ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA

CAPTURA Y GESTIÓN DE OPEN DATA EN ENTORNOS DE SMART CITY

ACQUISITION AND MANAGEMENT OF OPEN DATA FOR SMART CITIES

Realizado por
Antonio Fenna Vílchez
Tutorizado por
Ismael Navas Delgado y José Manuel García Nieto
Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, SEPTIEMBRE 2016

Fecha defensa:
El Secretario del Tribunal

Resumen

Este trabajo va enfocado a la aplicación de Big Data en Smart Cities (Ciudades Inteligentes). Se pretende facilitar la gestión el tráfico de una ciudad y en este caso se ha escogido la ciudad de Santander.

Santander dispone de una colección de datos abiertos (Open Data), accesibles para cualquier usuario y actualizados en tiempo real. En esta colección de datos podemos encontrar, entre otros, conjuntos de datos referentes a:

- sensores de la intensidad del tráfico
- sensores de parking
- señales de tráfico
- calles de la ciudad
- condiciones meteorológicas
- paradas de taxis

El trabajo, también, tiene como objetivo analizar, comparar y estudiar el comportamiento de tres sistemas de gestión de bases de datos idóneos para grandes volúmenes de datos: MongoDB, Apache Cassandra y Apache Hive.

El sistema desarrollado tiene tres funcionalidades:

- Recolección en tiempo real de los recursos proporcionados por las fuentes de datos.
- Almacenamiento en base de datos de los recursos procesados y evaluación del comportamiento de los distintos sistemas de gestión de bases de datos.
- Representación gráfica de los datos almacenados.

Para la recolección y el almacenamiento de los datos se hará uso de la herramienta Apache Spark. Esta herramienta nos permite realizar descargas de datos en tiempo real (streaming), así como procesarlos y almacenarlos de manera muy rápida.

Por último, para que la representación de los datos sea más comprensible e intuitiva, se hará uso de la API de Google, la cual nos permitirá ver los datos representados en el mapa de Santander.

Palabras claves: Big Data, Open Data, Smart City, Apache Spark, MongoDB, Apache Cassandra, Apache Hive.

Abstract

This project is focused on the implementation of Big Data solutions in Smart Cities (Smart Cities). The goal is to facilitate the management of a city traffic and in this case, we have chosen the city of Santander.

Santander has a collection of open data, accessible to any user and updated in real time. In this collection of data we can find, among others, data sets concerning:

- traffic intensity sensors
- parking sensors
- traffic signals
- city streets
- weather conditions
- taxi ranks

The project also aims to analyze, compare and study the behaviour of three database management systems suitable for large volumes of data: MongoDB, Apache Cassandra and Apache Hive.

The developed system has three functions:

- Real-time collection of resources provided by the data source.
- Database processed resources storage and performance evaluation of the various database management systems.
- Graphical representation of stored data.

The project will use Apache Spark for the collection and storage of data. This tool allows us to download data in real time (streaming), process them and store them very quickly.

Finally, to develop a more intuitive and understandable representation of data, we will make use of Google API, which will allow us to see the data represented on the map of Santander.

Keywords: Big Data, Open Data, Smart City, Apache Spark, MongoDB, Apache Cassandra, Apache Hive.

Agradecimientos

A mi familia por apoyarme durante toda la carrera, proporcionándome valores y orientándome durante toda mi época universitaria. Gracias a vosotros he aprendido a tener constancia, seguridad en mí mismo y lo más importante, a no rendirme pese a cualquier circunstancia.

A mis amigos y compañeros de universidad, por haber realizado tantos grandes trabajos juntos, por haberme aportado buenos consejos y por haberme apoyado en todo momento.

A mis profesores, por haberme aportado tantos conocimientos nuevos, por haberme corregido cuando era necesario y haberme solventado cualquier duda o problema que surgían.

A mis tutores Ismael Navas Delgado y José Manuel García Nieto, por haber hecho posible el desarrollo de este proyecto, por atenderme cuando lo he necesitado y por ayudarme con todos los problemas que han surgido durante la elaboración de este proyecto.

Índice de Contenidos

1 Introducción 1

1.1 Contexto 1

1.2 Motivación 2

1.3 Objetivos del proyecto 3

1.4 Estructura del documento 3

2 Tecnologías Utilizadas 5

2.1 Apache Spark 5

2.2 Java 7

2.3 Netbeans 8

2.4 MongoDB 9

2.5 Apache Cassandra 10

2.6 Apache Hive 11

2.7 Wordpress 11

3 Metodología 13

4 Análisis del sistema 15

4.1 Definición del sistema 15

4.2 Requisitos Funcionales 17

4.3 Requisitos No Funcionales 17

4.4 Diagrama de Casos de uso 18

4.4.1 Descripción de los casos de uso 19

5 Diseño 23

5.1 Arquitectura del Sistema 23

5.2 Diagrama de Clases 25

5.2.1 Módulo de Descargas 25

5.2.2 Módulo de Almacenamiento 26

5.2.3 Módulo Estadístico 27

5.2.4 Módulo de Representación Gráfica 27

5.2.5 Módulo de Envío y Actualización de Datos 28

5.3 Modelo de Datos 29

5.3.1 Modelo de Datos de Apache Cassandra 29

5.3.2 Modelo de datos de Apache Hive 31

5.3.3 Modelo de datos de MongoDB 32

6 Implementación 35

6.1 Módulo de Descargas 35

6.1.1 Clase Recibidor 35
6.1.2 Clase Aplicación 37
6.1.3 Clase Lanzador 37
6.1.4 Clase Procesar 38
6.2 Módulo de Almacenamiento 38
6.2.1 Clase Conector 39
6.2.2 Clase AlmacenamientoBD 39
6.2.3 Clase EscribeFichero 42
6.3 Módulo Estadístico 43
6.4 Módulo de Representación Gráfica 43
6.4.1 Taxis y Paradas de Taxis 44
6.4.2 Sensores Smart Traffic y Mediciones 45
6.4.3 Zonas 30 46
6.4.4 Sensores Smart enviroment monitoring 46
6.5 Módulo de Envío y Actualización de Datos 47
6.5.1 Clase FusionTables 47
6.5.2 Clase Uploader 47
6.6 Aplicación Web 47
7 Evaluación de los Sistemas de Almacenamiento 49
7.1 Necesidades de los sistemas de Almacenamiento 49
7.2 Pruebas de Almacenamiento 49
8 Interfaz de Usuario 53
8.1 Módulo de Descargas 53
8.2 Módulo de Almacenamiento 55
8.3 Módulo Estadístico 56
8.4 Módulo de Representación Gráfica 57
8.5 Módulo de Envío y Actualización de Datos 59
8.6 Representación en Fusion Tables 60
8.7 Aplicación Web 62
8.7.1 Login 62
8.7.2 Nuevo Usuario 63
8.7.3 Mapas y Zonas 30 64
9 Conclusiones y Trabajos Futuros 67
9.1 Conclusiones 67
9.2 Trabajos Futuros 68
10 Bibliografía 69

Índice de Ilustraciones

Figura 2.1 Componentes Apache Spark 6
Figura 3.1 Modelo en cascada 13
Figura 3.2 Modelo incremental e iterativo 14
Figura 4.1 Descripción del sistema 16
Figura 4.2 Diagrama de casos de uso 18
Figura 5.1 Diagrama de Clases Módulo de Descargas 25
Figura 5.2 Diagrama de Clases Módulo de Almacenamiento 26
Figura 5.3 Diagrama de Clases Módulo Estadístico 27
Figura 5.4 Diagrama de clases Módulo de Representación Gráfica 28
Figura 5.5 Diagrama de Clases Módulo de Envío y Actualización de Datos 29
Figura 5.6 Estructura de Apache Cassandra 30
Figura 5.7 Modelo de Datos Apache Hive 31
Figura 5.8 Modelo de datos MongoDB 33
Figura 6.1 Hebras Recibidores 36
Figura 7.1 Gráfica número de recursos con respecto al tiempo 50
Figura 7.2 Gráfica tamaño de recursos con respecto al tiempo 51
Figura 8.1 Ejecución Módulo de Descargas 54
Figura 8.2 Almacenamiento del Módulo de Descargas 54
Figura 8.3 Ejemplo de ejecución del módulo de almacenamiento 55
Figura 8.4 Almacenamiento en MongoDB 56
Figura 8.5 Ejecución Módulo Estadístico 57
Figura 8.6 Ejecución del Módulo de Representación Gráfica 58
Figura 8.7 Resultados ejecución Módulo de Representación Gráfica 58
Figura 8.8 Ejecución del Módulo de Envío y Actualización de Datos 59
Figura 8.9 Credenciales Fusion Tables 60
Figura 8.10 Mapa de paradas de taxis en Fusion Tables 61
Figura 8.11 Página de Inicio 62
Figura 8.12 Formulario de Registro 63
Figura 8.13 Página de Login nuevo usuario 64
Figura 8.14 Mapa de Sensores de Tráfico 64
Figura 8.15 Zoom e información de los marcadores 65
Figura 8.16 Mapa estático Zonas 30 66

1 Introducción

En este capítulo se va a proporcionar al lector una visión general sobre el proyecto desarrollado. Para ello se ha dividido en los siguientes apartados: contexto, motivación, objetivos establecidos y estructura de la memoria.

1.1 Contexto

El proyecto desarrollado se encuentra en el campo del Big Data. Al ser un campo novedoso, vamos a comenzar con una explicación del concepto de Big Data y algunas de sus posibles aplicaciones.

Big Data[1] es un término inglés que hace referencia al almacenamiento masivo de datos, así como a los procedimientos usados para encontrar patrones repetitivos en estos datos. Este fenómeno a veces también se denomina datos a gran escala.

Este concepto ha nacido debido a que actualmente, gracias a los avances tecnológicos, se ha producido un enorme crecimiento de información, hablamos incluso de volúmenes de datos de petabytes y hexabytes. Este crecimiento de información ha dado lugar a que mediante los métodos y herramientas convencionales, no sea posible que los datos sean capturados, administrados y procesados en un tiempo razonable. Esto se debe a que han superado los límites y capacidades de las herramientas software utilizadas para estos propósitos.

En cuanto a las posibles aplicaciones del Big Data, podemos encontrar numerosas aplicaciones a lo largo de una gran variedad de ámbitos. Estos ámbitos pueden ser el empresarial, deportivo, investigación, etc.

Dentro del ámbito empresarial, podemos encontrar aplicaciones en redes sociales, ya que actualmente cada vez hay más tendencia a subir información a estas redes. Las empresas utilizan esta información para por ejemplo cruzar los datos de los candidatos a un trabajo. Otra aplicación bastante extendida dentro de este ámbito es la de las ventas cruzadas, para crear un mayor índice de ventas. Este tipo de técnica consiste en utilizar minería de datos masiva basándose en los patrones de compra de los usuarios. De esta manera se consiguen crear anuncios personalizados y así aumentar el índice de ventas.

Dentro del ámbito deportivo, Big Data puede ayudar tanto a nivel de toma de decisiones de los entrenadores como al de predicción de partidos para las apuestas. Se analizan el rendimiento de los jugadores y se crean estadísticas y predicciones. Gracias a esto los entrenadores pueden tener una mejor aproximación del avance de los jugadores.

Dentro del ámbito de la investigación, podemos encontrar aplicaciones muy diversas. Se han aplicado técnicas de Big Data para predecir la pandemia de

enfermedades, la vigilancia y seguridad de fronteras, lucha contra el terrorismo, planteamiento táctico y misiones, etc.

1.2 Motivación

Como se ha comentado anteriormente, Big Data tiene numerosas aplicaciones, pero en este caso en concreto, se ha optado por el ámbito de Smart City (Ciudad Inteligente). En este ámbito, el Big Data puede ser de mucha ayuda, y un claro ejemplo sería la gestión del tráfico. Esta aplicación podría ser de ayuda tanto a los ciudadanos como al Estado, puesto que podría ahorrar mucho tiempo perdido en atascos, restricciones de tráfico, optimizar la colocación de las distintas señales de tráfico, o cualquier otra incidencia.

Este aspecto comentado, también tiene una repercusión importante en la logística y el transporte, puesto que podría aprovechar el análisis de todos los datos de la ciudad para obtener la ruta más óptima bajo cualquier tipo de adversidades en el tráfico.

Esta aplicación es muy interesante para el Estado, puesto que podría hacer más efectiva la actuación de las fuerzas del estado, ya que tomarían la ruta más óptima, teniendo en cuenta muchos de los posibles factores que afectan al tráfico. Además no solo ahorraría tiempo en desplazamientos, sino que a la vez, podría ayudar a que las fuerzas del Estado sepan donde hay mayor necesidad de agentes a nivel de tráfico. Y así conseguir que haya menos congestión de tráfico en la ciudad.

Por otro lado, hay circunstancias impredecibles que a veces ocurren, como es el caso de fenómenos meteorológicos, accidentes de tráfico, obras y restauraciones, cortes de carreteras, etc. Estas circunstancias son muy difíciles de tener en cuenta, pero mediante el uso del análisis de Big Data, podemos conseguir atenuar en muchos casos, estas situaciones adversas.

Además de lo comentado, el almacenamiento de estos datos se realizaría de manera histórica, por lo que, no solo tendríamos en cuenta los datos actuales, sino que podríamos realizar comparaciones y estadísticas teniendo en cuenta datos referentes a otros años. Por lo tanto, además de poder realizar análisis más precisos, nos permite evaluar si las decisiones que se tomaron en un determinado momento de la historia fueron oportunas ante determinadas circunstancias del tráfico. Así que, también es posible evitar cometer los mismos errores que se pudieron cometer al realizar la gestión del tráfico en un determinado momento de la historia.

Para poder realizar un análisis predictivo del tráfico de una ciudad es necesario un sistema que capture datos cada poco tiempo desde una fuente de datos y los almacene. La realización de este proyecto surge de la necesidad de proveer un software capaz de realizar esta captura datos masivos en tiempo real desde la fuente de datos y el almacenamiento de los mismos de una manera rápida, puesto que este tipo de datos cambia constantemente.

Para este propósito es necesario realizar una evaluación de los sistemas de almacenamiento. Este proyecto también comprende este aspecto, ya que el rendimiento, capacidad de almacenamiento y rapidez son factores determinantes para conseguir la solución óptima.

1.3 Objetivos del proyecto

El objetivo principal del proyecto es desarrollar un sistema capaz de recolectar un gran volumen de datos en tiempo real, procesarlos y almacenarlos en una base de datos. Una vez realizada dicha recolección y almacenamiento, el sistema también tiene que extraer y representar los datos de forma gráfica, para que sean fácilmente interpretables.

Otro objetivo es analizar, comparar y estudiar el comportamiento de tres sistemas de gestión de bases de datos idóneas para grandes volúmenes de datos: MongoDB, Apache Cassandra y Apache Hive.

Una vez realizada la evaluación y el estudio del funcionamiento de estos tres sistemas de gestión de bases de datos, se procederá a la elección de uno de ellos, teniendo en cuenta el análisis realizado y el rendimiento obtenido.

1.4 Estructura del documento

El presente documento se encuentra divido en varios capítulos, recogiendo toda la información relacionada con el desarrollo de este proyecto. A continuación, se ofrece una breve descripción de cada uno de ellos.

CAPÍTULO 1 – INTRODUCCIÓN.

En este capítulo se establece el contexto que abarca el desarrollo del proyecto, describiendo su temática, objetivos que se desean alcanzar y las motivaciones que han llevado a su desarrollo.

CAPÍTULO 2 – TECNOLOGÍAS UTILIZADAS.

Durante este capítulo se describen las tecnologías utilizadas en el desarrollo del proyecto, así como las características principales de cada una de ellas. Además, se explican las razones por las que se ha decidido el uso de esa tecnología.

CAPÍTULO 3 – METODOLOGÍA.

Este capítulo describe la metodología seleccionada para llevarlo a cabo el proyecto, explicando los motivos de su elección.

CAPÍTULO 4 – ANÁLISIS DEL SISTEMA.

Durante este capítulo se recoge toda la información relativa a la fase de análisis del sistema. En esta fase se definen los requisitos necesarios para la elaboración del proyecto, así como los distintos casos de uso del sistema. Toda esta etapa sirve de base para la realización de la siguiente fase del proyecto.

CAPÍTULO 5 – DISEÑO DEL SISTEMA.

Este capítulo está dedicado a la fase de diseño del proyecto. Durante esta fase, se realiza el diseño de la arquitectura del sistema, se exponen los diagramas de clases, el modelo de datos utilizado y se describen los componentes que conforman el sistema.

CAPÍTULO 6 – IMPLEMENTACIÓN.

En este apartado se recogen los aspectos más importantes del desarrollo del proyecto y se exponen algunos fragmentos de código para una mejor comprensión.

CAPÍTULO 7 – EVALUACIÓN DE LOS SISTEMAS DE ALMACENAMIENTO

Durante este capítulo se describe la evaluación realizada a los sistemas de almacenamiento, detallando las pruebas de almacenamiento realizadas y explicando los resultados obtenidos.

CAPÍTULO 8 – MANUAL DE USUARIO

En este capítulo se detalla la interacción del usuario con el sistema, de modo que se explican los argumentos que necesitan cada módulo y se ofrecen ejemplos de cómo ejecutarlos. Además se detalla cómo usar la aplicación Web para poder ver los resultados finales.

CAPÍTULO 9 – CONCLUSIONES Y TRABAJOS FUTUROS.

Durante este capítulo se aportaran las conclusiones obtenidas a lo largo del desarrollo del proyecto, teniendo en cuenta los objetivos y el todo el proceso de desarrollo. Además, se describen los conocimientos aprendidos a lo largo del proyecto y algunos aspectos a tener en cuenta en trabajos futuros.

CAPÍTULO 10 – BIBLIOGRAFÍA.

Referencias a documentos y webs que se han usado tanto en el desarrollo de este trabajo como para realizar la redacción de este documento.

2 Tecnologías Utilizadas

En este capítulo se describen las tecnologías utilizadas durante el desarrollo de este proyecto. De cada tecnología se describen sus características principales y se explican los motivos de su elección para la elaboración del proyecto.

2.1 Apache Spark

Apache Spark[2] es un framework de código abierto (Open Source) para el procesamiento de datos masivos, y diseñado con tres prioridades: velocidad, facilidad de uso y capacidades avanzadas de analítica. Fue desarrollado en la Universidad de Berkeley (California), y posteriormente donado a la fundación Apache.

Spark proporciona una interfaz para la programación de clusters con paralelismo de datos y tolerancia a fallos. También cabe decir que actualmente, se trata de una de las herramientas más innovadoras y eficaces en todo lo relacionado con el mundo de Big Data.

MapReduce fue el antecesor de Spark, el cual en su momento ya revolucionó el mundo de Big Data, puesto que permitía ejecutar programas de forma paralela con un gran conjunto de datos y en varias máquinas al mismo tiempo. Gracias a esto, MapReduce produce una complejidad lineal con respecto a la escalabilidad de los datos, puesto que si el volumen de datos es mayor, basta con ampliar el número de máquinas disponibles.

Spark mantiene estas características de MapReduce y además aporta una mayor funcionalidad, a través de sus componentes DAG (Directed Acyclic Graph) y RDD (Resilent Distributed Dataset).

DAG (Grafo Acíclico Dirigido)

DAG es un grafo dirigido sin ciclos, esto significa que para cada vértice V del grafo, no hay un camino directo que empiece y termine en V. Un vértice se conecta a otro pero nunca se conecta consigo mismo.

Spark divide el trabajo en tareas, cada una de estas tareas crea un DAG con varias etapas de trabajo que se ejecutan en un cluster. A diferencia de MapReduce, Spark es mucho más rápido puesto que no tiene que escribir los resultados en disco después de cada etapa. Esto nos permite procesar una mayor cantidad de datos de una manera mucho más rápida, puesto que cada vez que se escribe en disco se ralentiza mucho el trabajo.

RDD (Resilent Distributed Dataset)

Los RDDs son una colección de elementos particionados a lo largo de los nodos de un cluster, que permiten realizar operaciones en paralelo y con tolerancia a

fallos. Mediante la API proporcionada por Spark, podemos realizar dos operaciones con los RDDs:

- Transformaciones: Son operaciones mediante las cuales obtenemos un nuevo RDD basado en una modificación del RDD original.
- Acciones: Son operaciones realizadas sobre un RDD que nos permiten obtener un valor como resultado.

A parte de estas características sobre el procesamiento de datos, Spark posee los siguientes módulos: Spark SQL, Spark Streaming, MLlib (Machine learning), GraphX (Figura 2.1). Todos estos módulos son muy interesantes, pero en concreto Spark Streaming y Spark SQL, han sido de gran ayuda para abordar este proyecto, por lo que vamos a comentar brevemente la funcionalidad de los mismos.

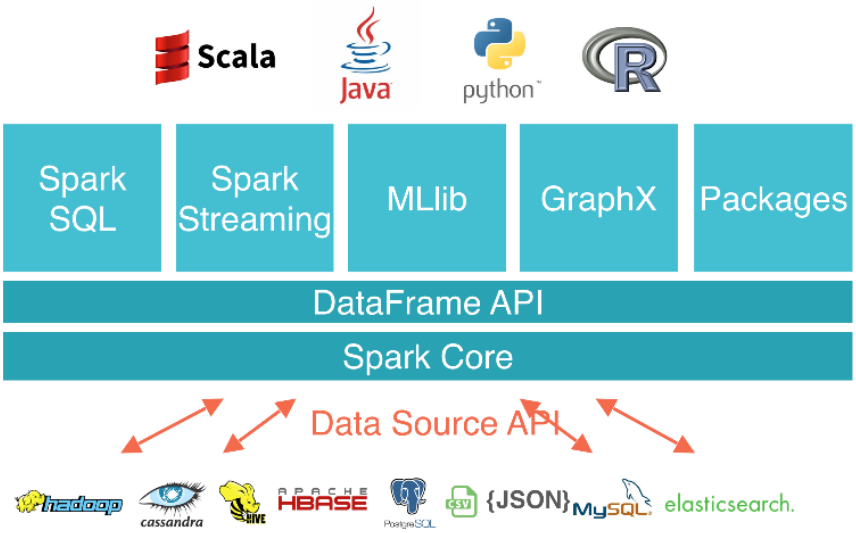

Figura 2.1 Componentes Apache Spark

Spark Streaming

Spark Streaming es un módulo que, como bien su propio nombre indica, nos permite realizar un "Streaming" de datos. Lo cual quiere decir, que nos permite recolectar datos en tiempo real. También comentar que ofrece una gran flexibilidad en cuanto a la fuente de datos, ya que soporta flujos de datos provenientes de Apache Kafka, Apache Flume, Amazon Kinesis, Twitter, sensores y dispositivos conectados a través de sockets TCP y datos almacenados en sistemas de archivos como HDF o Amazon S3.

