Universidad Politécnica de Cataluña

Facultad de informática de Barcelona

Creación de un sistema de monitorización y seguimiento de errores de aplicaciones ubicuas

Trabajo de Final de Grado

Memoria

Antonio Arellano Moral
Grado en Ingeniería informática

2018

LudiumLab S.L.

Índice general

Índice de figuras

Índice de tablas

Resumen

Este documento contiene el Trabajo de Final de Grado para el Grado en Ingeniería Informática, especialidad en Tecnologías de la Información por la Facultad de Informática de Barcelona, siendo este trabajo desarrollado en LudiumLab S.L.[46]
Este proyecto tiene como objetivo aportar a la empresa un sistema para el monitoreo y la detección de errores en aplicaciones ubicuas, diseñando, implementando y desplegando las diferentes partes de tal sistema.
Las aplicaciones ubicuas se han convertido ya en una constante en la vida diaria. En especial las aplicaciones dirigidas a los smartphones. Cada día aparecen nuevas aplicaciones que hacen que la informática se integre en el entorno de la persona, apareciendo en cualquier lugar y en cualquier momento, este hecho genera nuevos retos para los desarrolladores, como pudiera ser la detección y seguimiento de errores que puedan producirse en tales aplicaciones.
Las aplicaciones ubicuas hoy día pueden ejecutarse en un gran número de dispositivos, este trabajo se centrará en los smartphones y en especial en aquellos que ejecutan Android como sistema operativo.

Abstract

This document contains the Master Thesis for the Bachelor's degree in Informatics Engineering, Major in Information Technologies by Facultat dÍnformàtica de Barcelona, developed at LudiumLab S.L. [46].

This project aims to bring to the company a system for monitoring and detect errors in ubiquitous devices by designing, implementing and deploying the different parts of that system.

Ubiquitous applications have already become a constant in daily life. Especially smartphones applications. Every day new applications appear and get integrated into the environment of the person, appearing anywhere and at any time. This fact generates new challenges for developers, such as the detection and monitoring of errors that may occur in such applications.

Today's ubiquitous applications can run on a large number of devices, this project will focus on smartphones and especially those that run Android.

Capítulo 1

Introducción y motivaciones

1.1. Introducción

Este proyecto es un Trabajo de Final de Grado de la modalidad B de la especialidad Tecnologías de la Información. El presente trabajo se ha hecho en colaboración con la empresa LudiumLab S.L. la cual es propietaria y creadora del servicio de cloud gaming Play Everywhere. Este servicio consiste en permitir a sus usuarios jugar a juegos diseñados para Windows PC desde otros dispositivos como podrían ser smartphones, tabletas o navegadores web.

Dado que la empresa da servicio a dispositivos ubicuos, tiene la necesidad de recolectar y procesar eventos para que sus programadores corrijan comportamientos anómalos de la aplicación que se ejecuta en dichos dispositivos. Por lo que este trabajo es un caso de estudio para dar solución a la problemática de la empresa de recolectar y procesar eventos para darle información de valor a sus programadores.

En la Figura 1.1 se puede ver como un dispositivo ubicuo está corriendo un videojuego diseñado para Windows PC, este videojuego realmente lo corre el Backend de Ludium y envía el video y el audio al dispositivo, el dispositivo ubicuo envía las acciones a realizar en el videojuego al Backend de Ludium. Este flujo de datos es el existente en la empresa. En la Figura 1.1 también se ve otro flujo de datos que va del dispositivo ubicuo al Sistema de monitorización y seguimiento de errores. Este flujo de datos es el que se ha desarrollado en este trabajo. Existe un flujo de datos entre el Sistema de monitorización y seguimiento de errores, y el Backend de Ludium

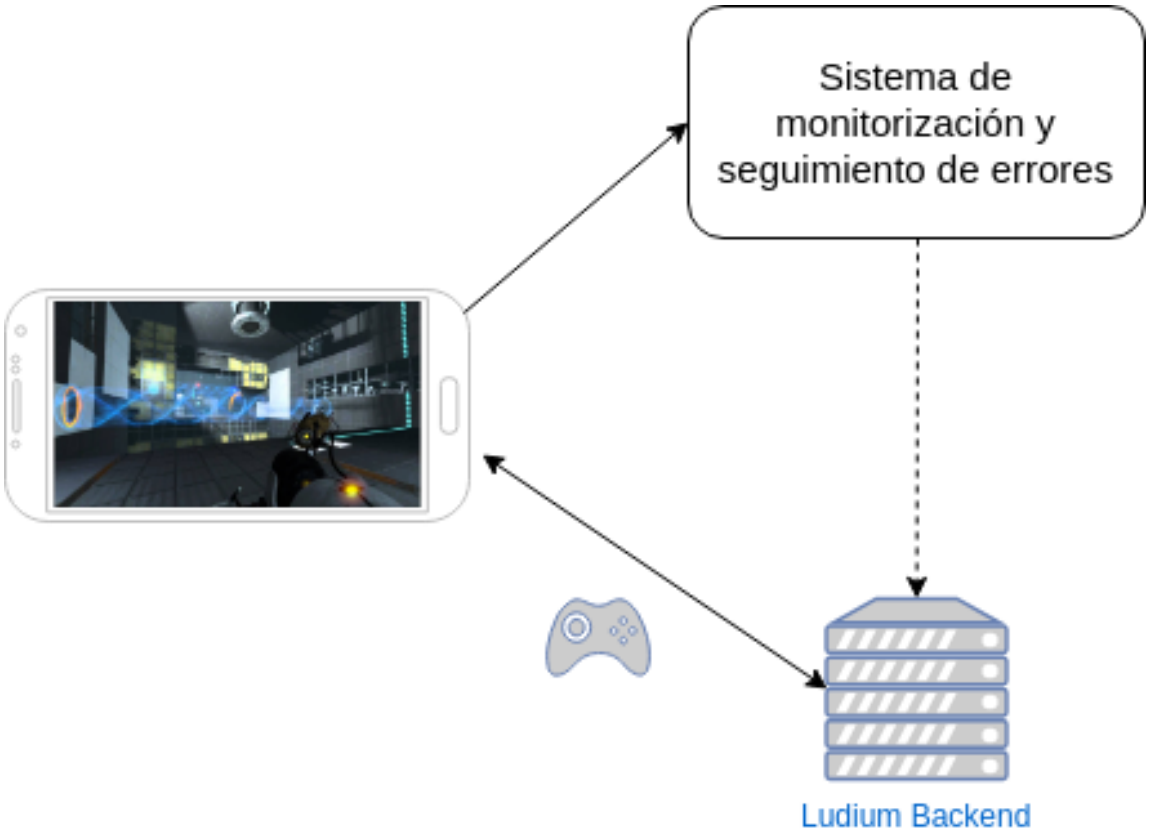


Figura 1.1: Visión introductoria del contexto del sistema

puesto que el sistema que se ha desarrollado se ha de integrado con ciertas partes del Backend de Ludium.

1.2. Contexto

Las aplicaciones ubicuas [28] se han convertido ya en una constante en la vida diaria. En especial las aplicaciones dirigidas a los smartphones. Cada día aparecen nuevas aplicaciones que hacen que la informática se integre en el entorno de la persona, apareciendo en cualquier lugar y en cualquier momento. Este hecho genera nuevos retos para los desarrolladores, como pudiera ser la detección y seguimiento de errores que puedan producirse en tales aplicaciones. Las aplicaciones ubicuas hoy día pueden ejecutarse en un gran número de dispositivos. Este trabajo se centra

en los smartphones y en especial en aquellos que ejecutan Android como sistema operativo.

Para controlar comportamientos anómalos en las aplicaciones, se suelen desarrollar para que, a parte de cumplir su función principal, informen sobre su estado. El registro de actividades de una aplicación es conocido como log [37]. Los sistemas que corren estas aplicaciones ubicuas también generan logs, en este trabajo para diferenciar los logs de las aplicaciones, de los logs del sistema, los últimos son llamados syslogs. Hay logs que directamente informan de un error y de que la aplicación ha dejado de funcionar, para diferenciar a estos logs de los syslogs y los logs, en este trabajo son llamados crashlogs. El conjunto de logs, syslogs y crashlogs es llamado alertas o eventos.

El hecho de que las aplicaciones ubicuas aparezcan en cualquier lugar y en cualquier momento, hace que muchas veces las aplicaciones ubicuas sean también aplicaciones distribuidas [43]. El controlar la comunicación y la integración de las aplicaciones con otros sistemas puede llegar a producir una cantidad de alertas (logs, syslogs y crashlogs) elevada. Al multiplicar la gran cantidad de alertas producidas por el número de instancias ejecutándose de la aplicación, se obtiene un número más que cuantioso de alertas, las cuales se han de procesar para obtener información sobre el comportamiento de las aplicaciones. Tomando el caso de Netflix[1] como ejemplo [41], pueden llegar a generar 8 millones de eventos por segundo, lo que supone 24 GB por segundo de eventos. Por este motivo, es cada vez más frecuente entre las empresas el procesar la gran cantidad de alertas producidas por las aplicaciones con soluciones de procesado de Big Data.

Big data [22] se refiere al concepto de conjuntos de datos tan masivos que las aplicaciones tradicionales de procesado de datos no son capaces de tratar con ellos. Por lo tanto, las soluciones de procesado de Big Data son todas aquellas capaces de tratar con tal volumen elevado de datos. El proceso que se suele utilizar para tratar Big Data es el de ETL (Extract, Transform and Load) [35], consiste en extraer los datos desde los sistemas de origen, luego transformarlos de manera oportuna y por último, una vez los datos ya han sido transformados, cargarlos en un sistema destino donde serán consumidos.

[1] https://www.netflix.com

Existen dos técnicas principales para aplicar el ETL en Big Data [52]. La primera es el Batch Processing, que consiste en transformar datos que ya han estado almacenados durante un tiempo. La segunda es el Stream Processing, la cual consiste en transformar los datos en tiempo real, es decir, sin que los datos pasen mucho tiempo almacenados antes de ser transformados.

Se encuentran dos arquitecturas de Big Data distinguidas que aplican las técnicas mencionadas.

Una es la arquitectura Lambda [31], la cual integra las dos técnicas. Consta de tres capas lógicas, una capa de Batch Processing, otra de Stream Processing y una última capa donde se sirven los datos transformados.

La otra es la arquitectura Kappa [51], en la cual se utiliza únicamente Stream Processing. Consta de tres capas lógicas, una capa de Stream Processing, otra de almacenamiento persistente de datos sin transformar y una última capa donde se sirven los datos transformados.

1.3. Descripción de los sistemas de alertas existentes en la empresa

En la empresa no existía un sistema de alertas como tal, pero sí que existen servidores con software de seguimiento de incidencias y servidores de búsqueda donde almacenar los datos procesados. En concreto, el software de seguimiento de incidencias que se utiliza es JIRA[10] y el servidor de búsqueda Elastic Search[16]. El sistema se integra tanto con JIRA como con Elastic Search.

1.4. Problema, objetivos y alcance

1.4.1. Problema

Para que los desarrolladores puedan ofrecer la mejor experiencia de usuario necesitan recoger información de manera y forma específica sobre el comportamiento de sus aplicaciones, luego procesar esa información para poder identificar posibles comportamientos inesperados de las aplicaciones y recibir avisos con información de valor

sobre los comportamientos anómalos encontrados para que puedan ser solucionados.

La empresa encuentra el problema de que esa recolección y procesado de eventos requiere de unos conocimientos sobre cómo hacer la recolección, el procesado, y que la manera y los medios que se vayan a utilizar se adapten al entorno ya existente en la empresa. Cabe destacar que no existe ningún sistema en la empresa, a parte del que se expone en este trabajo, que ya supla la necesidad de recolectar y procesar eventos, por lo que otro problema puede ser la complejidad, ya que el empezar de cero es más complejo que no aprovechar partes existentes. Otro problema se encuentra en el hecho de que la empresa tenderá a crecer y a adquirir nuevas necesidades, por lo que la escalabilidad es otro problema.

1.4.2. Objetivos

Objetivo principal

El objetivo principal del trabajo es diseñar, implementar y desplegar un sistema que sea capaz de recolectar eventos de aplicaciones ubicuas, procesarlos e integrarse con herramientas utilizadas en la empresa.

Objetivos específicos

Para efectuar el objetivo principal, se realizarán los siguientes objetivos específicos:

a) Diseño, implementación y despliegue de un protocolo para la recolección y envío de eventos (logs, syslogs y crashlogs) transparente al usuario.

b) Diseño, implementación y despliegue de un sistema capaz de recibir y almacenar de forma asíncrona un volumen elevado de datos y capaz de integrarse con una herramienta de Big Data Processing.

c) Configuración de una herramienta de Big Data Processing para que se integre en el sistema y almacene los datos transformados donde puedan ser consumidos.

d) Integración de JIRA y Elastic Search con el sistema. Se ha de ser capaz de publicar tanto en JIRA como en Elastic Search la información transformada.

1.4.3. Resultados esperados

Los resultados que se esperan obtener con este Trabajo de Final de Grado son los siguientes:

1. Integración de una o más librerías que sean capaces de recolectar logs y crash-logs en un dispositivo ubicuo y enviarlos a un sistema externo. Esta recolección y envío se ha de realizar de forma transparente al usuario en el momento que se considere oportuno.

2. Un sistema, accesible por red, de recepción asíncrona y capaz de almacenar un volumen elevado de eventos, que sea escalable, resiliente a fallos en la red, independiente al dispositivo donde se ejecuta la aplicación y que se comunica con el sistema de transformación de eventos.

3. Un sistema de transformación de eventos en tiempo real, siendo el núcleo del sistema una herramienta de Big Data Processing, que transforma en información útil para los desarrolladores, los datos recibidos por el sistema de recepción de eventos. Este sistema deberá almacenar en el lugar más conveniente la información transformada ya sea en JIRA o en Elastic Search. La latencia de este sistema será baja en comparación con los sistemas de Batch Processing. Las herramientas que compondrán al sistema serán modernas en la medida de lo posible. La escalabilidad del sistema ha de ser muy elevada ya que luego ha de poder ser utilizado para procesar otro tipo de datos como los derivados del Business Intelligence además unificar de todos los logs que puedan generarse en la empresa en una misma capa.

1.5. Estado del arte

En esta sección se muestra el estado del arte de dos partes fundamentales del sistema. Por un lado, la recolección de eventos en Android, y por otro el procesado de eventos. Al final de la sección, se muestran las conclusiones extraídas para la creación del sistema.

1.5.1. Recolección de eventos en Android

En Android es posible ver crashlogs, syslogs o logs de aplicaciones si se trabaja conectado al dispositivo en modo depuración, este sistema para el propósito del proyecto no es útil dado que lo que se pretende es enviar tales eventos sin necesidad de conectar de forma física el dispositivo ubicuo a otro sistema.

Existen librerías que pueden ser utilizadas para añadirle funcionalidades a las aplicaciones Android con el objetivo de que recolecten eventos y los redirijan a otros canales diferentes del estándar (el canal estándar suele ser la consola de un PC).

Por un lado encontramos librerías especializadas en recolectar crashlogs como ACRA [1], que recolecta crashlogs y es capaz de redirigirlos a diferentes destinos, en esta categoría también encontramos a Breakpad [29] que es un conjunto de cliente y servidor, aunque esta es más compleja de acoplar con Android.

Por otro lado encontramos librerías para generar y redirigir logs de aplicaciones. Existen librerías que tienen como base la clase Log [8] propia del sistema Android como es el caso de HyperLog [34]. Tales librerías suelen mejorar la clase Log consiguiendo redirigir los logs a diferentes destinos como podría ser un servidor. También encontramos librerías que no utilizan la clase Log de Android y que son propias del lenguaje Java en vez de Android. Este es el caso de las librerías basadas en log4j o su nueva versión log4j2 [20]. Han habido intentos de utilizar de forma pura log4j o log4j2 en Android, pero dado que Android utiliza de manera especial ciertos aspectos de Java no se suelen utilizar de forma pura, sino que se suelen utilizar librerías adaptadas como Logback [38] o Blitz4j [42] que se adecuan mejor a Android.

1.5.2. Procesado de eventos con arquitecturas de Big Data Processing

Como referencia principal en la indústria es destacable el sistema que utiliza Netflix[2] para explotar sus eventos. La conocida como V2.0 Keystone pipeline [41] utiliza una arquitectura basada en Lambda para integrar en una sola capa tanto el procesado de eventos procedentes de logs como el procesado de eventos procedentes de su BI (Business Intelligence) [40]. Utiliza Apache Kafka [19], herramienta de moda

[2] https://www.netflix.com

en el mundo del Big Data Processing.

Otra referencia bastante importante en la industria, es el sistema que utiliza LinkedIn[3] para explotar los eventos que producen sus aplicaciones remotas y servidores. El conocido como Inception [48] unifica en una sola capa todos los logs de sus dispositivos y aplicaciones para luego transformarlos y consumirlos por un software de seguimiento de errores como JIRA [10]. Utiliza también Apache Kafka dado que el creador de dicha herramienta es trabajador de LinkedIn.

