollo de esta herramienta basada en un servicio Windows proporciona una oportunidad única para corregir esas lagunas.

Desde un punto de vista económico, tras realizar un análisis de distintas herramientas que ofrece el mercado como, por ejemplo, Paessler, SolarWinds o New Relic [3], hemos visto que son herramientas muy completas, pero el coste de dichas soluciones era impensable para una start-up como Bigle Legal, por lo que la solución in-house parece la más razonable.

A su vez es una oportunidad para que el desarrollador del proyecto se familiarice con los lenguajes y las herramientas utilizadas en la empresa (JAVA, JS, TypeScript [4], Angular [5] y Maven [6]) y potencie sus habilidades para poder dar un mayor rendimiento dentro de la organización.

Además la presencia de un agente propio en todos los equipos de la empresa permitirá tanto la monitorización de dichos equipos como el diagnóstico y la implementación de futuras mejoras.

Al tratarse de un agente propio, permitirá extraer toda la información que deseemos acerca del uso y rendimiento de los equipos a fin de tener una visión tanto in live como histórica del estado de nuestra red de equipos así como de su nivel de obsolescencia a fin de poder proceder a su renovación según las necesidades de cada departamento y usuario. Debido a que se instalará como un servicio de Windows con permisos de administrador, se podrán descargar paquetes de actualización de manera remota para que los equipos estén siempre actualizados y protegidos.

Finalmente, al ser un agente que solo se instalará en los equipos de Bigle Legal, en el futuro, permitirá gestionar el acceso a las herramientas del cloud mediante la implementación de un sistema de SingleSignOn que unifique todos los accesos bajo un sistema único que esté vinculado a cada uno de los equipos, evitando así el acceso desde equipos externos a las herramientas online de la empresa.

4 METODOLOGÍA Y PLANIFICACIÓN

El desarrollo del proyecto se divide en 4 grandes bloques:

- **El Recolector de datos**. El servicio comenzará con la creación de una aplicación JAVA que ejecute comandos CMD en el equipo monitorizado para recoger los KPI (Key Performance Indicator) deseados (voltaje, temperatura, rendimiento de la RAM, consumición del propio servicio). Una vez recogidos los datos, se enviarán en formato JSON a un controlador vía https. Esta aplicación se transformará en un ejecutable .exe con la ayuda de prunsrv.exe [7], nsi y unos ficheros .bat para que el equipo que se desea monitorizar únicamente necesite este archivo.

- **El Controlador**. El controlador dispondrá de los KPI recogidos y los transformará para poder enviarlos a una BDD cloud.

- **La BDD**. La BDD cloud almacenará los datos de interés recogidos en los equipos y mandados por el controlador. Se ha escogido Firestore [8] de Firebase, Google.

- **La Interfaz de usuario**. Finalmente se procederá a crear una interfaz de usuario donde se podrán visualizar todos los KPI recogidos. Dicha interfaz será una aplicación web desarrollada con el framework de Angular con lenguaje TypeScript.

Se han definido los tres casos de uso que podemos ver en la Figura 2.

En primer lugar se recogen los KPI del sistema accediendo al CMD de éste. Con la ayuda de RegEx podemos recoger exactamente el KPI y su valor eliminando espacios entre medio para así poder transformarlos en formato JSON y poder enviarlos vía http al controlador.

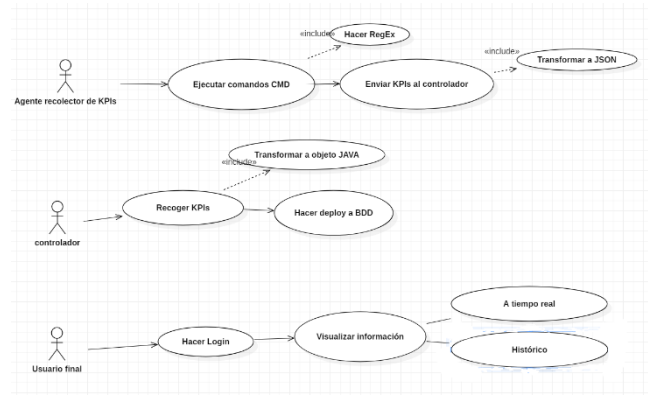


Fig. 2: Diagrama casos de uso del sistema

Una vez se recoge en el controlador estos KPI, se transforman de nuevo para poder mandarlos a la BDD.

Por último, un usuario con autorización puede visualizar estos datos en una plataforma Web.

4.1 Metodología de trabajo

La metodología que se ha utilizado en el desarrollo del TFG ha sido Critical Path Method (CPM) [9] con la cual se ha estalecido la duración estimada de cada actividad, el orden de implementación y la relación entre ellas. No se ha comezado la siguiente actividad hasta que la anterior ha finalizado, ya que se necesita utilizar la funcionalidad de ésta para ver si la salida esperada de la recién finalizada es la correcta.

