UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Departamento de Sistemas Informáticos y Computación

Universitat Politècnica de València

APLICACIÓN PARA EL ANÁLISIS, ESTUDIO Y PUBLICACIÓN DEL COMPORTAMIENTO EN LA UTILIZACIÓN DE VALENBISI

Trabajo Fin de Master

Master Universitario en Computacion Paralela y Distribuida

Autor: Jordi Llinares Llinares

Tutores: Pedro Alonso Jordá

Pau Arce Vila

2018-2019

Resumen

El trabajo realizado en este TFM ha consistido en realizar una aplicación que mediante el análisis de datos de los datos recopilados mediante la plataforma API APP CIUDAD obtenga unos resultados y los proporcione al público, su función consiste en recopilar un histórico de datos en tiempo real del estado de las estaciones de Valenbisi para poder prever su estado en el futuro, actualizar estas previsiones en función de los nuevos datos que van siendo obtenidos y proporcionar estos resultados a usuarios y propietarios de este servicio para la posible utilización o para la optimización de los recursos disponibles por la empresa.

Palabras clave: Apache Hadoop, Hadoop Streaming, Python, Valenbici, Big Data, Docker, Internet of Things.

Abstract

This Master Thesis consists of implementing an application that, upon analyzing the data collected through the API APP CITY platform, obtains some results and provides them to the public. The main function of the application is to compile a real-time data history of the status of Valenbisi stations in order to be able to forecast their status in the future, update these forecasts based on the new data that is being obtained. This results will be provided to the users, but also to service providers for its possible use in the optimization task of resources provision of new facilities.

Keywords : Apache Hadoop, Hadoop Streaming, Python, Valenbici, Big Data, Docker, Internet of Things.

Resum

El treball realitzat en aquest TFM ha consistit en la realització d'una aplicació que mitjançant el anàlisis de dades, de les dades recopilades mitjançant la plataforma API APP CIUDAD obtinga uns resultats i els proporcione al públic, la seva funció consisteix en recollir un històric de dades en temps real del estat de les estacions de Valenbisi per a poder preveure el seu estat en el futur, actualitzar aquestes previsions en funció de les noves dades que van següent obtingudes y proporcionar aquests resultats a usuaris y propietaris d'aquest servei per a la possible utilització o per a la optimització dels recursos disponibles per l'empresa.

Paraules clau : Apache Hadoop, Hadoop Streaming, Python, Valenbici, Big Data, Docker, Internet of Things.

Tabla de contenidos

1. Introducción

Actualmente el mundo del IoT (Internet de las cosas) está teniendo una gran relevancia puesto que proporciona la posibilidad de mejorar tanto los entornos empresariales como la vida cotidiana.

Grandes ciudades como Madrid o Valencia cuentan con un gran número de sensores incorporados en la ciudad con el fin de obtener datos en tiempo real en referencia al estado del tráfico, uso de transporte público, puntos de interés, contaminación, etc. Los datos son ofrecidos al público mediante una API REST[1], estos datos pueden ser utilizados por otras aplicaciones con el fin de poder ser servidos a usuarios de estas.

Este trabajo se centra en los datos de Valenbisi ofrecidos por la API, vamos a realizar una aplicación para poder obtener el estado de las estaciones un día y hora especifico, también obtendremos la saturación y la no utilización óptima de los vehículos.

1.1 Planteamiento del problema

El problema de este proyecto consiste en el desarrollo de una aplicación que realice un análisis de datos, sobre los datos recopilados de la plataforma API APP CIUDAD, y obtenga el estado futuro de las estaciones de Valenbisi así como la publicación de este.

Como sistema sobre el que se ha implementado la aplicación se ha elegido la distribución Linux Ubuntu 16.04, se ha escogido esta distribución puesto que, en el proyecto se ha decidido la utilización de Docker[2] y Apache Hadoop[3] y resulta más fácil trabajar con distribuciones de Linux.

Como lenguaje de programación, he escogido Python[4] puesto que, es muy versátil, completo y de fácil aprendizaje, además de que el tratamiento de los datos resulta muy sencillo mediante su utilización.

Como herramienta para el tratamiento y análisis de los datos se ha utilizado Apache Hadoop, que es una implementación de código abierto formada por varios proyectos, en los cuales tenemos Hadoop Streaming[5], que permite escribir programas bajo el modelo de programación MapReduce[6], independientemente del lenguaje de programación utilizado.

Como modo de almacenamiento de los datos obtenidos se ha optado por la utilización de MySQL[7] puesto que en el grado se utilizaba esta base de datos, es open-source (código abierto) y es una de las más populares en todo el mundo.

En referencia a la publicación de los resultados obtenidos mediante el análisis de los datos, se ha decidido la utilización de HTML[8], que es el elemento de construcción más básico de una página web y se usa para crear y representar visualmente una página web, y de Flask[9] que es un micro-framework para Python basado en Werkzeug[10] y Jinja 2[11].

Para la construcción del entorno de la aplicación se ha utilizado Docker, puesto que es más ligera su utilización que la de las máquinas virtuales, y permite una fácil puesta en marcha del entorno con docker-compose.

Como posibles vías de solución, tenemos la realización de este sistema conjuntamente en una instancia o separar las diferentes secciones estructurales en instancias distintas.

La solución mediante la utilización de una sola instancia implicaría, crear una imagen Docker en la que se incluirían los software MySQL, Apache Hadoop y Python. Esta solución no requeriría de comunicación para resolución del problema.

La solución mediante el separado en diferentes nodos o instancias las diferentes partes de la aplicación, implicaría la creación de diferentes imágenes Docker, una para el almacenamiento de los resultados con MySQL, una para el tratamiento y análisis de los datos con Apache Hadoop y por ultimo una con Python y usando el módulo Flask para la publicación de los datos.

En este proyecto se ha decidido realizar la aplicación mediante esta última solución, que implica la separación de las funciones de la aplicación en distintas instancias, puesto que las imágenes creadas para las instancias serán pequeñas en comparación a la otra solución y porque al estar separadas las partes hace más fácil su posible modificación, replicación y solución de errores.

1.2 Motivación

La motivación principal para la realización de este proyecto, es ofrecer un servicio a los usuarios y a los propietarios que proporcione información sobre el estado de un recurso en el futuro, en este caso las estaciones de Valenbisi, para su posible utilización u optimización.

Como motivación secundaria para la selección de este proyecto sobre este tema ha sido que están implicados temas como son IoT, que tiene una gran relevancia debido a la domótica y a la integración de estos dispositivos en las empresas, y Big Data que tiene una gran relevancia para realizar estudios obtener ideas e identificar o prever problemas.

1.3 Objetivos

El objetivo principal del proyecto consiste en la implementación de una aplicación, que realice un estudio sobre un dataset[12] de datos recopilados de Valenbisi, que obtenga como resultados el estado en un futuro, de las diferentes estaciones y que estos resultados sean publicados. A partir de esta idea se deben definir los objetivos que expliquen la intención del problema.

Los objetivos son:

- Realizar el tratamiento de los datos
- Obtener los resultados.
- Almacenar los resultados obtenidos.
- Actualizar los resultados.
- Publicar los resultados obtenidos.
- Recopilar datos de Valenbisi para crear un dataset.
- Crear una imagen Docker para los módulos anteriores.

Estos son los objetivos principales. Se han implementado de la mejor forma posible teniendo en cuenta las restricciones que han ido surgiendo.

Como objetivos concretos o secundarios que la aplicación debía tener son los siguientes:

- Realizar el Map de los datos para los resultados de bicis disponibles y lugar de estacionamiento.
- Realizar el Reduce de los datos para los resultados de bicis disponibles y lugar de estacionamiento.
- Realizar el Map de los datos para los resultados de saturación y no utilización.
- Realizar el Reduce de los datos para los resultados de saturación y no utilización.
- Insertar los resultados obtenidos en la base de datos
- Actualizar periódicamente los registros de la base de datos.
- Publicar los datos mediante una API REST.
- Crear los recursos necesarios, para poder realizar el lanzamiento de la aplicación mediante el uso de docker-compose.

Como podemos observar, estos objetivos sirven como complemento a los objetivos principales, es decir hacen posible el funcionamiento y la resolución de los objetivos principales.

1.4 Estructura

Este proyecto está compuesto de distintos capítulos, primeramente comprobaremos el estado del arte actual del ámbito del proyecto, realizaremos una crítica al estado del arte actual y se realizara una propuesta de proyecto. De este modo podremos ver las carencias de los anteriores trabajos y proponer aportaciones o ampliaciones son las que añadirá nuestra aplicación.

En segundo lugar, se realizara el análisis del problema a afrontar en este proyecto mediante el establecimiento de los requisitos necesarios para el funcionamiento de la aplicación, la identificación y análisis de las posibles soluciones y la solución escogida para realizar el proyecto. En este punto se obtendrá una visión más concreta del problema y de su solución.

En tercer lugar se realizara el diseño de la solución, obtendremos y estudiaremos la arquitectura del sistema, se estudiara en detalle el diseño de la aplicación y se verán las herramientas y tecnologías utilizadas en este proyecto. En este apartado podrá obtenerse la estructura general de la aplicación, así como también un nivel con más detalle del diseño de la aplicación y las tecnologías que han sido elegidas para la implementación de esta.

En cuarto lugar, se realizara la descripción del desarrollo de la solución propuesta, para ello se hará una descripción de la aplicación y un descripción del flujo de esta. Mediante este apartado se podrá ver como se ha pasado de la propuesta a la solución final.

Seguidamente se presentara la etapa de implantación de la solución y las pruebas realizadas para verificar el correcto funcionamiento de la aplicación. Finalmente, se realizaran: las conclusiones a las que se ha llegado en la realización de este proyecto, la relación del trabajo desarrollado con los estudios cursados y los trabajos futuros que podrían realizarse a partir de este proyecto.

2. Estado del arte

Los objetivos se han alcanzado utilizando diferentes tecnologías. Estas tecnologías han sido estudiadas para su correcta utilización dentro de la aplicación.

2.1 Estado del arte actual

El estado del arte en referencia al Internet de las Cosas (IoT), se encuentra en un estado muy desarrollado, puesto que cada vez tenemos más dispositivos capaces de conectarse tanto a una red propia, como a Internet. Esto hace que tengamos una gran cantidad de datos proporcionados por este tipo de dispositivos, ya sea en nuestra casa, en las empresas o en las ciudades.

El estado del arte, en referencia al uso del análisis de datos para poder prever comportamientos futuros, se encuentra en pleno auge debido a que muchas empresas ya cuentan entre sus departamentos uno específico para este problema. Estos análisis les ayudan a poder solventar fallos, optimizar procesos y recursos, prever futuros problemas, etc.