Luego es posible encontrar sistemas más enfocados al Business Intelligence pero también interesantes a considerar como es el caso de los sistemas de Spotify [39] o el de Twitch [44]. Lo destacable en el sistema de Spotify es el uso extenso de Kafka además de una integración de la capa de BI con la de logs. Lo destacable del sistema de Twitch es el modo en que hace la ingesta de datos dentro del sistema, una forma diferente con respecto a otros sistemas en la industria.

Otro sistema a destacar es el sistema de Pinterest [32], la arquitectura en sí es la clásica arquitectura Lambda sin variaciones, lo interesante de este sistema es la recolección de eventos que hace en los dispositivos, consiste en un agente (Singer [23]) que recolecta logs en diferentes formatos y los envía a la puerta de entrada del sistema de transformación de datos, en este caso Apache Kafka.

En conclusión, encontramos que la industria lo que suele hacer para solucionar el problema de la recolección de eventos en aplicaciones ubicuas, procesado y mostrado de datos transformados es utilizar algún tipo de librería en el dispositivo ubicuo que sea capaz de recopilar los datos, y enviarlos a un agente intermedio entre el dispositivo ubicuo y el sistema de transformación de datos, una vez los datos han llegado al sistema de transformación, los datos de eventos se suelen procesar utilizando técnicas de Stream Processing. Estos sistemas de transformación de datos suelen integrar eventos y BI. Para mostrar los datos transformados suelen integrar con el sistema algún software de seguimiento de incidencias ya utilizado en la compañía y que los trabajadores ya están acostumbrados a él.

[3] https://es.linkedin.com

Tecnologías de transformación en tiempo real

Las tecnología expuestas en la sección 1.5.2 siguen el patrón ETL, donde la extracción es diferente para cada tipo de dispositivo al cual se le quiera extraer datos, luego tales datos se envían a un elemento intermedio (Apache Kafka) antes de transformarlos. Se puede decir que una vez los datos traspasan tal elemento intermedio empieza la transformación de los datos. Existen diferentes tecnologías con las que llevar a cabo la transformación en tiempo real, pero se van a presentar las dos más significativas y actuales.

Apache Spark Streaming:

Es la tecnología más popular para hacer transformaciones en tiempo real, aunque realmente no hace Stream Processing sino Micro Batching[36], para latencias no muy pequeñas no se diferencia el utilizar Stream Processing a Micro Batching. Es una solución ampliamente testeada y con una gran comunidad detrás, por lo que se puede encontrar bastante documentación y ejemplos, pero tiene la limitación de que realmente no hace Stream Processing. El lenguaje de implementación es en alto nivel, por lo que es bastante sencillo.

Apache Kafka Streams:

Es una tecnología relativamente nueva y no hace Micro Batching sino que hace un Stream Processing real. Tal solución está basada en Apache Kafka por lo que si ya se utiliza Kafka puede ser una buena opción puesto que no habrá ningún problema de integración. Una curiosidad de esta tecnología es que para procesar consume de Apache Kafka y una vez los datos han sido procesados los devuelve a Kafka, por lo que todo lo que se pueda integrar con Kafka podrá ser procesado. Aunque la documentación podría ser mayor, el lenguaje de implementación es bastante amigable y la curva de aprendizaje no es muy pronunciada.

Logstash:

Tanto Spark Streaming como Kafka Streams son tecnologías de propósito general, es decir, se pueden procesar todo tipo de datos ya sean eventos, mensajes o mero texto sin formatear. Existen tecnologías de propósito específico que se encargan de procesar un tipo de datos en concreto, en el caso de este proyecto el tipo de datos que interesa son los eventos (Logs, Syslogs, Crashlogs). Este es el ejemplo de Logstash parte de la ELK Stack (Logstash, Elastic Search, Kibana), que permite un procesado

específico para eventos antes de almacenar los eventos.

1.5.3. Procesado de eventos como servicio

Es posible hallar empresas que ya ofrecen la solución del seguimiento y procesado de errores en las aplicaciones como un servicio, este es el caso de Crashlytics [30] o Datadog [15]. Existe una solución de Stream Processing como servicio de Amazon, Amazon Kinesis [3], pero no resuelve el tema de la recolección de eventos, sólo su procesado.

1.5.4. Conclusiones

Una vez analizado el estado del arte se concluye que para la recolección de eventos se han de utilizar librerías ya existentes adaptando ciertos aspectos de estas para que se ajusten a las necesidades de la empresa, tales librerías ya implementan los requisitos buscados y se adaptan fácilmente al uso que se quiere hacer con ellas. Las librerías ACRA y Logback son buenas candidatas para ser utilizadas.

Para el procesado de eventos no se ha de contratar ningún servicio sino que tomando como base casos de uso ya existentes se ha de diseñar, implementar y desplegar una nueva arquitectura de Big Data que se adapte a las necesidades de la empresa. No se ve conveniente contratar ningún servicio dado que es preferible ser dueño de los datos y la arquitectura para ser más flexibles a cambios en el modo de tratar los datos y el uso que se va a hacer de ellos. El uso de una arquitectura basada en el Stream Processing parece una buena aproximación para procesar los eventos, dado que tales eventos pueden contener información crítica y un procesado de baja latencia puede ayudar a corregir errores en las aplicaciones ubicuas en poco tiempo.

Capítulo 2

Planificación inicial del proyecto

En este Capítulo se muestra la planificación que se hizo antes de iniciar el proyecto. En el Capítulo 9 se presenta la planificación final y se analizan las desviaciones con respecto la planificación inicial. En el Capítulo 10 se muestra la gestión económica final teniendo en cuenta los cambios respecto la planificación inicial.

2.1. Identificación de tareas

Aprendizaje: El desarrollador del proyecto es la primera vez que se enfrenta a elementos de Big Data. Necesita un tiempo para analizar el estado del arte del campo del Big Data antes de ponerse a diseñar y organizar el sistema, por lo que la tarea se basa en adquirir los conocimientos esenciales del mundo del Big Data processing y del mundo de la recolección de eventos en aplicaciones ubicuas, en especial dispositivos Android. Del mundo del Big Data processing se profundizará sobretodo en los diferentes agentes que intervienen para poder luego diseñar una solución al problema propuesto por el proyecto. Esta tarea no necesita ningún recurso dado que se utilizará información libre de Internet.

Desarrollo del sistema: Delimitar el sistema en subsistemas y estos subsistemas en módulos reconociendo dependencias entre ellos. Se puede delimitar el sistema en tres subsistemas, el de recolección y envío, el de recepción e ingesta y el de transformación. Para cada subsistema se tendrán que realizar las tareas siguientes:

- **Delimitación de módulos:** Establecer las funcionalidades del módulo y cómo

interacciona con los módulos adyacentes.

- **Estudio de desarrollo de los módulos:** Investigar y diseñar la forma que mejor se adapta en el contexto de la empresa las delimitaciones del módulo, es decir, que servicios y herramientas utilizar para llevar a cabo la configuración e implementación del módulo.
- **Configuración e implementación de los módulos:** Puesta a punto de servicios y programación del código que haga cumplir las funcionalidades de cada módulo en base al estudio de desarrollo que se ha hecho sobre este. A nivel de recursos se necesitarán máquinas para lleva a cabo dicha configuración e implementación.
- **Testeo del subsistema predespliegue:** Antes de desplegar el subsistema en producción se le someterá a una serie de escenarios posibles en local
- **Despliegue del subsistema:** Selección de máquinas y llevar a cabo las configuraciones pertinentes para que el subsistema esté en producción. A nivel de recursos se necesitaran una serie de servidores.
- **Testeo del subsistema postdespliegue:** Una vez el subsistema esté en producción someterlo a una serie de escenarios posibles.

Integración del sistema con JIRA: El sistema se ha de integrar con JIRA para ofrecer información de valor a la empresa. Esta integración se tendrá que hacer una vez el sistema de procesado de eventos esté en marcha y produzca datos. A nivel de recursos, esta tarea necesita una licencia de JIRA. Esta tarea se puede dividir en las subtareas siguientes:

- **Aprendizaje:** Documentarse sobre cómo consumir datos generados por el sistema en JIRA.
- **Análisis de implementación:** Especificar las funcionalidades y requisitos de la integración así como, concebir la mejor manera para llevarla a cabo.
- **Implementación:** Programación del código que hará posible la integración de JIRA con el sistema de procesado de eventos.

- **Testeo:** La integración de JIRA se someterá a una serie de escenarios posibles en local para comprobar si cumple el comportamiento esperado.

Despliegue del sistema: Despliegue de las partes del sistema que aún no estén en producción. A nivel de recursos, esta tarea necesita los servidores necesarios para que el sistema esté totalmente desplegado.

Testeo del sistema en producción: Todo el sistema desplegado, incluida la integración de JIRA, será sometido a una serie de escenarios posibles para comprobar que los diferentes elementos del sistema se relacionan correctamente entre ellos y se producen los resultados esperados.

2.2. Tabla de tiempo

En la Tabla 2.1 se muestran, de forma general, las tareas a realizar y el tiempo que se va a emplear para realizar tales tareas, en la sección 2.4 se especificarán con más detalle.

Tarea	**Horas empleadas**
Aprendizaje	**150 horas**
Subsistema de recolección y envío de eventos	**90 horas**
Subsistema de recepción e ingesta de eventos	**90 horas**
Subsistema de transformación de eventos	**90 horas**
Integración con JIRA	**90 horas**
Despliegue del sistema	**15 horas**
Testeo del sistema en producción	**15 horas**
Total	**540 horas**

Tabla 2.1: Tiempo de realización esperado de las tareas

2.3. Restricciones

Las restricciones no intuitivas aparecen en las tareas a realizar dentro del desarrollo del sistema.

Por razones de diseño, es necesario empezar por el subsistema de recepción e ingesta de eventos ya que al ser la entrada al sistema de procesado de eventos puede marcar la manera en que se recolectan y envían eventos. Por lo que el subsistema de recepción ha de preceder en el tiempo al subsistema de recolección y envío.

El sistema de recolección y envío de eventos ha de preceder al sistema de transformación ya que depende de cómo se recolectan los datos hará falta un tipo concreto de sistema u otro.

La integración con JIRA no tiene sentido hasta que todos los subsistemas no se hayan llevado a cabo.

2.4. Diagrama de Gantt

En el Anexo A la Figura A.2 muestra el diagrama de Gantt de la planificación inicial, en él se especifican cada una de las tareas, su fecha de inicio y fin, su duración en horas y el coordinador o responsable de cada tarea. Se ven también las restricciones temporales de cada tarea.

Las tareas marcadas en rojo indican un riesgo alto a que causen posibles desviaciones. Las tareas marcadas en azul un riesgo medio a que causen posibles desviaciones.

Las tareas marcadas en verde marcan las reuniones de seguimiento con el director, el espacio temporal entre ellas es de quince días de media, estas reuniones empiezan después de la fase de aprendizaje y duran hasta la semana anterior de la entrega del proyecto. En tales reuniones se irán revisando aspectos claves del desarrollo del proyecto así como resolviendo posibles dudas.

Capítulo 3

Especificación de la arquitectura

3.1. Arquitectura del sistema

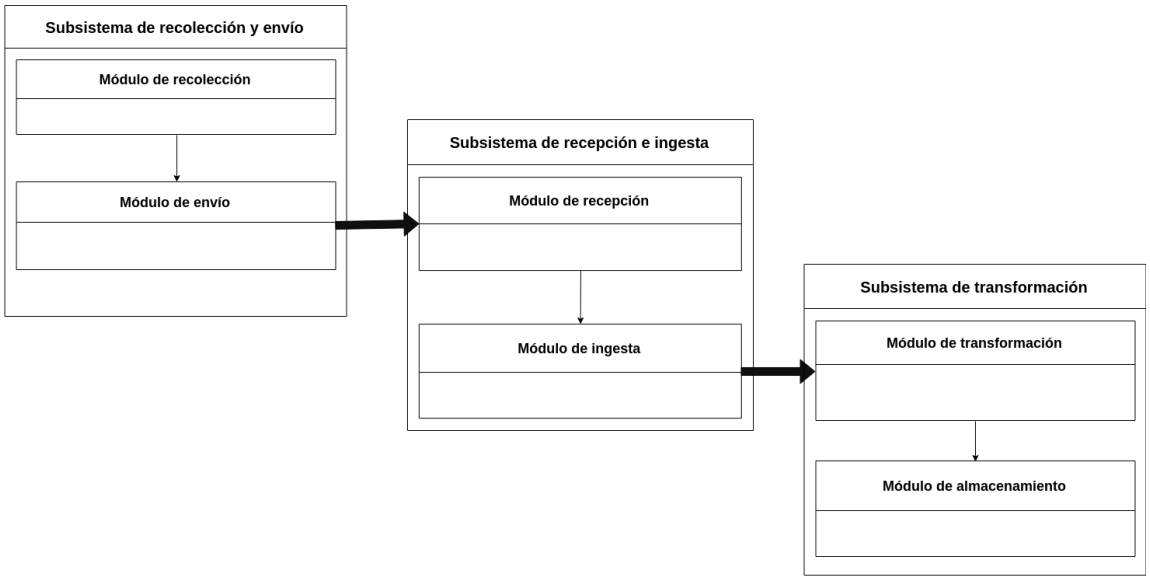


Figura 3.1: Vista parcial de la arquitectura del sistema

La arquitectura del sistema se divide en tres partes o subsistemas, cada subsistema tiene una tarea diferenciada dentro del sistema. En la figura 3.1 se puede ver una vista parcial de como se relacionan los subsistemas y los módulos del sistema. Las flechas indican el sentido del flujo de datos.

Los subsistemas son los siguientes:

- *Subsistema de recolección y envío*: Está compuesto de dos módulos, el de recolección y el de envío. El módulo de recolección se encarga de recoger los eventos en el dispositivo ubicuo. El módulo de envío se encarga de enviar los eventos recogidos por el módulo de recolección al módulo de recepción del subsistema de recepción e ingesta.

- *Subsistema de recepción e ingesta*: Está compuesto por dos módulos, el de recepción y el de ingesta. El módulo de recepción se encarga de centralizar la ingesta de eventos a un único punto de entrada y de recibir los eventos que le envía el módulo de envío. El módulo de ingesta se encarga de recibir los eventos del módulo de recepción y almacenarlos como mínimo hasta que el módulo de transformación los consuma.

- *Subsistema de transformación*: Está compuesto por dos módulos, el de transformación y el de almacenamiento. El módulo de transformación se encarga de transformar los eventos que consume del módulo de ingesta y de enviarlos al módulo de almacenamiento. El módulo de almacenamiento recibe los eventos transformados del módulo de transformación y los almacena.

La arquitectura propuesta cumple con las fases del proceso ETL. Los encargados de la Extracción son el subsistema de recolección y envío junto con el subsistema de recepción e ingesta. El encargado de la Transformación y la Carga es el subsistema de transformación.

Esta arquitectura está basada en la arquitectura Kappa[51], siendo el módulo de ingesta el que implemente la capa de almacenamiento de datos, el módulo de transformación el que implemente la capa de Stream Processing y el módulo de almacenamiento el que implemente la capa en la cual se sirven los datos.

Capítulo 4

Subsistema de recolección y envío

4.1. Arquitectura del subsistema

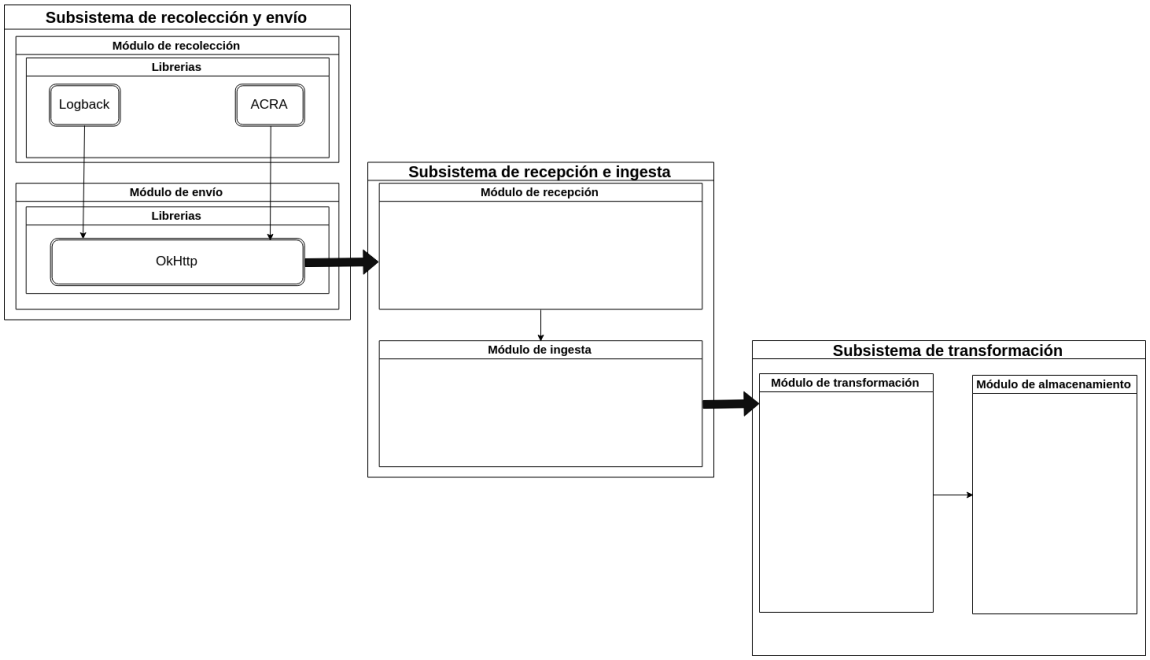


Figura 4.1: Vista general del sistema hasta el subsistema de recolección y envío

El subsistema de recolección y envío está formado por dos módulos, los cuales actúan en el dispositivo ubicuo. Esta es una división a nivel lógico y existe para reducir a problemas más simples el problema general, aunque nada impide que una

herramienta implemente dos módulos. En la figura 4.1 se puede ver como se relacionan los módulos y las herramientas del subsistema. Las flechas indican el sentido del flujo de datos.