Después de finalizar la implementación de cada actividad se ha sometido a un testeo para ver que todo funcionaba correctamente.

Los lenguajes utilizados han sido JAVA (recolector de KPIs y controlador) y TypeScript (interfaz de usuario).

Los IDE y las herramientas utilizadas han sido Microsoft Visual Studio [10] como IDE y el Framework de Angular.

Me he decantado por el lenguaje JAVA a la hora de implementar el recolector y el controlador porque es el lenguaje con el que me siento más cómodo programando y contiene librerías que encajan perfectamente en las necesidades de este proyecto como, por ejemplo, las librerías java.util.concurrent.Executors y java.util.concurrent.ScheduledExecutorService para ejecutar de forma periódica un Thread.

Respecto a utilizar TypeScript y el framework de Angular, el primero es una muy buena forma de tipificar JavaScript y el segundo es un framework perfecto para construir aplicaciones web creando componentes y módulos para la reutilización de código y un desarrollo de código

más ágil.

Como IDE de desarrollo, Microsoft Visual Studio Code es una herramienta sensacional ya que contiene plugins tanto para desarrollar aplicaciones en Angular y TypeScript como un plugin para desarrollar aplicaciones Spring Boot [11]. Para este segundo caso, inicialmente utilicé el IDE Spring Tools Suite [12] para la parte de recogida y envío de los KPI, pero al conocer el plugin reciente que ha introducido Microsoft VS Code de aplicaciones Spring Boot para JAVA y Maven, he escogido esta opción para así tenerlo todo en un mismo IDE de trabajo y coseguir más homogeneidad. Además la apariencia visual de la interfaz es mucho más moderna y utiliza la distribución del Sistema de explorador de archivos que ofrece Microsoft.

4.2 Planificación

Los 4 módulos a desarrollar se han englobado en tres grandes tareas con las que será necesario un diseño previo, su implementación y un testeo posterior, tanto a nivel funcional como de integración con el resto de módulos.

El orden de las implementaciones ha sido el siguiente:

- Aplicación que recolecta los KPIs accediendo al CMD del equipo y su construcción como servicio Windows.
- Controlador que recoge estos KPIs, los transforma y los envía a la BDD.
- Plataforma de interacción con el usuario para que pueda visualizar los atriutos recogidos.

Nombre actividad	Fecha inicio	Fecha fin	Duración en horas
Búsqueda de lenguajes e IDEs que utilizar	01-feb	19-feb	no estimadas
Diseño del recolector de KPIs	20-feb	22-feb	16
Listar Kpis que se quieren monitorizar	22-feb	23-feb	8
Diseño del controlador de KPIs	23-feb	24-feb	16
Implementación del recolector	26-feb	01-mar	56
Implementación del controlador	01-mar	02-mar	60
Implementación del Servicio Windows	02-mar	15-mar	80
Diseño plataforma web para el cliente	15-mar	18-mar	16
Implementación plataforma web	25-mar	15-abr	80
Total			332

Fig. 3: Tabla de planificación

En la Figura 3 podemos ver la planificación de tareas definida inicialmente para el desarrollo del proyecto, que se ha seguido de forma bastante estricta para poder conseguir los resultados esperados en el tiempo establecido para el desarrollo de un TFG. Podemos ver que el número total de horas previsto es un poco superior a las horas estipuladas en un TFG.

5 Desarrollo

A continuación, se resume qué pasos se han llevado a cabo para conseguir el resultado final y una breve reflexión sobre los logros coseguidos.

5.1 Establecer el Servicio en el equipo

Para llevar a cabo el establecimiento del Servicio Windows se han necesitado los siguientes elementos:

- Una aplicación JAVA en un proyecto Maven.
- Un fichero batch donde se establecen los comandos que debe ejecutar el CMD del equipo.
- El ejecutable prunsrv.exe.

Una vez se dispone de estos 3 elementos se tiene que exportar dicha aplicación como un .jar, editar el fichero batch estableciendo todas las variables de entorno necesarias y guardarlas en el mismo directorio junto con el administrador de instalación y desinstalación de servicios Windows prunsrv.exe. Dicho directorio se denomina "Installable" y está dentro del proyecto.

Para instalar el servicio en el Sistema Operativo se tiene que ejecutar en el CMD del equipo (siempre como administrador) *installService.bat* y para desinstalarlo *uninstallservice.bat*.