Realizando una búsqueda de posibles aplicaciones y sitios web que tengan el mismo propósito, se ha conseguido encontrar una serie de aplicaciones para dispositivos móviles llamadas: Mapas Valenbisi, Valenbisi Valencia y Valencia Valenbisi. También se han encontrado 3 sitios web: el propio de la plataforma Valenbisi*, Valencia al minut* y biciv.com*. No obstante estos servicios se centran únicamente en la obtención y publicación del estado de las estaciones de bicis en tiempo real.

En referencia al funcionamiento de las aplicaciones para dispositivos móviles anteriores, hemos comprobado que la aplicación de nombre Valencia Valenbisi se centra en proporcionar únicamente el estado de las estaciones de Valenbisi y su localización. El uso dado a esta aplicación es únicamente el de conseguir una bici o aparcamiento cercano, es decir, se centra en la publicación de datos únicamente para el usuario.

El funcionamiento de la aplicación Valenbisi Valencia es mucho más completo que el de la aplicación anterior, también se centra en la publicación de datos para el usuario de las bicis, pero ofrece más funcionalidades. Esta aplicación además de tener las funcionalidades de la aplicación anterior tiene también la función de muestra de carril bici, así como, la función para personalizar nuestro perfil añadiendo estaciones preferidas, estaciones frecuentes y buscador de estaciones por nombre, número o calle.

Como último caso de aplicaciones orientadas a dispositivos móviles tenemos, la aplicación Mapa para ValenBisi, esta aplicación al igual que las otras se centra

en publicar datos sobre las estaciones, es decir está centrada en obtener datos para el usuario de las bicis. Mapa para ValenBisi destaca por ser la más completa de las aplicaciones encontradas para dispositivos móviles, además contiene el mayor número de descargas de aplicaciones de este ámbito.

Esta aplicación tiene todas las funcionalidades ofrecidas por las otras dos, pero añadiendo una interfaz más simple e intuitiva, integrando sus servicios con Google Maps para poder indicar y realizar rutas y además añade la hora de la última actualización del estado de las estaciones.

Por otra parte en las aplicaciones de los sitios web, vemos que el sitio web oficial de la plataforma, cuenta con una interfaz parecida a la de la primera aplicación para dispositivos móviles. Este sitio web ofrece la vista en un mapa indicando la posición de todas las estaciones y el estado de estas. Se observa que al igual que las aplicaciones móviles se centra en la publicación de los datos de relevancia para los usuarios de las bicis.

La aplicación de biciv.com solo contiene la funcionalidad de ver la posición de las estaciones y únicamente las bicis y los sitios disponibles, este sitio contiene una interfaz bastante complicada de manejar para un usuario normal.

Por último, tenemos la aplicación del sitio web Valencia al minut, este sitio web no se centra exclusivamente en la publicación de datos de Valenbisi, sino que, también publica datos obtenidos mediante los sensores de polución y del estado del tráfico. No obstante, esta aplicación contiene funcionalidades que no encontramos en las otras, como son: el porcentaje de estaciones saturadas o el porcentaje de estaciones vacías, así como también el número de bicis en uso y el número de bicis disponibles.

Todos estos datos se ofrecen en tiempo real, es decir no contienen información histórica del estado de las estaciones en el pasado.

2.2 Crítica al estado del arte

Existen muy pocas aplicaciones que estando orientadas a la publicación de datos para uso del consumidor, como son datos sobre el estado del tráfico, el estado de la contaminación, la humedad, el viento o el uso de las bicicletas en nuestro caso, utilicen el análisis de datos que ya está siendo utilizado por grandes empresas para la mejora de sus sistemas, infraestructuras y solución de errores.

Realizando un análisis de datos sobre grandes volúmenes de datos, seriamos capaces de prever el estado del tráfico en un momento del día determinado, la posibilidad de una tormenta, la saturación de un aparcamiento, etc. Esto es debido a que este tipo de información proporcionada por sensores de carácter digamos público, debe ser impulsada por los distintos ayuntamientos de cada

ciudad, como posible solución estaría la realización de inversiones en proyectos de este carácter.

Se ha podido observar que las empresas están dispuesta a realizar estos estudios, puestos que reconocen que la inversión en esta área contribuiría a su mejora.

Los trabajos realizados anteriormente por mis compañeros de la universidad, se observa que su estudio, se basa en el análisis de los datos para obtener estadísticas sobre coste temporales de movilidad por barrios, pero deja de lado el estudio de la saturación o la previsión de estado de las estaciones.

2.3 Propuesta

En este proyecto se va a realizar una recopilación de datos de los estados de las estaciones de Valenbisi con una frecuencia de obtención de datos determinada, un posterior análisis de estos datos recopilados para obtener estadísticas sobre la saturación, el uso y la disponibilidad de los recursos proporcionados por la empresa, una actualización de dichas estadísticas mediante consultas en tiempo real del estado de las estaciones y una publicación de dichos resultados en función de la latitud, la longitud y la acción que se requiera.

Este proyecto tiene algunas similitudes con los proyectos nombrados anteriormente, como publicar datos en referencia al estado de las estaciones, con la ampliación de que los datos mostrados pueden referirse tanto al estado actual como al estado futuro de los recursos.

La novedad que incluye este proyecto además de la nombrada anteriormente es la de obtención de saturación o no utilización de los recursos disponibles, afectando esta información a la empresa.

3. Análisis del problema

En este apartado, se pretende dar explicación de los diferentes aspectos de este proyecto previamente planificados a su implementación, debido a que hay algunas posibles soluciones mediante las cuales poder realizar la aplicación.

3.1 Requisitos

Para la correcta realización de este proyecto se deben llevar a cabo una serie de requisitos los cuales pueden ser requisitos del sistema o requisitos funcionales.

3.1.1 Requisitos del sistema

Como requisitos del sistema, contemplamos los diferentes componentes que el sistema debe poseer para realizar cada una de las tareas necesarias que nos lleven solución al problema.

Para la correcta solución del problema, el sistema debe contar con soporte para los siguientes softwares:

Python 2.7, Docker, MySQL y los módulos de Python urllib3, mysql-conector, Flask y WTForms.

3.1.2 Requisitos funcionales

Como requisitos funcionales o casos de uso, contemplamos las diferentes funcionalidades que tendrá el sistema y obtenemos 4 casos:

Caso obtener-aparcamientos:

Este caso consiste en introducir la latitud y la longitud sobre la que se quiere consultar las posibles estaciones con plazas disponibles, introducir el día y la hora sobre la que se querrá obtener dicha información, y se recibirá las estaciones cercanas con sitios disponibles.

Caso coger-bicis:

Este caso consiste en introducir la latitud y la longitud sobre la que se quiere consultar las posibles estaciones con plazas disponibles, introducir el día y la hora sobre la que se querrá obtener dicha información, y se recibirá las estaciones cercanas con bicis disponibles.

Caso consultar-saturación:

Este caso está orientado a las empresas y los administradores de sus recursos, consiste en la obtención del identificador de todas las estaciones que suelen estar saturadas en un día concreto a una hora concreta.

Caso consultar-desuso:

Este caso está orientado a las empresas y los administradores de sus recursos, consiste en la obtención del identificador de todas las estaciones que suelen tener muchos recursos desaprovechados en un día concreto a una hora concreta.

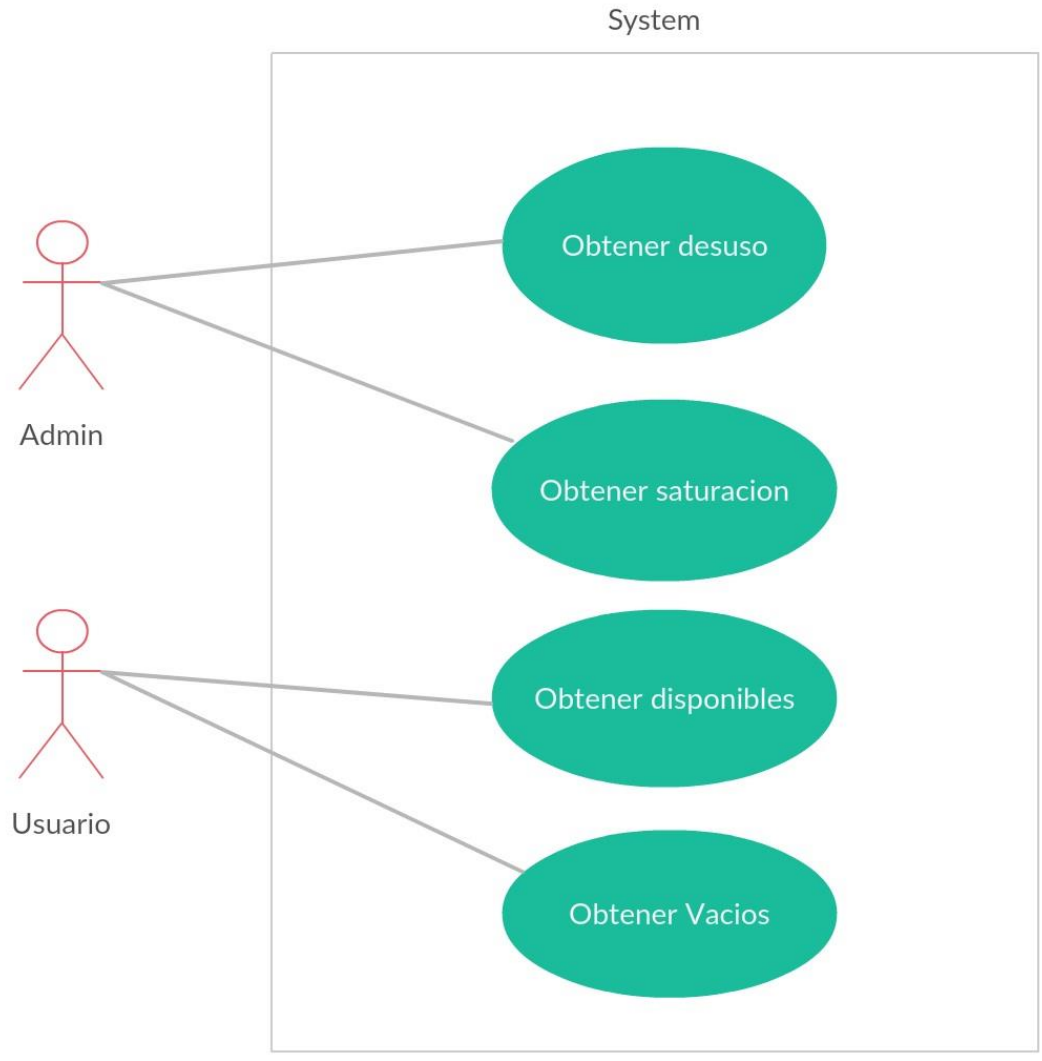


Figura 1 - Casos de uso

3.2 Identificación y análisis de las soluciones posibles

Las posibles soluciones que se han obtenido para la resolución del problema han sido las siguientes: creación de una única imagen Docker, que contara con todo el software necesario para la resolución del problema instalado, o la creación de 3 imágenes Docker una para cada parte estructural del problema.