4.1.1. Módulo de recolección

El módulo de recolección es el encargado de recoger los eventos en el dispositivo ubicuo, para que luego el módulo de envío pueda consumirlos.

En concreto para este proyecto los eventos que se recogen son logs y crashlogs. Los logs servirán para llevar a cabo la monitorización y los crashlogs el seguimiento de errores.

Para generar logs se han de explicitar en el código ahí donde el programador considere adecuado, los crashlogs se generan automáticamente cuando la aplicación termina inesperadamente.

Los logs se recolectan en tiempo de ejecución y se almacenan en el dispositivo ubicuo para que el módulo de envío los consuma cuando sea pertinente.

Los crashlogs se recolectan cuando la aplicación termina inesperadamente y no se almacenan en el dispositivo, sino que de forma inmediata el módulo de envío los consume. Se ha decidido que no se almacenen en el dispositivo puesto que, se considera que tienen más prioridad que los logs, y porque no se puede asegurar que se lleguen a almacenar en el dispositivo o que la aplicación pueda volverse a abrir después de que el crash haya sucedido. Así pues se invoca directamente al módulo de envío para que consuma tal crashlog.

4.1.2. Módulo de envío

El módulo de envío es el encargado de consumir los eventos que ha recolectado el módulo de recolección y enviar tales eventos al módulo de recepción del subsistema de recepción e ingesta.

El envío de los logs lo hace cuando la aplicación se cierra correctamente. Es entonces cuando consume los logs almacenados en el dispositivo y los envía al módulo de recepción.

El envío de los crashlogs lo hace cuando la aplicación se ha cerrado inesperadamente. Es entonces cuando consume el crash generado y lo envía al módulo de recepción.

4.2. Estructura del sistema

4.2.1. Módulo de recolección

Tal y como se aprecia en la Figura 4.1, el módulo de recolección está integrado por dos librerías. Una librería se encarga de recolectar los logs y otra los crashlogs.

La librería encargada de recolectar logs como mínimo ha de permitir que el programador defina logs en el punto del programa donde este considere oportuno, y que se pueda definir el destino de los logs.

La librería encargada de recolectar crashlogs como mínimo ha de identificar cuando se ha producido un cierre inesperado de la aplicación, recuperar el error que ha notificado el sistema sobre el cierre inesperado, formar un crashlog con la información recopilada y permitir definir el destino del crashlog.

4.2.2. Módulo de envío

Tal y como se aprecia en la Figura 4.1, el módulo de envío está integrado por una librería. Esta librería se encarga de enviar los logs y los crashlogs recolectados mediantes peticiones HTTP. Como mínimo esta librería ha de poder formar una petición HTTP conteniendo los eventos recolectados, y enviarla a un destino.

4.3. Herramientas utilizadas

Las herramientas utilizadas para llevar a cabo los módulos han sido librerías para Android.

4.3.1. Módulo de recolección

Para llevar a cabo la recolección de logs se ha utilizado la librería Logback[50] en su versión para Android. Esta librería permite definir en que punto del programa se quiere generar un log y redirigir la salida del log a diferentes destinos. Esta librería también nos permite editar el formato de salida del log así como la metainformación que se añade al log.

Para llevar a cabo la recolección de crashlogs se ha utilizado la librería ACRA[1]. Esta librería detecta cuando una aplicación ha acabado inesperadamente y genera un crashlog con información de valor para reconocer que puede haber pasado. Esta librería también permite redirigir la salida del crashlog a diferentes destinos, editar el formato de salida del crashlog y la metainformación que se añade al crashlog.

4.3.2. Módulo de envío

Para llevar a cabo el envío de logs y crashlogs se ha utilizado la librería OkHttp[47] que permite hacer peticiones HTTP[18]. Se ha escogido esta librería por la rapidez con la que se puede programar lo que se desea para este proyecto, a parte del uso eficiente que hace de los recursos del sistema.

4.4. Configuración del subsistema

4.4.1. Módulo de recolección

Del módulo de recolección se han configurado los siguientes factores:

Metadatos

Metadatos hace referencia a toda la información que se recolecta a parte de la información de valor del evento. La información de valor en el caso de los logs es el

mensaje que define el programador, en el caso de los crashlogs es el mensaje de error del crash.

De los logs se recogen metadatos porque se consideran útiles para entender el contexto del log, los metadatos que se recogen son:

- **Hora del evento**: Hora en la que se produce el evento
- **Tiempo que transcurre desde que se inició la aplicación**: Una resta entre la hora que se produce el evento y la hora que se inició la aplicación.
- **Nivel**: Los logs pueden tener varios niveles, nivel informativo, de debug, de error, y más niveles que se pueden definir.
- **Nombre del thread**: Nombre del thread que lanza el log para que sea recolectado.

De los los crashlogs se recogen metadatos porque se consideran útiles para entender el contexto del crashlog y poder solucionar el error cuanto antes, los metadatos que se recogen son:

- Versión de la aplicación
- Modelo del dispositivo
- RAM total del dispositivo
- RAM disponible en el momento del crash
- Versión de Android
- Tipo de CPU
- Tipos de CPU soportadas por el dispositivo

Formato

El formato escogido para representar tanto logs, como crashlogs ha sido un objeto JSON puesto que aporta el poder tratar el evento de manera fácil y para cumplir con los requisitos de formato que exige el módulo de recepción. Cada dato que se quiera incluir en el evento es un par clave-valor, donde la clave es fija y el valor depende del estado del dispositivo. Las claves de los crahslogs no se han podido definir, sino que ACRA define las que han de ser. En el caso de los logs sí que se han podido definir, para reducir el uso de red y de almacenamiento estas claves son tan solo un carácter, el módulo de transformación se configurará para que sea capaz de enriquecer los logs y de reconocer que significa cada carácter.

Destino

Destino hace referencia a dónde se envían los eventos una vez se han recolectado. Los logs no pasan directamente al módulo de envío, sino que se almacenan en un archivo de texto en el dispositivo ubicuo para que cuando la aplicación se cierre, el módulo de envío los consuma de ese archivo. Los crashlogs pasan directamente a disposición del módulo de envío una vez se recolectan. No se almacenan en persistencia.

4.4.2. Módulo de envío

Del módulo de envío se han configurado los siguiente factores:

- **Cuándo se hace el envío**: En el caso de los logs, el envío se hace una vez la aplicación se ha cerrado, se hace así para no interferir en el uso de red mientras la aplicación está corriendo. En Android es posible detectar cuando se va a cerrar una aplicación y ejecutar código.
  En el caso de los crashlogs el envío se hace justo cuando el crash sucede, se hace así ya que nada asegura que la aplicación pueda volverse a abrir después de que un crash haya sucedido. La librería ACRA nos permite llevar a cabo este comportamiento con los crashlogs.

- **Destino**: El destino tanto de los logs como de los crashlogs es el módulo de recepción, por lo que se ha configurado para que sean enviados ahí. Dentro del

módulo de ingesta, logs y crashlogs van a destinos lógicos diferentes, por lo que también se ha configurado su destino dentro del módulo de ingesta.

- **Cómo se hace el envío**: En el caso de crashlogs no hay más opción a que cada petición HTTP POST hacía el módulo de recepción contenga tan solo un crashlog. En el caso de los logs se puede enviar una petición HTTP POST por cada log o envíar una petición HTTP POST con todos los logs recolectados durante la sesión. Se ha escogido enviar una sola petición HTTP POST con todos los logs recolectados para no saturar al módulo de recepción atendiendo peticiones.

Capítulo 5

Subsistema de recepción e ingesta

5.1. Arquitectuta del subsistema

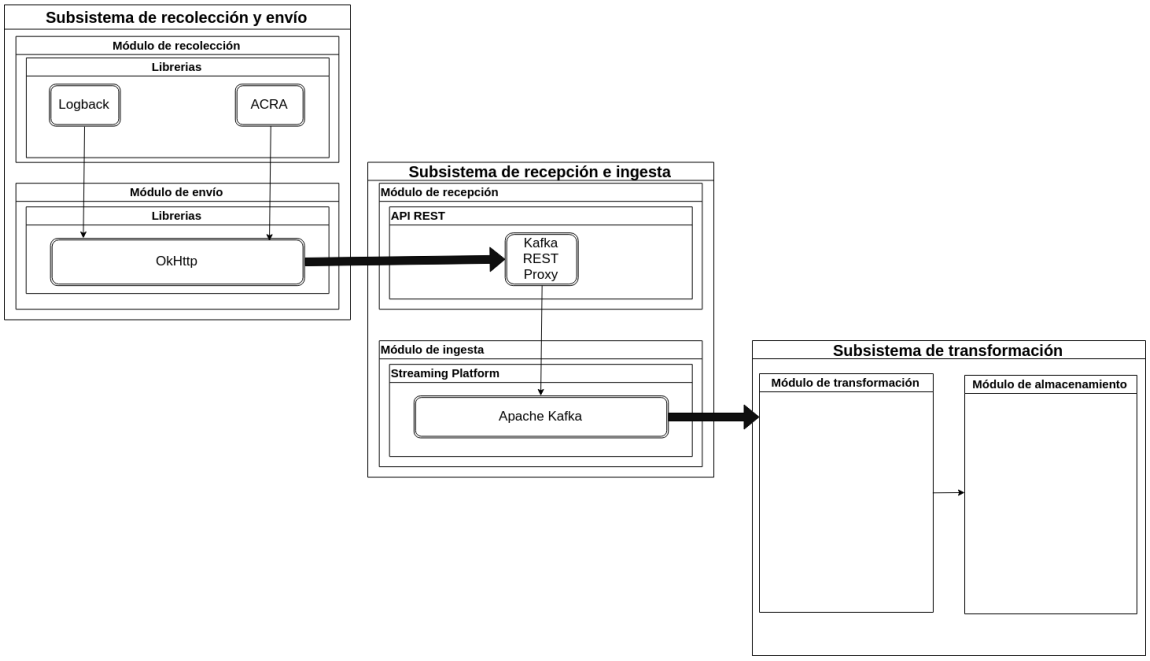


Figura 5.1: Vista general del sistema hasta el subsistema de recepción e ingesta

El subsistema de recepción e ingesta está formado por dos módulos, los cuales actúan fuera del dispositivo ubicuo. Esta es una división a nivel lógico y existe para reducir a problemas más simples el problema general, aunque nada impide

que una herramienta implemente dos módulos. En la figura 5.1 se puede ver como se relacionan los módulos y las herramientas del subsistema. Las flechas indican el sentido del flujo de datos.

5.1.1. Módulo de recepción

El módulo de recepción es el encargado de recibir los eventos que le envía el módulo de envío y de enviárselos al módulo de ingesta. Este módulo no almacena de forma persistente los eventos recibidos.

Se ha añadido este módulo para centralizar a un único punto de entrada y tener una metodología de envío universal para todos los dispositivos ubicuos de la empresa. De esta manera, también se facilita el que nuevos dispositivos ubicuos, diferentes a los que la empresa utiliza, puedan integrarse con el sistema.

5.1.2. Módulo de ingesta

El módulo de ingesta es el encargado de recibir los eventos del módulo de recepción y dejar a disposición los datos al módulo de transformación. Este módulo almacena tanto de forma persistente como volátil los eventos recibidos.

Se ha añadido un módulo entre el módulo de recepción y el módulo de transformación por cuatro razones:

1. Permitir el reprocesado de los datos.
2. No saturar al módulo de transformación.
3. No perder eventos si el módulo de transformación está caído.
4. Facilitar la integración de nuevos módulos.

Al añadir un módulo que almacene los datos sin procesar antes del módulo de transformación, se permite que el módulo de transformación pueda volver a consumir datos que ya había procesado antes. Tal módulo querría reprocesar los datos

por algún error humano en el programa que se encarga de transformarlos. Estos datos sin procesar no se almacenarán por siempre en el módulo de ingesta, sino que se irán eliminado del módulo siguiendo una política de limpieza establecida por el administrador del sistema.

Puesto que el módulo de ingesta sirve los datos para que sean consumidos, el módulo de transformación los consume cuando puede, de esta manera si estaba caído o estaba saturado, y puesto que módulo de ingesta almacena los datos de forma persistente, los consumirá cuando vuelva a estar arriba o pueda consumirlos.

Añadir este módulo de ingesta facilita la integración de nuevos módulos ya que transforma parcialmente los datos recibidos para que sean compatibles con los diferentes módulos que se le puedan acoplar. A parte, facilita el enrutamiento de los datos hacia los diferentes módulos que se le quieran acoplar.

5.2. Estructura del subsistema

5.2.1. Módulo de recepción

Tal como muestra la 5.1, el módulo de recepción lo integra una API REST que es capaz de recibir los eventos y enviarlos al módulo de ingesta. Se ha escogido una API REST, puesto que la comunicación mediante peticiones REST permite una fácil integración con los dispositivos ubicuos. Utilizar una API REST también ayuda a que el sistema se pueda integrar rápidamente con nuevos dispositivos ubicuos diferentes a los que usa actualmente la empresa.

5.2.2. Módulo de ingesta

Tal como muestra la 5.1, el módulo de ingesta lo integra una streaming platform, una especie de buffer que almacena los datos sirviéndolos para que otro módulo los consuma. Suple las funciones de un message broker[24] y le añade ciertas funcionalidades. Esta streaming platform es capaz de recibir los eventos y almacenarlos en memoria y en disco.

5.3. Herramientas utilizadas

5.3.1. Módulo de ingesta

Para el módulo de ingesta se ha escogido utilizar Apache Kafka[19], a parte de porque posee las funcionalidades requeridas, por las siguientes razones:

- **Compatibilidad**: Ofrece una gran compatibilidad con las posibles herramientas a utilizar en el módulo de transformación, aparte de con otras herramientas de otros posibles módulos que se quieran acoplar.
- **Documentación**: Existe una documentación extensa sobre la herramienta y la comunidad es bastante activa.
- **Confiabilidad**: Kafka es capaz de tener múltiples consumidores y productores subscritos. En caso de fallo de alguno de los nodos componentes de Kafka, es capaz de balancear automáticamente los consumidores y los productores a nodos componentes funcionales. Aparte, Kafka replica los datos que recibe entre los diferentes nodos componentes.
- **Durabilidad**: Los mensajes que recibe Kafka los almacena en disco, por lo que aparte de la replicación de datos que hace el propio Kafka, es fácil replicar los datos hacia otro sistema.
- **Escalabilidad**: Kafka es capaz de escalar rápido, de forma fácil y en caliente ya que es un sistema distribuido.
- **Rendimiento**: Debido al uso de los recursos que hace Kafka, es capaz de ofrecer un throughput elevado tanto a productores como consumidores.

5.3.2. Módulo de recepción

Para el módulo de ingesta se ha escogido utilizar Kafka Rest Proxy[13], a parte de porque posee las funcionalidades requeridas, la elección de esta herramienta ha dependido de la elección de la herramienta del módulo de ingesta, ya que se ha tenido que buscar una herramienta que sea integrable con Apache Kafka. La herramienta

escogida ha sido creada por los mismos creadores que Apache Kafka y la integración está completamente asegurada. A parte de la integración, esta herramienta nos aporta ciertos beneficios de confiabilidad y escalabilidad que serán más evidentes en su despliegue.

5.4. Configuración del subsistema

5.4.1. Módulo de recepción

En el Capítulo 7 se hablará con más detalle de que este módulo a parte de máquinas tiene un balanceador de carga que distribuye el trabajo entre las máquinas. Configurar ese balanceador de carga también forma parte de la configuración de este módulo. A ese balanceador se le ha tenido que decir que las peticiones REST que vengan por el puerto *8080*las redirija a las máquinas que forman el módulo de recepción.