Además de esto, también existe la posibilidad de crear un recibidor customizado, lo cual nos permitiría recolectar datos prácticamente desde cualquier fuente.

Spark SQL

Spark SQL es un módulo que nos permite realizar consultas sobre los datos, así como realizar lecturas y escrituras en bases de datos, y del mismo modo en disco. Todas estas operaciones, se realizan a través de los DataFrames, los cuales son un conjunto de datos organizados en columnas. Esta manera de encapsular los datos nos abstrae enormemente de cualquier dificultad con los datos, ya que crea de forma automatizada todas las columnas con sus nombres y datos correspondientes. Además, también crea de forma automatizada los tipos de datos que cree conveniente para el conjunto de datos que se está procesando.

Para realizar este encapsulado, no importa si los datos provienen de un fichero o de una base de datos ya que si los datos provienen de un fichero (No importa su formato, si este está soportado por la API de Spark), las columnas tendrán los mismos nombres y organización que la dispuesta en el fichero. En cambio, si los datos provienen de una base de datos, las columnas tendrán los mismos nombres y organización que en la tabla correspondiente.

Además de lo comentado, para las operaciones de bases de datos, la API de Spark nos proporciona drivers de conexión y una gran variedad de métodos para una gran diversidad de sistemas de gestión de bases de datos.

Este módulo es bastante interesante puesto que nos abstrae tanto de la problemática ocasionada por la interacción con las bases de datos, como de la causada por los ficheros.

Para finalizar, comentar también que su API está disponible para Java, Scala, Python y R.

Se ha decidido el uso de esta herramienta porque dispone de múltiples cualidades y características interesantes para el desarrollo de este proyecto. Además de las herramientas más rápidas para el procesado y almacenamiento de datos.

2.2 Java

Java[3] es un lenguaje de programación que fue desarrollado por James Gosling de Sun Microsystems en 1995. Fue diseñado como una simplificación del lenguaje C++, lo que lo hace un lenguaje sencillo de aprender. Las aplicaciones Java son generalmente compiladas a bytecode, que pueden ejecutarse en cualquier máquina virtual (JVM) sin importar la arquitectura subyacente. Sus principales características son:

- **Lenguaje orientado a objetos.**
- **Contiene un amplio conjunto de bibliotecas a disponibilidad del programador.**
- **Lenguaje interpretado y compilado:** Java es compilado ya que su código es transformado a una especie de código máquina, los bytecodes.

Pero por otro lado es interpretado, ya que los bytecodes se pueden ejecutar en cualquier JVM.

- **Distribuido:** Es un lenguaje distribuido, puesto que nos permite desarrollar aplicaciones de red, permitiéndonos crear y gestionar sockets de conexión, facilitando la creación de aplicaciones distribuidas.
- **Robusto:** Java fue diseñado para crear software altamente fiable, para ello proporciona numerosas comprobaciones en compilación y tiempo de ejecución.
- **Seguro:** Dado que se trata de un lenguaje distribuido, muchas aplicaciones se pueden descargar desde la red, por lo que decidieron implementar algunos mecanismos de seguridad en el lenguaje y en el sistema de ejecución en tiempo real.
- **Independiente de la arquitectura:** Como ya hemos comentado anteriormente, el código Java se puede ejecutar en cualquier JVM, por lo que es posible ejecutar el mismo código sobre distintas arquitecturas.
- **Alto rendimiento.**
- **Multihebrado:** Java soporta la sincronización de múltiples hilos de ejecución.
- **Dinámico:** Tanto el lenguaje como su sistema de ejecución en tiempo real son dinámicos en la fase de enlazado. Las clases solo se enlazan cuando son necesarias. Esto facilita también, por ejemplo, la extensión de nuevos módulos.
- **Produce Applets:** Las applets son pequeños programas que aparecen embedidos en páginas Web y tienen capacidad de ejecutar acciones complejas.

Los motivos de la elección de este lenguaje son que es el lenguaje del que dispongo más experiencia, es uno de los lenguajes soportados por la API de Spark y porque contiene todas las características mencionadas.

2.3 Netbeans

Netbeans[4] es un entorno de desarrollo de código abierto, diseñado principalmente para trabajar con Java, aunque es un IDE multilenguaje. Es una herramienta bastante popular, ya que cuenta con una gran comunidad a nivel mundial, que se encuentra en constante crecimiento.

En cuanto a sus características principales, Netbeans dispone de un editor de texto con funciones de autocompletado y resaltado de funciones. Ofrece soporte para JavaScript, estructuras sprint, drivers para distintos sistemas de gestión de bases de datos, Java Beans, web APIs, RESTFUL web servis, Java Moviliti, etc. Además de ofrecer soporte para múltiples lenguajes y aplicaciones, uno de los aspectos más positivos de Netbeans es que que hay disponibles múltiples plugins y permiten extender su funcionalidad.

Para el desarrollo de este proyecto ha sido también bastante útil la integración que dispone con Maven en las últimas versiones, ya que han sido necesarias

bastantes librerías externas y Maven gestiona automáticamente todas las dependencias entre las mismas.

Se ha decidido el uso de este framework por todo lo comentado, y debido a que es uno de los entornos de desarrollos mejor adaptados a la programación en lenguaje Java.

2.4 MongoDB

MongoDB[5] es un sistema de gestión de bases de datos NoSQL de código abierto y orientado a documentos. Fue desarrollado en 2007 por la compañía 10gen (renombrada actualmente a MongoDB inc), y está disponible para los sistemas operativos Windows, Linux, OS X y Solaris.

A diferencia de los sistemas de datos relacionales, MongoDB no guarda los datos en tablas, sino que en lugar de tablas, guarda los datos en documentos BSON. Los documentos BSON son similares a los documentos JSON pero con un esquema dinámico, con esto se consigue que las integraciones con datos de distintas aplicaciones, sea sencilla y rápida.

Las características principales de MongoDB son:

- **Consultas Ad Hoc:** soporta búsqueda por campos, consultas de rangos y expresiones regulares.
- **Indexación:** permite crear índices para cualquier campo y también ofrece la posibilidad de crear índices secundarios.
- **Replicación:** La replicación es un mecanismo de seguridad que permite tener acceso al sistema de información en casos de que surja algún problema con el servidor. MongoDB soporta el tipo de replicación primario-secundario, este tipo de replicación consiste en que existe un nodo primario y el resto secundarios. El nodo primario puede ejecutar comandos de lectura y escritura, los secundarios replican los datos del primario y solo se pueden usar para lectura o para copias de seguridad. En caso de que el nodo primario caiga, los nodos secundarios tienen la capacidad de elegir a un nuevo nodo primario. A este sistema de nodos primario-secundarios, se le denomina “replica set”.
- **Balanceo de carga:** MongoDB ofrece la posibilidad de escalar de forma horizontal empleando el concepto de “shard”. Esto permite que los datos sean distribuidos en distintos servidores, balanceando la carga entre ellos, de manera que ninguno de ellos tenga sobrecarga de datos. De esta manera, se pueden incorporar nuevos servidores proporcionando mucha flexibilidad.
- **Almacenamiento:** Es posible utilizar MongoDB como un sistema de archivos, permitiendo utilizar la ventaja de capacidad proporcionada por el balanceo de carga, así como la replicación de datos para el almacenamiento mediante múltiples servidores. Esto permite que los

datos puedan ser distribuidos y replicados varias veces, proporcionando un sistema eficiente, con tolerancia a fallos y balanceos de carga.

- **Agregación:** Proporciona un framework de agregación que permite realizar operaciones similares a las operaciones “GROUP BY” de SQL. Además, MongoDB también ofrece la función MapReduce que puede ser utilizada para el procesamiento por lotes de datos y las operaciones de agregación.
- **Ejecución de JavaScript desde el lado del servidor:** Tiene la capacidad de realizar consultas usando JavaScript, enviándolas directamente a la base de datos.

2.5 Apache Cassandra

Apache Cassandra[6] es una base de datos NoSQL de código abierto, distribuida y basada en un modelo de almacenamiento clave-valor. Está escrita en Java y permite el almacenamiento de grandes volúmenes de datos de forma distribuida. Fue desarrollada en 2012 por Apache Software Foundation y su objetivo principal es la escalabilidad lineal y la disponibilidad. Cassandra consigue aportar el máximo rendimiento para el número máximo de nodos, pero tiene una alta latencia en operaciones de escritura y lectura.

Su modelo de datos consiste en la partición de filas reorganizadas en tablas. Estas tablas se pueden crear, modificar y eliminar sin bloqueos en tiempo de ejecución.

Sus características principales son:

- **Escalabilidad:** Como hemos comentado anteriormente, las operaciones de lectura y escritura aumentan a medida que se añaden nuevos nodos. Pero estos nuevos nodos se pueden añadir sin necesidad de interrumpir la ejecución de la aplicación.
- **Tolerancia a fallos:** Los datos se replican automáticamente a varios nodos de manera que pueda haber una recuperación ante posibles fallos. Estos nodos se pueden reemplazar sin que se produzcan tiempos de inactividad o interrupciones.
- **Modelo descentralizado:** Todos los nodos tienen el mismo rol, por lo que cada nodo puede responder ante cualquier solicitud.
- **Consistencia:** Ofrece distintas posibilidades distribuidas en niveles para la consistencia en lecturas y escrituras.
- **Ofrece soporte para MapReduce, Apache Pig y Apache Hive:** Cassandra posee una integración con Apache Hadoop, por lo que ofrece soporte para MapReduce, Apache Pig y Apache Hive.
- **Lenguaje CQL (Cassandra Query Language):** Cassandra posee un lenguaje propio de consulta, similar a SQL. Posee drivers disponibles para Java (JDBC), Python (DBAPI2), Node.JS (Helenus), Go (gocql) y C++.

2.6 Apache Hive

Apache Hive[7] es una infraestructura de almacenamiento de datos construida sobre Apache Hadoop. Proporciona agrupaciones, consulta y análisis de datos. Es un sistema multiplataforma y programado en Java. Fue desarrollado inicialmente por Facebook, pero actualmente es utilizado y desarrollado por Netflix y la Financial Industry Regulatory Authority. Amazon posee una derivación de Apache Hive incluido en sus servicios Amazon Web Services.

Sus características principales son:

- **Indexación:** Permite la creación de índices que incluyen compactación e índices bitmaps.
- **Funciones definidas por el usuario (UDF):** Permite el uso de funciones definidas por el usuario para manipulación de fechas, textos, y otras herramientas de minería de datos.
- **Almacenamiento:** Dispone de distintos tipos de almacenamiento, como texto, RCFile, HBase, ORC y otros.
- **Almacenamiento de metadatos:** Permite el almacenamiento de metadatos en bases de datos relacionales. Esto consigue disminuir el tiempo de verificaciones semánticas realizadas durante las consultas.
- **Operaciones sobre datos comprimidos:** Permite el uso de algoritmos sobre datos comprimidos como DEFLATE, BWT, snappy,etc.
- **Lenguaje HiveQL:** Apache Hive posee un lenguaje propio para consultas, similar a SQL llamado HiveQL. Este lenguaje posee esquemas para leer y convertir consultas directamente en MapReduce, Apache Tez o tareas Spark.

Se han decidido utilizar estos sistemas de gestión de bases de datos debido a las características comentadas, y a que son sistemas muy utilizados y pioneros en el mundo de Big Data.

2.7 Wordpress

Wordpress[8] es un sistema de gestión de contenidos o CMS que en principio estaba pensado para la creación de blogs, pero actualmente permite la creación de cualquier sitio web. Se trata de una herramienta de software libre, basada en PHP para entornos que ejecuten Mysql y Apache. Su primer lanzamiento fue el 9 de Mayo de 2003 y su creador fue Matt Mullenweg.

Algunas de sus características principales son:

- Permite realizar actualizaciones de manera rápida y sencilla, ya que cada vez que creamos nuevas publicaciones, sus páginas y artículos se generan de forma dinámica.
- Dispone de soporte para una gran cantidad de idiomas.

- Su instalación se realiza localmente en el propio servidor, lo que proporciona un mayor control de su configuración.
- Es fácilmente integrable con distintas redes sociales.
- Tiene un enorme repositorio de plugins, que son añadidos por distintos desarrolladores y que permiten extender la funcionalidad del sitio web. Además continuamente se realizan actualizaciones de estos plugins y se añaden nuevos.
- Permite gestionar usuarios de una manera sencilla, permitiendo establecer distintos permisos para restringir las acciones que pueden realizar.
- Dispone de una herramienta de búsqueda integrada que facilita la navegación de los usuarios.
- Permite la moderación de comentarios y la obtención de información de los visitantes del sitio web.
- Proporciona plantillas y temas que diseñar de una manera rápida y fácil la estética del sitio web.
- Funciona sobre PHP y MySQL, que son herramientas muy utilizadas actualmente.
- Se pueden programar las publicaciones de artículos y páginas, lo que facilita la organización del sitio web.
- Los artículos se organizan en categorías, lo que facilita la gestión de los mismos.

Se ha seleccionado esta herramienta, por las características comentadas y porque se trata de una herramienta gratuita, muy popular y de fácil manejo.

3 Metodología

La metodología es un aspecto muy importante en un desarrollo software, puesto que nos permite estructurar, planificar y controlar todo el desarrollo del proyecto. Además proporciona un marco de trabajo compuesto por estándares probados en numerosos proyectos, lo que facilita conseguir alcanzar las metas propuestas.

La metodología está presente durante todo el ciclo de vida del proyecto, por lo que comprende todas las fases del mismo. Durante el ciclo de vida se planifican una serie de actividades, hitos y procesos que se deben realizar para lograr cumplir con los objetivos establecidos.

Existe un alto porcentaje de proyectos software que fracasan por realizar una mala estimación de la planificación que se debe seguir, seleccionar una metodología errónea o por no cumplir la metodología durante el desarrollo del proyecto. A la hora de seleccionar una metodología hay que tener en cuenta los distintos aspectos y características que presenta el proyecto en cuestión. Existe una gran variedad de metodologías y algunas se pueden adaptar mejor que otras a las características que deseamos.

La metodología de desarrollo en cascada[9], es la metodología que más se utilizaba tradicionalmente para el desarrollo de proyectos software. Es una metodología que establece las etapas del proyecto de una manera ordenada y secuencial. Esto significa que el desarrollo se realiza desde las primeras etapas hasta las últimas, de modo que para realizar la última etapa es necesario que estén terminadas todas las anteriores.

En la siguiente figura podemos observar las fases que comprende el modelo en cascada.

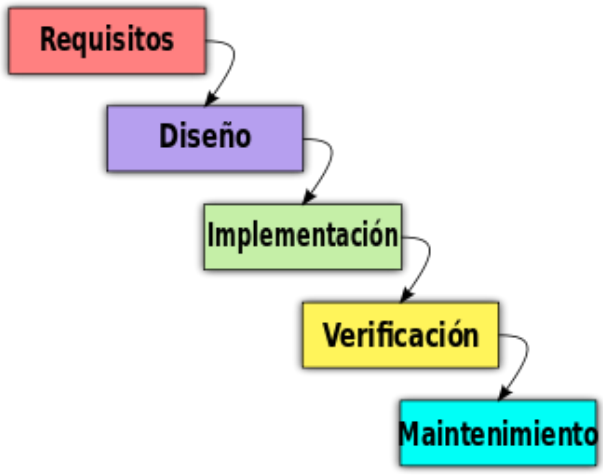

Figura 3.1 Modelo en cascada

Esta manera de enfocar un desarrollo software ofrece muy poca flexibilidad, puesto que no existe retroalimentación en el modelo. De forma que lo que se acordó en un principio, es lo que finalmente se llevará a cabo.

Al ocurrir a este inconveniente, surge el modelo de desarrollo iterativo e incremental[10]. Este modelo consiste en realizar una serie de entregables o incrementos, que se van entregando al cliente cada cierto tiempo. De esta manera, el cliente puede ver como se está desarrollando el producto y a la vez puede ofrecer retroalimentación para mejorar las próximas entregas. En cada incremento se realizan tres etapas: Análisis, diseño, codificación y pruebas.

En la siguiente figura se puede observar el esquema del modelo incremental e iterativo.

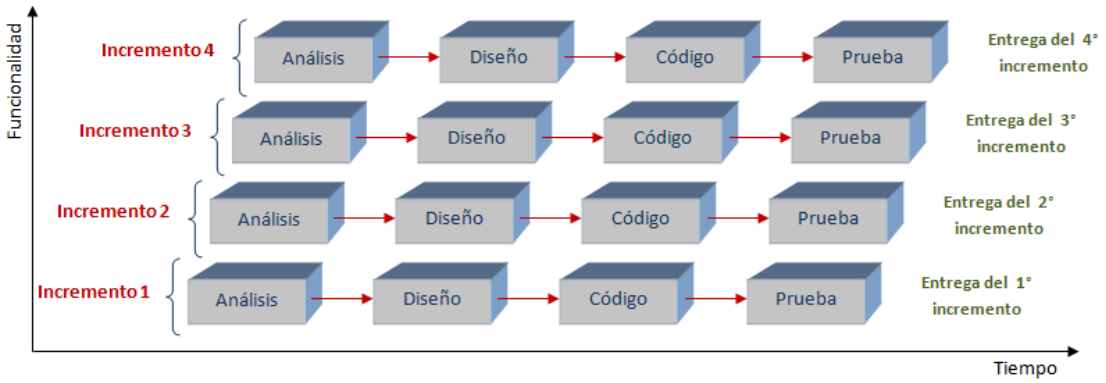

Figura 3.2 Modelo incremental e iterativo

Para la elaboración de este proyecto se ha optado por una metodología incremental e iterativa, puesto que se adapta bastante bien al proyecto. En este proyecto necesitamos que algunos módulos estén desarrollados y en funcionamiento antes que otros. De modo que si algún módulo no funciona correctamente, es necesario volver a diseñarlo y desarrollarlo antes de continuar con el desarrollo del resto de módulos.

Además al tratarse de un proyecto de investigación, los requisitos van cambiando a la vez que el proyecto evoluciona. Por lo que es muy importante que exista retroalimentación con cada incremento desarrollado.

4 Análisis del sistema

Durante este capítulo se detallan los aspectos relacionados con el análisis del sistema. El análisis es una parte fundamental de un proyecto, puesto que se establece como debe comportarse el sistema.

En la fase de análisis, se identifican los requisitos necesarios para el desarrollo del sistema. Este es uno de los aspectos más importantes puesto que se establecen las funcionalidades que debe tener el sistema, así como los objetivos y el alcance que debe tener el mismo.

4.1 Definición del sistema

Para el desarrollo de este sistema el primer paso a realizar es buscar una fuente de datos adecuada para este propósito. Esta tarea no es nada fácil, puesto que no solo son necesarios un conjunto de datos abiertos al público y referentes al tráfico, sino también es necesario que estos estén actualizados cada poco tiempo.

Finalmente, después de una exhaustiva búsqueda de fuentes de datos apropiadas, se ha optado por el catálogo de datos de la ciudad de Santander. Esta ha sido la elección final debido a que es la fuente de datos que más se ajusta a los requisitos necesarios para este proyecto. Adicionalmente se disponen de datos referentes a la temperatura, humedad y presión de Santander, obtenidos desde otra fuente de datos.

El sistema se divide en varios módulos, de modo que cada módulo se encarga de una funcionalidad distinta del sistema. Además de estos módulos, el sistema también dispone una aplicación web para poder visualizar con mayor facilidad y comodidad la representación gráfica de los datos.

En la siguiente figura podemos ver una representación del sistema completo:

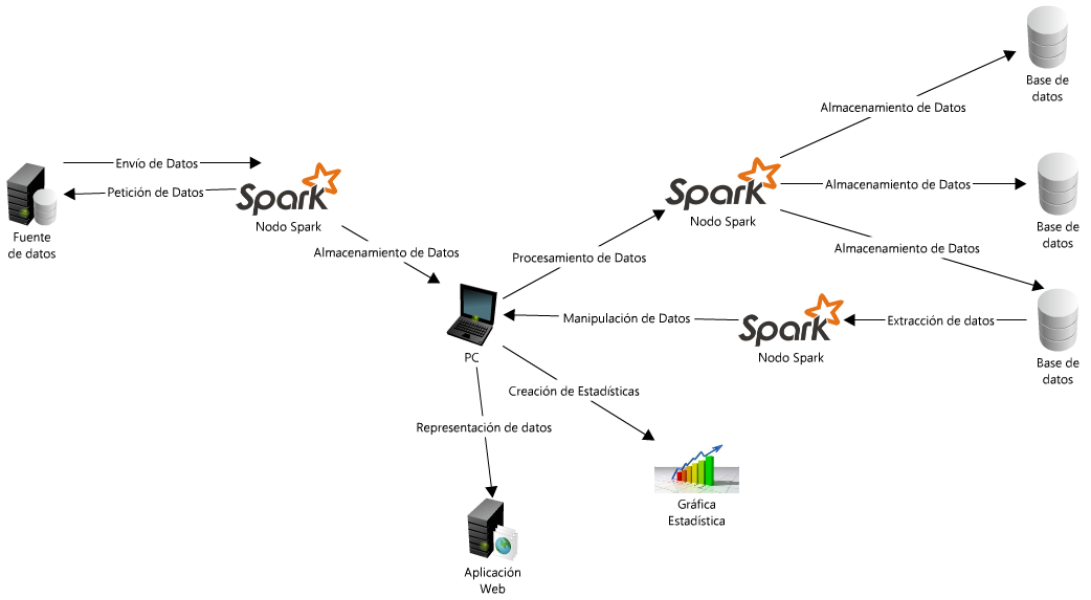

Figura 4.1 Descripción del sistema

A continuación, se describen todos los componentes que forman parte del sistema.