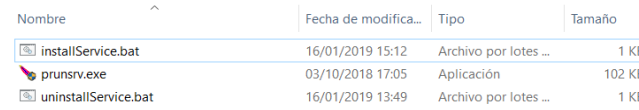


Fig. 4: Directorio con el administrador y los ficheros de instalación y desinstalación

A continuación, para exportar el proyecto se tiene que clicar el botón derecho, escoger la opción "Show in Local Terminal" y escribir en el terminal *mvn package*. También se puede ejecutar el comando *mvn package* desde el CMD del equipo en el directorio donde tengamos el proyecto creado. El jar exportado se encuentra en la carpeta target del proyecto (se tiene que coger siempre el de tamaño más grande ya que es el que exporta las librerías). Por último, se debe modificar el nombre del jar a *monitorization- agent.jar*.

Finalmente, para que el proyecto exporte las librerías se necesita modificar el *pom.xml* para que contenga el plugin *maven-assembly-plugin* [13]. Para ejecutar el servicio instalado hay que ir al apartado Servicios del equipo, buscar el nombre del servicio e iniciarlo. El procedimiento para detenerlo será el mismo.

El fichero *pom.xml* está configurado de tal forma que cuando ejecutas el comando *maven package* detro de la carpeta del proyecto en un Terminal crea el .jar, coge los ficheros batch de instalación y desinstalación dentro de la carpeta *batch* junto con el *prunsrv.exe* y los aloja en la carpeta *Installable* anteriormente comentada.

Esto sirve para que cualquier persona con acceso al código (desarrolladores de la empresa) pueda crear esta carpeta de instalación. Para poder instalar el servicio en un equipo únicamente se necesita tener esta carpeta, es decir, con un pen drive, envíandola por mail o por cualquier otra vía puedes obtener los elementos para istalar el servicio en cualquier equipo.

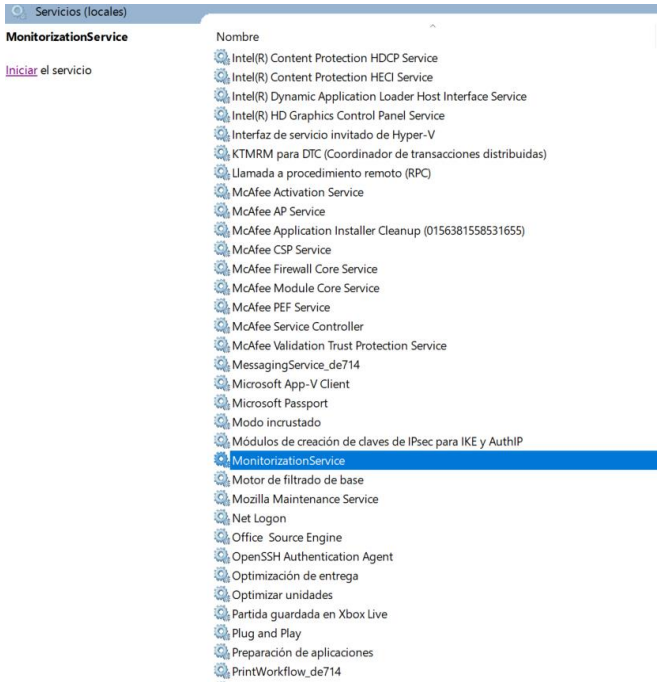


Fig. 5: Servicio instalado en la sección de Servicios del equipo

El fichero *installService.bat* contiene, entre otras, la variable PR_INSTALL que indica donde está el configurador *prunsrv.exe*, una variable PR_CLASSPATH que indica cual es nuestra aplicación .jar que debe instalar (servicio), las variables PR_STARTCLASS y PR_STOPCLASS que indican la ruta dentro de la aplicación .jar del método *main*, las variables PR_STARTMETHOD y PR_STOPMETHOD que indican los métodos que se tienen que llamar dentro del *main* (*start* para iniciar el servicio y *stop* para pararlo).

Finalmente, se ejecuta //IS// que es el parámetro instalador de *prunsrv* indicando cómo queremos que se llame nuestro servicio con la variable SERVICE_NAME para instalarlo.

El fichero *uninstallService.bat* lo único que necesita es ejecutar //DS// que es el parámetro desinstalador de *prunsrv* indicando el nombre del servicio a desinstalar.