A las máquinas que forman el módulo de recepción se le ha tenido que configurar el destino de los mensajes recibidos, es decir, que envíen los mensajes que reciben del módulo de envío al módulo de ingesta. También se le ha tenido que dotar de una id única para que el módulo de ingesta reconozca las máquinas.

5.4.2. Módulo de ingesta

El módulo de ingesta tiene una forma particular de almacenar los datos. Almacena los datos en *topics*, los *topics* son similares a las tablas en SQL. Se han creado dos *topics*, el *topic* logs, donde se almacenarán los logs, y el *topic* crashlogs donde se almacenarán los crashlogs. Los *topics* se dividen en particiones para explotar al máximo el paralelismo y permitir que diversas máquinas consuman diferentes particiones del mismo *topic*. Estas particiones se pueden replicar entre las máquinas con el fin de aumentar la durabilidad y la confiabilidad del sistema. Por lo que en resumen la configuración escogida de los *topics* es la siguiente:

- Topic *logs*, 3 particiones, 3 replicas de cada partición. De esta manera 3 máquinas pueden consumir los datos a la vez y pueden fallar 2 máquinas del cluster que aún los datos serán accesibles.

- Topic *crashlogs*, 3 particiones, 3 replicas de cada partición. De esta manera 3 máquinas pueden consumir los datos a la vez y pueden fallar 2 máquinas del cluster que aún los datos serán accesibles.

A parte, el sistema nos permite definir una política de limpieza para los datos almacenados. Se ha establecido que los datos permanezcan almacenados en el sistema de forma persistente una semana. Es un tiempo más que suficiente para detectar un mal procesado y tener que volver a reprocesar los datos.

Como en Kafka se puede forman un cluster de máquinas, a cada máquina se le ha tenido que configurar a qué cluster pertenece.

Capítulo 6

Subsistema de transformación

6.1. Arquitectura del subsistema

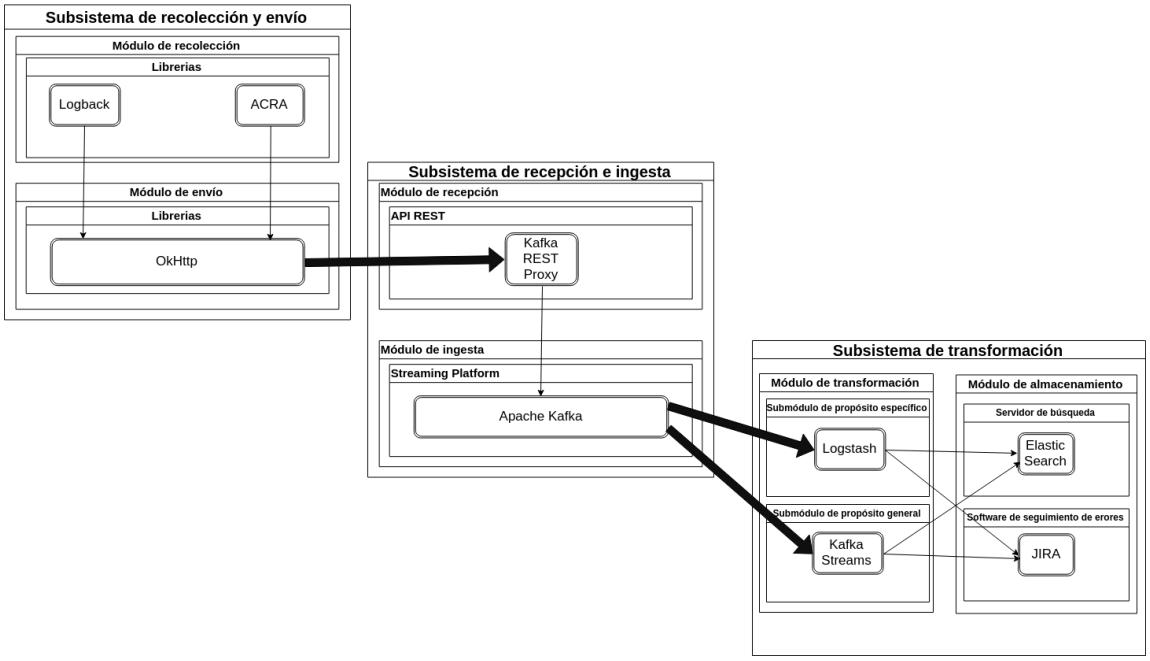


Figura 6.1: Vista general del sistema hasta el subsistema de transformación

El subsistema de transformación está formado por dos módulos, los cuales actúan fuera del dispositivo ubicuo. Esta es una división a nivel lógico y existe para reducir a problemas más simples el problema general, aunque nada impide que una herra-

mienta implemente dos módulos. En la figura 6.1 se puede ver como se relacionan los módulos y las herramientas del subsistema. Las flechas indican el sentido del flujo de datos.

6.1.1. Módulo de transformación

El módulo de transformación es el encargado de consumir los datos del módulo de ingesta, hacer las transformaciones deseadas a los datos y publicarlas en el módulo de almacenamiento. Este módulo no almacena de forma persistente los datos.

Este módulo es necesario ya que nos ayuda a crear información de valor con los eventos extraídos, a parte de generar la información en un formato que el módulo de almacenamiento acepte.

6.1.2. Módulo de almacenamiento

El módulo de almacenamiento es el encargado de recibir los datos transformados del módulo de transformación y almacenarlos para que puedan ser consumidos por otros sistemas o aplicaciones.

Este módulo es necesario ya que es el encargado de almacenar de forma persistente los datos transformados y sin él otros sistemas no podrían consumir los datos transformados.

6.2. Estructura del subsistema

La estructura de este subsistema puede variar mucho dependiendo de los datos que se quieran procesar, en nuestro casos son logs y crashlogs, por lo que la estructura utilizada ha de ser capaz de transformar de forma eficaz logs y crashlogs, pero la elección de la estructura no ha de condicionar el propósito del sistema. Aunque el sistema del proyecto ha de trabajar con eventos, se ha buscado la solución más general para tener lo más cercano a un sistema de propósito general, de esta manera se podrán procesar otro tipo de datos cuando sea necesario. Se ha de tener en cuenta que la elección del módulo de almacenamiento no se ha tomado en este proyecto, sino que este sistema se ha tenido que acoplar con un módulo de almacenamiento existente

en la empresa, por lo que la elección estructural del módulo de transformación se ha visto en parte condicionada por el módulo de almacenamiento existente.

6.2.1. Módulo de transformación

Para el módulo de transformación se ha buscado por un lado, que sea eficaz transformando logs y crashlogs y por otro que sea lo más general posible para soportar en el futuro la transformación de otro tipo de datos, por lo que la solución encontrada pasa por aprovechar la existencia de un módulo de ingesta extremadamente versátil y dividir en submódulos, que pueden colaborar entre ellos, el módulo de transformación. Así pues, los elementos a nivel estructural han de encajar en esta división.

Como se puede apreciar en la Figura 6.1, el módulo de transformación se divide en dos submódulos:

Submódulo de propósito general

El submódulo de propósito general es capaz de transformar todo tipo de datos, ya sean eventos u otra cosa. Este submódulo se ha añadido para dotar al sistema de generalidad transformando datos. Este submódulo es capaz de hacer transformaciones stateful como agregaciones, joins y windowing, o transformaciones stateless como filtering y mapping, todas ellas en tiempo real. A parte, este submódulo es fácilmente escalable, desplegable e integrable con el submódulo de ingesta. La solución escogida ha sido, en vez de una estructura pasiva a la que se le envían lotes de trabajo para que los procese, un solución activa en la que una aplicación con el uso de alguna librería sea capaz de consumir los datos del módulo de ingesta, transformarlos, y publicarlos en el módulo de almacenamiento. Por lo que a nivel estructural este submódulo es una aplicación que utiliza una librería de stream processing.

Submódulo de propósito específico

El submódulo de propósito específico es capaz de transformar un tipo de datos en concreto, en el caso de este proyecto eventos. Este módulo hace transformaciones que requieren una potencia de cálculo menor que las transformaciones que puede hacer

el submódulo de propósito general. Las transformación que principalmente efectúa es un enriquecimiento para que los datos sean aceptados en el módulo de destino o para que el submódulo de propósito general haga transformaciones más potentes con los datos ya enriquecidos.

Estos dos submódulos pueden cooperar entre ellos. Por ejemplo, los datos podrían pasar primero por el submódulo de propósito general aplicarles la transformación pertinente y luego pasarlos al submódulo de propósito específico enriquecerlos y pasarlos al módulo de almacenamiento.
Esta división en submódulos hace que la aplicación sea viable para todos los casos de uso, ya sea un caso de uso que requiera poca potencia de procesado o mucha potencia de procesado. Con esta división se ajustan bastante los costes a la complejidad del procesado, si el procesado es simple, el precio será bajo, si el procesado es complejo el precio será alto, por lo que no existe un umbral a partir del cual salga a cuenta hacer el procesado de los datos, siempre lo es. Es más, no es necesario utilizar los dos submódulos, quizás con las transformaciones que ofrece uno de los dos módulos es suficiente.
Para que la división en submódulos sea tan versátil es importante que el submódulo de propósito general se pueda adaptar a la complejidad del problema, es decir, que para problemas simples no utilice una cantidad elevada de recursos que hagan que ese procesado no salga a cuenta. No se habla del submódulo de propósito específico ya que este submódulo depende en gran medida del módulo de almacenamiento, por lo que habrán situaciones en las que este submódulo tenga que estar sobredimensionado por culpa del módulo de almacenamiento, pero se pueden compensar los costes gracias al submódulo de propósito general.

6.2.2. Módulo de almacenamiento

El módulo de almacenamiento nos venía dado por la empresa. La empresa ya utilizaba un módulo de almacenamiento en concreto y el sistema se ha de integrar con él. El módulo de almacenamiento lo integra una aplicación de seguimiento de errores y un servidor de búsqueda. Los dos elementos tienen bases de datos, pero

para acceder a ellas se ha de hacer a través de los puntos de entrada que disponen, API REST en este caso. El sistema se ha de integrar con este módulo.

6.3. Herramientas utilizadas

6.3.1. Módulo de transformación

Submódulo de propósito general

Para el submódulo de propósito general se ha decidido utilizar Kafka Streams[21] ya que a parte de cumplir las necesidades estructurales, por las siguientes razones:

- **Proporciona una abstracción de alto nivel para hacer las transformaciones**. Repercute en el tiempo necesario para programar la solución deseada.
- **Hace un Stream Processing real, no hace micro-batching**. Por lo que las latencias de procesado pueden ser muy bajas.
- **Se integra sin problemas con Kafka**. No se produce un cuello de botella entre las dos herramientas.
- **Es una librería de Java**. Una aplicación en Java que utiliza Kafka Streams es la que hace la transformación de los datos, por lo que la aplicación se podrá desplegar en máquinas que soporten Java, no es necesario desplegar un cluster para realizar las transformaciones, aunque nada impide desplegar uno. Es altamente escalable puesto que si se necesita más potencia de cálculo tan solo se ha de desplegar otra instancia de la aplicación en otra máquina o en una en la que ya está corriendo la aplicación.

Submódulo de propósito específico

Para el submódulo de propósito general se ha decidido utilizar Logstash puesto que se integra perfectamente con Kafka y Elastic Search, y hace una transformación específica para eventos, ofreciendo transformaciones bastante útiles para el caso de los eventos.

6.3.2. Módulo de almacenamiento

Las herramientas que componen el módulo de almacenamiento son Elastic Search, en el cual se almacenan logs y crashlogs, y JIRA, en el cual se alamacenan tan solo crashlogs. Estas herramientas ya estaban presentes en la empresa.

6.4. Configuración del subsistema

6.4.1. Módulo de transformación

La configuración de este módulo depende de la transformación que se quiera hacer a los datos. Para comprobar el funcionamiento del módulo y simplificar el problema, se hacen transformaciones simples que puedan ser demostrables el día de la demostración.

Para los logs, lo que se pretende es que se enriquezcan. En concreto, los logs entran como un objeto JSON, y las claves de este objeto son tan solo un carácter, lo que se quiere es reemplazar las claves por una palabra que identifique mejor que es cada parte del objeto JSON y luego publicarlo en Elastic Search. Para este caso haciendo uso del submódulo de propósito específico nos basta. Se ha configurado Logstash para que sea capaz de consumir los datos del topic *logs* de Kafka, transformarlos y publicarlos en Elastic Search.

Para los crashlogs, lo que se pretende es que cuando se produzca el mismo crashlog X veces durante un periodo de tiempo de Y segundos, se genere un ticket en JIRA que contenga los datos más relevantes de este crashlog. A parte, también se quiere que todos los crashlogs que se produzcan, se publiquen en Elastic Search. Para este caso se hace uso de los dos submódulos. El submódulo de propósito general se encarga de controlar si se produce el mismo crashlog X veces durante un periodo de tiempo de Y segundos, y de generar el ticket en JIRA. El submódulo de propósito específico se encarga de publicar todos los crashlogs en Elastic Search.

6.4.2. Módulo de almacenamiento

Este módulo ya está configurado por la empresa, por lo que la configuración que se va a hacer es una que permita ver que el sistema funciona como se espera el día de la presentación.

En JIRA se ha creado un proyecto, en el cual se publican los tickets con la información sobre los crashlogs. En Elastic Search se han creado dos índices, uno para los logs y otro para los crashlogs.

6.5. Integración con el entorno

El módulo de transformación se ha de integrar con un módulo de almacenamiento ya existente en la empresa, en concreto se ha de integrar con JIRA y con Elastic Search. De la integración con JIRA se espera que el sistema sea capaz de crear un ticket en JIRA con información obtenida de los eventos. De la integración con Elastic Search se espera que el sistema sea capaz de publicar información, obtenida de las alertas, en dos índices de Elastic Search con el objetivo de que la información publicada sea visible en Kibana.

Capítulo 7

Despliegue

7.1. Entorno

El entorno que se ha escogido para desplegar las partes del sistema que no residen en el dispositivo ubicuo ha sido Amazon Web Services (AWS)[1]. La razón es porque la empresa actualmente trabaja con este servicio y es más fácil integrarse con los módulos con los que se ha de integrar el sistema. AWS nos ofrece una serie de servicios que encajan con lo que se pretende desplegar.

El entorno que se ha escogido para desplegar las partes del sistema que residen en el dispositivo ubicuo ha sido Android. La razón es porque la empresa trabaja actualmente con dispositivos Android y tenía la necesidad de recolectar eventos en estos dispositivos.

7.2. Máquinas

El servicio que se ha escogido para desplegar las máquinas ha sido EC2[2], ya que ofrece diferentes máquinas virtuales con diferentes características a nivel de hardware que pueden adaptarse notablemente a las necesidades de las diferentes partes del sistema.

[1] https://aws.amazon.com

7.2.1. Subsistema de recolección y envío

Para desplegar este subsistema tan solo se necesita un dispositivo Android. La versión mínima de Android ha de ser la 4.0 para que las librerías funcionen correctamente. El dispositivo ha de tener acceso a Internet para que pueda comunicarse con el subsistema de recepción e ingesta.

7.2.2. Subsistema de recepción e ingesta

Módulo de recepción

Este módulo hace un uso intensivo de CPU y red[12]. El uso de red es elevado, puesto que ha de ser capaz de soportar diversas conexiones concurrentes en las cuales se pueden transmitir gran cantidad de datos. El uso de CPU es intensivo ya que cada petición POST la atiende un thread. Kafka Rest Proxy explota el paralelismo de la CPU, a más cores, más peticiones podrá atender. El uso de RAM que hace este módulo es modesto y con que pueda soportar 1GB de heap es suficiente. El uso de disco es mínimo, ya que no almacena ninguna información en ellos.

La máquina escogida para desplegar este módulo ha sido la c5.2xlarge, en la Tabla 7.1 se pueden ver sus recursos hardware.

Máquina	**c5.2xlarge**
CPUs	8 cores
RAM	16.0 GiB
Rendimiento de red	Hasta 10 Gbps
HDD tamaño	30 GiB
HDD IOPS máximos	3000 IOPS

Tabla 7.1: Recursos hardware de la máquina virtual c5.2xlarge

Para asegurar la confiabilidad del módulo y de paso aumentar el número de cores del sistema, se ha decidido desplegar 3 máquinas c5.2xlarge y un balanceador de carga que distribuya las conexiones entre las máquinas. Para desplegar el balanceador de carga se ha escogido utilizar el servicio que presta AWS de Aplication Load

Balancer[4].
Con tal despliegue se puede soportar que caigan dos máquinas y aún el módulo podrá continuar haciendo su trabajo. Si ninguna máquina está caída se tendrán 24 cores y se podrán soportar conexiones de hasta 30 Gbps de tráfico agregado.

Módulo de ingesta

Este módulo hace un uso intensivo de RAM, red y disco. Para que Apache Kafka pueda funcionar necesita un orquestador que dirija y balancee los diferentes nodos (brokers) de Apache Kafka. Este orquestador es Apache Zookeeper. Apache Kafka y Apache Zookeeper hacen un uso diferente de los recursos, para no sobredimensionar las máquinas y reducir costes, Kafka y Zookeeper se desplegarán en máquinas con diferentes recursos.