- **Módulo descarga:** Este módulo es el encargado de la recolección de datos. La funcionalidad de este nodo es la de descargar cada minuto, una serie de recursos relacionados con el tráfico desde la fuente de datos de Santander (http://datos.santander.es/), así como de la fuente de datos referente a las condiciones meteorológicas (https://www.netatmo.com/es-ES/weathermap). Una vez realizada la descarga, todos los recursos serán almacenados en disco.
- **Módulo de almacenamiento:** Este módulo es el encargado del almacenamiento de los recursos descargados por el módulo anterior. Este componente tiene como objetivo almacenar los recursos en tres sistemas de gestión de bases de datos: MongoDB, Apache Hive y Apache Cassandra.
- **Módulo estadístico:** Este módulo tiene como objetivo la elaboración de gráficas estadísticas que ofrezcan una visión acerca de la evaluación destinada a los sistemas de gestión de bases de datos.
- **Módulo de representación gráfica:** Este módulo tiene como objetivo manipular los datos y reordenarlos de manera adecuada para su representación en Google Maps. Ya que el análisis predictivo de los datos no es un objetivo de este proyecto, el módulo se limitará a extraer algunos datos de prueba para realizar dicha representación.
- **Módulo de envío y actualización de datos:** Este módulo tiene como objetivo enviar los datos que se van a representar a Google. Además, permitirá actualizar estos datos cada cierto tiempo.

- **Aplicación Web:** La aplicación Web tiene como objetivo proporcionar al usuario la posibilidad de ver todas las representaciones realizadas en Google Maps, de una manera simple y cómoda.

4.2 Requisitos Funcionales

Los requisitos funcionales describen las funcionalidades que debe proporcionar el sistema. Dichas descripciones indican como debe comportarse el sistema ante un conjunto de entradas o ante determinadas situaciones.

Los requisitos funcionales de este sistema son los siguientes:

RF-001: El sistema realizará descargas de datos desde las fuentes de datos cada minuto, y almacenará estos en disco.

RF-002: El sistema permitirá al usuario elegir los formatos de las descargas entre los disponibles en la fuente de datos, así como la dirección de destino de los archivos descargados.

RF-003: El sistema permitirá al usuario almacenar los datos en formato CSV en MongoDB, Apache Cassandra y Apache Hive.

RF-004: El sistema elaborará gráficas estadísticas relacionadas con la evaluación de MongoDB, Apache Cassandra y Apache Hive. Almacenará dichas gráficas en disco y permitirá al usuario indicar el lugar de almacenamiento.

RF-005: La aplicación Web permitirá registrar usuarios nuevos.

RF-006: Cada usuario registrado en la aplicación Web podrá visualizar la representación gráfica de los datos.

RF-007: Los usuarios podrán ingresar a la aplicación Web mediante un sistema de login, con un nombre de usuario y contraseña.

RF-008: El sistema almacenará en disco los datos manipulados, para su posterior representación gráfica y permitirá al usuario elegir la ubicación de estos datos.

RF-009: El sistema permitirá al usuario enviar la información que se va a representar a Google, así como actualizar dichos datos cada minuto.

4.3 Requisitos No Funcionales

Los requisitos no funcionales son aquellos requisitos que no describen funciones que debe de tener el sistema, ni información que debe almacenar, sino características de funcionamiento que debe tener el sistema.

Los requisitos no funcionales de este proyecto son los siguientes:

RNF-001: Es necesario tener instalado Apache Hadoop para ejecutar los módulos que utilizan Apache Spark.

RNF-002: El usuario tiene que tener acceso a internet para poder utilizar el módulo de descargas.

RNF-003: Para usar el sistema será necesario tener instalado al menos un sistema de gestión de bases de datos, ya sea MongoDB, Apache Cassandra o Apache Hive.

RNF-004: Los sistemas de gestión de bases de datos MongoDB, Apache Cassandra y Apache Hive deben estar configurados permitiendo conexiones externas.

RNF-005: La interfaz web debe ser sencilla y comprensible para proporcionar una buena interacción con el usuario.

RNF-006: La aplicación web deberá ser accesible desde cualquier navegador.

RNF-007: Para el uso del sistema es necesario disponer de espacio en disco, ya que se pretende descargar un gran volumen de información.

RNF-008: El sistema de gestión de bases de datos necesitará disponer de suficiente espacio para albergar todo el volumen de información.

4.4 Diagrama de Casos de uso

Mediante los diagramas de casos de uso, representamos la interacción del usuario con el sistema. De modo que podemos visualizar de una manera gráfica todas las funcionalidades del sistema.

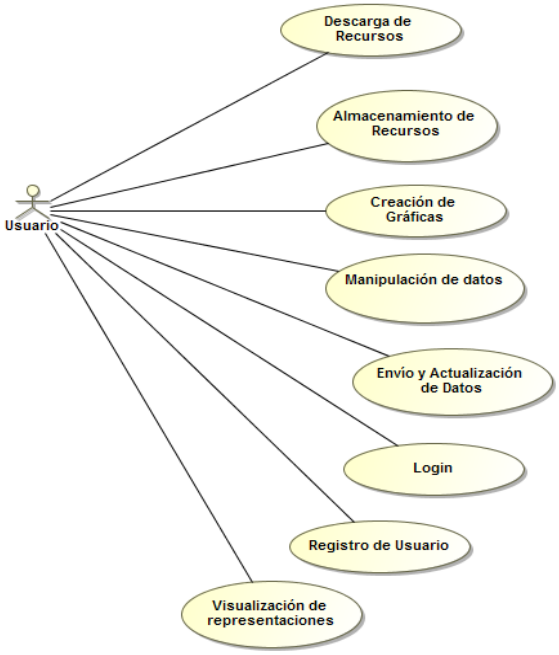

Figura 4.2 Diagrama de casos de uso

4.4.1 Descripción de los casos de uso

Caso de Uso	Descarga de Recursos
Actor principal	Usuario
Precondiciones	1. El usuario tiene que tener acceso a internet para poder descargar los recursos.
Garantía de éxito	Los recursos se descargan cada minuto y son almacenados en disco en el o los formatos deseados.
Escenario de éxito	1. El usuario introduce el o los formatos que desea descargar. 2. El usuario introduce la ruta de almacenamiento de los recursos. 3. El sistema descarga todos los recursos en el o los formatos deseados. 4. El sistema procesa y almacena en disco los recursos.
Extensiones	3 a. Si alguno de los recursos no se encuentra disponible. 1. El sistema informará del código de error de arrojado por la petición del recurso. 2. Se continuará con la descarga de los demás recursos.

Caso de Uso	Almacenamiento de Recursos
Actor principal	Usuario
Precondiciones	1. El usuario debe de haber descargado uno o más recursos. 2. El usuario tiene que tener instalado y configurado al menos un sistema de gestión de bases de datos.
Garantía de éxito	Los recursos se almacenarán en el sistema de gestión de bases de datos correspondiente.
Escenario de éxito	1. El usuario introduce la ruta donde se encuentran los recursos descargados. 2. El usuario introduce los datos de conexión del sistema de gestión de bases de datos. 3. El usuario introduce la ruta donde se almacenarán las anotaciones estadísticas. 4. El sistema lee cada recurso, lo procesa y almacena en el sistema de gestión de bases de datos indicado. 5. El sistema almacena las anotaciones estadísticas relativas al almacenamiento de los recursos.
Extensiones	4 a. Un archivo se encuentra vacío. 1. El sistema informará con un mensaje de que el recurso se encuentra vacío.

	2. El sistema continuará procesando y almacenando el resto de archivos.

Caso de Uso	Creación de Gráficas
Actor principal	Usuario
Precondiciones	1. El usuario ha debido de almacenar recursos en al menos en un sistema de gestión de bases de datos.
Garantía de éxito	El sistema crea gráficas en base a anotaciones estadísticas y las almacena en disco.
Escenario de éxito	1. El usuario introduce la ruta donde están almacenadas las anotaciones estadísticas. 2. El usuario introduce la ruta donde quiere que se almacenen las gráficas. 3. El sistema lee las anotaciones y genera las gráficas correspondientes a la representación de estas anotaciones.
Extensiones	Este caso de uso no dispone de extensiones.

Caso de Uso	Manipulación de Datos
Actor principal	Usuario
Precondiciones	1. El usuario tiene que tener recursos almacenados en una base de datos.
Garantía de éxito	El sistema extrae recursos desde base de datos, los manipula para que puedan ser representados y los almacena en disco.
Escenario de éxito	1. El usuario introduce los datos de conexión del sistema de gestión de bases de datos. 2. El sistema extrae los datos desde el sistema de gestión de bases de datos indicado. 3. El sistema procesa y manipula los datos para que puedan ser representados. 4. El sistema almacena los datos manipulados en disco.
Extensiones	Este caso de uso no dispone de extensiones.

Caso de Uso	Envío y Actualización de Datos
Actor principal	Usuario
Precondiciones	1. El usuario tiene que tener los recursos manipulados almacenados en disco.
Garantía de éxito	El sistema envía los recursos manipulados a Google, y los actualiza cada minuto.
Escenario de éxito	1. El usuario introduce la ruta donde se encuentran los datos manipulados. 2. El sistema envía los datos a Google. 3. El sistema actualiza los datos cada minuto.
Extensiones	Este caso de uso no dispone de extensiones.

Caso de Uso	Login
Actor principal	Usuario
Precondiciones	1. El usuario tiene que disponer de usuario y contraseña. 2. El sistema tiene que tener almacenados los credenciales del usuario.
Garantía de éxito	El sistema permite ingresar en la aplicación web a un usuario registrado.
Escenario de éxito	1. El usuario ingresa en la aplicación web. 2. El sistema muestra el formulario de login. 3. El usuario introduce su usuario y contraseña y presiona iniciar sesión. 4. El sistema comprueba que los datos introducidos son correctos. 5. El usuario puede acceder al contenido de la aplicación web.
Extensiones	3 a. El usuario ingresa datos incorrectos. 1. El sistema notificará al usuario de que los datos no son correctos. 3 B. El usuario presiona iniciar sesión con campos vacíos. 1. El sistema informará al usuario de que existen campos vacíos.

Caso de Uso	Registro de Usuario
Actor principal	Usuario
Precondiciones	Este caso de uso no dispone de precondiciones.
Garantía de éxito	El sistema permite al usuario registrarse en la aplicación web.
Escenario de éxito	1. El usuario ingresa en la aplicación web y pulsa en nuevo usuario. 2. El sistema muestra el formulario de registro al usuario. 3. El usuario rellena los campos del formulario y hace presiona en registro. 4. El usuario queda registrado en el sistema.
Extensiones	3 a. El usuario deja vacíos campos del formulario. 1. El sistema notificará al usuario de que es necesario rellenar los campos vacíos.

Caso de Uso	Visualización de Representaciones
Actor principal	Usuario
Precondiciones	1. El usuario tiene que disponer de acceso a la aplicación web.
Garantía de éxito	El sistema permite visualizar todas las representaciones de los recursos elaboradas.

Escenario de éxito	1. El usuario ingresa en la aplicación web, mediante su usuario y contraseña. 2. El usuario selecciona una representación. 3. El sistema muestra al usuario la representación seleccionada.
Extensiones	Este caso de uso no dispone de extensiones.

5 Diseño

Durante este capítulo se va a detallar el diseño de la arquitectura de la aplicación. Esta etapa es una fase crucial del proyecto, puesto que forma parte de la base de la implementación del sistema.

5.1 Arquitectura del Sistema

La arquitectura del sistema se encuentra dividida en módulos, con el fin de que en un futuro pueda ser ampliada la funcionalidad, sin que esta ampliación suponga demasiada complejidad.

Por otro lado, si deseamos realizar modificaciones sólo en un módulo, es posible realizarlas sin tener que modificar todo el código del proyecto. Este aspecto es muy positivo, puesto que si no estuviera modulado, la modificación podría afectar a todo el proyecto. Este incidente podría tener repercusiones realmente graves, puesto que se pondría en riesgo el proyecto en su totalidad.

Otro aspecto positivo de esta arquitectura es que los fallos se encuentran localizados, de modo que si alguna parte del sistema falla sea posible corregirla de una manera más rápida y eficiente.

Ya que uno de los objetivos de este proyecto es la evaluación de los sistemas de gestión de bases de datos, es necesaria la instalación y configuración de cada uno de ellos. Para ello, con el fin de consumir la menor cantidad de recursos posibles, dicha instalación y configuración se realizará en máquinas virtuales independientes para cada sistema de gestión de bases de datos.

Por otra parte, comentar también que Apache Spark nos permite procesar y realizar distintas operaciones sobre un gran volumen de datos de una manera muy rápida, por lo que todo el tratamiento de datos del sistema se realizará a través de esta herramienta.

En cuanto a la organización y el esquema del sistema, es el siguiente:

- El módulo de descargas será un nodo Spark que realizará peticiones “GET” a la fuente de datos cada minuto, dicha comunicación se realizará a través del protocolo HTTP. Se usará este protocolo debido a que los datos se suministran a través de una RESTFUL API, que actúa sobre este protocolo. Dado que hay varios formatos disponibles, el nodo descargará todos los formatos disponibles en la fuente de datos (siempre se incluirá el formato CSV, debido a que el resto de la aplicación trabajará con dicho formato). Una vez descargados y procesados los distintos recursos, se almacenarán en disco en una dirección suministrada por el usuario. De modo que lo único necesario por parte del usuario sea especificar el o los formatos deseados y la dirección de almacenamiento. Los datos se almacenarán en directorios con la fecha en la que se ha descargado el

recurso, con el formato de fecha: dd-mm-yyyy. En cada uno de estos directorios se almacenarán los recursos de la siguiente forma: Hora-minuto-recurso.formato. De esta manera, podemos disponer de una forma bien diferenciada todos los recursos correspondientes a un día, hora y minutos determinados. Se ha realizado de esta manera debido a que para el objetivo de este proyecto, es necesario ser bastante precisos.

- El módulo de almacenamiento será otro nodo Spark que leerá los archivos con formato CSV descargados en el módulo de descargas y los almacenará en los tres sistemas de gestión de bases de datos (MongoDB, Apache Cassandra y Apache Hive). Se elige el formato CSV porque es un formato que admiten la mayoría de sistemas de gestión de bases de datos y que por regla general, no suele generar problemas. Para realizar este proceso, el nodo tiene que realizar las siguientes operaciones: establecer una conexión con cada uno de los sistemas de gestión de bases de datos, leer los datos descargados, manipular los datos para su correcto almacenamiento y almacenar los datos. Si es necesario, también se crearán las tablas necesarias para dicho almacenamiento. A su vez, el módulo realizará anotaciones estadísticas para la evaluación de estos tres sistemas de gestión de bases de datos.
- El módulo estadístico leerá los datos producidos por el módulo de almacenamiento, los analizará y finalmente creará una gráfica correspondiente a la representación de dichos datos. Este módulo simplificará el proceso de evaluación de los sistemas de gestión de bases de datos, de modo que ofrecerá al usuario la posibilidad de poder ver las conclusiones de una manera más comprensible.
- El módulo de representación gráfica extraerá datos almacenados en base de datos, los manipulará y los almacenará en disco para su posterior representación gráfica en un mapa de Santander. Para la elaboración de dicha representación se hará uso de la herramienta Fusion Tables que proporciona Google. Esta herramienta permite crear tablas con los datos que queramos, y a partir de los datos almacenados en estas tablas, la herramienta crea una representación en un mapa. Fusion Tables dispone de una API para Java mediante la cual podemos crear nuevas tablas e interactuar con las tablas que ya tenemos creadas. Además nos ofrece la posibilidad de compartir las tablas, así como de incrustar el mapa en nuestra web.
- El módulo de envío y actualización de datos enviará a las Fusion Tables los ficheros producidos por el módulo de representación gráfica. Estos ficheros se enviaran cada minuto con el fin de actualizar los datos que se encuentran almacenados, y que estos cambios se vean reflejados en el mapa que contiene la representación de dichos datos.
- La aplicación Web proporcionará un medio para poder visualizar las representaciones realizadas. Debido a que la naturaleza de los datos es diversa, es necesario elaborar varios mapas en función a dicha naturaleza, de esta forma la representación quedará mucho más clara y comprensible para el usuario. Mediante la aplicación Web será posible

visualizar todas estas representaciones de forma conjunta, con el objetivo de proporcionar una interfaz amigable que nos permita visualizar todo de una manera sencilla.

5.2 Diagrama de Clases

En este apartado se van a describir los diagramas de clase correspondientes a cada módulo del proyecto. Se ha realizado un diagrama de clases por cada módulo.

5.2.1 Módulo de Descargas

La siguiente figura (Figura 5.1) corresponde al diagrama de clases del módulo de descargas. Este módulo está formado por las siguientes clases: Lanzador, Aplicación, Procesar, Recibidor, CSVWriter, JSONFlatenner. Las clases principales son Procesar y Recibidor, ya que son las encargadas de descargar, procesar y almacenar todos los recursos. El resto de clases son auxiliares que nos ayudan a procesar los datos, interactuar con el usuario y ejecutar el módulo de forma correcta.

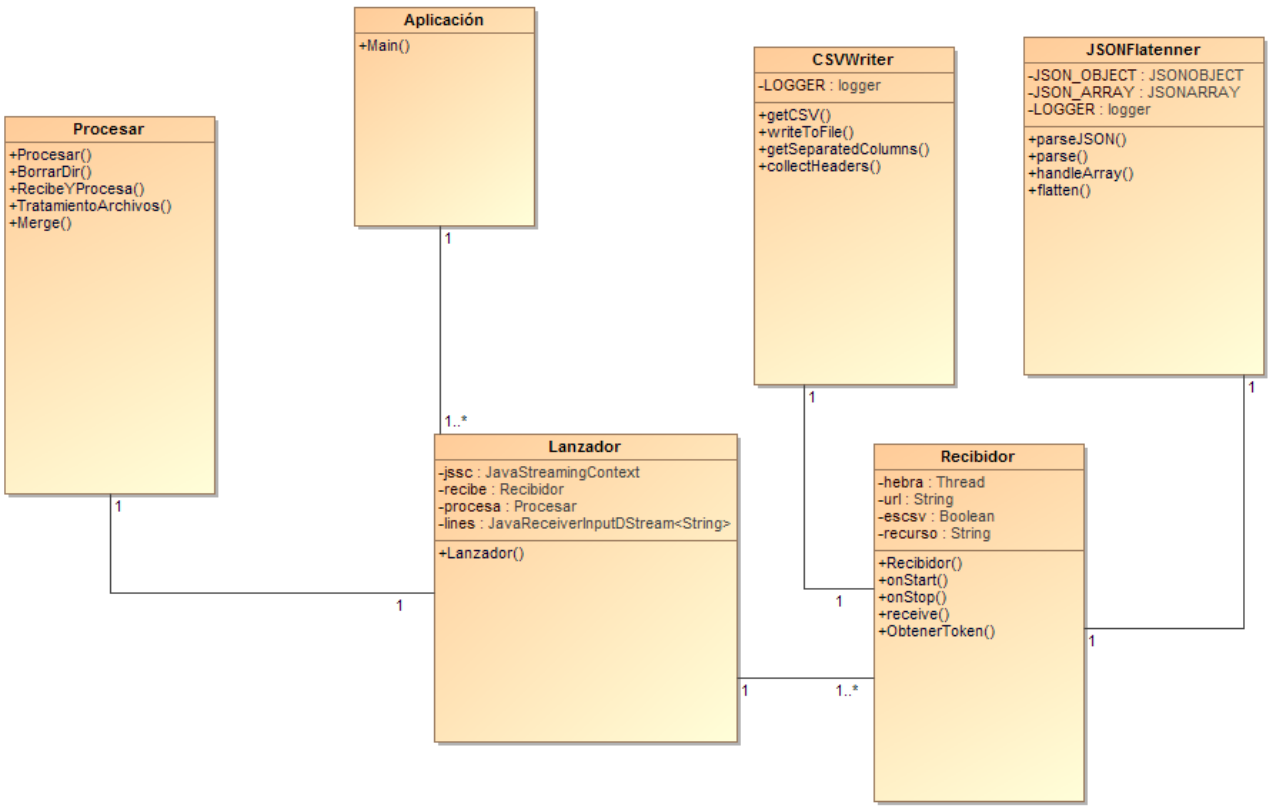

Figura 5.1 Diagrama de Clases Módulo de Descargas

5.2.2 Módulo de Almacenamiento

La siguiente figura (Figura 5.2) muestra el diagrama de Clases del módulo de almacenamiento. Este módulo está formado por tres clases: Conector, EscribeFichero y AlmacenamientoBD.

- La clase Conector se encarga de establecer la conexión con MongoDB, Apache Cassandra y Apache Hive.
- La clase EscribeFichero es la encargada de escribir las anotaciones estadísticas que se apuntan cada vez que almacenamos.
- La clase AlmacenamientoBD es la clase principal, puesto que se encarga de almacenar todos los recursos producidos por el módulo de Descargas.