En la Figura 6 podemos ver la configuración del fichero de instalación y en la Figura 7 la del fichero de desinstalación del servicio.

```
set SERVICE_NAME=MonitorizationService
set PR_INSTALL=%cd%\prunsrv.exe

REM Service log configuration
set PR_LOGPREFIX=%SERVICE_NAME%
set PR_LOGPATH=%cd%\logs
set PR_STDOUTPUT=%cd%\logs\stdout.txt
set PR_STDERROR=%cd%\logs\stderr.txt
set PR_LOGLEVEL=Error

REM Path to java installation
set PR_JVM=%JRE_HOME%\bin\server\jvm.dll
set PR_CLASSPATH=monitorization-agent.jar

REM Startup configuration
set PR_STARTUP=auto
set PR_STARTMODE=jvm
set PR_STARTCLASS=com.bigle.core.SomeService
set PR_STARTMETHOD=start

REM Shutdown configuration
set PR_STOPMODE=jvm
set PR_STOPCLASS=com.bigle.core.SomeService
set PR_STOPMETHOD=stop

REM JVM configuration
set PR_JVMMS=256
set PR_JVMMX=1024
set PR_JVMSS=4000
set PR_JVMOPTIONS=-DMONITORIZATION_AGENT=%cd%\logs

REM Install service
%PR_INSTALL% //IS//%SERVICE_NAME%
```

Fig 6: Configuración del fichero de instalación

```
set SERVICE_NAME=MonitorizationService

REM Uninstall service
%cd%\prunsrv.exe //DS//%SERVICE_NAME%
```

Fig 7. Configuración fichero de desinstalación

5.2 Recoger datos de rendimieto de cada màquina

Para recolectar los KPI se ha creado una clase con un método estático el cual llama al comando que se ejecuta en el CMD del agente y extrae el KPI deseado.

La clase necesita:

- Un array de strings denominado keys que contendrá el/los KPI/s deseado/s. Se llama keys porque es la key que guardamos en el mapa como llave.
- Un array de strings que contendrá el/los comando/s que se ejecutaría/n en el CMD.
- Un mapa donde se guarda el KPI deseado y su valor.

Para obtener el valor del KPI se utiliza una función de filtrado *RegEx* [14]. Ésta necesita una llave en formato texto y una línea del resultado de ejecutar el comando del array de strings del segundo punto de las necesidades nombradas anteriormente.

Los KPI recogidos actualmete son:

- nombre y modelo del equipo,
- número de serie que sirve para identificar cada equipo,
- zona horaria,
- arquitectura del SO,
- versión del SO,
- fecha en que se inicializó por primera vez el equipo,
- última actualización del equipo,
- capacidad de memoria,
- información del procesador,
- memoria virtual utilizada
- y por último, el consumo del propio servicio instalado en el equipo.

Según la empresa vaya necesitando recabar más información se puede ir ampliando el abanico de KPI.

```
private static String filterSystemInfo(String line, String key) {

    final String regex = "(" + key + "):\\s*([\\S| ]*)";
    final String string = line;
    String value = "";

    final Pattern pattern = Pattern.compile(regex);
    final Matcher matcher = pattern.matcher(string);

    if (matcher.find()) {
        value = matcher.group(2);
    }
    return value;
}
```

Fig 8: Función de filtrado RegEx.

El recolector recoge los datos cada cierto tiempo. Para llevar a cabo la periodicidad de la aplicación se debe utilizar el concepto de threading el cual permite controlar el flujo de un programa.

Para definir el thread se tiene que crear una clase que herede de la clase de java Thread y desarrollar su método *run()*, el cual llama a los métodos estáticos recolectores de los KPI.

```
public void run() {
    logger.info("Hello from the thread!");
    try {
        agentData.setCreationTime(new Date().getTime());
    } catch (Exception e) {
        logger.error("Fallo en el System Info", e);
    }
    try {
        agentData.setListKpis(SystemInfo.getSystemInfo());;
    } catch (Exception e) {
        logger.error("Fallo en el System Info", e);
    }

    try {
        agentData.add(TaskList.getTaskList());
    } catch (Exception e) {
        logger.error("Fallo en el Task List", e);
    }

    try {
        agentData.add(Wmic.getWmic());
    } catch (Exception e) {
        logger.error("Fallo en el wmic", e);
    }

    try {
        agentData.setAgentIdentification(Wmic.getSerialnumber());
    } catch (Exception e) {
        logger.error("Fallo en la conexi\u0000n con el Servidor", e);
    }

    try {
        this.respuesta = connection.startConnection(agentData);
    } catch (Exception e) {
        logger.error("No se han podido enviar los datos al Servidor", e);
    }

    logger.info("Datos enviados al Servidor: " + this.respuesta);

}
```

Fig. 9: Método run para llamar a las clases recolectoras de KPI

Después de implementar el thread se utiliza la interfaz *ScheduledExecutorService* de JAVA para poder ejecutar el *run()* del thread cada determiado tiempo.

Para iniciar:

```
executor.scheduleWithFixedDelay(startThread, 1, 5, TimeUnit.SECONDS);
```

Donde:

- *startThread* es el *Runnable* donde tendremos la ejecución del servicio
- *5* es la periodicidad con la que se ejecutará
- *TimeUnit* es la unidad de tiempo del delay (segundos, minutos u horas).