El uso que hace Kafka de los recursos es el siguiente[11]:

- **CPU**: Hace un uso ligero de CPU, si se ha de elegir entre CPUs rápidas o con muchos cores, mejor una con muchos cores.
- **RAM**: Hace un uso intensivo de RAM, menos de 16 GB es improductivo.
- **Red**: Hace un uso intensivo de red, menos de 1 GbE es improductivo.
- **HDD**: Hace un uso intensivo de disco, se necesita espacio dependiendo del volumen de datos a tratar y un throughput elevado.

El uso que hace Zookeeper de los recursos es el siguiente[14]:

- **CPU**: Hace un uso ligero de CPU.
- **RAM**: Hace un uso ligero de RAM, menos de 8 GB es improductivo.
- **Red**: Hace un uso intensivo de red, menos de 1 GbE es improductivo.
- **HDD**: Hace un uso intensivo de disco, se necesita un throughput elevado.

Con tales premisas, la máquina escogida para desplegar Apache Kafka ha sido la r4.xlarge, en la Tabla 7.2 se pueden ver sus recursos hardware. La máquina escogida para desplegar Apache Zookeeper ha sido la r4.large, en la Tabla 7.1 se pueden ver sus recursos hardware.

Máquina	**r4.large**	**r4.xlarge**
CPUs	2 cores	4 cores
RAM	15.25 GiB	30.5 GiB
Rendimiento de red	Hasta 10 Gbps	Hasta 10 Gbps
HDD tamaño	30 GiB	500 GiB
HDD IOPS máximos	3000 IOPS	3000 IOPS

Tabla 7.2: Recursos hardware de la máquina virtual r4.large y r4.xlarge

Zookeeper necesita un mínimo de 3 máquinas para funcionar, por lo que se han desplegado 3 máquinas r4.large. Kafka no tiene un número mínimo de máquinas para funcionar, pero para aumentar la confiabilidad, la durabilidad y el rendimiento, se han desplegado 3 máquinas r4.xlarge.

7.2.3. Subsistema de transformación

El despliegue del subsistema de transformación es particular puesto que un módulo ya está desplegado en la empresa. Para dar completud al trabajo, en aquellos módulos que ya estén desplegados, se hablará de las máquinas que se desplegarán para hacer la demostración el día de la lectura del trabajo.

Módulo de transformación

Al estar dividido el módulo de transformación en dos submódulos y encargarse de tareas diferentes, el uso de los recursos es diferente en los dos submódulos.
Para el submódulo de propósito específico se ha de desplegar Logstash, en la documentación de la herramienta no se deja del todo claro los requerimientos hardware[17], pero por su naturaleza se ha supuesto que hace un uso intensivo de CPU, RAM y red, y un uso ligero de disco. De momento se ha desplegado en una r4.xlarge, en la Tabla 7.2 se puede ver sus recursos hardware. Como trabajo futuro, se ha monitorear

el rendimiento de este submódulo para ver el uso de los recursos que hace, y cambiar si fuera necesario la máquina utilizada.
Para el submódulo de propósito general se ha de desplegar la aplicación Java que utiliza Kafka Streams. El despliegue de máquinas para este submódulo depende mucho del volumen de datos que se hayan de procesar, para la demostración se va a suponer un volumen de datos pequeño, tan solo para verificar que la solución funciona. La máquina escogida es una t2.micro en la Tabla 7.3 se pueden ver sus recursos hardware.

Máquina	**t2.micro**
CPUs	1 cores
RAM	1.0 GiB
Rendimiento de red	Moderado
HDD tamaño	30 GiB
HDD IOPS máximos	3000 IOPS

Tabla 7.3: Recursos hardware de la máquina virtual t2.micro

Se han desplegado 3 máquinas r4.xlarge para el submódulo de propósito específico, con el objetivo de distribuir la carga entre los servidores y aumentar la potencia de procesado.
Se ha desplegado 1 máquina t2.micro para el submódulo de propósito general, con el objetivo de comprobar la solución propuesta.

7.2.4. Módulo de almacenamiento

Este módulo ya está desplegado en la empresa, por lo que el despliegue que se ha hecho es exclusivo para la demostración. Por un lado se ha desplegado un cluster de Elastic Search y una máquina con Kibana, y por otro una máquina con JIRA.

Para el despliegue del cluster de Elastic Search se han utilizado máquinas parecidas a las que utiliza la empresa. La máquina escogida es una r4.xlarge, en la Tabla 7.2 se puede ver sus recursos hardware. Para el despligue de Kibana se ha utilizado una t2.micro, ya que JIRA no consume grandes recursos, en la Tabla 7.3 se puede ver los recursos hardware de la máquina escogida.

Para el despligue de JIRA se va a utilizar una t2.small ya que JIRA como mínimo requiere 2 GiB de RAM para funcionar fluido. En la Tabla 7.4 se puede ver los recursos hardware de la máquina escogida.

Máquina	**t2.small**
CPUs	1 cores
RAM	2.0 GiB
Rendimiento de red	Moderado
HDD tamaño	30 GiB
HDD IOPS máximos	3000 IOPS

Tabla 7.4: Recursos hardware de la máquina virtual t2.small

Se van a desplegar 3 máquinas r4.xlarge para Elastic Search ya que se quiere comprobar el correcto funcionamiento del sistema cuando Elastic Search está distribuido en más de una máquina. Se va desplegar 1 máquina t2.micro para Kibana. Se va a desplegar una máquina t2.small para JIRA.
Estas máquinas no aparecen en el presupuesto hardware de la Sección 10.1.2 del Capítulo 10 puesto que estos elementos ya existen en la empresa y no se han de contar como nuevos. El despliegue de estas máquinas es tan solo para la demo que se hará el día de la lectura del trabajo.

7.2.5. Resumen de servicios a desplegar

Módulo	**Aplicación**	**Servicio**	**Detalle**	**Número**
Recepción	Kafka Rest Proxy	Amazon EC2	c5.2xlarge	3
Recepción	Kafka Rest Proxy	Elastic Load Balancing	Application Load Balancer	1
Ingesta	Apache Kafka	Amazon EC2	r4.xlarge	3
Ingesta	Apache Zookeeper	Amazon EC2	r4.large	3
Transformación	Logstash	Amazon EC2	r4.xlarge	3
Transformación	Kafka Streams	Amazon EC2	t2.micro	1
Almacenamiento	Elastic Search	Amazon EC2	r4.xlarge	3
Almacenamiento	Kibana	Amazon EC2	t2.micro	1
Almacenamiento	JIRA	Amazon EC2	t2.small	1

Tabla 7.5: Recursos de servicios a desplegar

En la Tabla 7.5 se puede ver el resumen de los servicios a desplegar en AWS. La columna *Módulo* nos dice el módulo del sistema que se despliega, la columna *Aplicación* nos marca la aplicación que se va a desplegar, la columna *Servicio* no dice el servicio de AWS que se va a utilizar, la columna *Detalle* nos informa sobre que servicio en concreto se va a desplegar y la columna *Número* no dice el número de instancias del *Servicio* que se van a desplegar. Cada fila corresponde a los servicios a desplegar.

7.2.6. Visión general del despliegue

En la Figura 7.1 se puede ver el como queda el despliegue de red de las máquinas, y una leyenda con el significado de los iconos. Para interconectar las diferentes máquinas, AWS son ofrece el servicio Amazon Virtual Private Cloud (Amazon VPC), esta VPC es una red virtual que interconecta las máquinas virtuales que se han desplegado, funciona parecido a una red local tradicional. En la Figura 7.1 se puede ver como existen 3 subredes que residen en 3 Availability Zones diferentes. AWS trabaja con los conceptos de región y zonas de disponibilidad. La VPC reside en una región, una región es una porción geográfica de un país. Las zonas de disponibilidad son diferentes data centers dentro de la región. En el caso de la figura, el sistema reside en una zona y está disperso entre los diferentes data centers de la zona. Se ha hecho así para aumentar la confiabilidad del sistema ya que tal y como se han configurado los diferentes módulos, el sistema es tolerante a caídas de servidores, y desplegarlos en diferentes zonas de disponibilidad disminuye la posibilidad de que caigan dos máquinas a la vez del mismo módulo.

La Figura 7.1 también muestra que la VPC está encabezada por un Aplication Load Balancer, este balanceador de carga es el que recibe las peticiones del dispositivo ubicuo y distribuye la carga entre los 3 Kafka Rest Proxy nodes. Por último, la Figura 7.1 muestra en la Subnet 3 una Internet Gateway, se ha añadido para que las máquinas de JIRA y de Kibana sean accesibles desde Internet.

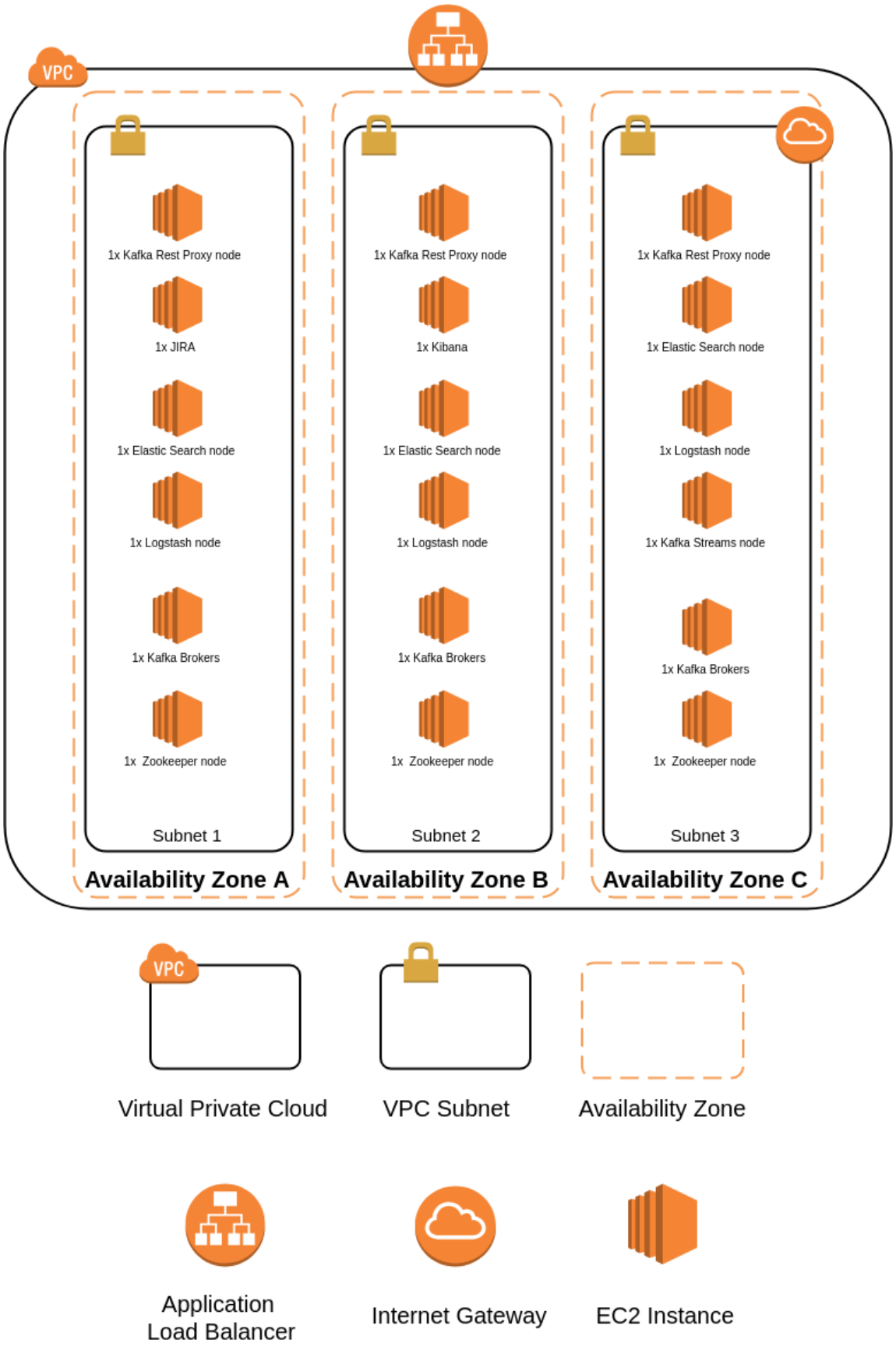


Figura 7.1: Networking del despliegue en AWS

Capítulo 8

Metodología y rigor

8.1. Método de trabajo

Se ha utilizado un modelo de desarrollo en cascada [33], ya que es una planificación sencilla, fácil de entender y utilizar por alguien que no está acostumbrado a trabajar siguiendo un método concreto. Para el éxito de este proyecto utilizando esta metodología, es muy importante la documentación, cosa que es positiva ya que este proyecto luego será utilizado en una empresa y ayudará a que los desarrolladores de la empresa tengan referencias e información sobre diferentes aspectos del proyecto. Al tener desde un principio la mayoría de requisitos del proyecto definidos, era poco probable que surgieran necesidades imprevistas, esto sumado al hecho de que el desarrollo del proyecto lo ha llevado a cabo una única persona y por lo tanto no ha tenido que sincronizarse con otros trabajadores, han hecho que el modelo cascada sea uno de los más adecuados para este proyecto.

A parte del uso del modelo de desarrollo cascada, la comunicación con el director del trabajo ha sido frecuente, ya sea para hablar de aspectos generales del proyecto, o de aspectos más concretos. La comunicación ha sido vía email o en persona. Se pactaron reuniones con el director cada quince días para comprobar los avances del proyecto.

8.2. Herramientas de seguimiento

Para el seguimiento del desarrollo del software se ha utilizado el sistema de control de versiones Git y estas versiones se almacenan en un repositorio remoto, se ha utilizado GitHub como repositorio remoto.

Para el seguimiento del diseño e implementación del sistema se ha generado documentación que se publicará en el JIRA de la empresa. Para asegurar que el diseño y la implementación es aparentemente correcta se ha consultará con el director antes de publicar la documentación en JIRA.

Para la generación de documentación de seguimiento se ha utilizado la suite ofimática LibreOffice, también LaTeX y se publicará en JIRA bajo el formato PDF.

8.3. Métodos de evaluación

Los métodos de evaluación corresponderían a la fase de verificación del desarrollo en cascada.

La parte de implementación del proyecto se puede dividir en las siguientes fases:

- Subsistema de recolección y envío de eventos
- Subsistema de recepción e ingesta de eventos
- Subsistema de transformación de eventos
- Integración con JIRA y ES Stack

Se ha aplicado el desarrollo en cascada de forma independiente en cada fase del proyecto por lo que se obtienen cuatro fases de verificación.

La fase de verificación del subsistema de recolección y envío de eventos consiste en la creación de una sencilla aplicación que integra las librerías escogidas en la fase de diseño con el fin de que recolecte y envíe eventos a un servidor desplegado en un entorno de pruebas. Tal servidor consiste en una máquina virtual que simula ser el subsistema de recepción e ingesta de eventos. Se ha modificado todo aquello de la aplicación que era erróneo hasta que se ha conseguido el resultado esperado en

el servidor. Una vez se ha conseguido el resultado esperado, se han exportado las configuraciones a los entornos de producción.

La fase de verificación del subistema de recepción e ingesta de eventos consiste por un lado en configurar en una máquina virtual el módulo de recepción para que sea capaz de recibir enventos y pasarlos al módulo de ingesta. Por otro lado se ha de configurar en otra máquina virtual el módulo de ingesta de eventos para que sea capaz de comunicarse con el módulo de recepción de eventos y almacenar de forma parcial los eventos. Una vez las dos máquinas virtuales se comuniquen entre si, y los mensajes enviados al módulo de recepción se almacenen en el módulo de ingesta, se ha considerado que la configuración principal era la correcta y se ha procedido a exportar tal configuración al entorno de producción.

La fase de verificación de verificación del subsistema de transformación de eventos, consiste en configurar en local una máquina virtual con el subsistema de recolección y envío de eventos a la cual se han ido enviando eventos, tales eventos van a ser consumidos por la aplicación de procesado que se ha programado y por Logstash, se ha redirigido la salida de tales programas a la consola para verificar que la transformación es la correcta, una vez se han obtenido los datos esperados, se ha procedido a desplegar las configuraciones en entornos de producción.

La fase de verificación de la integración con JIRA y ES Stack consiste en montar en local los tres subsistemas, un nodo con JIRA y un nodo con toda la ES Stack, refinar las configuraciones de los módulos hasta que se consiga, recolectar, enviar, recibir, ingerir, transformar y cargar un evento en JIRA y en la ES Stack. En JIRA se tendrá que generar un ticket con información sobre el evento. En ES Stack, Logstash ya ha hecho una transformación del evento, lo pasará a ElasticSearch, y Kibana será capaz de consumir el evento apuntando a ElasticSearch. Una vez se ha conseguido el comportamiento deseado, se han exportado las configuraciones a los entornos de producción.