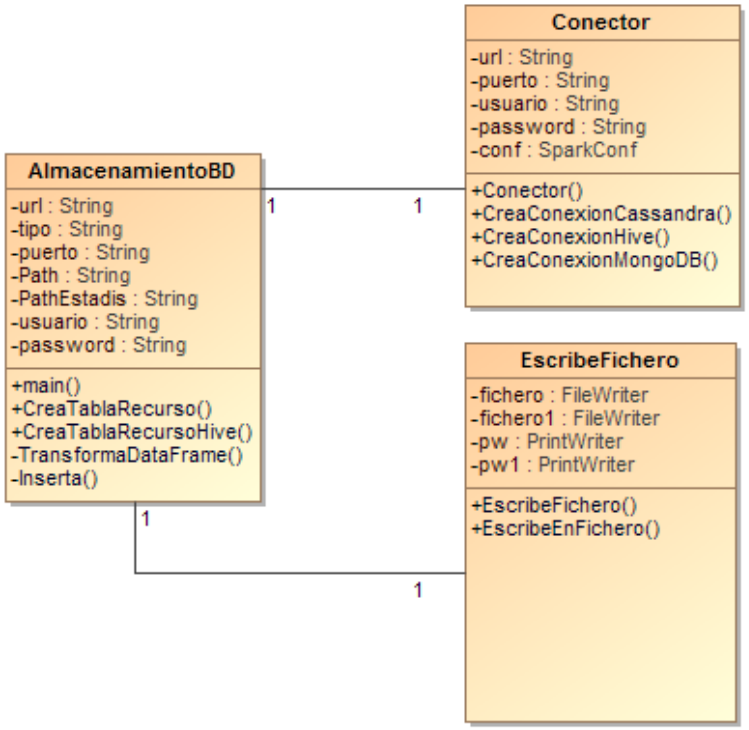

Figura 5.2 Diagrama de Clases Módulo de Almacenamiento

5.2.3 Módulo Estadístico

En la siguiente figura (Figura 5.3) podemos observar el diagrama de clases del módulo estadístico. Este módulo está formado por dos clases: Estadística y ScreenImage.

- La clase Estadística es la clase principal del módulo, ya que es la encargada de leer, analizar y elaborar estadísticas mediante las anotaciones realizadas en los módulos que interactúan con los sistemas de gestión de bases de datos.
- La clase ScreenImage es una clase auxiliar, que nos permite almacenar las gráficas en disco.

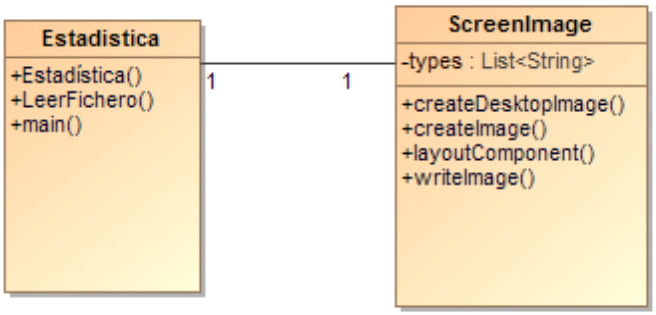

Figura 5.3 Diagrama de Clases Módulo Estadístico

5.2.4 Módulo de Representación Gráfica

En la siguiente figura (Figura 5.4) podemos ver el diagrama de clases correspondiente al módulo de Representación Gráfica. Este módulo está formado por las siguientes clases:

- La clase RepresentacionGrafica es la clase principal, encargada de extraer los datos, manipularlos y almacenarlos en disco.
- La clase ConverterUTM es una clase auxiliar, se utiliza para convertir coordenadas que pertenecen a distintos sistemas de coordenadas.
- La clase Conector tienen la misma función que en el módulo de Almacenamiento (Figura 5.2). En este caso, se utiliza para establecer la conexión con MongoDB.

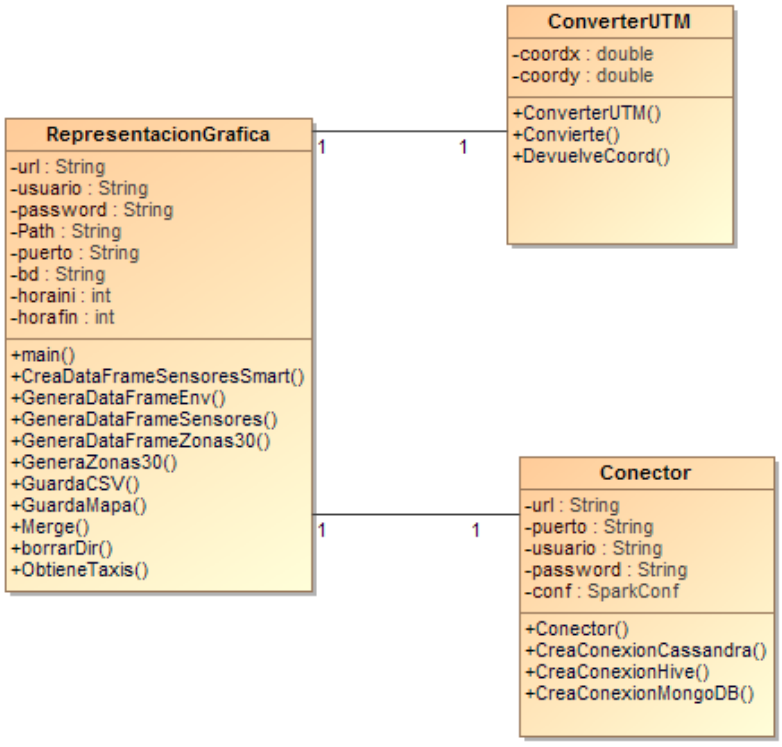

Figura 5.4 Diagrama de clases Módulo de Representación Gráfica

5.2.5 Módulo de Envío y Actualización de Datos

La siguiente figura (Figura 5.5) contiene el diagrama del módulo de Envío y Actualización de Datos. Este módulo está compuesto de las siguientes clases:

- La clase FusionTables es la principal, se encarga de interactuar con Google para crear las tablas y obtener los identificadores de las mismas.
- La clase Uploader es la encargada de almacenar los datos en las tablas. Esta clase es una hebra, ya que queremos que los datos de cada recurso se suban simultáneamente y no de manera secuencial.

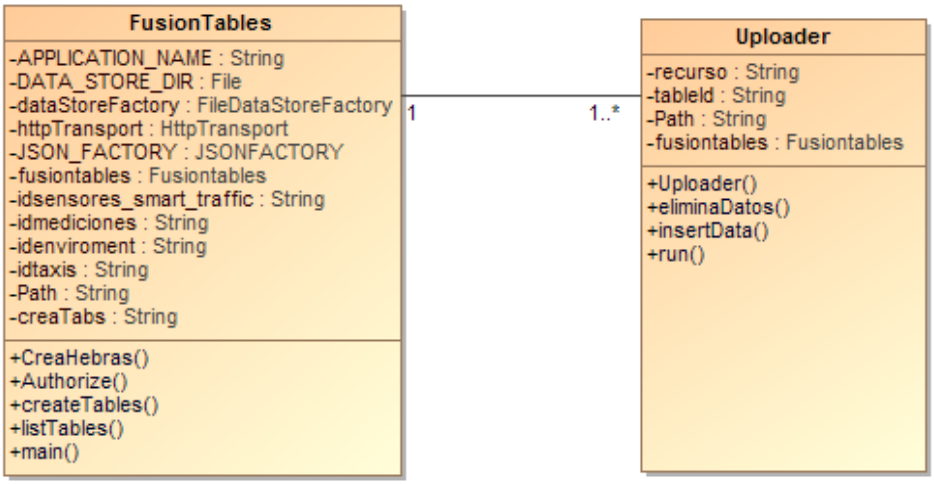

Figura 5.5 Diagrama de Clases Módulo de Envío y Actualización de Datos

5.3 Modelo de Datos

Durante esta sección se va a explicar el modelo de datos de cada sistema de gestión de base de datos usado en el proyecto. Para ello vamos a dividir este apartado en tres secciones, en las que comentaremos las características de cada modelo de datos.

5.3.1 Modelo de Datos de Apache Cassandra

El Modelo de Datos de Apache Cassandra combina propiedades de un modelo orientado a columnas y de un modelo clave-valor. La información se organiza de manera que todas las columnas poseen una clave única y pares de claves, valor de columnas.

Los elementos que forman parte del modelo de datos de Cassandra son los siguientes:

- Keyspaces: Es un espacio de nombres para un conjunto de tablas. Se suelen vincular a las bases de datos de los modelos relacionales.
- Tablas: Contienen varias columnas.
- Columnas: Compuestas de un nombre, valor y timestamp.

El modo de referirnos a un registro de datos es: Un keyspace, una tabla, una clave y finalmente la columna.

En la siguiente figura (Figura 5.6) podemos ver una representación de la estructura de una fila en Apache Cassandra.

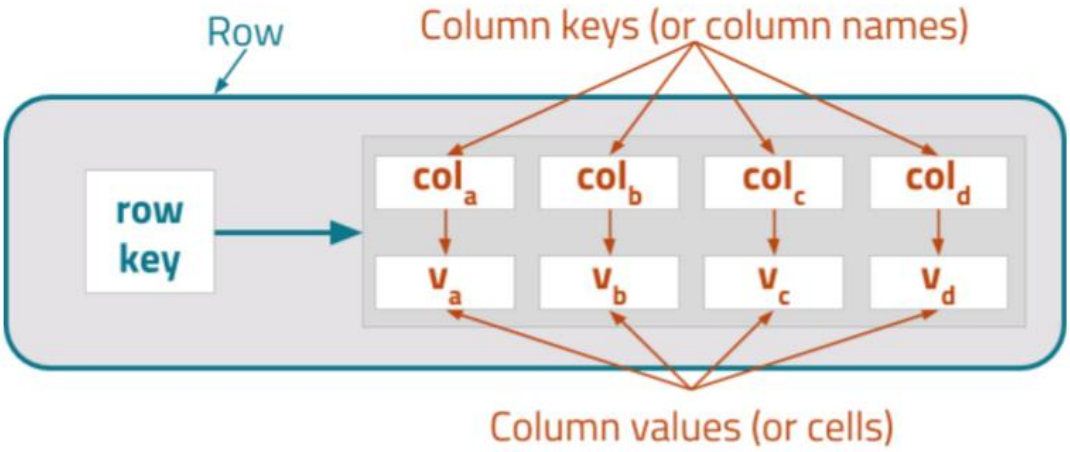

Figura 5.6 Estructura de Apache Cassandra

Una vez explicado el esquema que sigue Apache Cassandra, vamos a comentar el modelo de datos que se ha elaborado en este proyecto.

El modelo seguido comprende la siguiente estructura:

- El nombre de cada keyspace está formado por el nombre del recurso, seguido del carácter "_" y la fecha en la que se descargaron los recursos (recurso_fecha). Se realiza de esta manera debido a que Cassandra no permite que los nombres empiecen por números.
- Las distintas tablas que conforman cada keyspace, son todos los recursos que se han descargado en ese día concreto. El nombre de estas tablas está formado por el nombre del recurso y el momento (hora y minuto) en que se descargó dicho recurso. De manera que la estructura seguida es: nombre del recurso_hora_minuto.
- La clave de cada tabla está formada por la columna "identificador" de cada recurso.
- Los nombres de las columnas son, en el caso del catálogo de datos de Santander (http://datos.santander.es/), los mismos nombres que tienen las columnas de los recursos (omitiendo las cadenas: "ayto:", "dc:", "callej:" y "rdf:" que contienen algunas columnas al principio de su nombre). En el caso de la fuente de datos referente a las condiciones meteorológicas (https://api.netatmo.com/), el nombre de las columnas serán nombres como temperatura o humedad, dependiendo de la condición meteorológica a la que haga referencia cada columna.

La manera de realizar consultas siguiendo este modelo es la siguiente:

- Identificar el keyspace mediante el nombre del recurso que queremos consultar y la fecha en la que se descargó.
- Identificar la tabla mediante el nombre del recurso junto con su hora y minuto de descarga.

- Identificar la columna o las columnas que queremos consultar, mediante los nombres de las mismas.

5.3.2 Modelo de datos de Apache Hive

El modelo de datos de Apache Hive comprende los siguientes elementos:

Base de Datos: Contienen un conjunto de tablas. Son equivalentes a las bases de datos relacionales.

- Tablas: Son análogas a las tablas de bases de datos relacionales. Todos los datos de una tabla se almacenan en un directorio en HDFS. Las filas de una tabla se organizan en columnas similares a bases de datos relacionales. Hive también permite la creación de tablas externas.
- Particiones: Cada tabla puede tener una o más claves de partición que determinan cómo se almacenan los datos.
- Buckets - Los datos de cada partición pueden dividirse en Buckets, los cuales se basan en el código hash de una columna de la tabla. Cada Bucket se almacena como un archivo.

El modo de referirnos a un registro de datos es: Una base de datos, una tabla, una clave y el campo que deseamos.

En la siguiente figura (Figura 5.7), podemos ver la estructura que sigue el modelo de datos de Apache Hive.

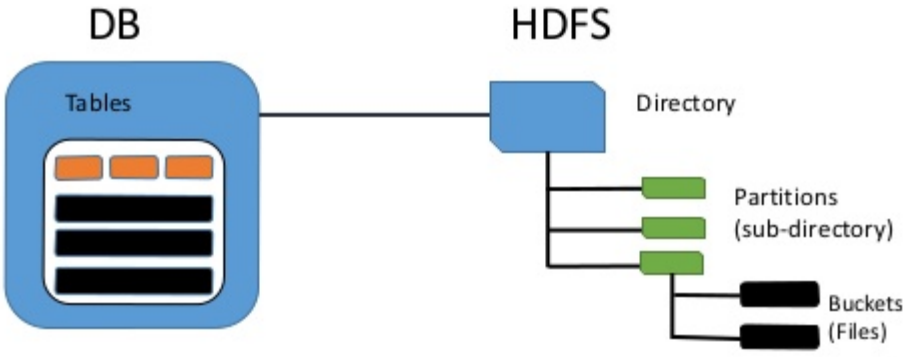

Figura 5.7 Modelo de Datos Apache Hive

El modelo de datos utilizado en este proyecto es el siguiente:

- El nombre de cada base de datos, es la fecha en la que se descargaron los recursos.
- Las tablas que forman parte de una base de datos, son todos los recursos que se han descargado en ese día concreto. El nombre de las tablas está formado por el nombre del recurso, junto con su hora y

minuto en que se descargó dicho recurso. De manera que la estructura seguida es: nombre del recurso_hora_minuto.
- La clave de cada tabla está formada por la columna “identificador” de cada recurso.
- Los nombres de las columnas son, en el caso del catálogo de datos de Santander (http://datos.santander.es/), los mismos nombres que tienen las columnas de los recursos (omitiendo las cadenas: “ayto:”, ”dc:”, ”callej:” y ”rdf:” que contienen algunas columnas al principio de su nombre). En el caso de la fuente de datos referente a las condiciones meteorológicas (https://api.netatmo.com/), el nombre de las columnas serán nombres como temperatura o humedad, dependiendo de la condición meteorológica a la que haga referencia cada columna.

La manera de realizar consultas siguiendo este modelo es la siguiente:

- Identificar la base de datos mediante la fecha en la que se descargó el recurso que queremos consultar.
- Identificar la tabla mediante el nombre del recurso junto con su hora y minuto de descarga.
- Identificar la columna o las columnas que queremos consultar, mediante la clave de la tabla y los nombres de las mismas.

5.3.3 Modelo de datos de MongoDB

MongoDB posee un modelo de datos orientado a documentos JSON con un esquema dinámico denominado BSON. Al poseer un esquema dinámico nos ofrece una enorme flexibilidad a la hora de realizar inserciones de datos. Puesto que no todos los documentos tienen porque tener la misma cantidad de campos, como ocurre en las bases de datos relacionales.

Los elementos que forman parte de este modelo de datos son los siguientes:

- Base de Datos: Están formadas por varias colecciones.
- Colecciones: Contienen varios documentos, son el equivalente a tablas en las bases de datos relacionales.
- Documentos: Contienen los datos, son el equivalente a registros en las bases de datos relacionales. La estructura de estos documentos está compuesta por “key-value pairs”.

El modo de referirnos a un registro de datos es: Una base de datos y una colección, y el campo o los campos que deseamos.

En la siguiente figura (Figura 5.8) podemos ver el esquema de la estructura que sigue MongoDB.

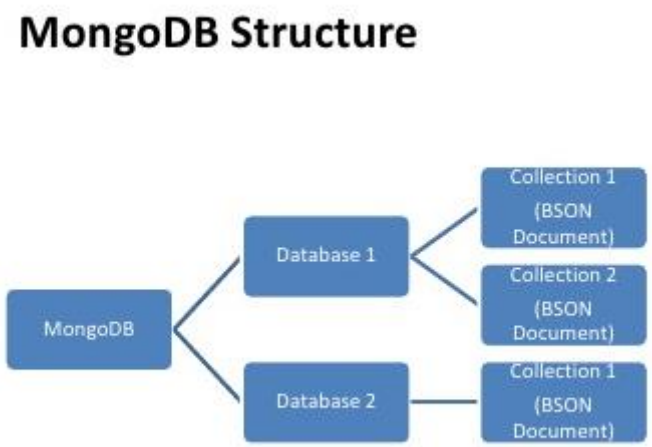

Figura 5.8 Modelo de datos MongoDB

El modelo de datos utilizado en este proyecto es el siguiente:

- El nombre de cada base de datos, es la fecha en la que se descargaron los recursos.
- Las colecciones que forman parte de una base de datos, son todos los recursos que se han descargado en ese día concreto. El nombre de las colecciones está formado por el nombre del recurso, junto con su hora y minuto en que se descargó dicho recurso. De manera que la estructura seguida es: nombre del recurso_hora_minuto.
- Los nombres de los campos son, en el caso del catálogo de datos de Santander (http://datos.santander.es/), los mismos nombres que tienen las columnas de los recursos (omitiendo las cadenas: “ayto:”, ”dc:”, ”callej:” y ”rdf:”que contienen algunas columnas al principio de su nombre). En el caso de la fuente de datos referente a las condiciones meteorológicas (https://api.netatmo.com/), el nombre de las columnas serán nombres como temperatura o humedad, dependiendo de la condición meteorológica a la que haga referencia cada columna.

La manera de realizar consultas siguiendo este modelo es la siguiente:

- Identificar la base de datos mediante la fecha en la que se descargó el recurso que queremos consultar.
- Identificar la colección mediante el nombre del recurso junto con su hora y minuto de descarga.
- Identificar la columna o las columnas que queremos consultar, mediante la clave de la colección y los nombres de las mismas.

6 Implementación

Durante este capítulo se detalla toda la implementación del proyecto. Dado que la arquitectura del sistema está dividida en módulos, dividiremos el capítulo en secciones, en las cuales explicaremos con detalle la implementación de cada módulo.

6.1 Módulo de Descargas

Como hemos comentado anteriormente en la fase de diseño, el módulo de descargas tiene como objetivo descargar, procesar y almacenar los archivos descargados desde las fuentes de datos.

El módulo está diseñado para descargar los recursos en todos los formatos disponibles en el catálogo de datos de Santander (http://datos.santander.es/), estos formatos son: JSON, CSV, RDF, ATOM, TURTLE, HTML, N3, XML y JSONLD. Dado que el resto del proyecto está pensado para trabajar con el formato CSV, si dicho formato no se encuentra incluido en los formatos de descarga, se incluirá automáticamente. Con respecto a la fuente de datos adicional (esta fuente es la referente a los datos meteorológicos, estos datos se encuentran en un recurso disponible en la web https://api.netatmo.com/), el recurso sólo está disponible en formato JSON, pero también se ha incorporado un parseador que transforma los datos a formato CSV. Esta incorporación se ha realizado debido a que el formato CSV ha sido el seleccionado para el resto del desarrollo del proyecto, puesto que es el formato que mejor se adapta con la mayoría de los sistemas de gestión de bases de datos. Por consiguiente, todos los recursos están disponibles en formato CSV.

Comentar también que el módulo está desarrollado de modo que el usuario solo tiene que insertar dos argumentos, el Path donde desea que se descarguen los recursos y el o los formatos deseados (Como se ha comentado anteriormente, el resto del desarrollo del proyecto está pensado para trabajar con el formato CSV).

A continuación se describen las clases principales que conforman este módulo.

6.1.1 Clase Recibidor

Para la descarga de recursos es necesario utilizar Apache Spark, a través de su componente Spark Streaming, que nos permite descargar y procesar datos en tiempo real. Lo primero que necesitamos es utilizar la clase Recibidor que proporciona Spark Streaming. Esta clase Recibidor, en principio, está diseñada solo para conexiones a través de socket, pero la API de Spark también ofrece la posibilidad de desarrollar un recibidor a medida. Debido a que en este caso la conexión no se realiza a través de un socket, sino que se realiza a través de una

RESTFUL API, se ha optado por el desarrollo de un recibidor customizado. La diferencia, básicamente, es que en vez de crear un socket de conexión, se crea una conexión mediante el protocolo HTTP y para poder realizar peticiones se realizan operaciones “GET” sobre este protocolo.

El recibidor es una hebra demonio, puesto que no queremos que deje de recoger datos, que necesita la implementación de los métodos run, receive y onStop. El método run realiza una llamada al método receive e inicializa la hebra. El método receive es el encargado de descargar, leer, dormir un minuto la hebra recibidor y almacenar “internamente” los recursos para su posterior procesado. Se duerme un minuto la hebra, debido a que los recursos tienen que ser solicitados cada minuto. Para el almacenamiento “interno” que realiza la hebra, la clase recibidor dispone del método store que nos permite realizar dicho almacenamiento. El método onStop se utiliza para detener la hebra.