La solución mediante la creación de una imagen Docker, se construiría basándose en las ideas de construcción de software como un monolito, es decir, todo el sistema estaría centrado en una máquina y resultaría en una imagen más pesada. Como ventaja obtendríamos que, resultaría más fácil de programar debido a que se omitirían las comunicaciones entre los elementos de la arquitectura, sin embargo a la hora de revisar fallos, corregirlos, instalar actualizaciones o permitir hacer un escalado, se obtendrían muchas complicaciones y se invertiría mucho tiempo.

La solución mediante la creación de tres imágenes Docker, se construiría basándose en la arquitectura de microservicios[13] es decir, en la separación de las funcionalidades de la aplicación cada una en un servicio y estos servicios en una imagen cada uno. Como desventaja tendríamos que, es un poco más complicada la implementación de este tipo de sistemas debido a las comunicaciones que se hacen entre sus componentes, pero, como ventajas obtendríamos un mejor tiempo y una mayor facilidad de corrección de errores, una mejor disposición a la escalabilidad y una mejor tolerancia a fallos.

En la actualidad, las aplicaciones creadas basadas en un único servicio como la primera solución, están en clara desventaja frente a las aplicaciones distribuidas y de microservicios como la segunda solución.

3.3 Solución propuesta

Para los diferentes problemas se han utilizado diferentes soluciones, puesto que no existe un problema común que pudiera ser resuelto con una única solución.

Como lenguaje de programación que se ha utilizado para implementar la solución, se ha escogido Python, puesto que es un lenguaje muy versátil y de fácil aprendizaje.

Como datos utilizados para el estudio se han estado recopilando durante unas semanas el estado de las estaciones, los cuales han sido guardados en formato JSON por su fácil manejo en el lenguaje escogido.

Fases de la aplicación

La solución de nuestro proyecto se divide en diferentes fases:

Fase de mapeo de datos A

En esta fase se mapean los datos en referencia a las dos primeras funcionalidades, que son, el número de bicis disponibles y el número de sitios de aparcamiento.

Fase de reducción de datos A

En esta fase se hace la reducción de los datos mapeados en el mapeo de datos A, obteniendo los resultados estadísticos que posteriormente serán insertados como registros en la base de datos.

Fase de mapeo de datos B

En esta fase se mapean los datos en referencia a las dos funcionalidades orientadas a la empresa, que son las estadísticas de saturación.

Fase de reducción de datos B

En esta fase se hace la reducción de los datos mapeados en el mapeo de datos B, obteniendo los resultados estadísticos que posteriormente serán insertados como registro en la base de datos.

Fase de inserción de datos

En esta fase se insertan los resultados obtenidos en las dos reducciones en sus respectivas tablas de la base de datos.

Fase de actualización de datos

En esta fase se actualizan los datos de las tablas de la base de datos con los nuevos datos recogidos mediante el servicio REST de la plataforma API APP CIUDAD.

Fase de publicación de datos

Esta fase consiste en la devolución de los datos guardados en la base de datos para cada petición realizada en nuestro servicio web.

Características de la solución

Se han creado tres imágenes diferentes públicas existentes en Docker Hub, una para Hadoop, otra para la base de datos y otra para el servicio web. Estas imágenes están hechas a medida del problema, es decir, no requiere del añadido o modificación de ninguno de sus ficheros o parámetros.

Se ha construido un sistema de directorios con sus Dockerfile[14] correspondientes, para poder lanzar la aplicación mediante el uso de Docker-compose[15].

Como modo de almacenamiento, se ha optado por la creación de dos tablas MySQL, una para para guardar los resultados ofrecidos al usuario común y otra para guardar los resultados de saturación y desuso de los recursos para el usuario administrador.

Como modo de tratar, mapear y reducir los datos se ha optado por la creación de 4 programas: dos Mapper y dos Reduce para obtener los registros que irán en cada tabla.

Como sistema operativo de las imágenes Docker referentes al análisis y publicación de los datos, se ha utilizado la distribución Ubuntu 16.04.

La imagen creada para contener una base de datos ha sido basada en la imagen mysql5.7.2 del registry de Docker Hub[16].

Como modo de publicación, se ha realizado un formulario HTML para poder poner los datos que serán proporcionados en el formato JSON.

Como modo de validación, se iniciará el sistema en una máquina, se realizarán los 4 casos de uso para los que cumple su funcionalidad y se verificará que los datos no contienen errores.

4. Diseño de la solución

El sistema requiere de tres máquinas o nodos, en este caso tres contenedores Docker para poder establecer las tres imágenes creadas para el trabajo. Estos contenedores interactuarán entre ellos para completar el correcto funcionamiento del proyecto.

4.1 Arquitectura del sistema

La aplicación presenta esta arquitectura a nivel estructural, no obstante, para la vista de los actores destinados a utilizarla, se ve como una arquitectura cliente-servidor puesto que, la interacción entre la aplicación y el usuario se realiza mediante una página web utilizando comunicaciones con protocolo HTTP.

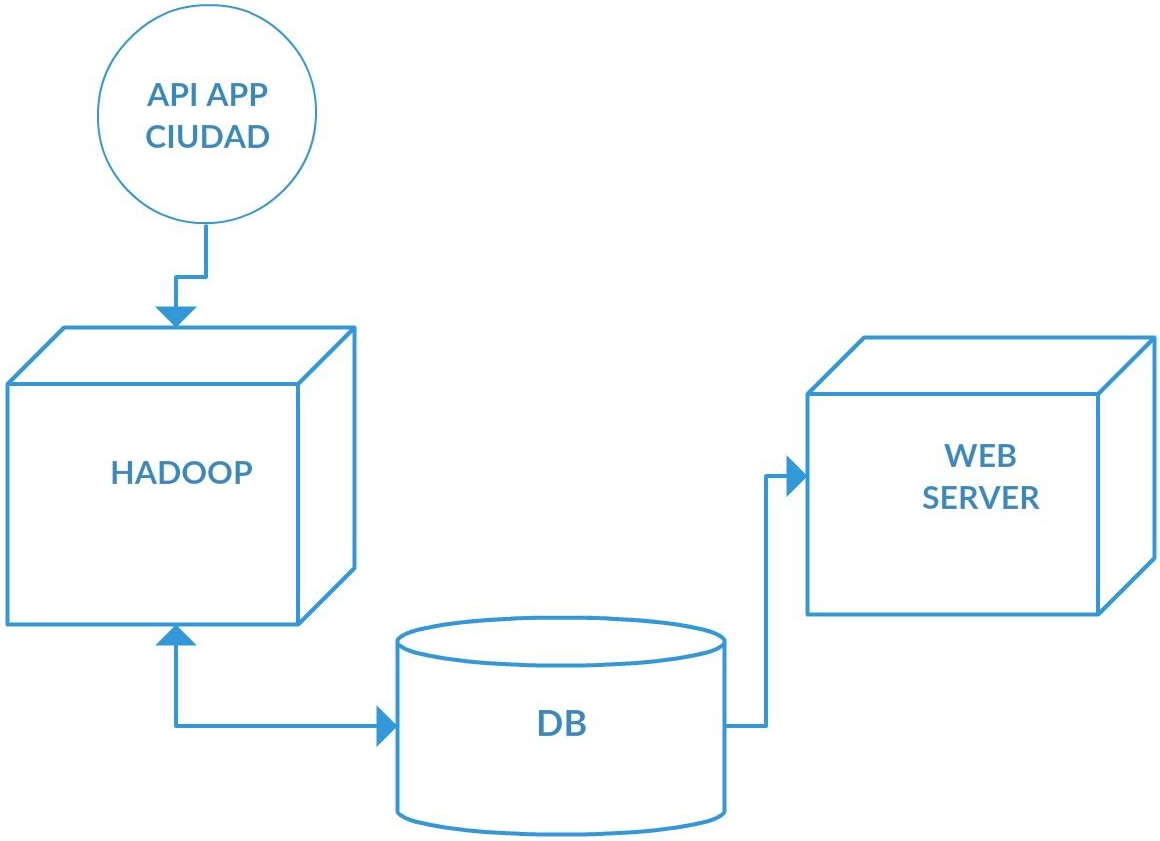
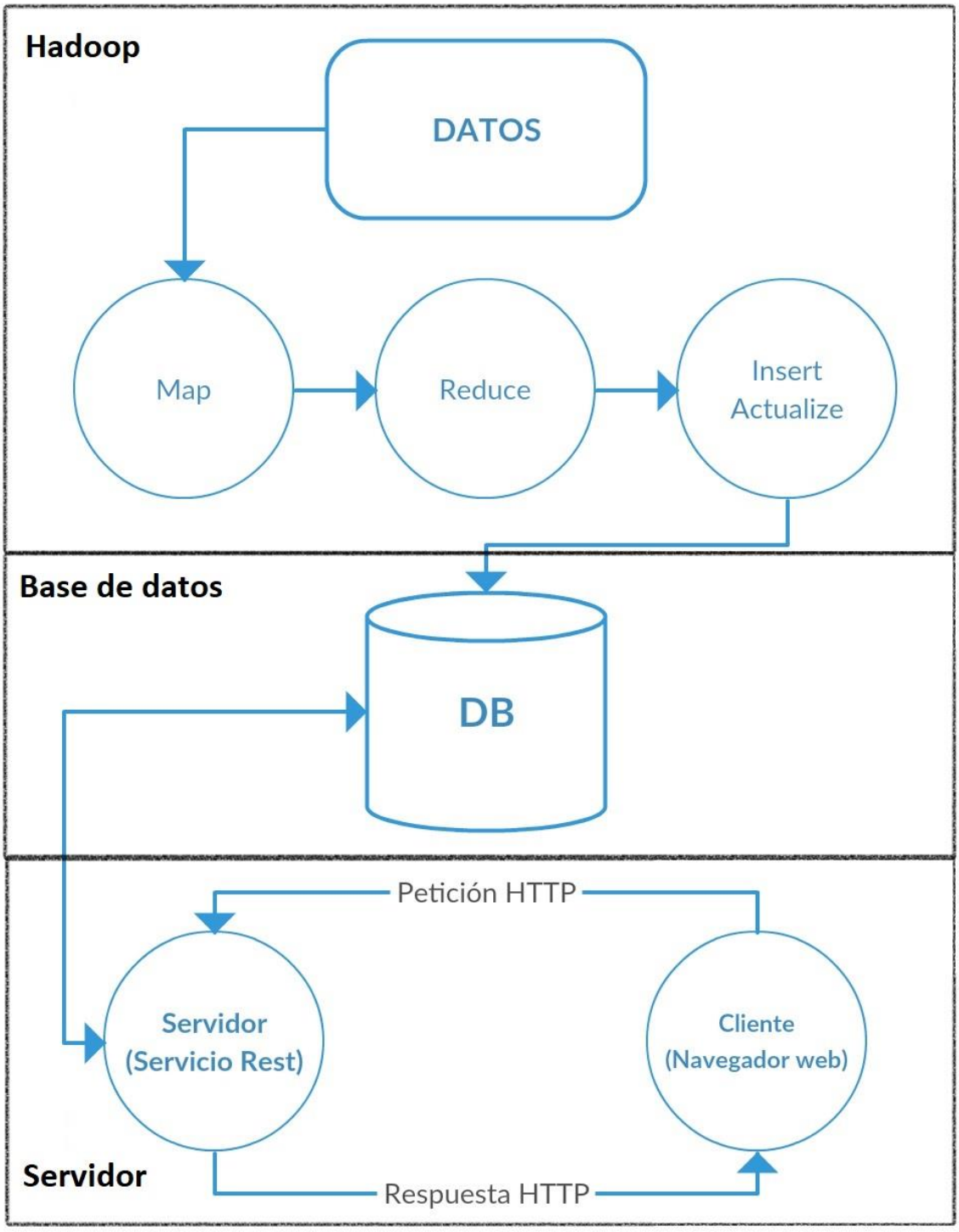