Para detener:

```
executor.shutdown();
```

Estos dos métodos son llamados desde la clase *main* de la aplicació Java.

5.3 Conexión con el servidor

Para enviar los datos monitorizados a nuestro servicio se ha creado una clase donde se abre una conexión http con la dirección del servidor y se manda el objeto en forma de JSON, con la librería Gson de Google [15].

Request Body	{
	agentIdentification: string,
	creationTime: long,
	"monitorizationAgentData": [{
	listKpis: [{
	name: string,
	kpi: string
	}],
	}]}

Success Response	Status: 200
	{
	ResponseEntity: {
	Message: string,
	ApiError: {
	timestamp: Date,
	message: string,
	debugMessage: string,
	subErrors: [{
	object: string,
	field: string,
	rejectedValue: Object,
	message: string
	}]
	}
	}
	}

Fig. 10: Cuerpo de envío de datos y respuesta esperada del servidor si no hay error.

5.4 Servidor para el tratamiento de datos y su envío a BDD

El servidor utiliza la estructura típica de servicio de *Spring boot* con las variables de anotación *RestController* y *EnableAutoConfiguration* para controlar y autoconfigurar la conexión, *RequestMapping* para establecer el método en el que se recibe el input y se devuelve el output y establecer la URL a la que hacer la petición y una clase para que el código del servidor se ejecute. Este servicio debe recoger un objeto y tiene que devolver otro.

La base de datos utilizada es *Firestore* de Firebase y el servidor está alojado en Heroku [16]. Heroku es uno de los PaaS (Plataforma como Servicio) más utilizados actualmente en entornos empresariales por sus facilidades de despliegue de aplicaciones y manejo de servidores y sus configuraciones.

Para crear la conexión a BDD se ha creado una classe con un método estático que se llama desde la clase recolectora y devuelve una conexión a Firestore. A este método se le pasa como parámetro un string que indica a qué base de datos nos conectamos. Una vez escogida la BDD a la que nos conectamos, se instancian la url donde está la BDD y el JSON SDK que ofrece Firebase. Este JSON SDK se encapsula en un *FileInputStream* y se guarda en una variable que junto con la url donde está la BDD se agregan al método *Builder().setCredentials* con la clave necesaria de Firebase para poder tener acceso a la BDD.

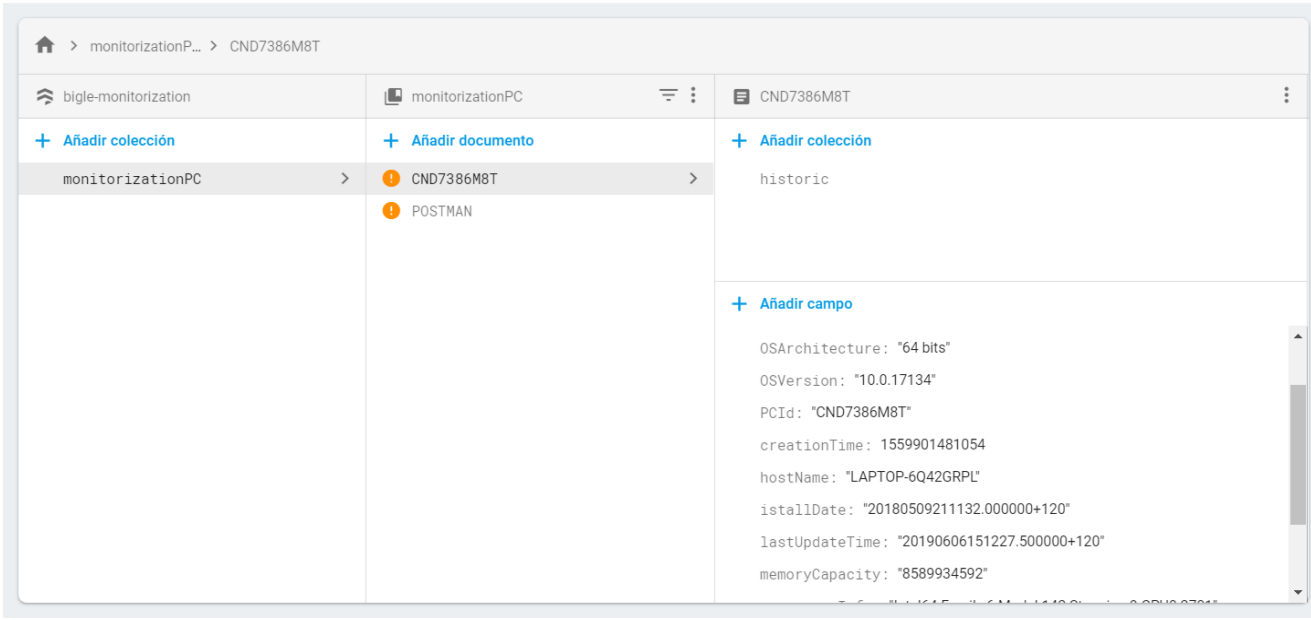


Fig. 11: BDD Firestore con los KPIs recogidos.