Debido a que existe otra fuente de datos adicional (https://api.netatmo.com/), en la que el recurso solicitado sólo se encuentra disponible en formato JSON, se ha realizado un proceso de descarga exclusivo. Este recurso tiene como nombre “TempSantander” y se realiza el mismo procedimiento pero, en el caso de que el formato solicitado sea CSV, se realiza una conversión de JSON a CSV. Para descargar el recurso es necesario disponer de un “token” de autorización. Ha sido necesario incorporar un método que permite obtener el “token” a través de una conexión sobre el protocolo HTTPS. Dicho token deberá ir en las cabeceras de las peticiones GET.

Para clarificar lo comentado, en la siguiente figura (Figura 6.1) podemos ver el esquema que siguen las hebras recibidores de Spark Streaming.

Figura 6.1 Hebras Recibidores

Como podemos observar en la Figura 6.1, Spark Streaming descarga los datos a través de los recibidores y posteriormente son fragmentados. Spark procesa estos fragmentos y se obtienen los resultados.

6.1.2 Clase Aplicación

La clase Aplicación es la clase que contiene el método main del módulo. Por lo que es la encargada de la ejecución del módulo. Además, el método main es el encargado de realizar las siguientes operaciones: crear las URLs de los recursos, configurar Spark, crear el objeto JavaStreamingContext e inicializar la hebra principal.

La configuración de Spark depende de los recursos que vayamos a solicitar, puesto que necesitamos una hebra independiente para cada recurso. Debido a que el usuario puede elegir el o los formatos que desee, para calcular el número de hebras necesarias, es necesario que haya una relación entre el número de recursos (siempre estático) y la cantidad de formatos solicitada (se le suma uno a la cantidad de formatos para poder cubrir el caso en el que el formato CSV no se encuentre en la lista de formatos solicitados). De esta forma, podemos saber el número de procesos que va a necesitar Spark sumándole siete procesos más para que pueda realizar también el procesado y almacenamiento de datos. Este dato calculado es el que establecemos en la propiedad de configuración de Spark; esta propiedad se denomina setMaster y se introduce de la siguiente manera: setMaster(local[número de procesos necesarios]).

Por otro lado, tenemos el objeto JavaStreamingContext; este objeto es la hebra principal del módulo que va a gestionar a todas las demás hebras. Va llamando a todos los recibidores y, a su vez, también gestiona todo el procesado y almacenamiento de los recursos.

6.1.3 Clase Lanzador

Esta clase únicamente hace de nexo entre las demás clases del módulo. Inicializa el recibidor, que recibe como argumento desde el método main, y recibe los datos almacenados en el almacenamiento “interno” mencionado anteriormente. Estos datos se almacenan en un objeto llamado JavaReceiberInputDStream: este objeto es un conjunto de RDDs que contiene los datos enviados por el recibidor. Como comentamos en el capítulo 2, Spark divide los datos en RDDs. Estas divisiones nos permiten realizar varias operaciones en paralelo y además nos aporta tolerancia a fallos. Gracias a los RDDs podemos procesar y almacenar todos los datos de una manera segura y más rápida.

Por último, la clase llama al método RecibeYProcesa de la clase procesar, para que procese y almacene los datos en disco.

6.1.4 Clase Procesar

La clase procesar es la encargada de procesar todos los recursos proporcionados por el recibidor. Una vez procesados, se almacenarán todos los recursos en disco.

Para procesar los datos, utilizamos el método foreachRDD de la API de Spark que nos permite realizar varias operaciones a todo el conjunto de RDDs. Este método tiene como argumento una cadena y un procedimiento denominado Call. Mediante el procedimiento Call realizamos todas las operaciones que necesitemos sobre los RDDs. En este caso, nos interesa obtener la fecha, hora y minutos para crear los directorios en base a la ruta proporcionada por el usuario, así como almacenar los datos en dicha ruta.

Para almacenar los datos es necesario crear dos rutas. Esto es debido a que Spark con motivo de la tolerancia a fallos guarda los datos divididos en varias partes, las cuales se almacenan en el formato del sistema de ficheros de Hadoop. Para realizar este almacenamiento de las divisiones de datos creadas por Spark, se ha creado una ruta adicional. Para guardar los datos, Spark proporciona el método saveAsTextFile, el cual nos permite guardar los datos en cualquier formato.

Como los datos se almacenan divididos y en el formato de sistemas de ficheros de Hadoop es necesario combinar todos estos archivos para poder obtener el fichero final, con el formato que deseamos. Para este propósito se ha implementado el método merge que recibe como parámetros la ruta donde se encuentran almacenados los datos divididos y la ruta final. Se combinan todos los datos divididos y se obtiene un único fichero que se almacena en la ruta de destino.

6.2 Módulo de Almacenamiento

El módulo de almacenamiento tiene que leer todos los datos descargados por el módulo de descargas y almacenarlos en base de datos. En este caso, otro requisito adicional es que puedan ser almacenados en cualquiera de los tres sistemas de gestión de bases de datos seleccionados y que se analizan en este proyecto: MongoDB, Apache Cassandra o Apache Hive.

También es necesario que el módulo anote datos estadísticos referentes al almacenamiento de datos en los diferentes sistemas de gestión de bases de datos para que, posteriormente, los datos sean analizados por el módulo estadístico y se cree una representación gráfica de los resultados. Se ha optado por analizar dos tipos de datos:

- tiempo de almacenamiento por número de recursos
- tiempo de almacenamiento por el tamaño de los recursos

A continuación se describen las clases que conforma este módulo.

6.2.1 Clase Conector

La clase conector es la encargada de establecer las conexiones con los tres sistemas de gestión de bases de datos. Dado que cada conexión contiene sus peculiaridades, se va a detallar como se han realizado cada una de ellas.

6.2.1.1 Conexiones con MongoDB, Apache Hive y Apache Cassandra

La primera operación que debemos realizar es crear las conexiones con los tres sistemas de gestión de bases de datos. La librería de Spark dispone de drivers de conexión para crear las conexiones que necesitamos, salvo para el caso de Apache Hive, para el cual ha sido necesario recurrir al componente DriverManager de la librería de Java.

Para poder realizar las inserciones ha sido necesario utilizar dos componentes: Statement y Session. El objeto Statement es un objeto que nos permite realizar queries SQL en Apache Hive. El objeto Session tiene la misma funcionalidad pero para el caso de Apache Cassandra. Ambos objetos se utilizan para el almacenamiento por lo que detallaremos más aspectos sobre ellos más adelante.

La configuración de conexión de MongoDB es mucho más sencilla que en los dos casos anteriores. Para establecer la conexión con MongoDB basta con el método desarrollado CreaOpciones. Este método simplemente crea un Map de opciones donde incluye la dirección de la base de datos y el usuario y la contraseña en caso de que la base de datos disponga de usuario y contraseña.

6.2.2 Clase AlmacenamientoBD

La clase AlmacenamientoBD es la clase principal este módulo. Es la clase encargada de realizar el almacenamiento de los todos los recursos que se han descargado a través del módulo de Descargas.

6.2.2.1 Creación de las bases de datos e inserciones de datos

Una vez realizadas las configuraciones de conexión, el siguiente paso a realizar es la lectura de ficheros en formato CSV. Recorreremos todos los directorios creados en el módulo de descargas leyendo todos los archivos almacenados en dichos directorios.

Para realizar el proceso de lectura nos apoyaremos en la herramienta DataFrames que ofrece Apache Spark a través de su componente SQLContext. (Hay una gran cantidad de desarrolladores que contribuyen a realizar mejoras para Apache Spark. En concreto, la comunidad databricks ha creado una librería denominada “com.databricks.spark.csv” que, entre otras cosas, nos permite leer y escribir ficheros CSV a través de los DataFrames).

Se vuelve a hacer uso de los objetos Statement, Session y el Map options. Esto es debido a que los datos se almacenarán creando bases de datos con la fecha, en el caso de MongoDB y Apache Hive, y creando un Keyspace con el nombre del recurso y la fecha, en el caso de Apache Cassandra. En el caso de Apache Cassandra ha sido necesario incluir el nombre del recurso antes de la fecha, debido a que no permite la creación de Keyspaces que empiecen por un número. Se ha realizado de esta manera para que queden bien diferenciados los datos de modo que estén identificados por su fecha de descarga.

Debido a las distintas peculiaridades de los tres sistemas de gestión de bases de datos, algunos de los sistemas no permiten los guiones en los nombres. De modo que ha sido necesario cambiar los guiones de las fechas por “_”, de modo que para todos los sistemas sea igual.

Así mismo, ha sido necesario renombrar las columnas de los recursos debido a que contienen caracteres como “:” o “.”, que no son admitidos en ninguno de los sistemas de gestión de bases de datos. Aprovechando esto, también se logra que los recursos queden almacenados de una forma más clara y con nombres de columnas más cortos.

De esta tarea se encargará el método TransformaDataFrame el cual recibe como parámetros el DataFrame que se quiere modificar y el recurso al que hace referencia. El método renombra todas las columnas que deseemos cambiar y devuelve un DataFrame con las columnas modificadas. Para realizar dicha operación, utilizaremos el método WithColumnsRenamed que nos proporciona la API de Apache Spark.

El siguiente paso a realizar es el almacenamiento de los datos en los sistemas de gestión de base de datos. Esta fase tiene ciertas complejidades debido a que los sistemas de gestión de bases de datos no operan de igual manera a la hora de realizar inserciones de datos. Para la inserción de los datos se ha creado un índice que será la hora y minuto en las que se descargaron los recursos. De

modo que el nombre de la tabla o de la colección, lo conforman el nombre del recurso junto a su hora y minuto (recurso_hora_minuto). Esto nos sirve para diferenciar los recursos que pertenecen a una misma fecha, pero a distintos minutos. Se ha realizado de esta forma para que la estructura de los recursos sea la misma en los tres sistemas de gestión de bases de datos.

MongoDB

Para el caso de MongoDB sólo ha sido necesario la incorporación de la librería "com.stratio.datasource.mongodb", la cual nos permite leer e insertar datos a MongoDB. MongoDB a diferencia de Apache Hive y Apache Cassandra, no requiere la creación de tablas. En MongoDB los datos se distribuyen en colecciones, en las que no importan los tipos de datos. Para poder completar el almacenamiento es necesario utilizar los métodos Write y Save de los DataFrames, además de especificar el nombre de la colección que deseamos para cada recurso. Este nombre está compuesto por el nombre del recurso junto a la hora y el minuto en el que fueron descargados.

Por ejemplo, supongamos que se va a insertar el recurso con nombre "mediciones", descargado el 08-07-2016 a las 15:04. La inserción se realizaría en la base de datos 08_07_2016 y en la colección mediciones_15_04.

Para consultar dicho recurso, tendríamos que especificar el nombre de la base de datos (08_07_2016), junto con el nombre de la colección que queremos consultar. En este caso, la colección sería mediciones_15_04.

Apache Cassandra

En el caso de Apache Cassandra el almacenamiento es similar a MongoDB, salvo que antes es necesario crear las tablas para cada recurso. Además, también es necesario especificar los tipos de datos para cada columna que componen las tablas. Para la realización de esta operación, se ha desarrollado el método CreaTablaRecurso. Este método recibe como parámetros el objeto Session, el recurso que queremos procesar, el DataFrame que contiene el archivo CSV y un índice para diferenciar los recursos.

Se crea una tabla distinta para cada recurso. En cada tabla especificamos los tipos de datos, clave y los nombres de cada columna. Una vez creada esta tabla, utilizamos los métodos Write y Save de los DataFrames para almacenar el recurso. Para poder utilizar el método write con Apache Cassandra, ha sido necesario utilizar la librería "org.apache.spark.sql.cassandra". Adicionalmente, hay que especificarle al método Write el nombre de la tabla a la que queremos destinar el recurso.

Por ejemplo, supongamos que, como en el caso anterior, va a insertar el recurso con nombre "mediciones", descargado el 08-07-2016 a las 15:04. La inserción se realizaría en el keyspace mediciones_08_07_2016 y en la tabla mediciones_15_04. La clave de esta tabla la formaría la columna "identifier" del recurso mediciones.

Para consultar dicho recurso, tendríamos que especificar el nombre del keyspace (mediciones_08_07_2016), junto con el nombre de la tabla que queremos consultar. En este caso, el nombre de la tabla sería mediciones_15_04.

Apache Hive

Para el último caso de Apache Hive es necesario realizar la inserción a través del objeto Statement. Esto es debido a que los DataFrames no disponen de ninguna librería adecuada para el almacenamiento de estos recursos en Apache Hive por lo que, tanto las operaciones de creación de tablas como las de inserción, se realizan a través de este objeto. Para este propósito se han desarrollado dos métodos: CreaTablaRecursoHive e Inserta.

El método CreaTablaRecursoHive crea las tablas para los distintos recursos. Este método recibe como parámetros el objeto Statement, el recurso que se desea almacenar, la fecha del recurso, el DataFrame que contiene el archivo CSV y el índice para almacenar el recurso. Para crear las tablas es necesario también incluir el tipo de dato y nombre para cada una de las columnas.

Una vez creadas las tablas, se realiza la inserción a través del método inserta. Este método ejecuta queries SQL de inserción, de modo que recorre todas las filas del archivo CSV y las une para ejecutarlas en una única query.

Por ejemplo, supongamos que nuevamente se va a insertar el recurso con nombre “mediciones”, descargado el 08-07-2016 a las 15:04. La inserción se realizaría en la base de datos 08_07_2016 y en la tabla mediciones_15_04.

Para consultar dicho recurso, tendríamos que especificar la base de datos con nombre (08_07_2016), junto con el nombre de la tabla que queremos consultar. En este caso, el nombre de dicha tabla sería mediciones_15_04.

6.2.3 Clase EscribeFichero

Esta clase es la encargada de escribir las anotaciones realizadas durante el almacenamiento en disco. Para ello recibe como parámetro en el propio constructor, el nombre de la ruta donde se van a almacenar las anotaciones y el tipo de sistema de gestión de bases de datos al que hacen referencia dichas anotaciones.

Una vez finalizada la ejecución del módulo de Almacenamiento, el método EscribeEnFichero almacena todos los datos en la ruta deseada. En este caso, se almacenan dos archivos correspondientes a los tiempos de almacenamiento por número y tamaños de recursos.

6.3 Módulo Estadístico

El módulo estadístico tiene como objetivo elaborar gráficas analizando los datos producidos en el módulo de almacenamiento. Se realizan dos tipos de gráficas: en una se representan el número de recursos almacenados con respecto al tiempo y en la otra se representa el tamaño almacenado con respecto al tiempo. El tiempo está expresado en segundos y el tamaño está expresado en KiloBytes.

Para la elaboración de dichas gráficas se ha utilizado la librería jfreechart. Esta herramienta nos permite crear gráficas complejas de una manera sencilla. Las gráficas pueden ser de distintos tipos, como: gráficos XY, gráficos en forma de tarta, gráficos de barra (horizontales y verticales, apilados e independientes), single valued (termómetro, brújula, indicadores de velocidad) y varias gráficas específicas (tablas de viento, gráficas polares, etc).

Se ha optado por dos gráficas XY en la cuales el eje Y corresponde al tiempo y el eje X varía según la estadística que representamos. En la primera estadística, el eje X representa el número de recursos almacenados y en la segunda, el tamaño de los recursos almacenados.

El constructor de la clase recibe como parámetros el título de la gráfica, la ruta donde están almacenados los ficheros con los datos que se van a representar y la ruta donde se almacenará la gráfica. Todos estos parámetros son especificados por el usuario.

Una vez elaboradas las gráficas se almacenarán en disco, como se especificó en la fase de diseño. Para este propósito se ha utilizado la clase ScreenImage. Esta clase complementa la librería jfreechart, y entre otras cosas, nos permite guardar en disco la gráfica elaborada por jfreechart en formato “jpg”.

6.4 Módulo de Representación Gráfica

El módulo de representación gráfica es el encargado de extraer los datos almacenados en base de datos, manipularlos y posteriormente almacenarlos en disco en formato CSV. Se elige el formato CSV porque es uno de los formatos aceptados por la API de Google. Este proceso se realiza para que el usuario pueda ver los datos de forma visual en un mapa correspondiente a la ciudad de Santander.

Muchos de los datos descargados no tienen la estructura adecuada para poder representarlos directamente. Los datos geográficos de algunos recursos, se encuentran en un sistema de coordenadas que no es el sistema que actualmente se usa en Europa o a nivel mundial. Los datos se encuentran dispersos en varios archivos o, sencillamente, contienen más información de la que nos interesa representar. Para conseguir poder realizar esta operación, se vuelve a hacer uso de los DataFrames, ya que nos permiten extraer, manipular y almacenar los datos en disco de una manera sencilla y rápida.

El sistema de gestión de bases de datos que mejor se comporta para el objetivo de este proyecto es MongoDB, por lo que este módulo de representación gráfica sólo está diseñado para extraer datos desde él.

Para la extracción de datos, solo es necesario especificar la librería que vamos a utilizar para leer los datos desde MongoDB. En este caso esta librería se denomina “com.stratio.datasource.mongodb”, la cual nos va a permitir extraer información desde MongoDB e incluirla en un DataFrame.

Dado que cada recurso tiene sus peculiaridades, vamos a dividir en distintos apartados los recursos representados, de modo que explicaremos el procedimiento seguido para transformar dichos recursos.

6.4.1 Taxis y Paradas de Taxis

En este recurso tenemos la información dispersa en los recursos taxis_taxis_en_parada y taxis_paradas. El recurso taxis_paradas contiene todas las paradas de taxi de Santander, tanto si hay algún taxi como si no. En cambio el recurso taxis_taxis_en_parada contiene el número de taxis que se encuentran en cada parada.

Para que la información quede clara es necesario combinar ambos recursos, de manera que podamos representar todas las paradas junto con el número de taxis que se encuentran en ellas. Esta información es mucho más útil combinada, puesto que en el caso de necesitar un taxi, sería de gran ayuda saber dónde se encuentran las paradas y si existe algún taxi disponible en ellas.

Esta combinación entre los dos recursos se realiza a través del método join que proporcionan los DataFrames. Este método nos permite realizar un join entre dos DataFrames del mismo modo que se realiza un join entre dos tablas en el lenguaje SQL. Basta con leer los recursos en dos DataFrames distintos, y utilizar el método join indicando la columna por la que se combinarán los DataFrames.

Además de lo comentado, los DataFrames nos permiten realizar consultas tipo select. Los DataFrames tienen un método denominado regisTempTable, que nos permite crear una tabla temporal con los datos que contiene el DataFrame. De modo que podremos seleccionar las columnas que deseamos, y desechar las columnas que consideremos innecesarias desde estas tablas temporales.

El método ObtieneTaxis selecciona las columnas con información útil, combina ambos recursos y llama al método GuardaCSV para almacenar el resultado.

El método GuardaCSV es el encargado de almacenar el DataFrame en un único fichero CSV. Este Método se repite con cada DataFrame que deseamos almacenar.

El método ObtieneTaxis almacena dos archivos. El primero hace referencia a todas las paradas de taxis donde hay algún taxi disponible. En cambio el

segundo hace referencia a todas las paradas de taxis donde no se encuentra ningún taxi disponible. Para conseguir esta diferenciación de paradas con taxis y sin taxis, es tan sencillo como realizar dos consultas SQL sobre el DataFrame que contiene toda la información.

Se realiza de esta forma para que en un mismo mapa podamos representar tanto las paradas donde podemos encontrar taxis, como aquellas en las que no.

En cuanto al método GuardaCSV, su función es almacenar en único fichero con formato CSV toda la información del DataFrame. Al igual que en el módulo de descargas, volvemos a hacer uso del método Merge para combinar los datos divididos que almacena Spark, y así conseguir un único archivo que incluya todos los datos divididos.

Para almacenar el DataFrame en formato CSV basta con llamar al método Write con la librería “"com.databricks.spark.csv", como hacíamos en el módulo de descargas. Pero en este caso no es tan sencillo, a la hora de llamar al método write de los DataFrames, se llama también al método repartition con el parámetro 1. Se realiza de esta forma porque el propio DataFrame se encuentra particionado en varias partes, por lo que es necesario que quede particionado en una sola parte.

6.4.2 Sensores Smart Traffic y Mediciones

Estos dos recursos hacen referencia a sensores de tráfico y su información nos indica la situación del tráfico en la ciudad de Santander.

Por un lado tenemos el recurso Sensores_smart_traffic, que contiene toda la información relativa a sensores de alta precisión situados en las entradas de Santander. Y por otro lado tenemos el recurso Mediciones, que contiene toda la información relativa a los sensores situados por toda la ciudad de Santander.

Para la manipulación y el almacenamiento de los sensores Smart traffic se ha desarrollado el método CreaDataFrameSensoresSmart. Dicho método extrae toda la información desde MongoDB, selecciona la información que deseamos representar y después la almacena en un fichero con formato CSV mediante el método GuardaCSV.

Para la manipulación y almacenamiento del recurso Mediciones, se ha desarrollado el método GeneraDataFrameSensores. En este método se realiza el mismo procedimiento que en el método CreaDataFrameSensoresSmart, salvo que en este caso, adicionalmente se lee un fichero denominado Espiras.csv. Esto es debido a que la situación geográfica de los sensores se encuentra almacenada en este fichero.

Mediante la combinación de ambos ficheros y la selección de la información que necesitamos obtenemos el resultado final, que es almacenado mediante el método GuardaCSV.

6.4.3 Zonas 30

El recurso zonas 30 es más especial, puesto que la información que necesitamos se encuentra en una cadena de caracteres. Esta cadena contiene todas las coordenadas geográficas de las zonas de Santander, donde hay que circular a 30 KM/H. Estas coordenadas se encuentran en el sistema de coordenadas ED50, el cual ha sido sustituido por el WGS84 a nivel mundial (ETRS89 en Europa). Debido a este inconveniente ha sido necesario incorporar un conversor de coordenadas, que transforme las coordenadas ED50 al sistema de coordenadas WGS84.