Figura 2 - Arquitectura del sistema

4.2 Diseño detallado

La arquitectura del sistema se divide en 3 grandes partes, que coinciden con las



3 imágenes preparadas para la aplicación, las cuales son las siguientes:

Figura 3 – Arquitectura desglosada del sistema

En primer lugar tenemos la imagen Hadoop (*Figura 3)*, como se puede observar, esta parte del proyecto se puede dividir en tres subpartes una encargada de realizar el Map, otra encargada de realizar el Reduce y la otra encargada de realizar las inserciones y las actualizaciones a partir de los elementos obtenidos en las otras partes.

En segundo lugar, tenemos la imagen MySQL (*Figura* 3), como estructura en esta base de datos, contamos con dos tablas diferentes las cuales contienen la siguiente estructura:

Tabla para la ocupación de las estaciones:

nEstacion	diaSem	Hora	Min	Disp.	Libr.	Lat	Lon	Año	Mes	media

Tabla para el estudio de la saturación:

nEstacion	diaSem	Hora	Min	diasSat	diasVacio	Lat	Lon	Año	Mes	media

En referencia a la ocupación de memoria de las tablas en la base de datos, se ha calculado que cada tabla contaría con el siguiente número de registros:

Para realizar el estudio se han recogido muestras del estado de las estaciones cada 15 minutos, lo que nos lleva a 4 muestras/hora por estación, teniendo en cuenta que el estudio se ha realizado sobre 100 estaciones diferentes de Valenbisi debido a las limitaciones de la plataforma que proporciona esta información, obtenemos 67200 registros por tabla.

$$7\frac{dias}{semana} * 24\frac{horas}{dia} * 4\frac{escrituras}{hora} * 100\ estaciones = 67200\ registros$$

En este caso, se coge como base de tiempo para el estudio, los diferentes días de la semana con horas determinadas, debido a lo cual se obtiene este número de registros.

Como se puede observar en la ecuación, no es un tamaño grande y se puede ver claramente que cuantas más estaciones se incorporen a nuestro sistema el número de registros aumentará linealmente. Así pues, el sistema tiene incorporada una función de actualización de los datos, que únicamente requerirá la modificación de 100 registros cada 15 minutos.

En último lugar, tenemos la imagen para el servidor web (*Figura 3)*, esta parte es la que tiene el contacto directo con los actores implicados en la aplicación, por este mismo motivo, obtenemos que esta parte de la aplicación contiene una propia arquitectura basada en el modelo Cliente-Servidor. Como podemos observar cliente y servidor se comunican mediante el protocolo HTTP.

El cliente solicita cierta información mediante una petición con ciertos parámetros utilizando el formulario proporcionado en la interfaz web. El servidor recibe esa petición, obtiene los parámetros enviados en la petición, utiliza dichos parámetros para realizar la consulta a la base de datos y devuelve los datos obtenidos en la respuesta.

4.3 Tecnología utilizada

En los siguientes apartados se expone las tecnologías utilizadas necesarias para realizar correctamente el proyecto.

Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multi-paradigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, de tipado fuerte y dinámico y multiplataforma.

En este proyecto se ha utilizado la versión Python 2.7

Apache Hadoop

Apache Hadoop es un framework que permite el procesamiento distribuido de grandes conjuntos de datos en grupos de computadoras utilizando modelos de programación simples.

Está diseñado para escalar desde servidores individuales a miles de máquinas, cada una ofrece computación y almacenamiento locales. En lugar de confiar en el hardware para ofrecer alta disponibilidad, la biblioteca en sí está diseñada para detectar y manejar fallos en la capa de la aplicación, por lo que ofrece un servicio de alta disponibilidad sobre un grupo de computadoras, cada una de las cuales puede ser propensa a fallos.

La versión de Apache Hadoop utilizada en este proyecto ha sido la 2.9.2

Hadoop Streaming

Hadoop Streaming es una característica de Apache Hadoop que permite escribir programas bajo el modelo de programación MapReduce independientemente del lenguaje de programación utilizado.

Consiste en escribir programas que implementan la función map y la función reduce, de manera que lean texto de la entrada estándar y escriban el resultado por la salida estándar, de ahí que el lenguaje en el que estén implementadas no sea relevante.

Ubuntu

Ubuntu es un sistema operativo de código abierto para computadores. Es una distribución de Linux basada en la arquitectura de Debian. Actualmente corre en computadores de escritorio y servidores, en arquitecturas Intel, AMD y ARM.

Está orientado al usuario promedio, con un fuerte enfoque en la facilidad de uso y en mejorar la experiencia del usuario. Está compuesto de múltiple software normalmente distribuido bajo una licencia libre o de código abierto.

En este proyecto se ha utilizado la versión de Ubuntu 16.04

Docker

Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos. Docker utiliza características de aislamiento de recursos del kernel Linux, tales como cgroups y namespaces para permitir que "contenedores" independientes se ejecuten dentro de una sola instancia de Linux, evitando la sobrecarga de iniciar y mantener máquinas virtuales.

MySQL

MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la base datos de código abierto más popular del mundo, y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web.

La versión utilizada en este proyecto ha sido MySQL 5.7.2

Docker Compose

Docker Compose es una herramienta que permite simplificar el uso de Docker, generando scripts que facilitan el diseño y la construcción de servicios. Con Docker Compose puedes crear diferentes contenedores y al mismo tiempo, en cada contenedor, diferentes servicios. En vez de utilizar una serie de comandos bash y scripts, Docker Compose implementa recetas similares a las de Ansible, para poder instruir al engine a realizar tareas.

5. Desarrollo de la solución

5.1 Descripción de la aplicación

La aplicación está dividida en 3 grandes núcleos que son las tres principales funcionalidades del proyecto: procesado de los datos, almacenamiento de los datos y publicación de los datos.

Procesado de los datos

Esta parte está dividida en tres subpartes: la parte del MapReduce, la parte de inserción y la parte de actualización.

Partimos de un dataset recolectado obtenido a partir de la plataforma Valenbisi.

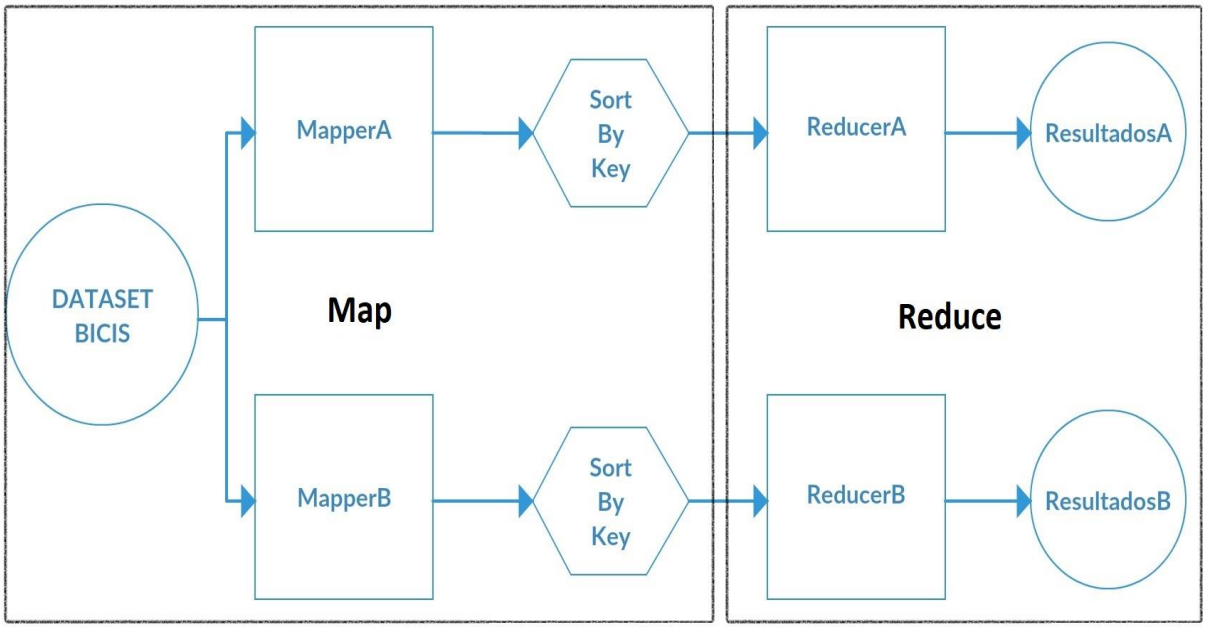


Figura 4 – Esquema del procesado de datos

La parte del Map (*Figura 4*), se ha completado con la implementación de dos programas que realizan el mapeo de los datos, estos programas parsean el dataset en datos en un formato específico para la aplicación.

Seguidamente el resultado de estos programas es ordenado a partir de los 4 primeros campos de cada objeto, que constan en el caso del MapperA de:

nEstacion,diaSem,Hora,Min,disponibles,libres,latitud,longitud,Año,Mes

Y en el caso del MapperB de:

nEstacion,diaSem,Hora,Min,latitud,longitud,Año,Mes,estado

Seguidamente los datos proporcionados como salida en la parte del Map y ya ordenados, son los que se utilizan como datos de entrada en la parte del Reduce.