Una vez han estado todos los módulos corriendo en producción, también se ha testeado que se cumpla el comportamiento esperado. Para ello se han generado una serie de eventos, los cuales han de seguir el pipeline hasta llegar a JIRA y Kibana. Cuando JIRA y Kibana han mostrado lo deseado se ha concluido el testeo.

Además de tales pruebas y para asegurar el correcto avance del diseño del pro-

yecto, se ha mantenido un contacto frecuente con el director para que vaya dando su visto bueno. Todas las pruebas y tests las ha llevado a cabo el desarrollador del proyecto.

Capítulo 9

Planificación final

9.1. Cambios en la planificación inicial y en la gestión económica

Respecto la planificación inicial han habido cambios en la duración del proyecto, ha pasado de durar 540 horas a 420 horas en total. Esta reducción de horas ha sido debida a que la dificultad con la que se había previsto algunas fases del proyecto ha sido menor. Las fases que han reducido sus horas son las siguientes:

- **Subsistema de recolección y envío de eventos** de 90 horas a 66, puesto que la integración de las librerías se ha llevado a cabo sin muchas complicaciones.
- **Subsistema de transformación de eventos** de 90 horas a 60, puesto que las herramientas con las que se ha desarrollado dicha transformación no tenían una curva de aprendizaje tan pronunciada como la que se previó.
- **Integración con JIRA y ES Stack** de 90 horas a 30, puesto que tanto JIRA como ES están pensados para que su integración sea rápida y fácil.
- **Despliegue del sistema** de 18 horas a 6, puesto que la mayoría de elementos ya se habían desplegado en las fases anteriores.
- **Testeo del sistema en producción** de 18 horas a 12, puesto que gran parte de las pruebas ya se habían realizado en las fases anteriores.

La tarea *Integración con JIRA y ES Stack* en el plan inicial tan solo era *Integración con JIRA*, en la planificación final también ha habido una integración con ES Stack.

La reducción de tiempo del proyecto también repercute positivamente en el coste del proyecto, ya que gran parte del coste va ligado a las horas de proyecto. Aunque se cometió un fallo en el cálculo del presupuesto inicial, esta reducción de horas, más la corrección de dicho error ha hecho que el coste total del proyecto disminuya de 33.499,55 € a 24.297,26 €.

En los capítulos 9 y 10 se expone la nueva planificación y los nuevos presupuestos contando esta reducción de horas. En el Anexo A se encuentra el diagrama de Gantt definitivo en la Figura A.1, así como el diagrama de Gantt que se planteó al inicio en la Figura A.2.

9.2. Tabla de tiempo

En el Tabla 9.1 se muestran, de forma definitiva, las tareas a realizadas y el tiempo que se ha empleado para realizar tales tareas, en la Sección 9.5 se especificarán con más detalle.

Tarea	**Horas previstas**	**Horas empleadas**
Aprendizaje	150 horas	**150 horas**
Subsistema de recolección y envío de eventos	90 horas	**66 horas**
Subsistema de recepción e ingesta de eventos	90 horas	**90 horas**
Subsistema de transformación de eventos	90 horas	**60 horas**
Integración con JIRA y ES Stack	90 horas	**30 horas**
Despliegue del sistema	15 horas	**6 horas**
Testeo del sistema en producción	15 horas	**12 horas**
Horas totales del proyecto	540 horas	**420 horas**

Tabla 9.1: Tiempo de realización esperado de las tareas

9.3. Restricciones

Las restricciones no han cambiado con respecto las restricciones del plan inicial que se encuentran en la Sección 2.3 del Capítulo 2.

9.4. Recursos

9.4.1. Recursos humanos

Se ha dispuesto de un desarrollador que es quién ha llevado a cabo el proyecto. El desarrollador de este proyecto ha asumido las tareas de jefe de proyecto, consultor y Site Reliability Engineer (SRE) [49]. El tiempo en horas por semana que ha dedicado el desarrollador al proyecto ha sido de 30 horas por semana. Es una medida aproximada ya que ha variado por la influencia de diversos factores tales como la carga de trabajo que tenga ha tenido desarrollador por causas externas al proyecto.

También se ha dispuesto de un director, el cual podrá ha sido consultado por el desarrollador.

9.4.2. Recursos físicos

Lenovo Y700 Computadora donde se ha desarrollado el proyecto y se han hecho las pruebas parciales antes de desplegar el sistema. Se ha utilizado también para escribir la memoria.

Xiaomi MI A1 Celular con el que se han hecho las pruebas de recolección y envío de eventos.

GitHub Servidor remoto de control de versiones. Se ha utilizado para almacenar las diferentes versiones del software que se ha desarrollado.

JIRA Herramienta para administración de tareas de un proyecto. Se publicará documentación de los diversos módulos en ella.

Servidores de producción En la Sección 10.1.2 se especifican las máquinas necesarias así como el coste de utilizarlas.

9.5. Diagrama de Gantt

En el Anexo A se encuentra el diagrama de Gantt final (Figura A.1) donde se especifican cada una de las tareas, su fecha de inicio y fin, su duración en horas y el coordinador o responsable de cada tarea. Se ven también las restricciones temporales de cada tarea.

Las tareas marcadas en rojo indican un riesgo alto a que causen posibles desviaciones. Las tareas marcadas en azul un riesgo medio a que causen posibles desviaciones.

Las tareas marcadas en verde marcan las reuniones de seguimiento con el director, el espacio temporal entre ellas es de quince días de media, estas reuniones empiezan después de la fase de aprendizaje y duran hasta la semana anterior de la entrega del proyecto. En tales reuniones se ha ido revisando aspectos claves del desarrollo del proyecto así como resolviendo posibles dudas.

Capítulo 10

Gestión económica final

10.1. Gastos directos

10.1.1. Presupuesto recursos humanos

El desarrollador de este proyecto ha asumido las tareas de jefe de proyecto, consultor y Site Reliability Engineer (SRE) [49]. Aún siendo así, los cálculos están hechos basados en precios actuales de mercado y diferenciando cada rol. En el Tabla 10.1 se ven tales cálculos.

Con relación a las tareas expuestas en el diagrama de Gantt A.1 de A, al jefe de proyecto le corresponden las tareas de aprendizaje (GEP también está incluido en esta tarea), al consultor le corresponden las tareas de delimitación y al SRE las restantes.

Rol	**Horas**	**Precio por hora**	**Precio total**
Jefe de proyecto	192 h	27 €/h [26]	5.184 €
Consultor	42 h	20 €/h [25]	840 €
SRE	258 h	34 €/h [27]	8.772 €
Total			**14.796 €**

Tabla 10.1: Presupuesto recursos humanos

Presupuesto total recursos humanos = 14.796 €

10.1.2. Presupuesto recursos hardware

Para llevar a cabo el proyecto y como se mostraba en la sección 9.4.2, se han utilizado una serie de recursos físicos. En el caso del hardware se han utilizado cuatro elementos, la computadora Lenovo Y700, el celular Xiaomi MI A1, el servidor remoto GitHub y los servidores de producción.

Es interesante comentar que para los servidores de producción se ha utilizado Amazon Web Service (AWS) [7], por lo que no se paga por la adquisición de los servidores sino por su uso por hora, se ha supuesto para hacer los cálculos que durante el desarrollo del proyecto se han utilizado todas las horas. No se contará nada más a parte de su coste por horas. Como el precio es bajo demanda, no se ha calculado el tiempo de amortización de los servidores de producción ya que el coste del hardware por hora real ya es el que se paga.

En el Tabla 10.2 se muestran las máquinas a utilizar con el gasto total contando que están siempre encendidas a partir de la etapa Desarrollo del sistema.

Instancia	**Número de instancias**	**Precio por hora**	**Horas**	**Precio total**
r4.xlarge	6	0,21 €/h [6]	270 h	342,9 €
r4.large	3	0,12 €/h [6]	270 h	97,2 €
c5.2xlarge	3	0,33 €/h [6]	270 h	267,3 €
t2.micro	1	0,01 €/h [6]	270 h	2,7 €
Total				**710,1 €**

Tabla 10.2: Presupuesto AWS

En el caso de GitHub, se ha utilizado la capa gratuita, por lo que tampoco se ha calculado la amortización.

En el Tabla 10.3 se muestra el presupuesto final de los recursos hardware con el cálculo de la amortización. Para calcular la amortización en la Tabla 10.3 como en la Tabla 10.4 se han hechos los siguientes supuestos y utilizado la siguiente fórmula:

$$Amortización = Hp * \frac{Pp}{Av * 365 * Hd}$$

$Hp = Horas\ del\ duración\ del\ proyecto.$

$Pp = Precio\ del\ producto.$

$Av = Años\ de\ vida\ útil\ del\ producto.$

$Hd = Horas\ al\ día\ que\ se\ utiliza\ el\ producto\ (suponemos\ siempre\ 8h).$

Producto	**Unidades**	**Precio**	**Vida**	**Amortización**
Lenovo Y700	1	1.099 € [5]	3 años	52,69 €
Xiaomi MI A1	1	229 € [53]	3 años	10,98 €
GitHub	-	0 €	-	0 €
Servidores producción	-	710,1 €	-	710,1 €
Total				**773,77 €**

Tabla 10.3: Presupuesto recursos hardware

Presupuesto total recursos hardware = 773,77 €

10.1.3. Presupuesto recursos software

El único software del que se ha tenido que pagar licencia es JIRA, necesario para la tarea Integración con JIRA y Elastic Search del diagrama de Gantt de la sección 9.5. La licencia que se va a comprar es una en que nosotros hemos de almacenar JIRA en servidores propios de la empresa y pueden utilizar el software hasta 10 personas. En la Tabla 10.4 se muestra el cálculo del presupuesto de los recursos software.

Producto	**Unidades**	**Precio**	**Vida**	**Amortización**
JIRA	1	8 € [9]	3 años	0,38 €
Total				**0,38 €**

Tabla 10.4: Presupuesto recursos software

10.1.4. Total presupuesto gastos directos

En la Tabla 10.5 se muestra el presupuesto total de los gastos directos del proyecto.

Tipo	**Coste**
Presupuesto recursos humanos	14.796 €
Presupuesto recursos hardware	773,77 €
Presupuesto recursos software	0,38 €
Total	**15.570,15 €**

Tabla 10.5: Presupuesto total gastos directos

10.2. Gastos Indirectos

Los principales gastos indirectos del proyecto han sido el consumo eléctrico y la conexión a Internet.

Para el cálculo del consumo eléctrico se han utilizado los siguientes datos:

- **Precio del kWh en España:** 0.10813 €/kWh (Como precio de consumo eléctrico se ha tomado el precio medio del 18 de marzo de 2018 [45]).
- **Horas de consumo eléctrico:** 420h.
- **Potencia consumida:** 60 + 6*58 = 408 W (Potencia del portátil 60 W, Potencia de los 6 fluorescentes del despacho 58 W).

$$0,408kW * 420h * 0,10813€/kWh = 18,5€$$

Para el cálculo de la conexión a Internet se han utilizado los siguientes datos:

- **Precio fibra óptica 100mb Movistar:** 45€/mes.
- **Horas de duración del proyecto:** 420h.

- **1 mes:** 30 días.
- **1 día:** 8h hábiles.

$$Coste\ de\ Internet = 420h * \frac{45€/mes}{30\ días * 8h} = 78,75€$$

10.2.1. Total presupuesto gastos indirectos

En la Tabla 10.6 se ve el presupuesto total de los gastos indirectos del proyecto.

Producto	**Coste**
Electricidad	18,5 €
Internet	78,75 €
Total	**97,25 €**

Tabla 10.6: Presupuesto total gastos indirectos

10.3. Total presupuesto gastos directos e indirectos

En la Tabla 10.7 se ve el presupuesto total de gastos directos e indirectos del proyecto.

Tipo	**Coste**
Gastos directos	15.570,15 €
Gastos indirectos	97,25 €
Total	**15.667,40 €**

Tabla 10.7: Presupuesto total de gastos directos e indirectos

10.4. Contingencia

Dado que el desarrollador del proyecto es la primera vez que se ha enfrentado a muchas de las tecnologías utilizadas para el desarrollo del proyecto y aunque las tareas están bien detalladas, se ha destinado un 20 % del valor total del presupuesto a contingencia. En la Tabla 10.8 se muestra lo que supone esta contingencia.

Contingencia	**Total**
20 %	**3.133,48 €**

Tabla 10.8: Calculo de contingencia

10.5. Control de gestión

Se han localizado dos posibles imprevistos, su solución y el gasto que supondrían. A parte de los dos imprevistos encontrados, podrían ocurrir otros, pero ninguna de ellos supondría un gasto de recursos hardware. El gasto extra a nivel de recursos humanos ya se ha contemplado en la Tabla 10.9 y el gasto a nivel de software sería nulo ya que se buscarían alternativas gratuitas si hiciera falta. La Tabla 10.9 muestra con que porcentaje pueden ocurrir los imprevistos y el impacto económico para el proyecto.

Imprevisto	Solución	Probabilidad	Coste	Coste total
Avería del equipo	Solución 1	10 %	1.099 €	109,9 €
Retraso de alguna fase del proyecto	Solución 2	10 %	11.696 €	1.169,6 €
Total				**1.279,5 €**

Tabla 10.9: Imprevistos del proyecto

Solución 1: Comprar otro similar

Solución 2: Aumento de las horas trabajadas por el SRE (8h/día). No se recalcula el coste de luz para facilitar los cálculos.

10.6. Coste total

En la Tabla 10.10 se ve el presupuesto del coste total del proyecto. El total con IVA es de **24.297,26 €**

	Tipo	**Coste**
	Directo e indirecto	15.667,40 €
	Contingencia	3.133,48 €
	Imprevistos	1.279,5 €
Total sin IVA		20.080,38 €
Total con el 21 % de IVA		**24.297,26 €**

Tabla 10.10: Coste total del proyecto

Capítulo 11

Identificación de leyes y regulaciones

Las dos normativas a las cuales este proyecto podría ser sensible de recibir la aplicación es la Ley Orgánica de Protección de Datos de Carácter Personal (LOPD) a nivel estatal y el Reglamento General de Protección de Datos (GDPR) a nivel europeo, pero a tal proyecto no se le aplican dichas normativas puesto que no se están recogiendo los datos que tales leyes consideran como sensibles.

La LOPD considera datos de carácter personal: cualquier información numérica, alfabética, gráfica, fotográfica, acústica o de cualquier otro tipo concerniente a personas físicas identificadas o identificables. En ningún caso se está recogiendo información que puede hacer al usuario identificable.

El GDPR considera datos personales todos aquellos que puedan revelar la identidad del usuario. También incluye direcciones IP pero en el presente proyecto no se recogen direcciones IP de los usuarios.

Capítulo 12

Sostenibilidad

12.1. Impacto económico

Los costes para este proyecto son más bajos que los de la competencia ya que las necesidades y los recursos también son más bajos. El hecho de que se plantee el problema de forma diferente que las demás compañías, puede hacer que la necesidad de recursos sea menor y este hecho puede ser el que sea mejor a las soluciones existentes a nivel económico, ya que se está diseñando un sistema desde cero en vez de adaptar una solución concreta para que resuelva un problema similar al que ya soluciona. Como resultado se es más eficiente a la hora de adquirir servidores y servicios.

12.2. Impacto ambiental

El indicador de impacto ambiental de este proyecto es principalmente el consumo eléctrico. En la sección 10.2 se ha detallado el consumo eléctrico de la realización del proyecto, en el momento en que el proyecto ha pasado a producción, el consumo eléctrico ya no depende íntegramente de la empresa. Se planteó el hecho de comprar servidores y colocarlos en la empresa en vez de alquilar por horas los servidores, esta idea se desestimó y una de las razones fue el consumo eléctrico. AWS es una empresa especializada en el alquiler de servidores y como tal busca su máximo beneficio, por lo que AWS intentará disminuir al máximo su consumo eléctrico dado que reduciendo

el consumo aumentan sus beneficios. Por lo tanto, con AWS se está seguro de que se tiene el mínimo consumo eléctrico y por ende el mínimo impacto ambiental. El proyecto se ha diseñado en vistas a que pueda ser utilizado para procesar otro tipo de datos diferente a los eventos, como pudiera ser los derivados del BI, por lo que ya en el planteamiento inicial del proyecto se pensó en reutilizar recursos a posteriori, a priori no se han podido reutilizar recursos dado que no existían. Actualmente las empresas que se disponen a procesar eventos tienden a unificar su capa de eventos con su capa de BI, algo muy parecido que lo que intenta este proyecto. La peculiaridad de este proyecto es el hecho de que primero se diseña un sistema para procesar eventos que luego tendrá que soportar procesar datos de BI, cuando las otras empresas plantearon, por diversas razones, el problema isómero, es decir, diseñaron un sistema de procesado de BI que luego están adaptando para procesar eventos. El hecho de no adaptar ningún sistema, sino diseñar un sistema que admita el procesado de los dos tipos de datos puede eliminar ciertos sobrecostes, como máquinas extras resultado de la adaptación y no de un diseño inteligente. Por lo que esta solución puede ser mejor ambientalmente que las existentes, dado que se pueden reducir el uso de máquinas extras y como consecuencia un menor gasto eléctrico.