Este conversor utiliza las librerías geotools y opengis. El método funciona de manera que tan solo proporcionando los sistemas de coordenadas de salida y de entrada, nos devuelve las coordenadas transformadas.

Además de esto, ha sido necesario restar una serie de decimales a las coordenadas. Esto se debe a que las coordenadas proporcionadas por el catálogo de datos de Santander se encuentran ligeramente desplazadas del lugar que le corresponden. De modo que estas operaciones consiguen que las coordenadas se ajusten a su lugar adecuado.

Adicionalmente, la aplicación de Google no permite incorporar un conjunto de coordenadas en forma de ruta al mapa a través de ficheros CSV. Por lo que se ha optado por realizar una representación de este recurso en un mapa estático, a partir de la herramienta StaticMap proporcionada por Google.

Esta herramienta funciona a través de peticiones HTTP tipo “GET”, las cuales mediante una serie de parámetros nos devuelve un mapa customizado. Esta API está algo limitada en cuanto a la longitud de las peticiones, por lo que ha sido imposible representar todas las zonas 30 del recurso. En lugar de representar todas, se ha representado una de las zonas a modo de ejemplo.

Para elaborar este mapa estático se ha desarrollado el método GeneraZonas30, el cual es el encargado de generar una lista con las cadenas que utilizaremos como parámetros de la API de Google. Una vez generada la lista con los parámetros, basta con realizar la petición “GET” y almacenar el mapa resultante. Para ello se han desarrollado los métodos GeneraMapa y GuardaMapa.

6.4.4 Sensores Smart enviroment monitoring

Este recurso hace referencia a sensores que miden el ruido y la temperatura. Para la representación de estos sensores se ha desarrollado el método GeneraDataFrameEnv. Dicho método extrae la información desde MongoDB, selecciona la información que necesitamos y lo almacena en un fichero CSV.

6.5 Módulo de Envío y Actualización de Datos

El módulo de Envío y Actualización de Datos es el encargado de interactuar con las Fusion Tables de Google. El módulo se encuentra formado por las clases FusionTables y Uploader. Vamos a dividir este apartado en dos secciones, para explicar el funcionamiento de ambas clases.

6.5.1 Clase FusionTables

La clase FusionTables, es la clase principal del módulo. Mediante esta clase creamos las tablas, obtenemos el identificador de cada una y creamos los uploaders. Para crear las tablas se ha desarrollado el método créateTables, el cual todas las tablas, con sus campos necesarios. Una vez creadas las tablas, es necesario obtener su identificador para operar con ellas. Para este propósito se ha desarrollado el método listTables, el cual obtiene todos los identificadores de las tablas que hemos creado a través de sus nombres.

Cuando tenemos todos los identificadores, podemos operar con las tablas de manera muy sencilla, mediante los métodos que proporciona la API de Google.

Para todo el proceso de subir y actualizar datos, hacemos uso de los uploaders. Los cuales son creados y lanzados en esta clase.

6.5.2 Clase Uploader

La clase Uploader es la encargada de insertar datos en las tablas y actualizarlos cada cierto tiempo. Lo que pretendemos es que mediante estas actualizaciones, podamos visualizar como van evolucionando los datos en los mapas. Debido a que los datos se corresponden con mapas y tablas distintas, es necesario que todos los datos se vayan insertando y actualizando de manera simultánea. Para ello, se ha diseñado esta clase como hebra. Al diseñarlo como hebras conseguimos que todos los recursos se suban al mismo tiempo y no de manera secuencial. Cosa que nos interesa bastante, para que no tengamos que esperar a que se termine de actualizar un recurso para empezar con el siguiente.

6.6 Aplicación Web

La aplicación Web nos permite visualizar todos los mapas elaborados en la aplicación de Google, además también se ha incorporado el ejemplo del mapa estático elaborado para las Zonas 30.

Para desarrollar esta aplicación se ha utilizado Wordpress, que es un CMS bastante sencillo de utilizar y mediante el cual podemos realizar aplicaciones

Web. En este caso se ha desarrollado un login, formularios de registro y un menú mediante el cual podemos acceder a todos los mapas.

Los mapas se han incorporado a la aplicación Web mediante código javascript, que permite crear mapas y actualizarlos a través de queries hacia las fusion tables.

7 Evaluación de los Sistemas de Almacenamiento

Este capítulo está destinado a la evaluación de los Sistemas de Almacenamiento, la cual es uno de los objetivos de este proyecto. Durante esta sección comentaremos los resultados de las pruebas de almacenamiento realizadas sobre MongoDB, Apache Hive y Apache Cassandra.

7.1 Necesidades de los sistemas de Almacenamiento

Antes de explicar las pruebas de almacenamiento realizadas y sus resultados, vamos a comentar brevemente las operaciones necesarias para realizar el almacenamiento de cada sistema de gestión de bases de datos.

MongoDB es el sistema de almacenamiento que requiere de menos operaciones a la hora de almacenar. Para poder almacenar datos en MongoDB debemos crear una base de datos (en caso de que no esté creada) y crear una colección, para especificar donde se almacenaran los datos. Para el caso de MongoDB estas operaciones son realmente sencillas, ya que no será necesario indicar el tipo de datos ni el esquema que siguen. De modo que lo único que necesitamos especificar es el nombre que queremos para la base de datos y la colección.

En cambio para Apache Cassandra y Apache Hive, es necesario especificar también los tipos de datos y el esquema. Por lo que aparte de indicar el nombre de la base de datos (keyspace en caso de tratarse de Apache Cassandra) y la tabla a la que hacemos referencia, también será necesario especificar las columnas que conforman cada recurso y los tipos de datos de estas columnas. Estas necesidades repercuten en las pruebas realizadas, puesto que crean una mayor complejidad a la hora de mapear e insertar los recursos en la base de datos correspondientes.

También es necesario comentar que Apache Spark posee muy buenos componentes para MongoDB y Apache Cassandra. Estos componentes nos permiten establecer la conexión con el sistema de almacenamiento, y realizar inserciones de datos de una manera sencilla y rápida. En el caso de Apache Hive, no ha sido posible realizar estas operaciones a través de componentes de Apache Spark. Este aspecto también ha repercutido en las pruebas realizadas, ya que estos componentes están diseñados para acelerar y facilitar las conexiones e inserciones con los sistemas de almacenamiento.

7.2 Pruebas de Almacenamiento

Se han realizado pruebas de almacenamiento para MongoDB, Apache Cassandra y Apache Hive. Estas pruebas han sido necesarias, puesto que uno de los objetivos de este proyecto es evaluar el comportamiento de los tres sistemas de gestión de bases de datos. Para elaborar dichas pruebas, se ha

instalado cada sistema de gestión de bases de datos en máquinas virtuales distintas. De este modo, hemos conseguido que todas las pruebas se realicen en igualdad de condiciones.

Para conseguir evaluar el comportamiento de los sistemas, se han descargado y almacenado un total de 100 recursos de diversos tamaños. Durante este almacenamiento se han estado anotando una serie de datos que son analizados y evaluados por el módulo estadístico. Una vez analizados y evaluados, el modulo estadístico elabora una gráfica con la representación de dichos datos.

En este caso se han elaborado dos gráficas correspondientes a dos estadísticas:

- tiempo de almacenamiento por número de recursos
- tiempo de almacenamiento por el tamaño de los recursos

Para elaborar ambas gráficas ha sido necesario aplicar un eje logarítmico, en este caso el eje Y que representa al tiempo. Esto es debido a que los valores que arroja Apache Hive son bastantes más dispersos que los de Apache Cassandra y MondoDB. De modo que si aplicamos el eje en forma logarítmica podemos ver los resultados de estos tres sistemas de gestión de bases de datos sin ningún problema.

A continuación en la siguiente figura (Figura 7.1) podemos ver la gráfica que hace referencia al tiempo consumido por el número de recursos almacenados.

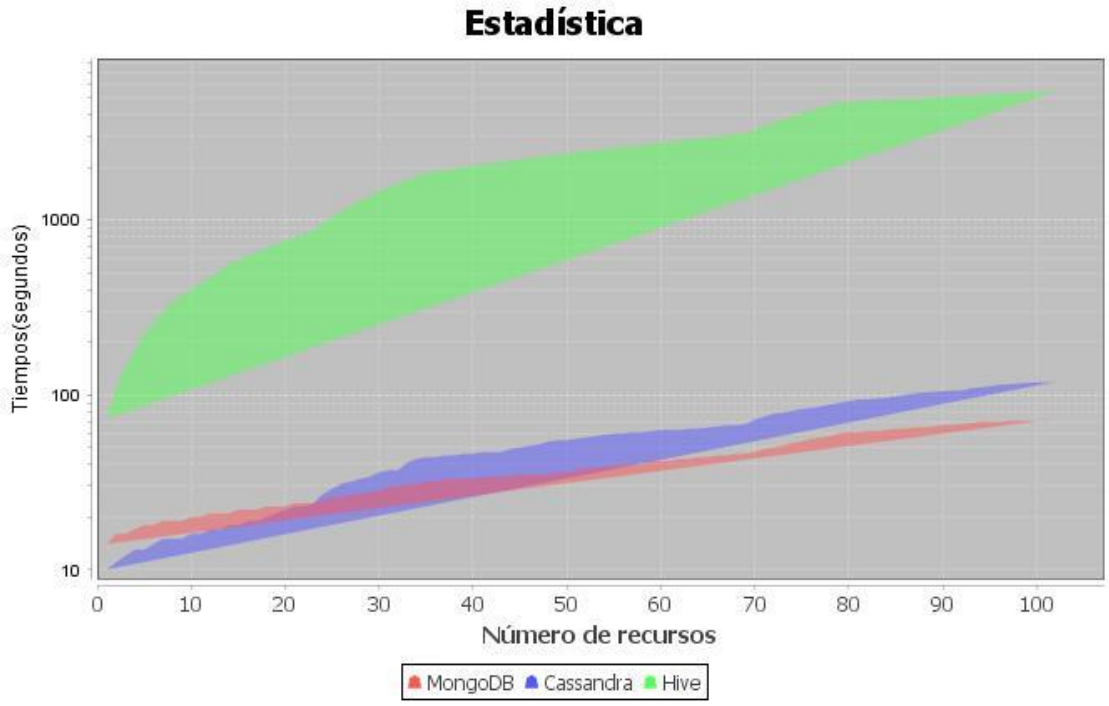

Figura 7.1 Gráfica número de recursos con respecto al tiempo

Como se puede observar, Apache Hive es el sistema de gestión de bases de datos que peor se comporta a la hora de realizar el almacenamiento, ya que en almacenar 100 recursos consume bastante más tiempo que MongoDB y Apache Cassandra.

Por otro lado, MongoDB y Apache Cassandra son más parecidos. En este caso Apache Cassandra se comporta mejor que MongoDB cuando la cantidad de recursos es menor que 20. Pero a medida que vamos aumentando el número de recursos almacenados, MongoDB es el sistema que consume menos tiempo.

Para el propósito de este proyecto necesitamos que el sistema de gestión de bases de datos sea realmente rápido, puesto que estamos tratando con un gran volumen de recursos que cambian constantemente.

En la siguiente figura (Figura 7.2) podemos ver la gráfica correspondiente al tiempo de almacenamiento por el tamaño de los recursos.

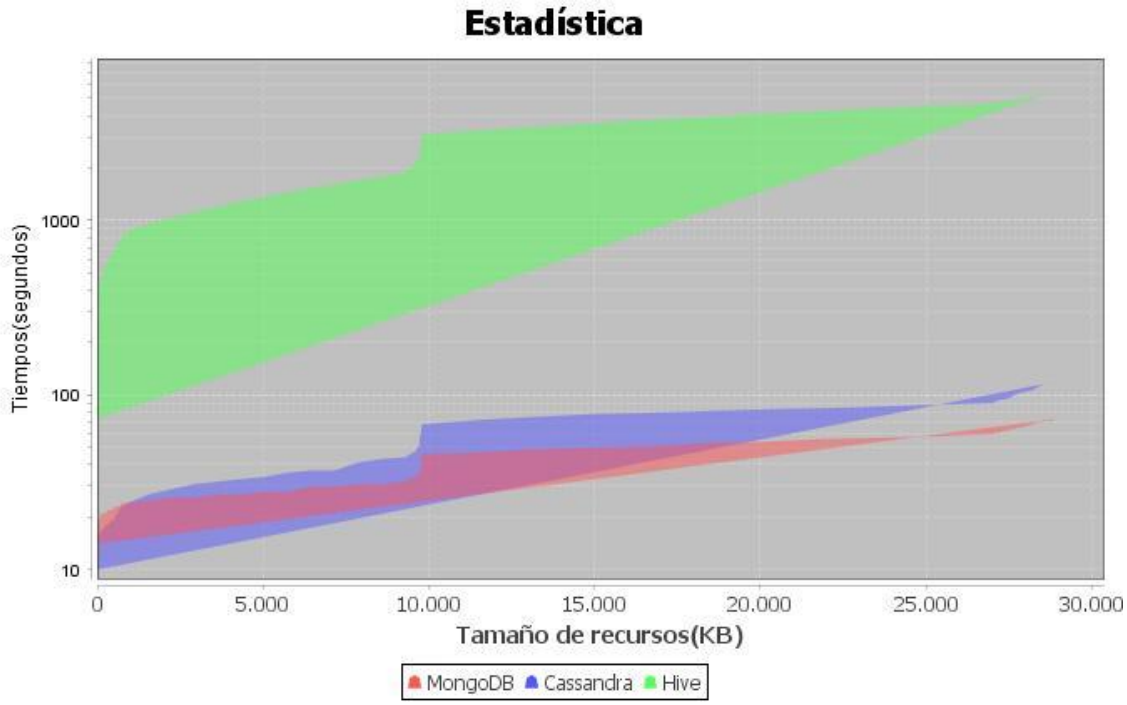

Figura 7.2 Gráfica tamaño de recursos con respecto al tiempo

En esta grafica también podemos observar que el comportamiento de los tres sistemas de gestión de bases de datos es similar al de la gráfica anterior. Apache Hive se encuentra muy por debajo del rendimiento que aportan Apache Cassandra y MongoDB. Estos dos últimos, se comportan de manera bastante parecida, pero cuando el tamaño almacenado va creciendo, MongoDB responde de mejor manera que Apache Cassandra.

Los tiempos de respuesta de las consultas de datos producen resultados similares a los analizados en estas dos gráficas (Figura 7.1 y Figura 7.2).

En conclusión, viendo el resultado obtenido en ambas gráficas, podemos afirmar que MongoDB es el sistema de gestión de bases de datos que mejor se ajusta a los objetivos de este proyecto. Puesto que es el sistema que menos tiempo consume por número y tamaño de recursos.

8 Interfaz de Usuario

En este capítulo comentaremos la interacción del usuario con el sistema, de modo que se expondrán ejemplos de cómo deben ejecutarse cada uno de los módulos que conforman el sistema. Adicionalmente, se explicará el funcionamiento de la aplicación web para visualizar todas las representaciones de los datos en el mapa de Santander.

Ya que el sistema se distribuye en módulos, dividiremos este capítulo en secciones, de modo que cada sección irá destinada a un módulo del sistema.

8.1 Módulo de Descargas

El módulo de descargas está desarrollado de modo que el usuario solo tiene que introducir como argumentos el directorio donde se desean almacenar los archivos descargados y el o los formatos deseados. Debido a que el resto de módulos trabajan con el formato CSV, aunque no se indique en la lista de formatos, los recursos también se descargaran en formato CSV.

Para la ejecución del módulo de descargas, es necesario que el usuario introduzca los siguientes parámetros:

- Directorio donde se almacenarán los archivos.
- Formato de los archivos, o en caso de ser varios formatos separados por comas.

Para la ejecución de este módulo debemos situarnos en el directorio a través de la consola de comandos y ejecutar la siguiente instrucción: java -Xmx512m -jar Modulo_de_Descargas.jar (directorio donde se almacenaran los archivos) (formato o en caso de ser varios formatos separados por comas).

Un ejemplo de ejecución válido sería el siguiente:

java -Xmx512m -jar Modulo_de_Descargas.jar c:/datos/ csv

y para varios formatos:

java -Xmx512m -jar Modulo_de_Descargas.jar c:/datos/ csv,json

A continuación en la siguiente figura (Figura 8.1) podemos observar un ejemplo de ejecución de este .jar.

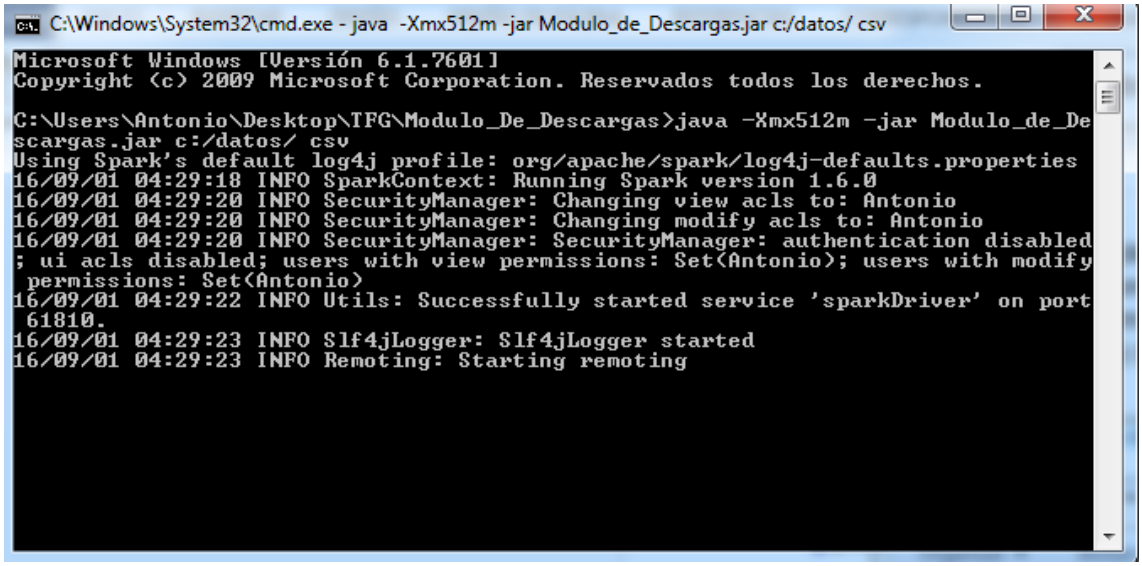

Figura 8.1 Ejecución Módulo de Descargas

Una vez que comience la ejecución, comenzará a descargar todos los recursos en el formato (o los formatos) indicados en la ruta de destino proporcionada. En este caso el formato es “csv” y la ruta de destino es “C:/datos”. En la siguiente figura (Figura 8.2) podemos observar un ejemplo del almacenamiento de uno de los recursos.

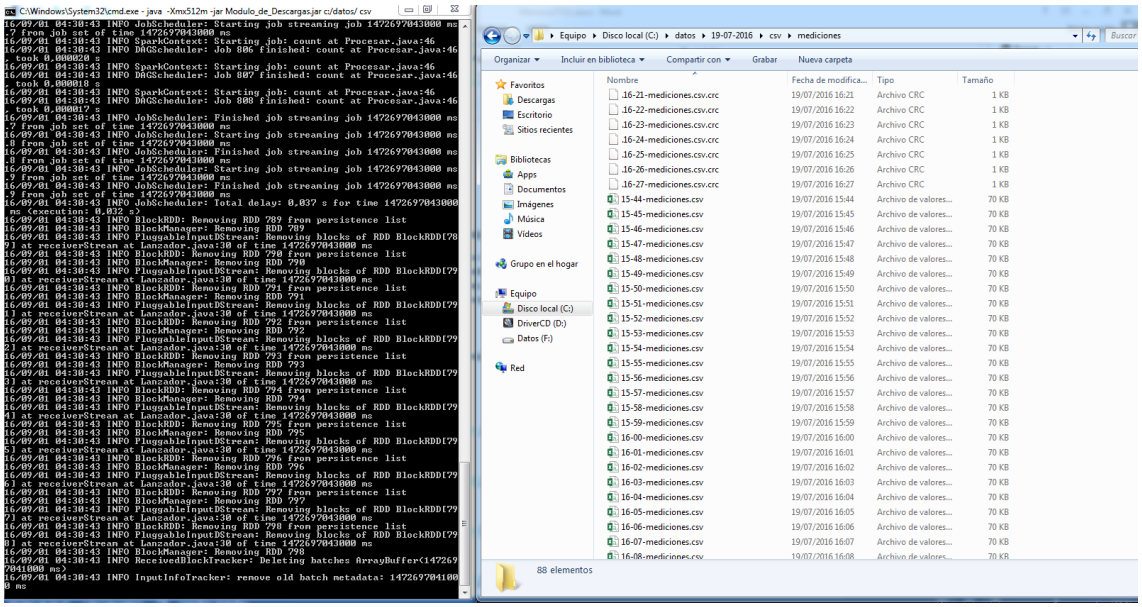

Figura 8.2 Almacenamiento del Módulo de Descargas

Para detener la ejecución del módulo basta con cerrar la ventana de la consola de comandos.

8.2 Módulo de Almacenamiento

El módulo de almacenamiento requiere que tengamos un sistema de gestión de bases de datos esperando conexiones entrantes. En este módulo el usuario debe introducir los siguientes argumentos, en este orden:

- Ruta donde se encuentran los archivos descargados por el módulo de descargas.
- Ruta donde se almacenarán las anotaciones estadísticas, para que el módulo estadístico pueda elaborar las gráficas.
- Tipo del sistema de gestión de bases de datos (mongodb, cassandra o hive).
- Dirección del sistema de gestión de bases de datos (ip:puerto).
- En caso de que el sistema de gestión de bases de datos disponga de usuario, introducir usuario.
- En caso de que el sistema de gestión de bases de datos disponga de contraseña, introducir contraseña.