La parte del Reduce *(Figura 4*), se ha completado con la implementación de dos programas que realizan la reducción de los datos, estos programas obtienen los resultados estadísticos a partir de los datos de salida de la parte del mapeo.

Estos programas obtienen como resultados objetos con esta estructura:

ReducerA

nEstacion,diaSem,Hora,Min,disponibles,libres,latitud,longitud,año,mes,media

ReducerB

nEstacion,diaSem,Hora,Min,diasSat,diasVacio,latitud,longitud,año,mes,media

Los resultado obtenidos en la parte del reduce son guardados en dos ficheros de texto, uno para los datos del ReducerA y otro para los datos del ReducerB.

Los datos obtenidos en la fase de reduce son los que sirven como datos de entrada a la fase de inserción.

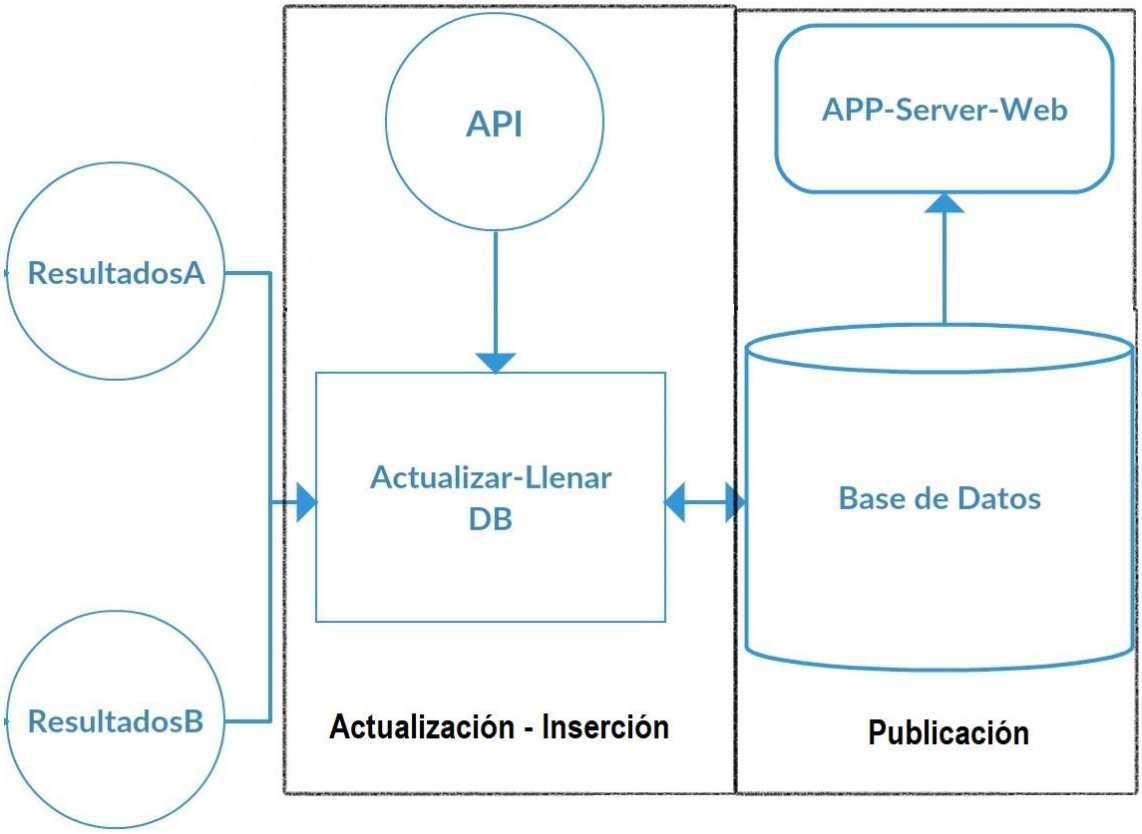


Figura 5 - Esquema de actualización y publicación

La fase de la inserción (*Figura 5*), se ha completado con la implementación de un programa que obtenga todos los resultados del análisis de los datos de los dos ficheros creados en la fase de Reducción y cree las dos tablas en la base de datos e inserte en estas los datos correspondientes al análisis.

La parte de actualización (*Figura 5*), se ha completado al realizar una petición REST al servicio API APP CIUDAD para obtener el estado de las estaciones, posteriormente se realiza una búsqueda de la base de datos y se obtienen los datos sin actualizar, se aplican los cálculos necesarios y se actualizan los correspondientes registros en la base de datos.

Almacenamiento de los datos

En esta parte (*Figura 5*) se construye una imagen basada en MySQL que sirve para el almacenamiento de los datos, esta base de datos es accedida por las otras dos partes principales de la aplicación, tanto para guardar datos, como para ser accedidos y publicados.

Publicación de los datos

En esta parte (*Figura 5*) se utiliza el framework web para Python denominado Flask conjuntamente con WTForms[17],que permite la creación de una aplicación web con un formulario para realizar peticiones REST, a partir de estas peticiones, obtenemos los parámetros necesarios para hacer las consultas a la base de datos y devolver los resultados obtenidos.

5.2 Descripción del flujo de la aplicación

En el siguiente apartado, podremos ver los flujos correspondientes a las acciones realizadas para la realización de la aplicación.

Tratamiento y análisis de los datos

Como primer caso, veremos el diagrama de flujo correspondiente a la parte de la aplicación encargada de realizar el tratamiento, análisis de los datos y la posterior obtención de los resultados generados.

En este proyecto, como se ha indicado anteriormente, se ha optado por el modelo de programación MapReduce para realizar este cometido.

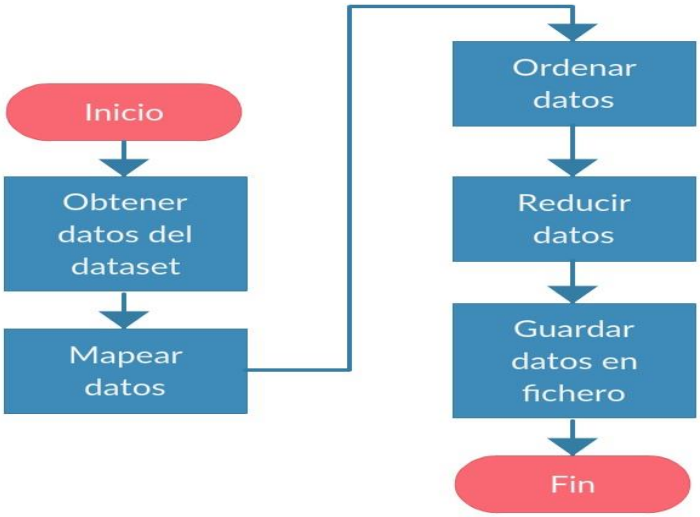


Figura 6 - Flujo del procesado de datos

Como podemos observar, este proceso de la aplicación consta de diferentes acciones:

En primer lugar, obtenemos los datos del dataset, previamente adquirido mediante la recolección de datos de la plataforma.

A continuación, estos datos deben ser mapeados, es decir, moldeados o convertidos en datos cuya estructura general sea adecuada para poder tratarlos en nuestra aplicación.

Como se ha utilizado el modelo de programación MapReduce, se impone que los datos generados en la actividad del Map deban ser ordenados a partir de cierta clave o claves. En nuestro caso son los elementos nEstacion, diaSem, Hora, Min.

Una vez los datos estén ordenados, serán transmitidos por entrada estándar al programa encargado de hacer el Reduce.

En la parte del Reduce, a los datos se les aplican ciertos análisis y funciones, de las que obtenemos todos los 67200 registros (por cada programa del Reduce), que son los que serán insertados en la base de datos.

Finalmente, estos datos generados son guardados en el sistema de ficheros del contenedor Docker correspondiente, contenedor Hadoop en este caso.

Inserción y actualización de la base de datos

Una vez terminado el flujo de la aplicación para el proceso de obtención de los resultados, continuamos con el proceso de inserción de los datos generados en este.

En este apartado, podremos contemplar el flujo de la aplicación para los procesos de inserción y actualización, este flujo es el correspondiente a ambos procesos debido a que estos están implementados en el mismo código.

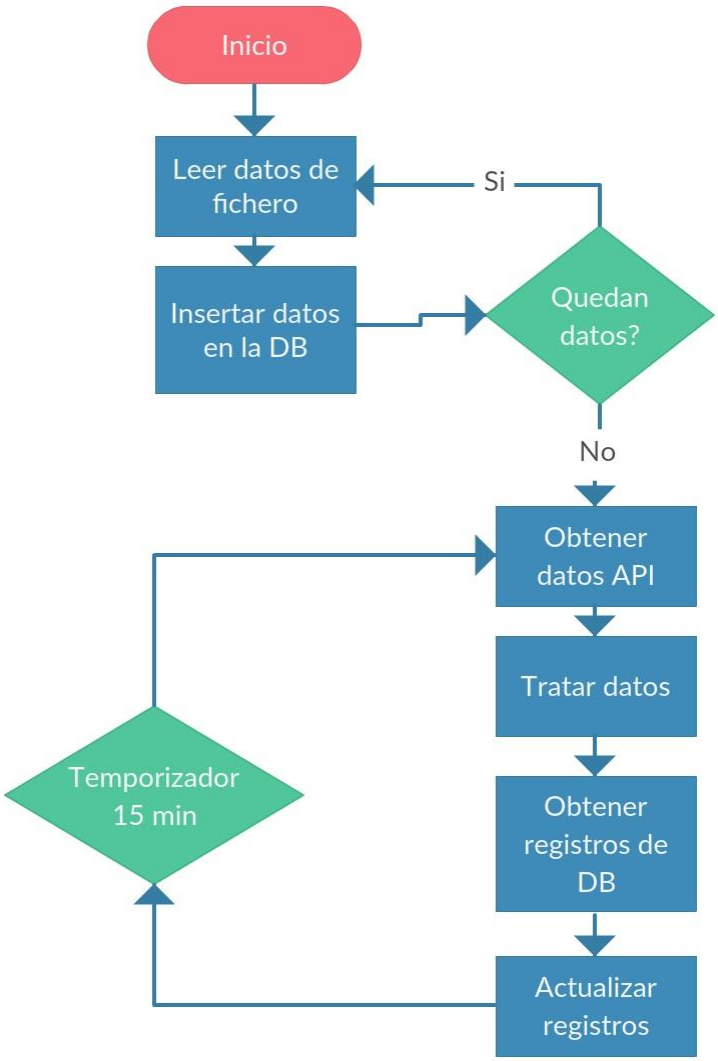


Figura 7 - Flujo de la inserción y actualización de los datos

Como podemos observar en el siguiente diagrama, este proceso de la aplicación da comienzo una vez terminado el proceso de MapReduce.