5.5 Plataforma Web

La plataforma web para la visualización de los datos recogidos se ha desarrollado en Angular con lenguaje TypeScript. Recoge los KPI de BDD y muestra por cada equipo que envía datos toda la información. La plataforma está protegida por una página de *Log in* en la que si no se tiene permiso no se puede acceder a la información. Las figuras 12 y 13 muestran un par de ejemplos de la interfaz de monitorización.

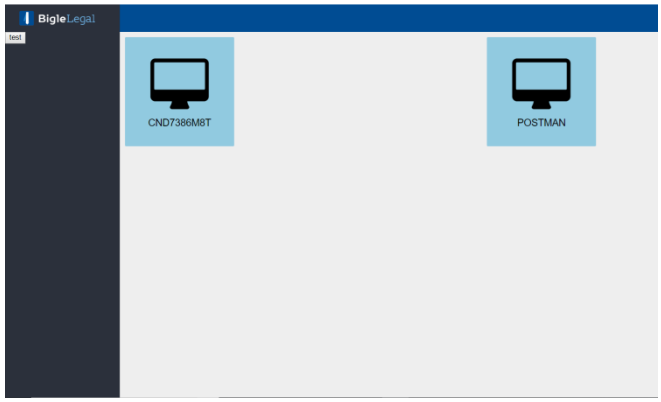


Fig. 12: Plataforma Web con 2 equipos monitorizados

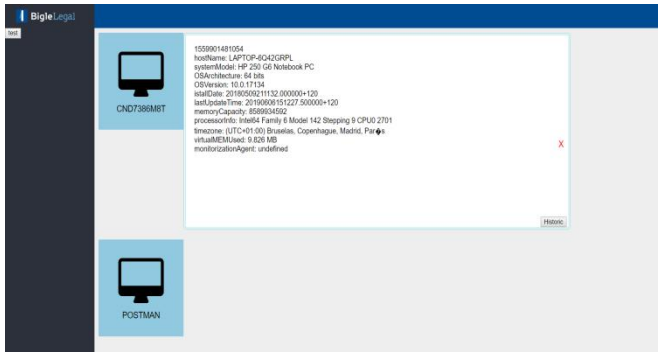


Fig. 13: KPI moitorizados de un equipo

6 Resultados obtenidos

Desde la empresa se planteaba la necesidad de crear un sistema de monitorización de los distintos equipos utilizados por los trabajadores. Para resolver esta necesidad se ha creado un servicio instalable para estos equipos que monitoriza el estado y el rendimiento frente a las operaciones que se llevan a cabo como pueden ser: peticiones constantes al cloud y a los servidores, operaciones dentro del terminal, deploys de aplicaciones, etc, que permite a cualquier responsable de la empresa con autorización visualizar la información monitorizada.

Se han creado con éxito cuatro elementos que hacen posible esta monitorización:

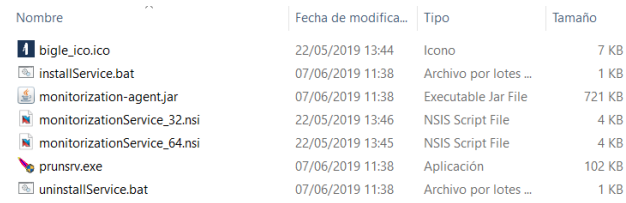
- una aplicación JAVA que accede al terminal del equipo y ejecuta los comandos pertinentes para recoger los datos y enviarlos a un servidor para que sean tratados;
- un servidor JAVA/Spring que transforma los datos y los despliega en la BDD;
- una plataforma web Angular/Typescript para visualizar la información recogida y,
- una serie de ficheros que permiten la instalación de esta primera aplicación en cualquier dispositivo Windows.

Se estudió en su día el funcionamiento y la posibilidad de integrar herramientas de monitorización que ofrece el mercado como, por ejemplo, Paessler, SolarWinds o New Relic en la empresa. Viendo lo que es capaz de hacer nuestro servicio se podría decir que no tiene nada que envidiarles en cuanto a funcionamiento. Sí que es verdad que estas herramientas están más trabajadas y tienen una mejor interfaz de visualización y experiencia de usuario. Sin embargo, al tratarse de un agente propio y personalizado, permite extraer toda la información que deseemos acerca del uso y rendimiento de los equipos a fin de tener una visión tanto in live como histórica del estado de nuestra red de equipos así como de su nivel de obsolescencia a fin de poder proceder a su renovación según las necesidades de cada departamento y usuario.