Para poder ejecutar el módulo debemos situarnos, mediante la consola de comandos, en el lugar donde se encuentre el archivo Modulo_de_Almacenamiento.jar. Una vez situados lanzaremos la sentencia java -Xmx512m -jar Modulo_de_Almacenamiento.jar (junto con los argumentos solicitados).

En la siguiente figura (Figura 8.3) podemos observar un ejemplo de ejecución del módulo sobre MongoDB.

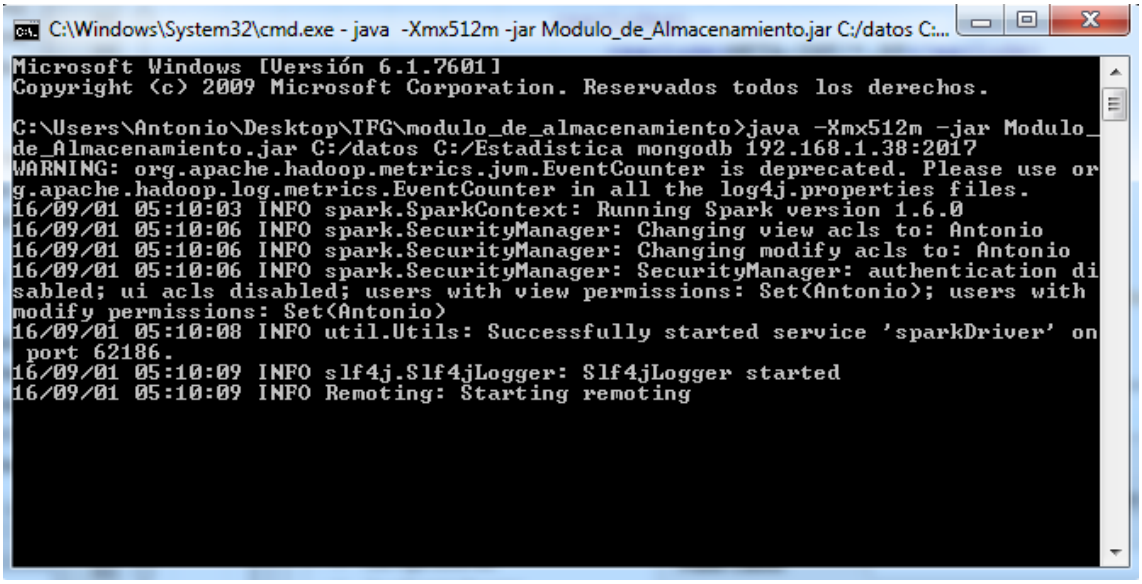

Figura 8.3 Ejemplo de ejecución del módulo de almacenamiento

Una vez que el módulo finalice su ejecución, habrá introducido todos los recursos en la base de datos correspondiente. A continuación en la siguiente figura (Figura 8.4) podemos ver el almacenamiento realizado en MongoDB.

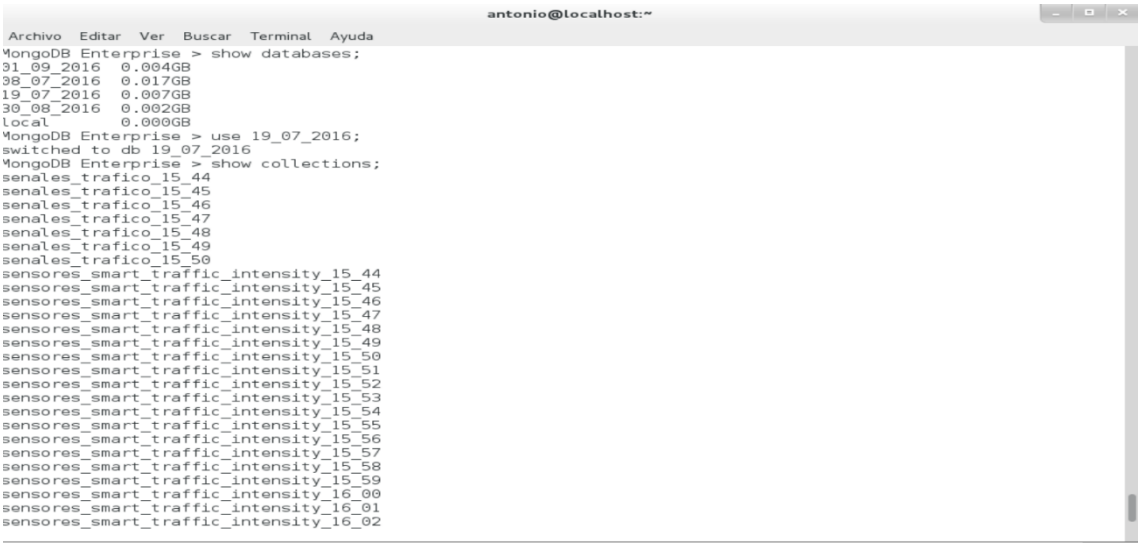

Figura 8.4 Almacenamiento en MongoDB

8.3 Módulo Estadístico

El módulo estadístico necesita que el usuario especifique los siguientes parámetros:

- Ruta donde se encuentran los ficheros con las notaciones estadísticas.
- Ruta de destino de las imágenes de las gráficas.

Para ejecutar este módulo, basta con situarnos mediante la consola de comandos en el directorio donde se encuentra el archivo Modulo_Estadistico.jar y ejecutar la siguiente sentencia:

java -Xmx512m -jar Modulo_Estadistico.jar (Ruta de las anotaciones) (Ruta de destino de las gráficas)

En la siguiente figura (Figura 8.5) podemos ver un ejemplo de ejecución del módulo estadístico.

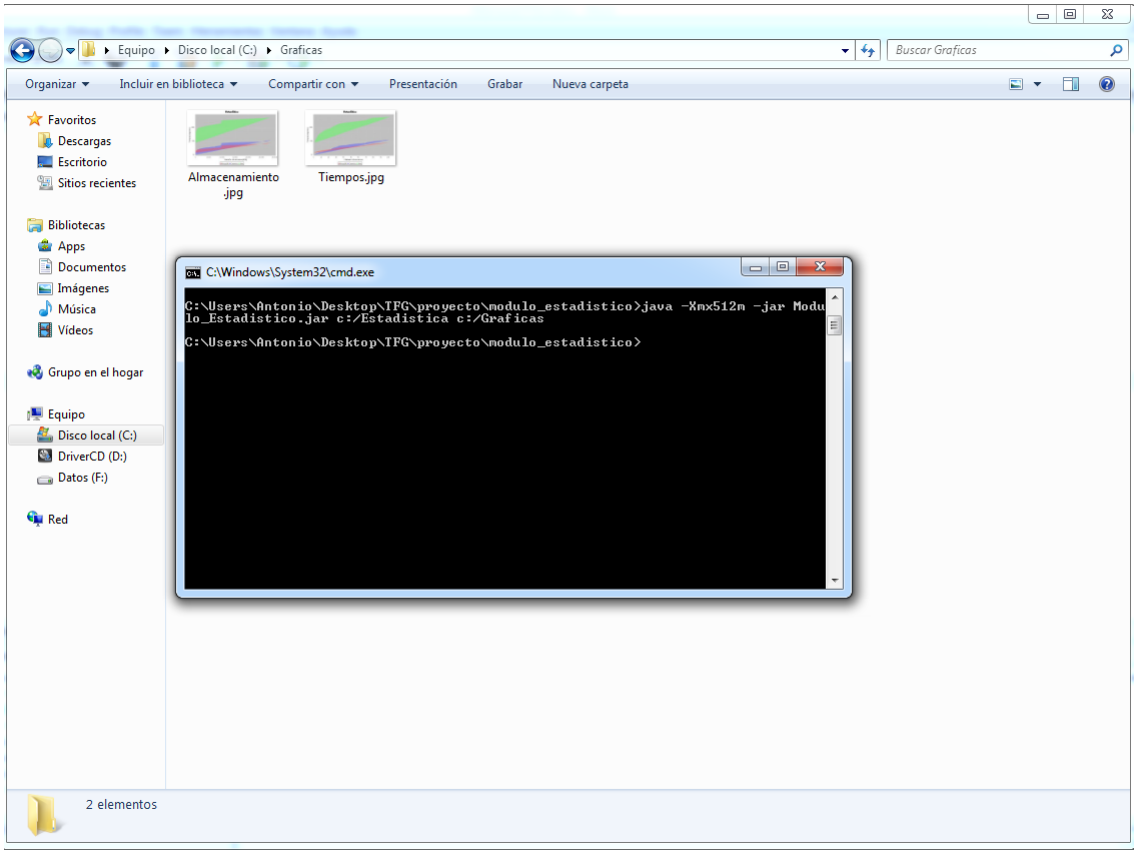

Figura 8.5 Ejecución Módulo Estadístico

8.4 Módulo de Representación Gráfica

El módulo de representación gráfica se ha desarrollado para representar una serie de datos de prueba desde MongoDB. Estos datos corresponden con una base de datos y una colección de cada recurso. Para que el módulo funcione correctamente es necesario tener en el mismo directorio que el archivo Modulo_de_Representacion_Grafica.jar, el archivo Espiras.csv.

Los argumentos que debe proporcionar el usuario son los siguientes:

- Ruta donde se almacenaran los ficheros finales.
- Base de datos desde la que queremos extraer los datos para su representación.
- Hora de inicio.
- Hora final.
- Dirección de MongoDB (ip:puerto).
- En caso de que MongoDB disponga de usuario, introducir usuario.
- En caso de que MongoDB disponga de contraseña, introducir contraseña.

Para ejecutar este módulo es necesario posicionarnos en el directorio donde se encuentra el archivo Modulo_de_Representacion_Grafica.jar y ejecutar la siguiente instrucción:

java -Xmx512m -jar Modulo_de_Representacion_Grafica.jar (junto con los argumentos solicitados).

En la siguiente figura (Figura 8.6) podemos ver un ejemplo de ejecución de este módulo.

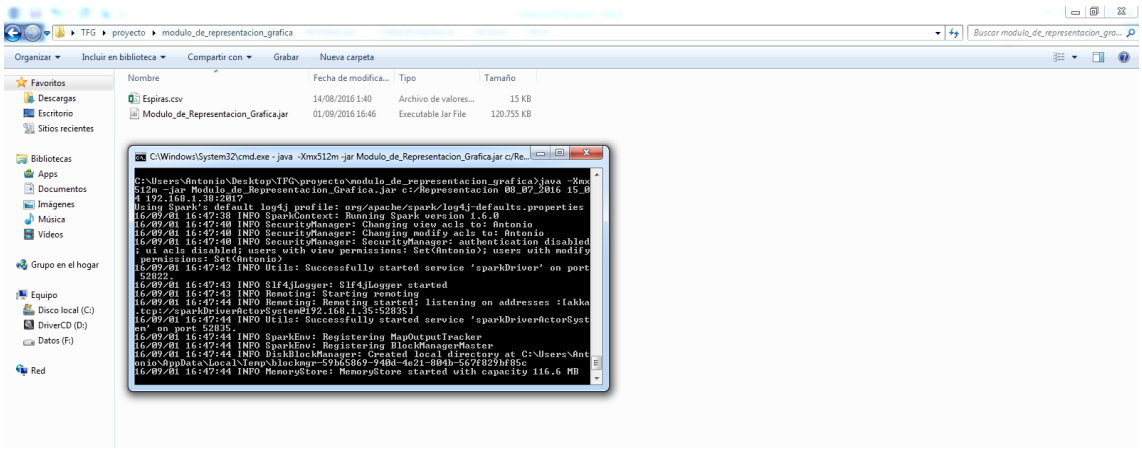

Figura 8.6 Ejecución del Módulo de Representación Gráfica

Una vez ejecutado, si nos vamos a la ruta donde se almacenan los archivos, tenemos todos los resultados en formato CSV. Excepto el recurso Zonas 30 que aparecerá como un mapa estático en formato png (Figura 8.7).

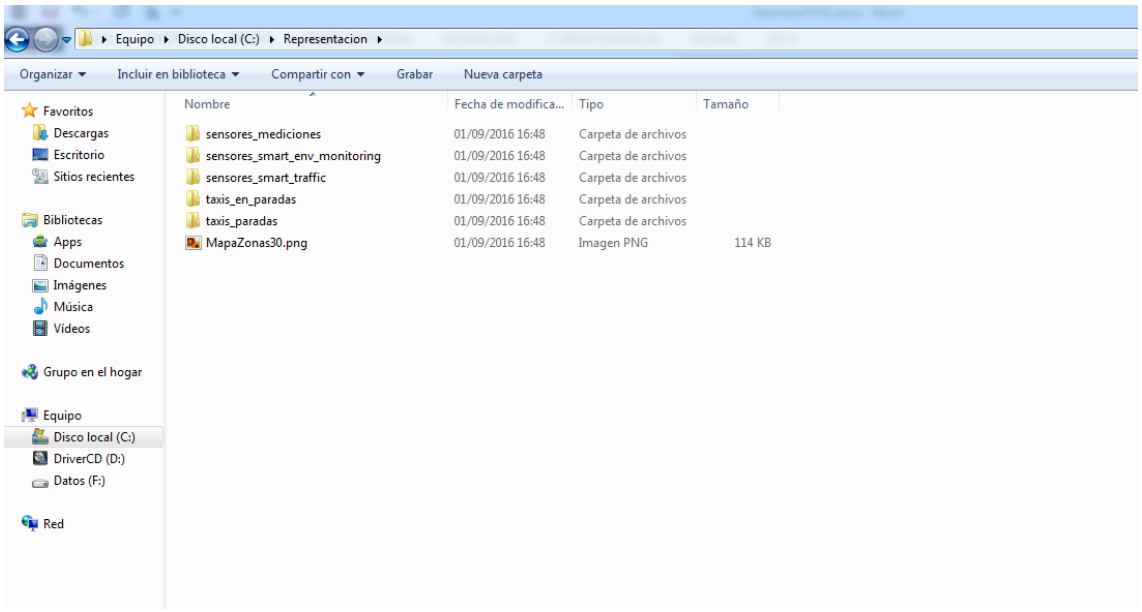

Figura 8.7 Resultados ejecución Módulo de Representación Gráfica

A continuación explicaremos los pasos a seguir para incluir un recurso en la herramienta Mis Mapas de Google.

8.5 Módulo de Envío y Actualización de Datos

El módulo de Envío y Actualización de datos se ha desarrollado para interactuar con la API de Google. De manera que podamos almacenar en unas tablas denominadas Fusion Tables, toda la información y actualizarla cada cierto tiempo. De esta manera, podremos ver en el mapa la evolución de los datos.

Los argumentos que debe proporcionar el usuario son los siguientes:

- Ruta donde se encuentran almacenados los archivos del módulo de Representación Gráfica.
- Si/No. En caso de querer crear tablas nuevas en Fusion Tables se indicaría mediante la palabra Si y en caso de querer actualizar las tablas ya creadas, se indicaría mediante la palabra No.

Para ejecutar este módulo es necesario posicionarnos en el directorio donde se encuentra el archivo Modulo_de_Representacion_Grafica.jar y ejecutar la siguiente instrucción:

java -Xmx512m -jar Modulo_de_Envío_y_Actualizacion_de_Datos.jar (junto con los argumentos solicitados).

En la siguiente figura (Figura 8.8) podemos ver un ejemplo de ejecución de este módulo.

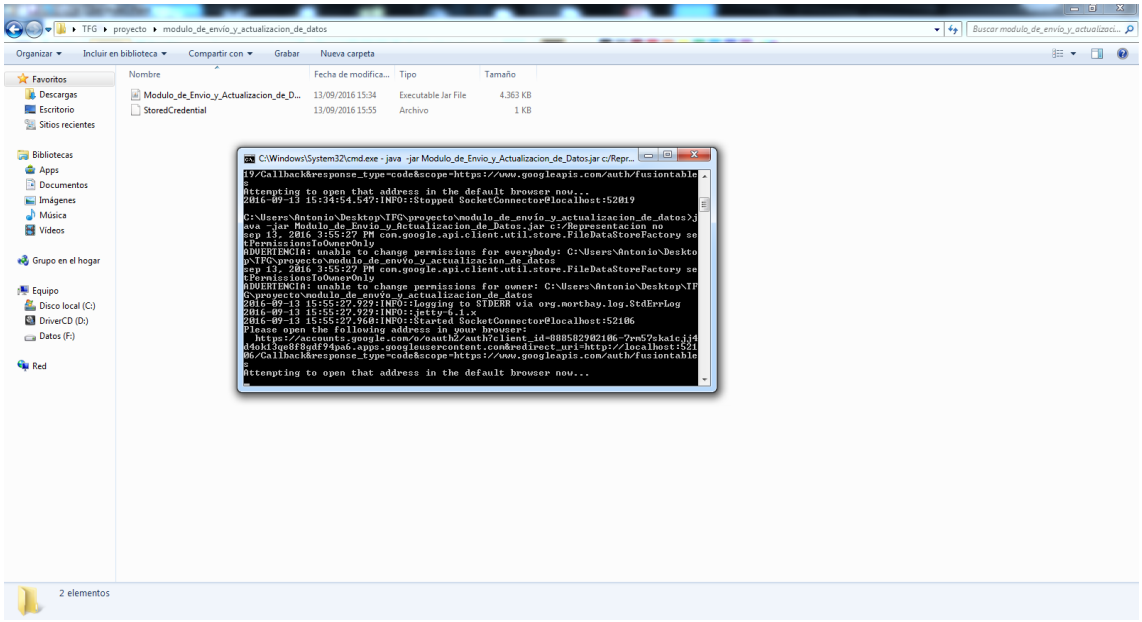

Figura 8.8 Ejecución del Módulo de Envío y Actualización de Datos

Como podemos observar en la Figura 8.8, hay un fichero que se genera cuando lo ejecutamos. Esto es debido a que es necesario solicitar credenciales para acceder a las tablas del usuario. Para ello, cuando ejecutemos el módulo se nos abrirá el navegador solicitándonos permisos para administrar las tablas dinámicas. En la siguiente figura (Figura 8.9) podemos ver el permiso solicitado a través del navegador.

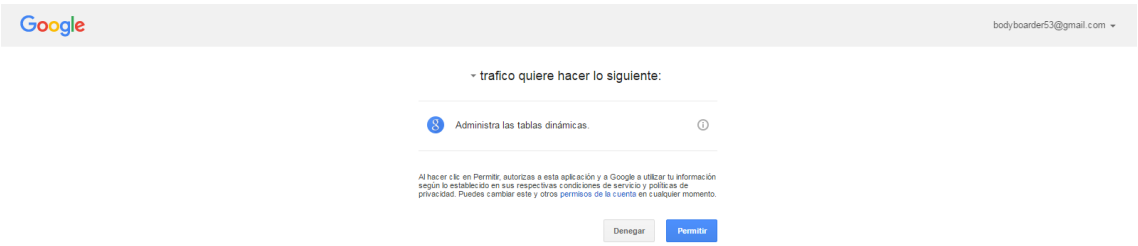

Figura 8.9 Credenciales Fusion Tables

A continuación explicaremos los pasos a seguir para ver las tablas y el mapa de los datos.

8.6 Representación en Fusion Tables

Cuando hayamos ejecutado el módulo de Envío y Actualización de Datos, ya tendremos nuestras tablas creadas con información. Para poder ver estas tablas, basta con irnos a la sección “My Drive” de Google y nos aparecerán todas en la sección mi unidad.

En la siguiente figura (Figura 8.10) podemos ver la sección My Drive, con las tablas. En este caso, me aparecen en una carpeta porque he decidido almacenarlas ahí, pero aparecen en el directorio raíz de Drive.

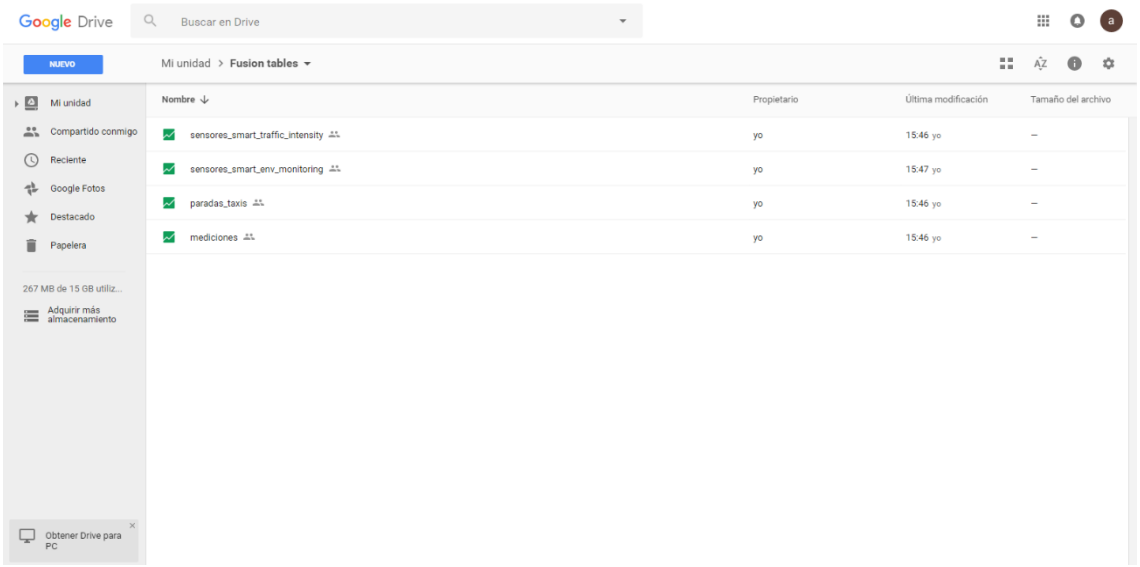

Figura 8.10 Drive con Fusion Tables

Para ver la información, basta con hacer doble click sobre la tabla que queramos y aparecerá la información. Para ver el mapa de los datos solo tenemos que hacer click en la pestaña de Map of Location.