Esta parte de la aplicación consta de diferentes procesos:

En primer lugar, se realiza una lectura de los ficheros generados por los Reduce, cuyos datos van siendo añadidos registro a registro.

Este proceso se repite hasta que se ha hecho la correcta inserción de todos los registros en la base de datos, cabe recalcar que este proceso se realizará completamente dos veces, puesto que se han generado dos ficheros de resultados en el apartado anterior, correspondientes a las dos tablas de la base de datos.

Una vez se ha comprobado que no quedan más datos por leer e insertar, la aplicación pasa al apartado de actualización de la base de datos.

Este proceso, empieza con la obtención de los datos en tiempo real de las estaciones de Valenbisi mediante una petición REST a la API APP CIUDAD.

Estos datos contienen más información de la que nosotros requerimos, por este motivo, son tratados para obtener únicamente los datos que son relevantes para nuestra aplicación.

Seguidamente, se realiza una búsqueda en la base de datos en la que obtenemos los registros que van a ser actualizados.

Una vez obtenidos estos registros, se aplican las funciones correspondientes a la actualización, utilizando los datos de estos y los datos actualizados de la API y estos resultados se aplican como actualización a los registros implicados.

Como se puede observar en el diagrama anterior, el flujo de este proceso no termina nunca, puesto que la base de datos requiere de ser actualizada para que los datos sean correctos. Se ha decidido que el proceso de actualización de datos se realizara cada 15 minutos.

Responder petición

Una vez en marcha el proceso de Inserción-Actualización ponemos en marcha el servicio web encargado de la publicación de los datos.

Este proceso es el que interactúa directamente con el usuario, y también consta de diferentes pasos para su correcta realización, como se muestra en el siguiente diagrama de flujo:

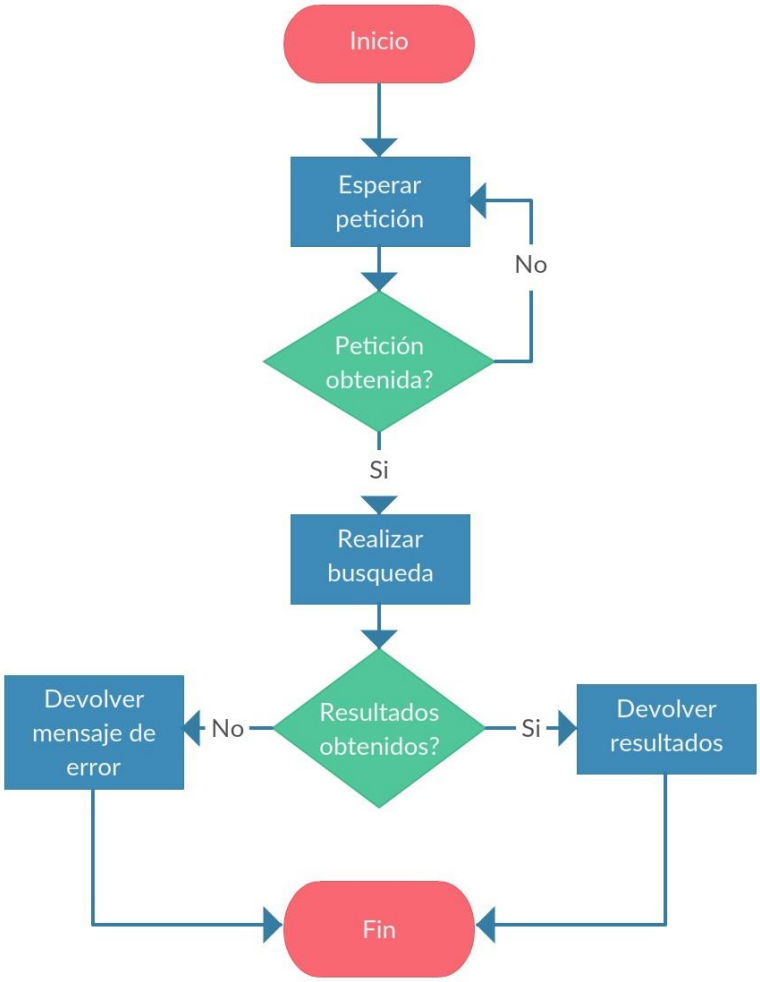


Figura 8 - Flujo de respuesta de bicis

Como podemos observar, este proceso queda a la espera de peticiones HTTP realizadas por el usuario.

En el caso de que el servidor reciba una petición y esta petición sea correcta se procede a realizar la búsqueda de los datos.

La petición consta de los siguiente parámetros: latitud, longitud, acción que se quiere realizar, día de la semana y hora (hora y minuto). De los anteriores parámetros son obligatorios latitud, longitud y acción.

En el caso de no rellenar los parámetros no obligatorios, la aplicación dará como valores por defecto a estos parámetros el día y la hora a la que se ha hecho la petición

El proceso de búsqueda en este caso, se realiza sobre la tabla con información destinada a los usuarios, es decir, la correspondiente al análisis de sitios libres y bicicletas disponibles.

A partir de los parámetros enviados por el usuario con la petición, se realiza la búsqueda en la base de datos de las 10 estaciones más cercanas a la latitud y longitud enviadas. Se comprueba que las estaciones cumplan con las condiciones mínimas establecidas por el programador, en este caso, se devolverán únicamente las estaciones cuyos datos analizados, den como resultado una buena probabilidad de que la acción del usuario pueda cumplirse.

Como ejemplo, para un usuario que quiera coger una bici, se devolverán estaciones en las que a la hora establecida la probabilidad de bicis disponibles es alta.

Cabe la posibilidad de que dados unos parámetros con una hora y día determinados, no se encuentren resultados que satisfagan dicha petición. En dichos caso, como se observa en el diagrama, se responderá con un mensaje al cliente de Error/Información en el que se le notificara que no se han podido encontrar resultados válidos.

Obtener informe saturación

Como se ha podido ver a lo largo de la memoria del proyecto, se han obtenido resultados en referencia a la saturación y a la no utilización de los recursos disponibles en las estaciones.

Estos análisis, se han realizado pensando en obtener información valida no solo para el usuario normal, sino también en el administrador de los recursos de la plataforma.

Como resultado, se ha creado una función similar a la anterior, en la que devolvemos la información referente a los informes de saturación y desuso de las estaciones.

Este proceso también contiene diferentes pasos a realizar para su correcto funcionamiento, los cuales vienen indicados en el siguiente diagrama:

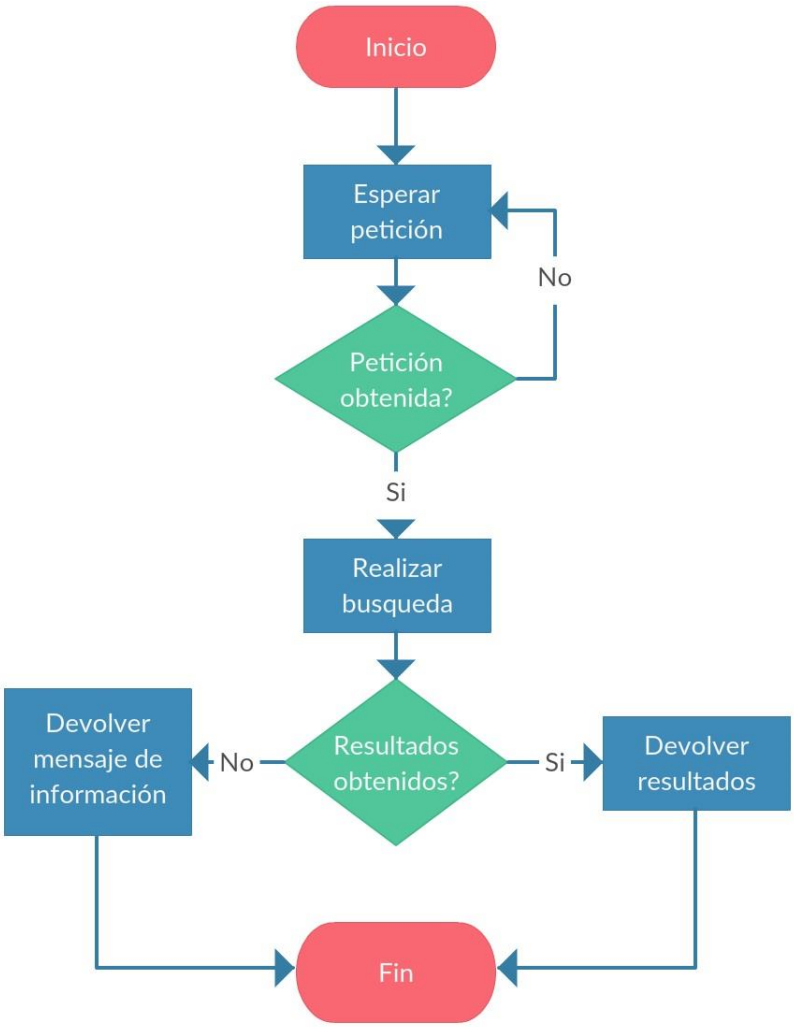


Figura 9 - Flujo de respuesta de saturación

Como podemos observar, este proceso queda a la espera de peticiones HTTP realizadas por el Administrador.

En el caso de que el servidor reciba una petición y esta petición sea correcta, se procede a realizar la búsqueda de los datos.

La petición consta de los siguientes parámetros: tipo de informe que se quiere obtener, ya sea informe de saturación o de desuso, junto con la contraseña de acceso a la base de datos del administrador. Ambos parámetros son obligatorios.

A partir de la verificación de la contraseña del administrador, se realiza la consulta a la base de datos de la cual sacaremos los resultados a devolver.

Si existen estaciones que cumplan las condiciones de saturación o desuso, estas son las que se devolverán como resultado, a partir de estos resultados el administrador podrá tomar las decisiones que crea oportuno para su producto.

Cabe la posibilidad de que ninguna estación cumpla dichas condiciones, en cuyo caso, se devolverá un mensaje de información que explique la situación.