12.3. Impacto social

El impacto personal del proyecto ha sido elevado, ya que documentándome para entender el estado del arte, me he dado cuenta de la cantidad de recursos que se invierten para proyectos parecidos para el mío y el uso que se le puede dar desde el punto de vista del dueño de los datos. Mi proyecto se centra en la recolección de eventos para mejorar la eficiencia del trabajo de los desarrolladores, pero también el sistema tendrá que ser compatible para procesar datos de BI. Los datos de BI al fin y al cabo lo que están midiendo es el comportamiento del usuario con respecto nuestro producto. Esto me genera ciertos dilemas éticos con respecto el trabajo futuro y cómo ha de evolucionar el proyecto. Es cierto que las empresas del sector suelen avisar de que recolectan datos del uso que se hace de sus productos, pero desde mi punto de vista los usuarios no son conscientes de qué datos se están recolectando ni cómo se están utilizando, las empresas del sector no acaban de asegurar que hacen

con tales datos una vez han sido procesados. Creo que tengo el deber ético de ser lo más transparente posible con el uso que haré de esos datos ya que con la información que se recolecta se está muy cerca de vulnerar una intimidad que los usuarios no saben que tienen. Por lo que algo que puede mejorar socialmente mi solución es la transparencia con respecto a los datos recolectados. Una vez acabado el Trabajo de Fin de Grado, este proyecto continuará por lo que mi deber debería ser también una labor de concienciación una vez el Trabajo de Fin de Grado haya terminado. Aunque el sistema planteado tiene sus peligros a nivel de privacidad, existe la necesidad de que exista, ya que analizando los eventos podemos mejorar las aplicaciones y corregir, por ejemplo, fallos de seguridad que podrían afectar más gravemente a la privacidad que el propio sistema. Este sistema nos proporciona un gran poder, por lo que conlleva una gran responsabilidad de que su uso sea el correcto y más beneficioso para la sociedad en general.

12.4. Matriz de sostenibilidad

La matriz de sostenibilidad resume las secciones 12.1, 12.2 y 12.3. Los criterios seguidos para determinar las puntuaciones son: cómo se ha llevado a cabo el proyecto, la utilidad del proyecto y los riesgos que puede suponer tal proyecto. El objetivo intrínseco del proyecto es conseguir un mejor rendimiento de los productos de la empresa para aumentar su beneficio económico, por lo que en la dimensión económica obtiene un 10/10 ya que es viable y se pueden apreciar ciertas mejoras con respecto la competencia. En la dimensión ambiental se ha intentado buscar soluciones que no afecten directamente sobre el medio-ambiente de forma negativa, pero al delegar a otras compañías ciertos servicios del proyecto no se puede estar completamente seguro de cumplan su compromiso ambiental, por lo que la puntuación es 6/10. En la dimensión social existente varios riesgos de impactar negativamente, pero el desarrollador del proyecto se compromete a ser transparente en este aspecto, cosa que puede ser diferencial con la competencia por lo que la puntuación que se obtiene es de 8/10.

Dimensión económica	**Dimensión ambiental**	**Dimensión social**
10/10	6/10	8/10
Total 23/30		

Tabla 12.1: Matriz de sostenibilidad

Capítulo 13

Conclusiones

13.1. Observaciones

Realizar este proyecto ha permitido a la empresa disponer de un sistema de seguimiento y monitorización de errores en tiempo real, agilizando así la resolución de problemas críticos en los dispositivos que actúan como clientes. Se han cumplido los objetivos planteados al inicio puesto que la prueba de concepto es funcional y desarrolla el comportamiento esperado.

El sistema se ha puesto en producción en la empresa, después de someterlo a una serie de test de validación, con el objetivo de testear el sistema en el entorno empresarial no en el académico.

A título personal este trabajo me ha servido para adquirir unos nuevos conocimientos sobre las arquitecturas de los sistemas de Big Data, en especial los sistema de Stream Processing. También he adquirido nuevos conocimientos de administración de sistemas en AWS así como administración del networking.

13.2. Trabajo futuro

El sistema cumple con los objetivos planteados al inicio del proyecto, pero para que esté listo en un entorno empresarial le falta pulir algunos aspectos. Así pues al sistema se le ha de dotar de monitorización en todos sus módulos, seguridad como mínimo en los módulos que están expuestos a Internet. Este sistema también ha de

servir para recoger eventos de otros dispositivos diferentes a los ubicuos, en este caso se han de adaptar los módulos de recolección y de envío al dispositivo objetivo.

Apéndice A

Diagrama de Gantt

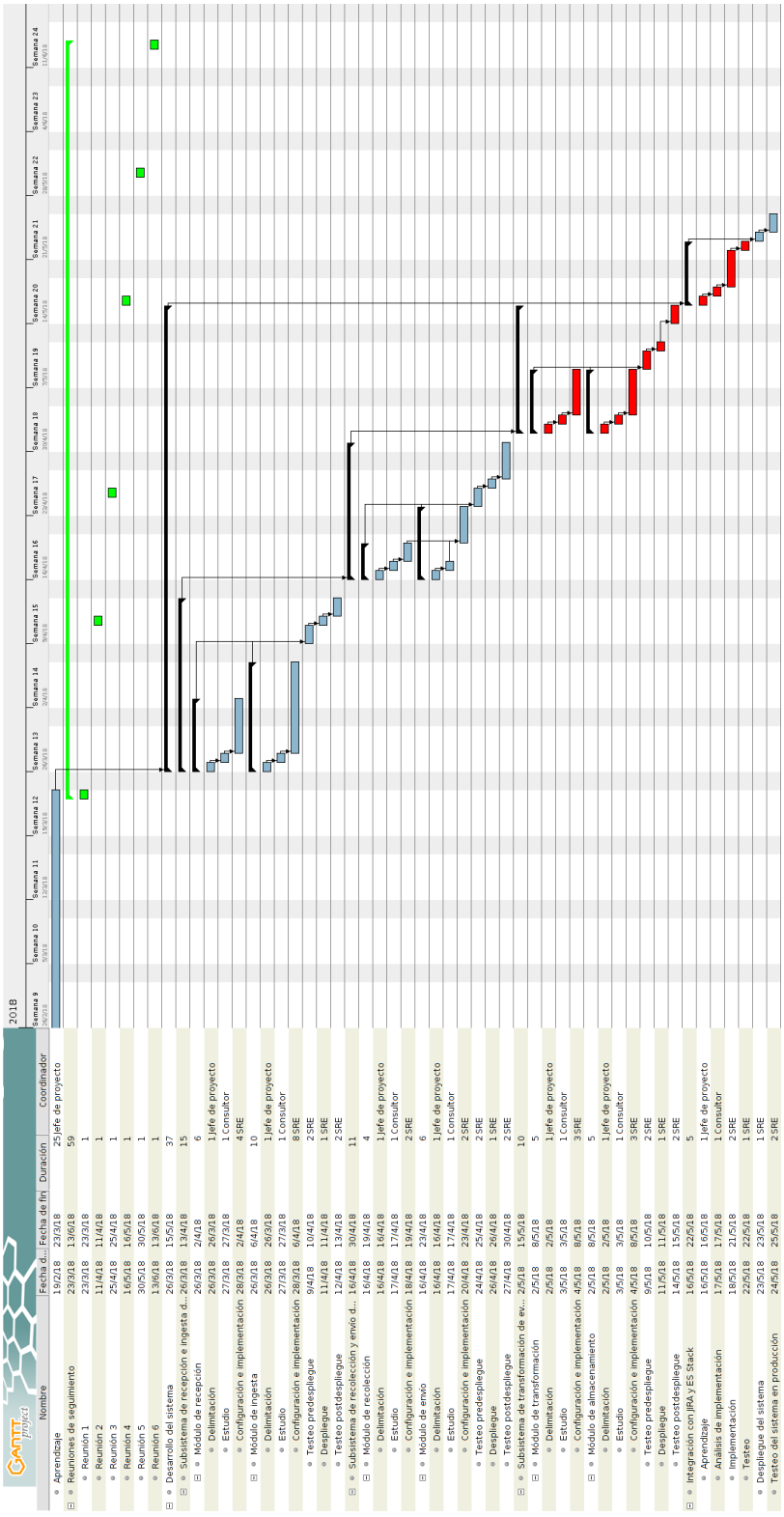
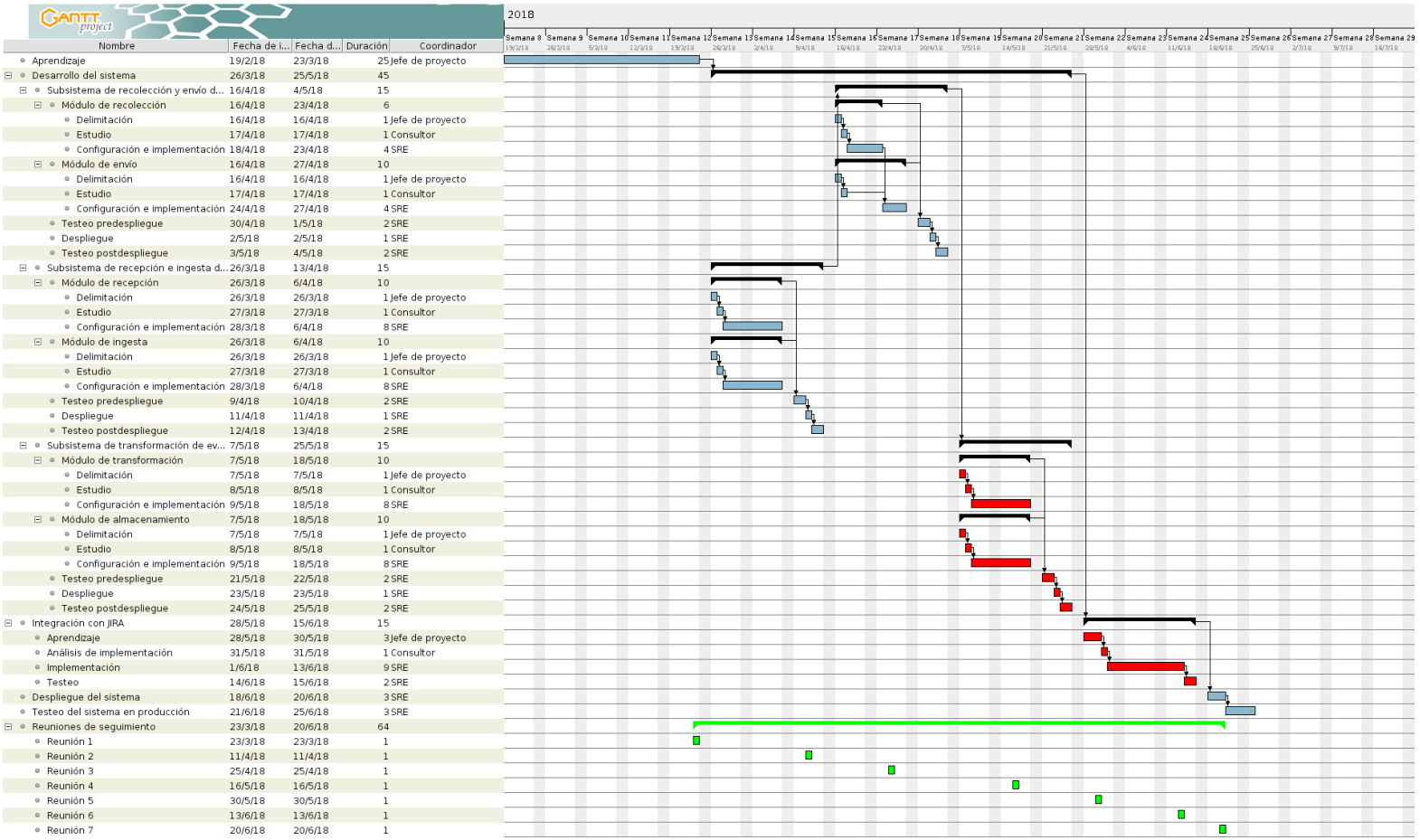


Figura A.1: Diagrama de Gantt Final



Nombre	Fecha de i...	Fecha d...	Duración	Coordinador
Aprendizaje	19/2/18	23/3/18	25	Jefe de proyecto
Desarrollo del sistema	26/3/18	25/5/18	45	
Subsistema de recolección y envío d...	16/4/18	4/5/18	15	
Módulo de recolección	16/4/18	23/4/18	6	
Delimitación	16/4/18	16/4/18	1	Jefe de proyecto
Estudio	17/4/18	17/4/18	1	Consultor
Configuración e implementación	18/4/18	23/4/18	4	SRE
Módulo de envío	16/4/18	27/4/18	10	
Delimitación	16/4/18	16/4/18	1	Jefe de proyecto
Estudio	17/4/18	17/4/18	1	Consultor
Configuración e implementación	24/4/18	27/4/18	4	SRE
Testeo predespliegue	30/4/18	1/5/18	2	SRE
Despliegue	2/5/18	2/5/18	1	SRE
Testeo postdespliegue	3/5/18	4/5/18	2	SRE
Subsistema de recepción e ingesta d...	26/3/18	13/4/18	15	
Módulo de recepción	26/3/18	6/4/18	10	
Delimitación	26/3/18	26/3/18	1	Jefe de proyecto
Estudio	27/3/18	27/3/18	1	Consultor
Configuración e implementación	28/3/18	6/4/18	8	SRE
Módulo de ingesta	26/3/18	6/4/18	10	
Delimitación	26/3/18	26/3/18	1	Jefe de proyecto
Estudio	27/3/18	27/3/18	1	Consultor
Configuración e implementación	28/3/18	6/4/18	8	SRE
Testeo predespliegue	9/4/18	10/4/18	2	SRE
Despliegue	11/4/18	11/4/18	1	SRE
Testeo postdespliegue	12/4/18	13/4/18	2	SRE
Subsistema de transformación de ev...	7/5/18	25/5/18	15	
Módulo de transformación	7/5/18	18/5/18	10	
Delimitación	7/5/18	7/5/18	1	Jefe de proyecto
Estudio	8/5/18	8/5/18	1	Consultor
Configuración e implementación	9/5/18	18/5/18	8	SRE
Módulo de almacenamiento	7/5/18	18/5/18	10	
Delimitación	7/5/18	7/5/18	1	Jefe de proyecto
Estudio	8/5/18	8/5/18	1	Consultor
Configuración e implementación	9/5/18	18/5/18	8	SRE
Testeo predespliegue	21/5/18	22/5/18	2	SRE
Despliegue	23/5/18	23/5/18	1	SRE
Testeo postdespliegue	24/5/18	25/5/18	2	SRE
Integración con JIRA	28/5/18	15/6/18	15	
Aprendizaje	28/5/18	30/5/18	3	Jefe de proyecto
Análisis de implementación	31/5/18	31/5/18	1	Consultor
Implementación	1/6/18	13/6/18	9	SRE
Testeo	14/6/18	15/6/18	2	SRE
Despliegue del sistema	18/6/18	20/6/18	3	SRE
Testeo del sistema en producción	21/6/18	25/6/18	3	SRE
Reuniones de seguimiento	23/3/18	20/6/18	64	
Reunión 1	23/3/18	23/3/18	1	
Reunión 2	11/4/18	11/4/18	1	
Reunión 3	25/4/18	25/4/18	1	
Reunión 4	16/5/18	16/5/18	1	
Reunión 5	30/5/18	30/5/18	1	
Reunión 6	13/6/18	13/6/18	1	
Reunión 7	20/6/18	20/6/18	1	

Figura A.2: Diagrama de Gantt Inicial

Apéndice B

Reproducción del sistema

En el siguiente anexo se detalla como reproducir los módulos de recepción, ingesta, transformación y almacenamiento en un solo nodo. Algunas configuraciones difieren de las del sistema en producción por el hecho de reproducirse en solo un nodo. Se supone un nodo con Ubuntu 16.04 Server Edition y con JRE 1.8.0 instalado.

Los módulos de recepción e ingesta están compuestos por librerías de Android, el siguiente enlace es un repositorio de una aplicación Android que se ha utilizado para testear que los módulos se integran correctamente:

https://github.com/pezmosca/LogBackToKafka.git

B.1. Módulo de recepción

B.1.1. Adquisición de la herramientas

Para adquirir la herramienta de este módulo se ha de rellenar un formulario en la siguiente web:

https://www.confluent.io/download

Se descargará en el home de Ubuntu para simplificar la reproducción.