7 Futuras mejoras

Durante el tiempo que me ha llevado poder finalizar lo que viene a ser la idea básica y original del proyecto se me han ido ocurriedo un conjunto de mejoras y amplaciones. A cotinuación se explica cada una de las mejoras que se pueden llevar a cabo en un futuro:

- En vez de tener la carpeta *Installable* con el contenido necesario para poder instalar el servicio en los equipos, tener un .exe que haga todo este trabajo. Actualmente esta mejora está creada utilizando NSIS. Sin embargo, se consigue crear el .exe e instalar el servicio ejecutándolo, pero al iniciarlo desde el SO no se comporta como debería. El error está detectado y ubicado pero no se ha podido solucionar. El error se encuentra dentro del fichero de configuración .nsi para crear ejecutables en un SO con arquitectura de 64 bits.



Nombre	Fecha de modifica...	Tipo	Tamaño
bigle_ico.ico	22/05/2019 13:44	Icono	7 KB
installService.bat	07/06/2019 11:38	Archivo por lotes ...	1 KB
monitorization-agent.jar	07/06/2019 11:38	Executable Jar File	721 KB
monitorizationService_32.nsi	22/05/2019 13:46	NSIS Script File	4 KB
monitorizationService_64.nsi	22/05/2019 13:45	NSIS Script File	4 KB
prunsrv.exe	07/06/2019 11:38	Aplicación	102 KB
uninstallService.bat	07/06/2019 11:38	Archivo por lotes ...	1 KB

Fig. 14: Directorio con los elementos necesarios para crear el .exe

- A parte de poder visualizar los datos recogidos en tiempo real, poder visualizar históricos mediante tablas y/o diagramas. Los datos nuevos y actuales se guardan en BDD manteniendo los datos anteriores, que se van guardando en una colección por cada equipo monitorizado llamada *histórico* (Figura 15) lo que facilita su posterior tratamiento.

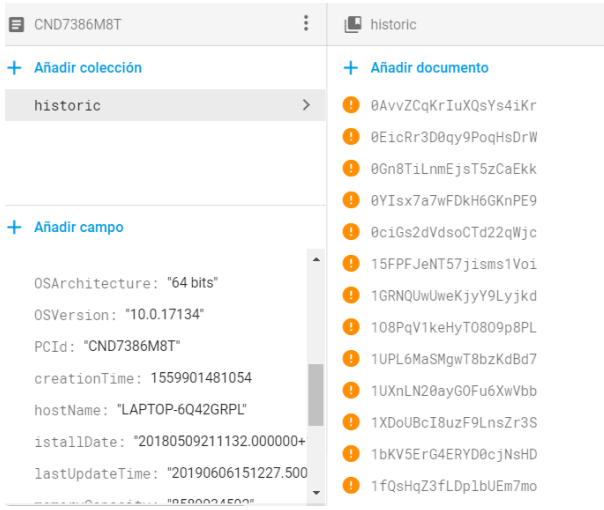


Fig. 15: Histórico de un equipo monitorizado

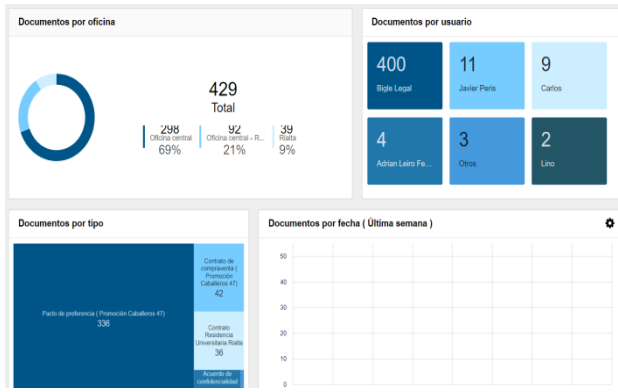


Fig. 16: Diagramas y tablas para el histórico

En la Figura 16 se puede apreciar un ejemplo de cómo se podría mostrar el histórico mediante el uso de librerías para crear tabblas y diagramas con Angular.

- Tener un monitor en la oficina el cual muestre la plataforma web con una distribución física de los equipos igual a la de la oficina. Esos equipos podrían verse de color verde si todos los KPI que determinan el rendimiento de la máquina monitorizada no superan ni se acercan a los parámetros de riesgo que se indiquen, amarillo si se acercan o rojo si los superan.

- Hacer la plataforma multilenguaje ya que actualmente solo está disponible en español.

- Parametrizar el tiempo de envío de los KPI. Si en un momento quiero ver los KPI como cambian cada 3 min en vez de 5 min, por ejemplo, poder hacerlo desde la plataforma sin tener que acceder al código fuente para modificarlo.