6. Implantación y Pruebas

Implantación

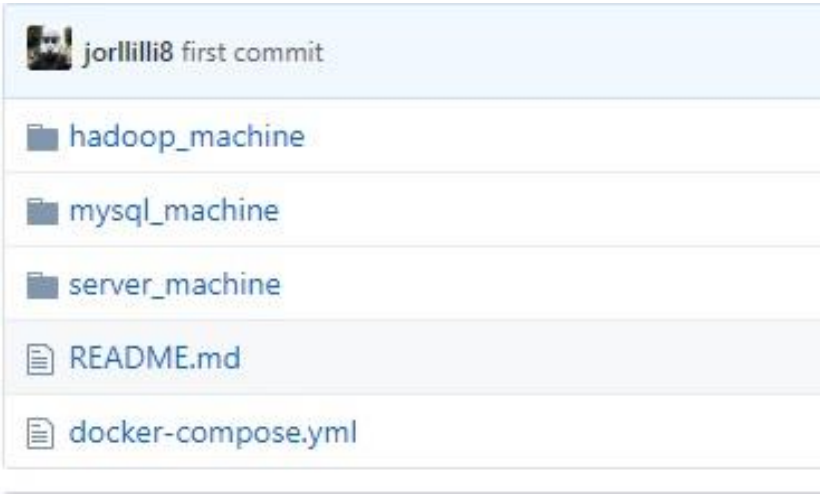
Para la implantación de nuestra solución en cualquier máquina se requiere, que dichas maquinas contengan instalada una de las últimas versiones estables de Docker, puesto que se requiere este software para poder poner en funcionamiento la aplicación.

También se requiere de la instalación de docker-compose puesto que el lanzamiento de esta aplicación se realiza con dicha tecnología.

Se ha creado un fichero con extensión .yml[18] para la definición de los distintos servicios.

```
version: '2'
services:
      hadoop:
         build: ./hadoop_machine/
         links:
          - db
         environment:
          - URL_DB=tcp://db:3306
         volumes:
          - "/var/run/docker.sock:/var/run/docker.sock"
      server:
         build: ./server_machine/
         ports:
          - 8888:5000
         links:
          - db
         environment:
          - URL_DB=tcp://db:3306
         volumes:
          - "/var/run/docker.sock:/var/run/docker.sock"
      db:
         build: ./mysql_machine/
         ports:
          - 8008:3306
         volumes:
          - "/var/run/docker.sock:/var/run/docker.sock"
```

Debido a la utilización de docker-comopose, es necesaria la disposición de los distintos ficheros Dockerfile en sus correspondientes directorios, para el correcto funcionamiento del despliegue de los componentes.

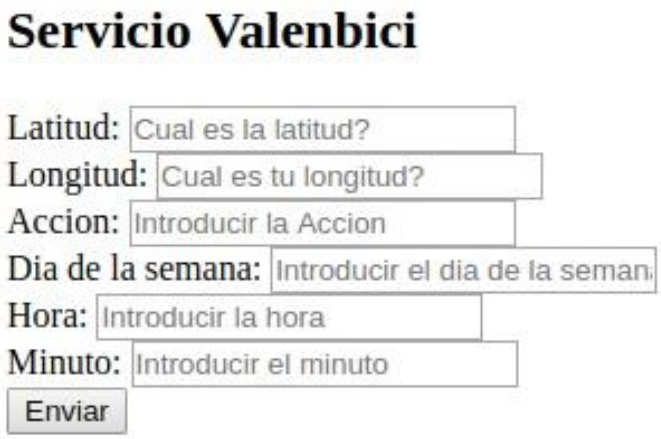


Se ha puesto a disposición de toda la estructura de ficheros en GitHub[19].

Así pues, se requerirá de acceso a internet por parte de los componentes de la aplicación en referencia a las comunicaciones entre estos y también para la fase de actualización de los datos.

Una vez puesto en marcha el proceso de inicialización de la aplicación, se requiere de un tiempo entre 15 y 20 minutos para poder contar con resultados fiables en las consultas, puesto que el proceso de análisis y de inserción de todos los registros en la base de datos ocupa ese intervalo de tiempo.

Cuando el sistema esté en correcto funcionamiento, podremos acceder a los datos mediante un formulario como los que se detallan a continuación o mediante una petición a la API:



Como comprobación de la correcta implantación de la aplicación, se ha verificado consultando la estructura de la base de datos:

Base de datos 1

nEstacion	diaSem	hora	min	latitud	longitud	disponibles	libres	year	mes	media

Base de datos 2

nEstacion	diaSem	hora	min	latitud	longitud	diasSat	diasVacio	year	mes	media

Pruebas

Para comprobar el correcto funcionamiento de la aplicación, se ha requerido de la realización distintas pruebas, dichas pruebas han consistido en la comprobación de la correcta ejecución de los 4 casos de uso, el visionado de la arquitectura a nivel de red de la aplicación, el visionado de los datos en la base de datos, y el estado de los contenedores y sus puertos.

Como primera comprobación, se ha visualizado el estado de los contenedores pertenecientes al sistema y los puertos expuestos de estos.

```
tfm-compose_server   "/bin/sh -c 'python …"   47 seconds ago      Up 42 seconds       0.0.0.0:8888->5000/tcp
tfm-compose_hadoop   "/bin/sh -c 'python …"   47 seconds ago      Up 43 seconds
tfm-compose_db       "docker-entrypoint.s…"   49 seconds ago      Up 47 seconds       8008/tcp, 0.0.0.0:8008->3306/tcp
```

Para la comprobación de la arquitectura de red se ha utilizado la función network inspect de Docker:


```
[
    {
        "Name": "tfm-compose_default",
        "Id": "0d0f310d789006353e4cdf8c83131d5c10fbf4e7cecd9b023fb865776f2a595c",
        "Created": "2019-06-26T18:41:53.596187763+02:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.20.0.0/16",
                    "Gateway": "172.20.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
```


Como se puede observar en la imagen anterior, la red en la que se trabaja en la aplicación es la 172.20.0.0/16 esta dirección de red es característica de la utilización de docker-compose.


```
"Containers": {
    "b4e54ee093ca534d293e0ac5908ff1f189f002c681a2e4772fb287a18cf5cad7": {
        "Name": "tfm-compose_hadoop_1",
        "EndpointID": "ae44e08317827de788acee6e58bb36953580889d044fcb2ac017c01115f2c78c",
        "MacAddress": "02:42:ac:14:00:03",
        "IPv4Address": "172.20.0.3/16",
        "IPv6Address": ""
    },
    "e2f3ec97903598ee72545479b8ddafe9fe11a9ab7d502e4381155085cc4fc797": {
        "Name": "tfm-compose_db_1",
        "EndpointID": "ff4b1646dada1159df29e92edc943a96a08c0c2d52bac03816238ddd7a30f499",
        "MacAddress": "02:42:ac:14:00:02",
        "IPv4Address": "172.20.0.2/16",
        "IPv6Address": ""
    },
    "f91d40b1538931e0c5bf4802f558190e8f2124861de631ee438b4d3411e2c545": {
        "Name": "tfm-compose_server_1",
        "EndpointID": "dbb173241b0cc25f5d2ec39f6659bc759b6a6453c72c69a2caba85471f6beb51",
        "MacAddress": "02:42:ac:14:00:04",
        "IPv4Address": "172.20.0.4/16",
        "IPv6Address": ""
    }
},
```

Mediante esta función de Docker, podemos ver que en la red creada existen tres elementos o nodos, los cuales son los que se han desplegado en nuestra aplicación. Se puede verificar el correcto despliegue de la aplicación puesto que, los componentes de la red son server, hadoop y db.

Como siguiente comprobación se han realizado 6 diferentes peticiones al servidor que representan todos los resultados posibles, de los casos de uso de la aplicación.

Consulta de estaciones para dejar bicis:

[{'nEstacion': 163, 'libres': 12.0}, {'nEstacion': 164, 'libres': 13.0}, {'nEstacion': 162, 'libres': 5.0}, {'nEstacion': 159, 'libres': 9.0}, {'nEstacion': 71, 'libres': 6.0}, {'nEstacion': 166, 'libres': 14.0}, {'n

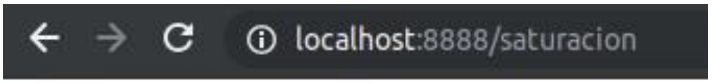
Consulta de estaciones para coger bicis:

[{'nEstacion': 163, 'disponibles': 7.0}, {'nEstacion': 276, 'disponibles': 11.0}, {'nEstacion': 159, 'disponibles': 7.0}, {'nEstacion': 71, 'disponibles': 13.0}]

Consulta de estaciones con saturación A:

[{'Hora': 9, 'nEstacion': 400, 'diaSem': u'Monday', 'Min': 30}, {'Hora': 9, 'nEstacion': 401, 'diaSem': u'Monday', 'Min': 30}, {'Hora': 9, 'nEstacion': 402, 'diaSem': u'Monday', 'Min': 30}]

Consulta de estaciones con saturación B:



No hay estaciones saturadas

Consulta de estaciones en desuso A:

[{'hora': 9, 'nEstacion': 400, 'diaSem': u'Monday', 'min': 30}, {'hora': 9, 'nEstacion': 401, 'diaSem': u'Monday', 'min': 30}, {'hora': 9, 'nEstacion': 402, 'diaSem': u'Monday', 'min': 30}]

Consulta de estaciones en desuso B:

Como se puede ver, se han obtenido resultados satisfactorios en referencia a las pruebas del correcto funcionamiento de los casos de uso de la aplicación.

Como pruebas finales, se ha realizado una visualización de los estados de la base de datos, en este caso de las dos tablas correspondientes a la aplicación.

Base de datos A:

nEstacion	diaSem	hora	min	latitud	longitud	disponibles	libres	year	mes	media
100	Wednesday	0	0	39471634	-338150	17	1	2019	6	4
100	Wednesday	0	15	39471634	-338150	17	1	2019	6	4
100	Wednesday	0	30	39471634	-338150	17	1	2019	6	4
100	Wednesday	0	45	39471634	-338150	16	1	2019	6	4
100	Wednesday	1	0	39471634	-338150	16	2	2019	6	4
100	Wednesday	1	15	39471634	-338150	16	2	2019	6	4
100	Wednesday	1	30	39471634	-338150	16	2	2019	6	4
100	Wednesday	1	45	39471634	-338150	16	2	2019	6	4
100	Wednesday	2	0	39471634	-338150	16	2	2019	6	4
100	Wednesday	2	15	39471634	-338150	16	2	2019	6	4
100	Wednesday	2	30	39471634	-338150	16	2	2019	6	4
100	Wednesday	2	45	39471634	-338150	16	1	2019	7	4
100	Wednesday	3	0	39471634	-338150	16	1	2019	6	4
100	Wednesday	3	15	39471634	-338150	16	2	2019	6	4
100	Wednesday	3	30	39471634	-338150	15	2	2019	6	4
100	Wednesday	3	45	39471634	-338150	15	2	2019	6	4
100	Wednesday	4	0	39471634	-338150	15	2	2019	6	4

Base de datos B:

nEstacion	diaSem	hora	min	latitud	longitud	diasSat	diasVacio	year	mes	media
10	Wednesday	0	0	39470816	-382610	0	0	2019	6	1
10	Wednesday	0	15	39470816	-382610	0	0	2019	6	1
10	Wednesday	0	30	39470816	-382610	0	0	2019	6	1
10	Wednesday	0	45	39470816	-382610	0	0	2019	6	1
10	Wednesday	1	0	39470816	-382610	0	0	2019	6	1
10	Wednesday	1	15	39470816	-382610	0	0	2019	6	1
10	Wednesday	1	30	39470816	-382610	0	0	2019	6	1
10	Wednesday	7	45	39470816	-382610	0	0	2019	6	2
10	Wednesday	8	0	39470816	-382610	0	0	2019	6	4
10	Wednesday	8	15	39470816	-382610	0	0	2019	6	3
10	Wednesday	8	30	39470816	-382610	0	0	2019	6	2
10	Wednesday	8	45	39470816	-382610	0	0	2019	6	2
10	Wednesday	9	0	39470816	-382610	0	0	2019	6	3
10	Wednesday	9	15	39470816	-382610	0	0	2019	6	4
10	Wednesday	9	30	39470816	-382610	0	0	2019	6	3
10	Wednesday	9	45	39471634	-338150	0	0	2019	6	4
10	Wednesday	10	0	39470816	-382610	0	0	2019	6	4
10	Wednesday	10	15	39470816	-382610	0	0	2019	6	4
10	Wednesday	10	30	39470816	-382610	0	0	2019	6	4

8. Conclusiones

Los objetivos propuestos en la introducción de esta memoria han sido completados satisfactoriamente, tanto los principales como los secundarios.