B.1.2. Instalación de las herramientas

La herramienta será instalada en el home de Ubuntu para simplificar la reproducción. Suponiendo que la herramienta está en el home y suponiendo la versión

4.1.1:

```
tar xzf confluent-oss-4.1.1-2.11.tar.gz
mv confluent-oss-4.1.1-2.11.tar.gz confluent

sudo systemctl enable kafka-rest.service
sudo systemctl start kafka-rest.service
```

kafka-rest.service

```
[Unit]
Description=Apache Kafka HTTP Proxy
Documentation=https://docs.confluent.io/current/kafka-rest/
    docs/index.html
Requires=network.target remote-fs.target
After=network.target remote-fs.target

[Service]
Type=simple
User=user
Group=user
ExecStart=/home/user/confluent/bin/kafka-rest-start /home/
    user/confluent/etc/kafka-rest/kafka-rest.properties
ExecStop=/home/user/confluent/bin/kafka-rest-stop

[Install]
WantedBy=multi-user.target
```

B.1.3. Configuración de las herramientas

La configuración por defecto es valida para este ejemplo.

B.2. Módulo de ingesta

B.2.1. Adquisición de las herramientas

Se descargará en el home de Ubuntu para simplificar la reproducción.

```
wget http://apache.rediris.es/kafka/1.1.0/kafka_2.11-1.1.0.tgz
```

B.2.2. Instalación de las herramientas

La herramienta será instalada en el home para simplificar la reproducción. Suponiendo que la herramienta está en el home y suponiendo la versión 1.1.0:

```
tar xzf kafka_2.11-1.1.0.tgz
mv kafka_2.11-1.1.0.tgz kafka

sudo systemctl enable zookeeper.service
sudo systemctl start zookeeper.service

sudo systemctl enable kafka.service
sudo systemctl start kafka.service
```

zookeeper.service

```
[Unit]
Description=Apache Zookeeper server (Kafka)
Documentation=http://zookeeper.apache.org
Requires=network.target remote-fs.target
After=network.target remote-fs.target

[Service]
Type=simple
User=user
Group=user
```

```
ExecStart=/home/user/kafka/bin/zookeeper-server-start.sh /
    home/user/kafka/config/zookeeper.properties
ExecStop=/home/user/kafka/bin/zookeeper-server-stop.sh

[Install]
WantedBy=multi-user.target
```

kafka.service

```
[Unit]
Description=Apache Kafka server (broker)
Documentation=http://kafka.apache.org/documentation.html
Requires=network.target remote-fs.target
After=network.target remote-fs.target

[Service]
Type=simple
User=user
Group=user
ExecStart=/home/user/kafka/bin/kafka-server-start.sh /home/
    user/kafka/config/server.properties
ExecStop=/home/user/kafka/bin/kafka-server-stop.sh

[Install]
WantedBy=multi-user.target
```

B.2.3. Configuración de las herramientas

La configuración por defecto es valida para este ejemplo.

B.3. Módulo de procesado

B.3.1. Adquisición de las herramientas

Se descargará en el home de Ubuntu para simplificar la reproducción. De Kafka Streams se descargarán los jars utilizados para testear la solución, hay diversos jars para poder testear por separado cada una de las acciones que hace la aplicación con Kafka Streams, pero se podría implementar con threads en un solo jar.

```
wget https://github.com/pezmosca/empty/raw/master/jars.tar.
    gz
wget https://artifacts.elastic.co/downloads/logstash/
    logstash-6.3.0.tar.gz
```

B.3.2. Instalación de las herramientas

La herramienta será instalada en el home para simplificar la reproducción. Suponiendo que la herramienta está en el home y suponiendo la versión 6.3.0:

```
tar xzf logstash-6.3.0.tar.gz
mv logstash-6.3.0.tar.gz logstash
tar xzf jars.tar.gz

sudo systemctl enable logstash.service
sudo systemctl start logstash.service

sudo systemctl enable kafkastreams.service
sudo systemctl start kafkastreams.service
```

logstash.service

```
[Unit]
Documentation=https://www.elastic.co/products/logstash
After=network.target
```

```
ConditionPathExists=/home/ec2-user/logstash/tfg-pipeline.
    conf

[Service]
User=user
ExecStart=/home/user/logstash/bin/logstash -f /home/user/
    logstash/tfg-pipeline.conf --config.reload.automatic

[Install]
WantedBy=multi-user.target
```

kafkastreams.service

```
[Unit]
Description=Kafka Streams Service
Requires=network.target remote-fs.target
After=network.target remote-fs.target

[Service]
User=user
Group=user
ExecStart=/home/user/jars/start.sh

[Install]
WantedBy=multi-user.target
```

B.3.3. Configuración de las herramientas

La configuración por defecto no es valida para este ejemplo, se ha de generar una pipeline para Logstash.

Logstash

Se ha de generar una pipeline para procesar los eventos. Tal pipeline se ha de crear en /logstash/tfg-pipeline.conf y el contenido ha de ser el siguiente:

```
input {
  kafka {
    bootstrap_servers => "localhost:9092"
    topics => ["log", "crashout"]
    group_id => "tfg-logstash"
    codec => json
    decorate_events => true
  }
}

filter {
  if [@metadata][kafka][topic] == "log" {
    mutate {
      rename => { "T" => "ExecutionTime"
      "T2" => "Timestamp"
      "L" => "Level"
      "TN" => "ThreadName"
      "LN" => "Class"
      "M" => "Message"
      }
    }
  }
}

output {

  if[@metadata][kafka][topic] == "log" {
    elasticsearch {
      hosts => [ "localhost:9200" ]
      index => "logs"
    }
  }
```

```
  if[@metadata][kafka][topic] == "crashout" {
    elasticsearch {
      hosts => [ "localhost:9200" ]
      index => "crashes"
    }
  }
}
```

B.4. Módulo de almacenamiento

B.4.1. Adquisición de las herramientas

Las herramientas para este módulo se descargarán en el home de Ubuntu para simplificar la reproducción.

```
wget https://artifacts.elastic.co/downloads/elasticsearch/
    elasticsearch-6.3.0.tar.gz
wget https://artifacts.elastic.co/downloads/kibana/kibana
    -6.3.0-linux-x86_64.tar.gz
```

B.4.2. Instalación de las herramientas

Las herramientas serán instalada en el home para simplificar la reproducción, suponiendo que las herramientas están en el home y suponiendo la versión 6.3.0 de Elastic Search, Logstash y Kibana, y la 7.10.1 de JIRA. Para adquirir JIRA se ha de acceder mediante un navegador web a https://es.atlassian.com/software/jira/download y descargarlo.

```
tar xzf elasticsearch-6.3.0.tar.gz
mv elasticsearch-6.3.0.tar.gz elasticsearch

tar xzf kibana-6.3.0.tar.gz
```

```
mv kibana-6.3.0.tar.gz kibana

sudo systemctl enable elasticsearch.service
sudo systemctl enable kibana.service

sudo chmod +x atlassian-jira-software-7.10.1-x64.bin
sudo ./atlassian-jira-software-7.10.1-x64.bin
```

elasticsearch.service

```
[Unit]
Documentation=https://www.elastic.co/products/elasticsearch
After=network.target

[Service]
User=user
Group=user
LimitMEMLOCK=infinity
LimitNOFILE=65536
ExecStart=/home/user/elasticsearch/bin/elasticsearch

[Install]
WantedBy=multi-user.target
```

kibana.service

```
[Unit]
Description=Kibana
After=network.target

[Service]
ExecStart=/home/user/kibana/bin/kibana
Type=simple
Restart=always
```

```
[Install]
WantedBy=default.target
```

B.4.3. Configuración de las herramientas

La configuración por defecto es valida para este ejemplo.

Bibliografía

[1] ACRA. *ACRA*. https://github.com/ACRA/acra, (2018). Último acceso: 4/03/2018.

[2] Amazon. *Amazon EC2*. https://aws.amazon.com/es/ec2/, (2018). Último acceso: 4/03/2018.

[3] Amazon. *Amazon Kinesis*. https://aws.amazon.com/es/kinesis/, (2018). Último acceso: 4/03/2018.

[4] Amazon. *Application Load Balancers*. https://docs.aws.amazon.com/es_es/elasticloadbalancing/latest/application/introduction.html, (2018). Último acceso: 4/03/2018.

[5] Amazon. *Lenovo Ideapad Y700*. https://www.amazon.es/Lenovo-Ideapad-Y700-17ISK-I7-6700HQ-operativo/dp/B01MZ9L7PT, (2018). Último acceso: 4/03/2018.

[6] Amazon. *Precios EC2*. https://aws.amazon.com/es/ec2/pricing/on-demand/, (2018). Último acceso: 4/03/2018.

[7] Amazon. *What is AWS?* https://aws.amazon.com/es/what-is-aws/, (2018). Último acceso: 4/03/2018.

[8] Android. *Log Class Android*. https://developer.android.com/reference/android/util/Log.html, (2018). Último acceso: 4/03/2018.

[9] Atlassian. *JIRA pricing*. https://es.atlassian.com/software/jira/pricing?tab=self-hosted, (2018). Último acceso: 4/03/2018.

[10] Atlassian. *Jira Software: La herramienta de desarrollo de software líder de los equipos ágiles*. https://es.atlassian.com/software/jira, (2018). Último acceso: 4/03/2018.

[11] Confluent. *Kafka Production Deployment*. https://docs.confluent.io/current/kafka/deployment.html#hardware, (2018). Último acceso: 4/03/2018.

[12] Confluent. *Kafka REST Production Deployment*. https://docs.confluent.io/current/kafka-rest/docs/deployment.html, (2018). Último acceso: 4/03/2018.

[13] Confluent. *Kafka REST Proxy*. https://docs.confluent.io/current/kafka-rest/docs/index.html, (2018). Último acceso: 4/03/2018.

[14] Confluent. *Zookeeper Production Deployment*. https://docs.confluent.io/current/zookeeper/deployment.html#hardware, (2018). Último acceso: 4/03/2018.

[15] Datadog. *Modern monitoring and analytics*. https://www.datadoghq.com/, (2018). Último acceso: 4/03/2018.

[16] Elastic. *Elastic Search*. https://www.elastic.co/products/elasticsearch, (2018). Último acceso: 4/03/2018.

[17] Elastic. *Logstash Production Deployment*. https://www.elastic.co/guide/en/logstash/current/deploying-and-scaling.html, (2018). Último acceso: 4/03/2018.

[18] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, y T. Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*. http://www.ietf.org/rfc/rfc2616.txt, (1999). Último acceso: 4/03/2018.

[19] The Apache Software Foundation. *Apache Kafka: A distributed streaming platform*. https://kafka.apache.org/, (2017). Último acceso: 4/03/2018.

[20] The Apache Software Foundation. *Apache Log4j 2*. https://logging.apache.org/log4j/2.x/, (2017). Último acceso: 4/03/2018.

[21] The Apache Software Foundation. *Kafka Streams*. https://kafka.apache.org/documentation/streams/, (2018). Último acceso: 4/03/2018.

[22] Ricardo Barranco Fragoso. *¿Qué es Big Data?* https://docs.oracle.com/cd/E13203_01/tuxedo/tux71/html/addist2.htm, (2012). Último acceso: 4/03/2018.

[23] Krishna Gade y Roger Wang. *Singer, Pinterest's Logging Infrastructure*. https://es.slideshare.net/DiscoverPinterest/singer-pinterests-logging-infrastructure, (2014). Último acceso: 4/03/2018.

[24] Gartner. *KIB (integration broker)*. https://www.gartner.com/it-glossary/ib-integration-broker, (2018). Último acceso: 4/03/2018.

[25] Glassdoor. *IT Consultant Salaries*. https://www.glassdoor.com/Salaries/it-consultant-salary-SRCH_KO0,13.htm, (2018). Último acceso: 4/03/2018.

[26] Glassdoor. *IT Project Manager Salaries*. https://www.glassdoor.com/Salaries/it-project-manager-salary-SRCH_KO0,18.htm, (2018). Último acceso: 4/03/2018.

[27] Glassdoor. *Site Reliability Engineer Salaries*. https://www.glassdoor.com/Salaries/site-reliability-engineer-salary-SRCH_KO0,25.htm, (2018). Último acceso: 4/03/2018.

[28] IGI Global. *What is Ubiquitous Computing*. https://www.igi-global.com/dictionary/ubiquitous-computing/30811, (2018). Último acceso: 4/03/2018.

[29] Google. *Breakpad: A set of client and server components which implement a crash-reporting system*. https://chromium.googlesource.com/breakpad/breakpad/, (2018). Último acceso: 4/03/2018.

[30] Google. *Crashlytics, Introducing the most powerful, yet lighttest weight crash reporting solutio*. https://try.crashlytics.com/, (2018). Último acceso: 4/03/2018.

[31] Michael Hausenblas, Nathan Bijnens, y Nathan Marz. *Lambda Architecture*. http://lambda-architecture.net/, (2017). Último acceso: 4/03/2018.

[32] Tongbo Huang. *Behind the Pins: Building Analytics*. https://medium.com/@Pinterest_Engineering/behind-the-pins-building-analytics-f7b508cdacab?s=hi-from-keen-io, (2014). Último acceso: 4/03/2018.

[33] Douglas Hughey. *The Traditional Waterfall Approach*. http://www.umsl.edu/~hugheyd/is6840/waterfall.html, (2009). Último acceso: 4/03/2018.

[34] HyperTrack. *HyperLog: Utility logger library for storing logs into database and push them to remote server for debugging*. https://github.com/hypertrack/hyperlog-android, (2018). Último acceso: 4/03/2018.

[35] IBM. *ETL*. https://www.ibm.com/support/knowledgecenter/en/SSWSR9_11.5.0/com.ibm.mdshs.initiateglossary.doc/topics/r_glossary_etl.html, (2015). Último acceso: 4/03/2018.

[36] J.Jenkov. *Micro Batching*. http://tutorials.jenkov.com/java-performance/micro-batching.html, (2018). Último acceso: 4/03/2018.

[37] Jay Kreps. *The Log: What every software engineer should know about real-time data's unifying abstraction*. https://engineering.linkedin.com/distributed-systems/log-what-every-software-engineer-should-know-about-real-time-datas-unifying, (2013). Último acceso: 4/03/2018.

[38] Logback. *Logback*. https://logback.qos.ch/, (2018). Último acceso: 4/03/2018.

[39] Igor Maravic. *Spotify Event Delivery, The Road to the Cloud, Part I*. https://labs.spotify.com/2016/02/25/spotifys-event-delivery-the-road-to-the-cloud-part-i/, (2016). Último acceso: 4/03/2018.

[40] Microsoft. *What is business intelligence?* https://technet.microsoft.com/en-us/library/cc811595(v=office.12).aspx, (2008). Último acceso: 4/03/2018.

[41] Netflix. *Evolution of the Netflix Data Pipeline.* https://medium.com/netflix-techblog/evolution-of-the-netflix-data-pipeline-da246ca36905, (2016). Último acceso: 4/03/2018.

[42] Netflix. *Blitz4j: Logging framework for fast asynchronous logging.* https://github.com/Netflix/blitz4j, (2018). Último acceso: 4/03/2018.

[43] Oracle. *What Is a Distributed Application.* https://docs.oracle.com/cd/E13203_01/tuxedo/tux71/html/addist2.htm, (2000). Último acceso: 4/03/2018.

[44] Mike Ossareh. *Twitch Data Analysis — Part 1 of 3: The Twitch Statistics Pipeline.* https://blog.twitch.tv/twitch-data-analysis-part-1-of-3-the-twitch-statistics-pipeline-51556a14c961, (2014). Último acceso: 4/03/2018.

[45] Selectra. *Tarifa Luz Hora.* https://tarifaluzhora.es/, (2018). Último acceso: 4/03/2018.

[46] LudiumLab S.L. *Ludium.* http://www.ludiumlab.es/, (2018). Último acceso: 4/03/2018.

[47] Inc. Square. *OkHttp.* http://square.github.io/okhttp/, (2018). Último acceso: 4/03/2018.

[48] Toon Sripatanaskul y Zhengyu Cai. *Inception: How LinkedIn Deals with Exception Logs.* https://engineering.linkedin.com/blog/2016/12/inception--how-linkedin-deals-with-exception-logs, (2016). Último acceso: 4/03/2018.

[49] Benjamin Treynor. *What is SRE (Site Reliability Engineering)?* https://www.oreilly.com/ideas/what-is-sre-site-reliability-engineering, (2018). Último acceso: 4/03/2018.

[50] Tony Trinh. *logback-android*. https://github.com/tony19/logback-android, (2018). Último acceso: 4/03/2018.

[51] Shu Uesugi. *Kappa Architecture*. http://milinda.pathirage.org/kappa-architecture.com/, (2018). Último acceso: 4/03/2018.

[52] Gowthamy Vaseekaran. *Big Data Battle : Batch Processing vs Stream Processing*. https://medium.com/@gowthamy/big-data-battle-batch-processing-vs-stream-processing-5d94600d8103, (2017). Último acceso: 4/03/2018.

[53] Xiaomi. *MI A1*. http://buy.mi.com/es/buy/product/mi-a1, (2018). Último acceso: 4/03/2018.