8 Conclusiones

Se ha conseguido desarrollar el proyecto dentro del periodo marcado implementando todos los objetivos planteados inicialmente, tal como hemos visto en el apartado 7 del informe. Sin embargo, como hemos descrito en el apartado anterior, todavía existen posibles ampliaciones y mejoras para conseguir un producto más útil para la empresa.

Al haber cambiado de IDE de trabajo para la parte de recolección y envió de los KPI se ha notado una mejora en cuanto a tiempo de codificación y ejecución de la aplicación porque VS Code funciona extremadamente más rápido y es más intuitivo que STS Tools. Además, al utilizar este primer IDE como herramienta de trabajo para la parte de Angular y TypeScript se ha logrado una homogeneidad en cuanto a instrumentos de trabajo. Esto facilitará las posibles actualizaciones futuras del sistema.

Dentro del marco académico comentr que existen una variedad de asignaturas en el Grado que han contribuido en mayor o menor medida en el desarrollo de este trabajo, pero podemos destacar dos que contribuyen en mayor medida: “Tecnologías de desarrollo para Internet y web” y “Sistemas y tecnologías web”. También tienen una contribución importante, pero menos directa, otras asignaturas que han aportado ayudas adicionales como técnicas de diseño de software, estándares y métodos de programación y codificación, por ejemplo.

A nivel personal el resultado más satisfactorio es el de haber interactuado con las herramientas, lenguajes y metodologías que utiliza la empresa y así mejorar en su manejo para poder ser cada vez un trabajador más importante dentro de ésta.

Agradecimientos

Primero de todo quiero agradecer a mi tutor Jordi Pons el tiempo dedicado durante todo el transcurso de proyecto a guiarme, aconsejarme y motivarme

También me gustaría agradecer a mi empresa por dejarme llevar a cabo el TFG con un proyecto importante para ella y en especial al que fue mi tutor de prácticas dentro de la empresa y actualmente superior y CTO de la compañía Daniel Tomás por todo el apoyo ofrecido y todo lo que me ha enseñado.

Me gustaría agradecer además a todos los profesores que he tenido durante todo el grado que de una forma u otra me han proporcionado lo necesario para poder realizar a día de hoy tal designio.

Por último, agradecer a mis padres la ayuda y la motivación que me han proporcionado, no únicamente para este trabajo, sino durante toda mi vida estudiantil.

BIBLIOGRAFÍA

[1] **Law Technology Today**. [artículo en línea]. [Fecha consulta: 25 de abril de 2019]. <https://www.lawtechnologytoday.org/>.

[2] **TechCrunch**. [artículo enn línea]. [Fecha consulta: 25 de abril de 2019]. <https://www.techcrunch.com/>.

[3] **Paessler, SolarWinds y New Relic**. [artículo enn línea]. [Fecha consulta: 5 de ferero de 2019].
<https://www.es.paessler.com/>,
<https://www.solarwinds.com/es/>,
<https://newrelic.com/>, respectivamet.

[4] **TypeScript.** [documentación].
<https://www.typescriptlang.org/>.

[5] **Angular**. [documentación]. <https://angular.io>/.

[6] **Maven**. [documentación]. <https://maven.apache.org/>.

[7] **Martin Farrell (31/10/2016). Procrun – Java Program as Windows Services.** [artículo enn línea]. [Fecha consulta: 15 de marzo de 2019] <https://www.javabullets.com/procrun-java-programs-as-windows-services/>.

[8] **Firebase Google Cloud**. [documentación].
<https://firebase.google.com/>.

[9] **Luz Mariela Espinal Torres (4/04/2013). CPM.** [artículo enn línea]. EOI. [Fecha consulta: 15 de febrero de 2018]
<https://www.eoi.es/blogs/madeon/2013/04/14/metodo-de-ruta-critica-cpm-critical-path-method/>.

[10] **Microsoft Visual Studio Code.** [documentación].
<https://code.visualstudio.com/docs/>.

[11] **Spring Boot.** [documentación].
<https://spring.io/projects/spring-boot/>.

[12] **Sping Tools Suite.** https://spring.io/guides/.

[13] **Apache Maven Assembly Plugin.** [documentación].
<http://maven.apache.org/plugins/maven-assembly-plugin/>.

[14] **RegEx.** [documentación]. <https://es.ryte.com/wiki/RegEx/>.

[15] **Arlandy Rodríguez, Miguel (2012). Jugando con JSON en Java y la librería Gson. Oracle.** [artículo enn línea].
[Fecha consulta: 25 de marzo de 2019]
<https://www.adictosaltrabajo.com/2012/09/17/gson-java-json/>.

[16] **Heroku.** [documentación].
<https://dashboard.heroku.com/apps/>.