Se ha completado correctamente la creación de la imagen Docker con Hadoop para el tratamiento de los datos.

Se ha completado correctamente la creación de la imagen Docker para realizar la función de servidor.

Se ha completado correctamente la creación de la imagen Docker para el almacenamiento de los resultados obtenidos.

El dataset correspondiente a la recolección de datos de la plataforma Valenbisi ha sido correctamente creado.

Los objetivos referentes al apartado del tratamiento y análisis de los datos del dataset, se han completado correctamente mediante la implementación de los siguientes programas:

MapperA: responsable del mapeado de los datos para el primer análisis de datos de la solución del problema.

ReducerA: mediante la implementación de este programa, se ha conseguido hacer la reducción y guardado de los datos de salida proporcionados por el MapperA.

MapperB: responsable del mapeado de los datos para el segundo análisis de datos de la solución del problema.

ReducerA: mediante la implementación de este programa, se ha conseguido hacer la reducción y guardado de los datos de salida proporcionados por el MapperB.

Como continuación de la resolución de los objetivos, se ha implementado correctamente un código capaz de leer los datos generados en el proceso de MapReduce, realizado por los 4 componentes anteriores, y guardarlos en la base de datos del contenedor db.

A su vez, se ha completado el siguiente objetivo mediante la implementación del código anterior, puesto que una vez terminado el proceso de almacenamiento, este programa es el encargado de realizar la actualización de los registros de la base de datos.

Se ha implementado un código en la imagen del servidor para la publicación de los datos guardados en la base de datos.

Se ha establecido una correcta disposición de los ficheros y los directorios para poder ejecutar la aplicación mediante el uso de docker-compose y también la creación de un fichero .yml para poder definir el despliegue de esta.

El trabajo realizado se ha visto afectado por algunas limitaciones, como son la de tiempo y la dificultad de obtención de los datos de la plataforma. Estas limitaciones han afectado en una pequeña parte al desarrollo final de la solución puesto que hay ciertos aspectos de la aplicación que hubieran sido realizados de una manera más eficiente y atractiva.

En primer lugar, debido a las limitaciones de tiempo se ha desarrollado un servidor web orientado a la funcionalidad y se ha dejado de lado la parte estética de los componentes de la página web. También debido a estas limitaciones se ha optado por realizar un servicio orientado a proporcionar los datos obtenidos en forma de respuesta de API REST.

Por otro lado, en referencia a las limitaciones obtenidas por parte de la plataforma, se encuentran: el estado de los datos de la plataforma, la dificultad de obtención y la rapidez a la hora de proporcionar acceso a dichos datos.

Desglosando los anteriores problemas, en el caso de los datos proporcionados por la plataforma, se puede observar que cuenta con limitaciones en el tamaño de los campos de latitud y longitud. El tamaño de los datos en dichos campos es de 8 y 7 caracteres respectivamente, además añadir que el campo de longitud únicamente contiene 6 caracteres numéricos, puesto que en la ciudad de valencia la longitud siempre viene representada por un número negativo.

En el caso de la dificultad de obtención, se encontraron estas limitaciones puesto que como máximo se puede obtener el estado de 50 estaciones por petición a la API. Debería proporcionarse un método para poder obtener el estado de todas las estaciones operativas, para que el análisis fuera más eficiente, debido a esta limitación se ha optado por realizar este trabajo con 100 estaciones.

Por último, para el acceso a los datos de la plataforma es necesaria la creación de un usuario con su respectiva contraseña por parte de los administradores de la plataforma, para ello se dispone de un formulario en la página web del Ayuntamiento de Valencia, el problema viene dado por que el tiempo de respuesta de dicha función ha sido de entre 2 y 3 semanas. Debido a esto, se ha limitado la cantidad de datos que han sido recogidos en la creación del dataset que ha servido para poner en funcionamiento y de entrenamiento de la aplicación.

En este proyecto he podido obtener conocimientos tanto de ámbito profesional como de ámbito personal.

Como se ha mencionado anteriormente en esta memoria, este proyecto se ha implementado con el lenguaje de programación Python como lenguaje base. Se ha escogido este lenguaje debido a la gran popularidad que tiene en estos

momentos y también por recomendación de los tutores del TFM. Se ha obtenido un nivel alto en la comprensión y desarrollo de código en este lenguaje.

Se ha podido obtener una mejor comprensión de la tecnología Docker y docker-compose.

Se ha obtenido un nivel alto en la gestión y creación de bases de datos asi como en la utilización del lenguaje SQL.

Se ha obtenido una mejor comprensión sobre el sistema operativo Ubuntu al realizar todo el trabajo en este.

Como conocimientos de ámbito personal, se ha podido obtener una visión más cercana a la realización de proyectos de un nivel más elevado y la experiencia obtenida al completar un proyecto solucionando los fallos y problemas que han ido surgiendo.

9. Relación del trabajo desarrollado con los estudios cursados

Este trabajo realizado ha requerido de la utilización de los conocimientos adquiridos durante el transcurso del master.

En este proyecto se ha utilizado la tecnología Docker, esta tecnología es la base de asignaturas de la parte del master de programación en la nube como son Cloud Computing (CC), Programación en Sistemas Cloud (PSC) o Plataformas de Gestión de Contenedores (PGC).

Concretamente los conocimientos en cuanto a la tecnología Docker y a la estructura del proyecto, han sido adquiridos en la asignatura Cloud Computing, puesto que se realizó un proyecto de asignatura cuya estructura y comunicaciones entre sus componentes era muy similar a la de este proyecto.

Se ha requerido de la creación de una arquitectura del sistema separada por funcionalidades, estos conocimientos son los que se han adquirido mediante la realización de asignaturas tales como Diseño y Arquitectura de Sistemas Escalables (DASE), Conceptos de la Computación en Grid y Cloud (CCGC), Infraestructura de Cloud Publico (ICP) y las nombradas en el apartado anterior.

Estas asignaturas han influido en la realización de este proyecto puesto que, en estas asignaturas se prepara al alumno para pensar en programación para sistemas distribuidos, escalables y tolerantes a fallos.

Se ha utilizado API REST cuyos conocimientos de tratamiento, utilización y estructura se han adquirido en la asignatura de Internet of Things (IoT).

Para este proyecto se ha requerido de la utilización de Apache Hadoop, concretamente la funcionalidad de Apache Streaming para el análisis del dataset. Esta herramienta junto con la forma de la gestión, tratamiento y análisis de datos en un dataset, se ha obtenido durante la realización de la asignatura Estrategias y Herramientas de Computación Big Data en la nube (CBD).

10. Trabajos Futuros

Debido al tiempo disponible que he tenido para realizar el trabajo ha quedado pendiente la realización e implementación de algunas funcionalidades y mejoras para la aplicación.

Las mejoras que podrían aplicarse en un futuro a esta aplicación serian:

- Establecer una interfaz web que a partir de los datos generados por la aplicación, se encargara de la publicación de los datos de una manera gráfica, de este modo los usuarios tendrían un modo más cómodo de visualizar los datos.
- Realizar el proyecto sobre el total de estaciones que contiene la plataforma.

Se ha observado durante la realización de este trabajo, que la plataforma API APP CIUDAD se encarga de administrar y publicar datos en referencia a otros sensores de la ciudad y que dichos datos, son similares en estructura a los de este proyecto.

Debido a ello, se han abierto distintas líneas de desarrollo para aplicar los resultados obtenidos en este trabajo a otras áreas, como son:

- Análisis de la congestión de tráfico.
- Análisis del estado de los aparcamientos públicos.
- Análisis de la polución en la ciudad.
- Análisis de humedad para probabilidad de tormentas.

Por último, tenemos las posibles ampliaciones o mejoras que podrían ser aplicables al trabajo, estas mejoras se centrarían en las funcionalidades y en la eficiencia.

En cuanto a las mejoras de eficiencia, una mejora significativa seria realizar las inserciones de los resultados obtenidos en el MapReduce y la posterior actualización de los registros mediante un código en paralelo

En el caso de este proyecto no era muy relevante la implementación de dicho código, puesto que el número distinto de elementos a actualizar era de 100 cada 15 minutos, pero si llegaran a incrementarse los elementos debería de realizarse.

Por otro lado, en cuanto a las mejoras en las funcionalidades, creo que sería muy interesante añadir una funcionalidad para poder visualizar la utilización de Valenbisi por áreas de la ciudad, es decir, poder prever los desplazamientos de

los usuarios de una parte de la ciudad a otra como podría ser de las afueras al centro y viceversa.

11. Referencias

[1] https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional

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

[3] https://hadoop.apache.org/

[4] https://www.python.org/

[5] https://hadoop.apache.org/docs/r1.2.1/streaming.html

[6] https://es.wikipedia.org/wiki/MapReduce

[7] https://www.mysql.com/

[8] https://developer.mozilla.org/es/docs/Web/HTML

[9] http://flask.pocoo.org/

[10] https://www.palletsprojects.com/p/werkzeug/

[11] http://jinja.pocoo.org/docs/2.10/

[12] https://en.wikipedia.org/wiki/Data_set

[13] https://es.wikipedia.org/wiki/Arquitectura_de_microservicios

[14] https://docs.docker.com/engine/reference/builder/

[15] https://docs.docker.com/compose/

[16] https://hub.docker.com/

[17] https://wtforms.readthedocs.io/en/stable/

[18] https://en.wikipedia.org/wiki/YAML

[19] https://github.com/