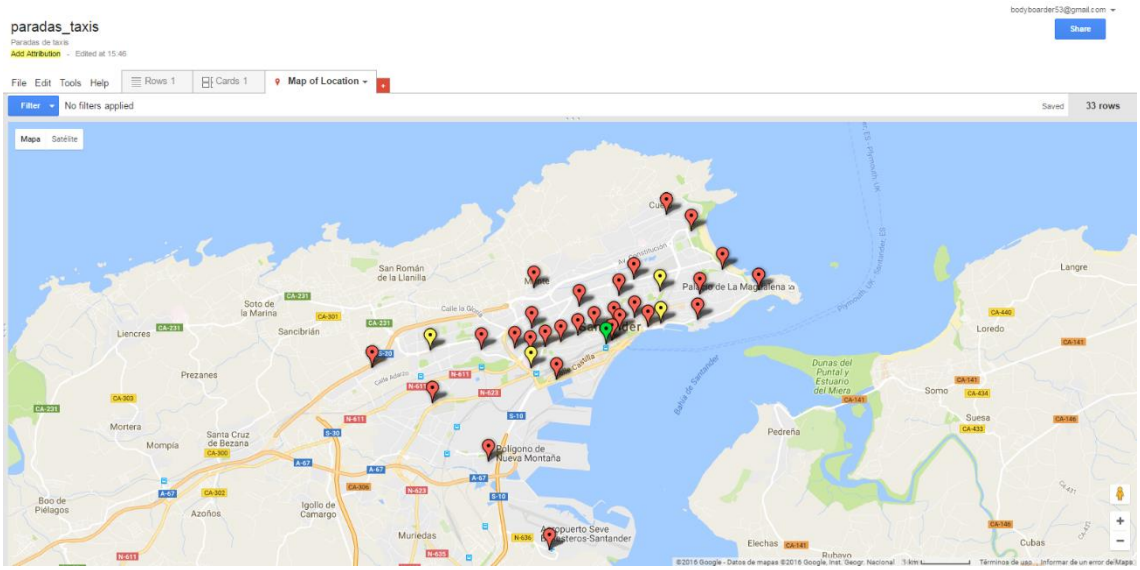

Figura 8.11 Mapa de paradas de taxis en Fusion Tables

Estos mapas (Figura 8.11) pueden ser compartidos entre los usuarios que se desee y también tenemos la opción de incorporarlos a nuestro sitio web. Como podemos observar en la figura 8.11, los marcadores se encuentran distribuidos en colores, de manera que nos informa con tan solo un vistazo de la situación de los datos. En este caso, se trata de la situación de las paradas de taxis de Santander. De manera que el color rojo simboliza que no hay ningún taxi, el color amarillo que hay un taxi y el color verde que hay más de uno.

Este aspecto es muy interesante ya que ofrece la posibilidad de que con un solo vistazo podamos obtener información. Pero hay un inconveniente, los datos de las Fusion Tables cuando cambian se cambian también en el mapa pero tarda un tiempo en mostrar estos cambios. Los datos se transmiten a la información del marcador de manera rápida, pero el aspecto visual tarda unos dos minutos en cambiar. Sin embargo si cambiamos el zoom del mapa (nos acercamos o nos alejamos), estos cambios aparecen. Esto es debido a problemas de caché de la aplicación de Google. Además los datos se procesan en los servidores de Google, lo que implica que se necesite algo más de tiempo en actualizarse el aspecto visual.

8.7 Aplicación Web

La aplicación Web es un medio para que el usuario pueda ver todas las representaciones en mapas de manera conjunta. La dirección de la aplicación es http://trafficsantander2016.esy.es/. Para poder acceder a la Web basta con navegar a la dirección desde cualquier navegador.

Dado que la aplicación web dispone de varias páginas con distinta funcionalidad, vamos a dividir esta sección en subsecciones para cada una de las páginas.

8.7.1 Login

Para acceder al contenido de la aplicación es necesario ser un usuario, para ello todos los usuarios dispondrán de un nombre de usuario y una contraseña.

En la siguiente figura (Figura 8.12) podemos ver la página de inicio, que nos permite acceder a todo el contenido.

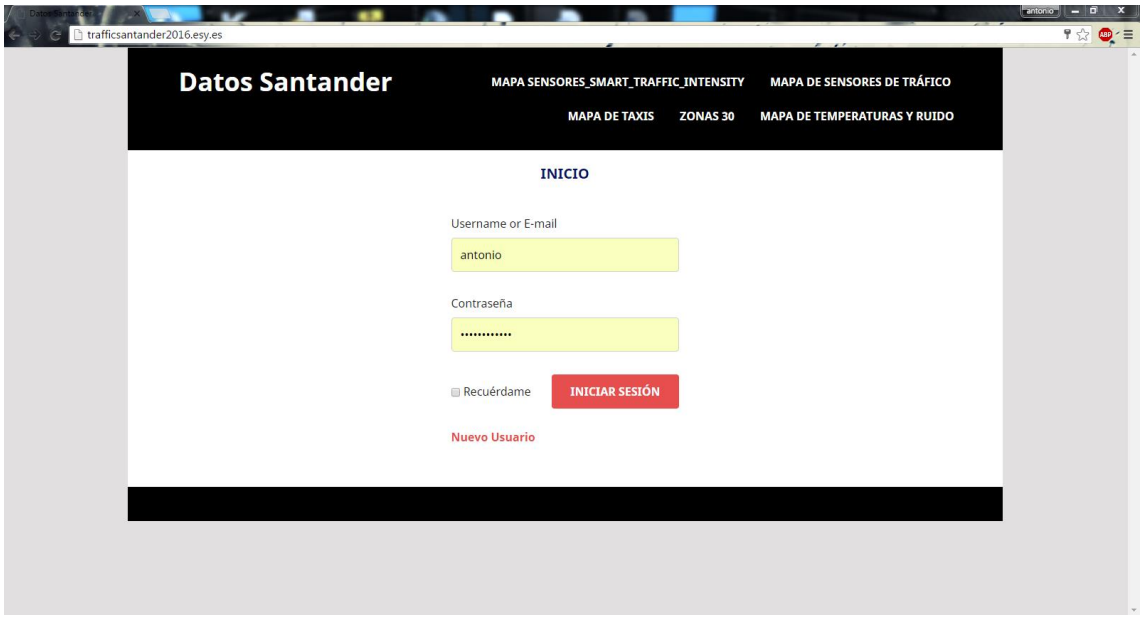

Figura 8.12 Página de Inicio

Para acceder al contenido bastaría con rellenar el formulario de login y clickear en iniciar sesión. También existe la opción de seleccionar la casilla Recuérdame, que nos permite que el nombre de usuario y la contraseña se autocompleten cada vez que ingresamos a la web. De manera que no sea necesario escribir ninguno de los campos del formulario.

8.7.2 Nuevo Usuario

En caso de ser un nuevo usuario, es necesario rellenar el formulario de registro. Para ello en la página de inicio (figura 8.12) hay una opción denominada nuevo usuario. Si hacemos click aquí se nos abrirá el formulario de registro, que podemos ver a continuación (Figura 8.13).

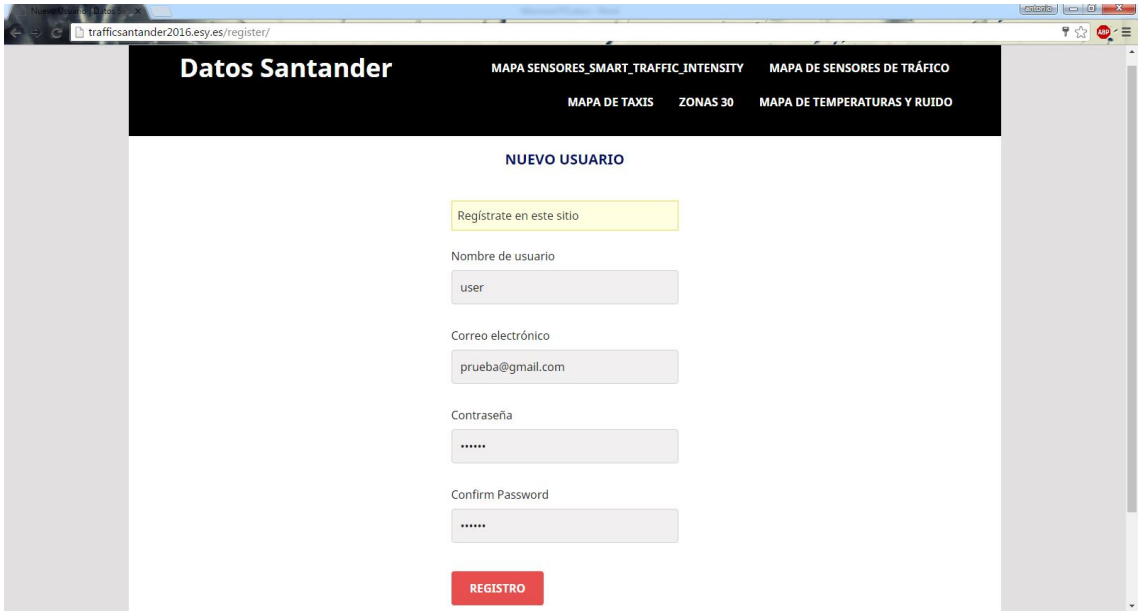

Figura 8.13 Formulario de Registro

Todos los campos de este formulario son obligatorios, en caso de no rellenarlos al pulsar registro nos saldrá un aviso de que es necesario rellenar los campos. Una vez pulsado el botón de registro, nos devolverá a la página de login para que podamos iniciar sesión(Figura 8.14).

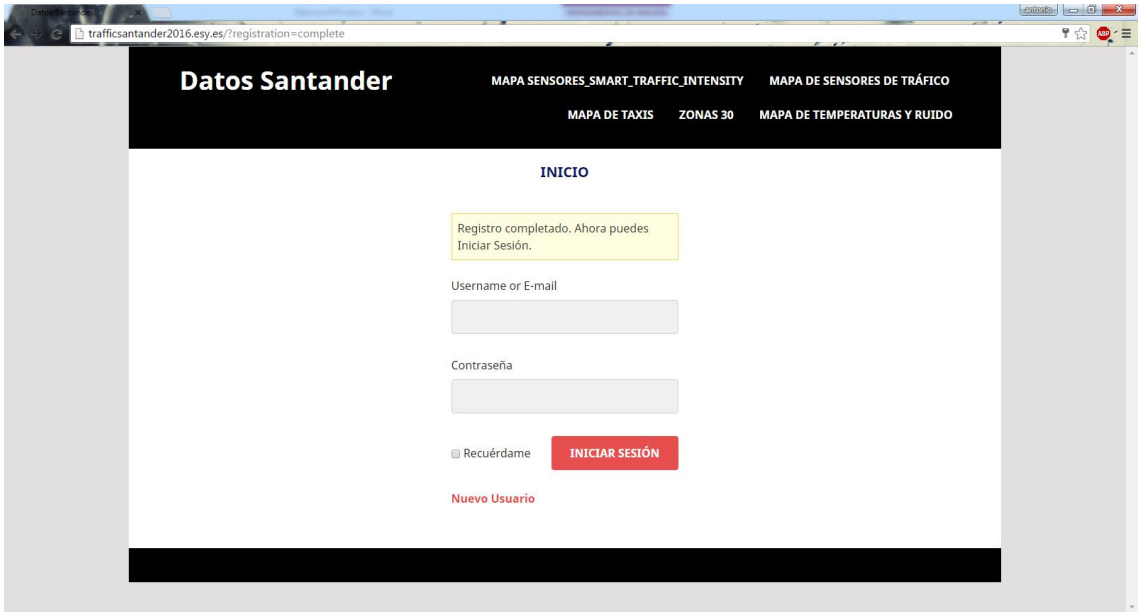

Figura 8.14 Página de Login nuevo usuario

8.7.3 Mapas y Zonas 30

El primer mapa que encontramos cuando iniciamos sesión, es el Mapa de tráfico. Está página es una sección de la web donde podemos visualizar el mapa con la representación de los datos referentes a los sensores de tráfico de Santander(Figura 8.15).

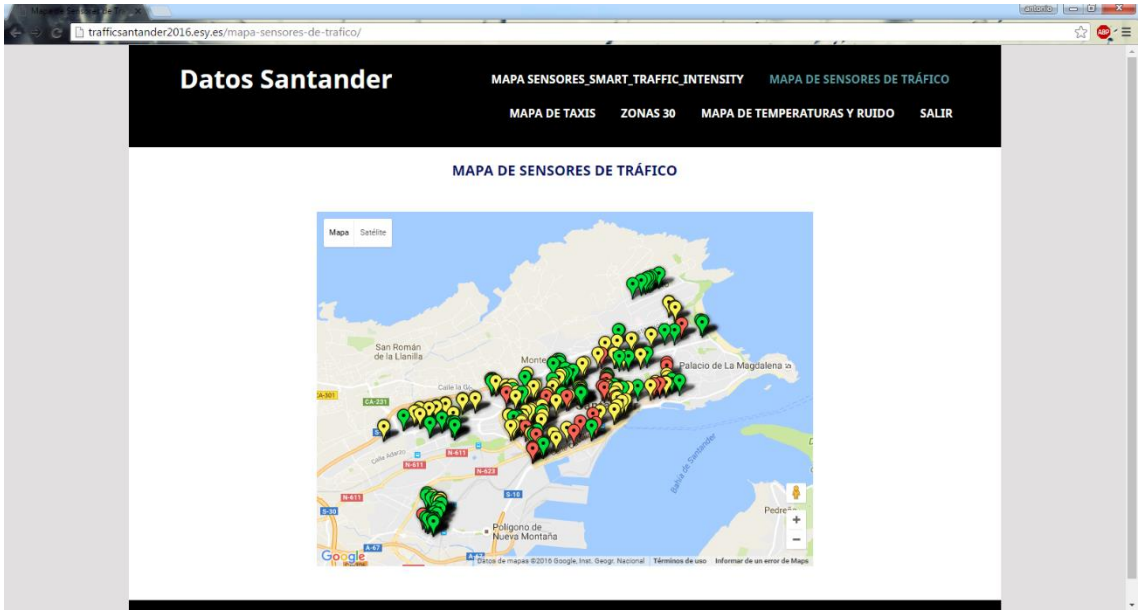

Figura 8.15 Mapa de Sensores de Tráfico

Además de mostrarnos la ubicación, estos mapas permiten visualizar las carreteras mediante Street view, hacer zoom y leer la información de los marcadores. Esta información es la que se encuentra en el las Fusion Tables de

Google. En la siguiente imagen podemos ver un ejemplo de lo comentado(Figura 8.16).

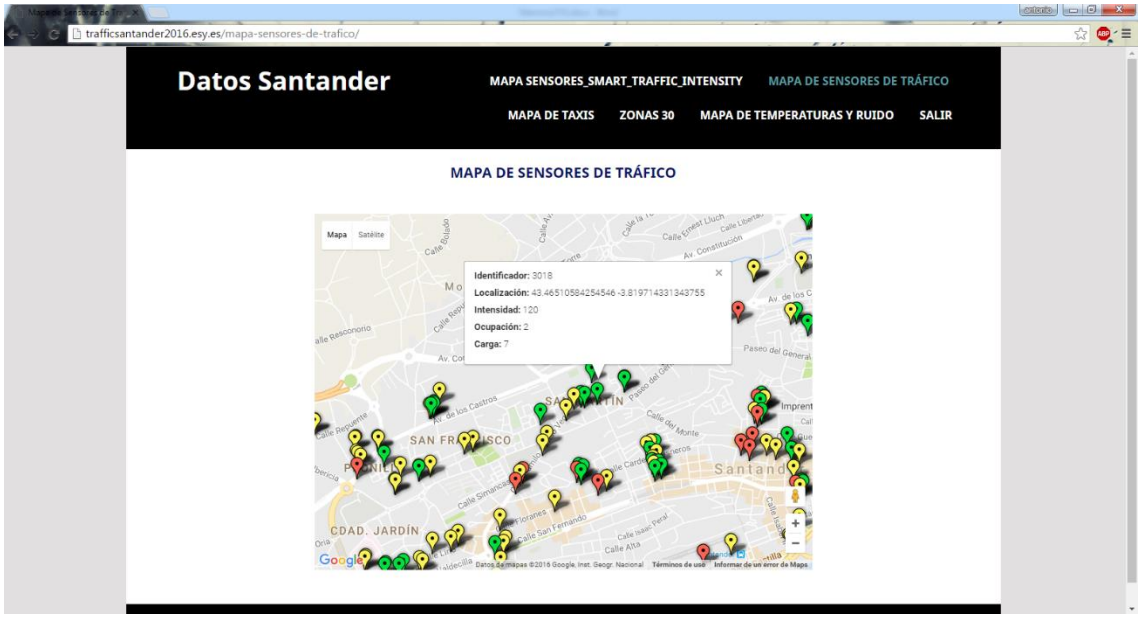

Figura 8.16 Zoom e información de los marcadores

Los mapas son dinámicos, de manera que cada cierto tiempo se realiza un refresco del mapa que actualiza los datos. Esto significa que cada vez que cambiamos los datos en las fusion tables, se verán reflejados los nuevos datos en la información que suministran los marcadores.

Las secciones Mapas de taxis y Mapas de Temperatura y Ruido, son secciones similares a la que acabamos de comentar. Con la diferencia de que los datos representados hacen referencia a otros recursos.

En el caso de Zonas 30, lo que visualizamos es un mapa estático con un ejemplo de una zona de Santander que solo se permite circular a 30 KM/H. En la siguiente figura (Figura 8.17) podemos observar el mapa estático.

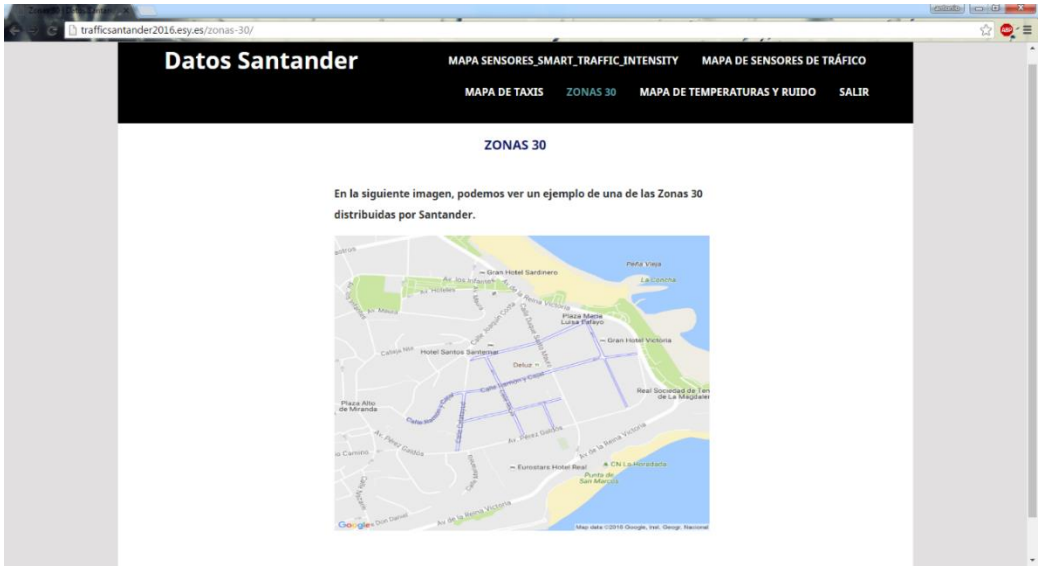

Figura 8.17 Mapa estático Zonas 30

Este mapa no permite hacer zoom, tan solo permite visualizar la ruta resaltada, que en este caso, hace referencia a una zona 30.

9 Conclusiones y Trabajos Futuros

Durante este capítulo se exponen las conclusiones obtenidas durante el desarrollo del proyecto, los conocimientos adquiridos y se añade alguna propuesta para trabajos futuros.

9.1 Conclusiones

Este proyecto ha sido una introducción al mundo de Big Data. Se han estudiado algunas herramientas que ayudan a procesar y gestionar datos masivamente. Los problemas del tratamiento de datos masivos cada vez se están convirtiendo en algo más común, debido al gran crecimiento de información que se está experimentando actualmente. Por este motivo, considero que el uso de herramientas de Big Data se está convirtiendo en algo esencial para el tratamiento de datos.

Big Data tiene numerosas aplicaciones, que son realmente ventajosas si tenemos que tratar con grandes volúmenes de datos. En este caso se ha hecho mayor hincapié en el concepto de Smart City o ciudades inteligentes. Este concepto se debe fomentar en todas las ciudades a nivel mundial, puesto que ofrece numerosas ventajas que favorecen a toda la población. Gracias a los avances que se están produciendo en esta dirección, cada día podemos tener más comodidades y facilidades.

Mediante la realización de este proyecto he podido experimentar la complejidad que supone gestionar los datos de tráfico de una ciudad. Es algo realmente complejo, puesto que es necesario tratar con una enorme cantidad de datos que están continuamente cambiando. Esta tarea sería inviable mediante los métodos tradicionales de tratamiento de datos, puesto que consumirían mucho tiempo y recursos.

La idea de predecir el tráfico puede ser de gran ayuda para todo el mundo, ya que nos ahorraría mucho tiempo en nuestras actividades diarias. Gracias a las herramientas de Big Data, tenemos más facilidades a la hora de realizar análisis predictivos del tráfico. Estas herramientas nos permiten cotejar y analizar de forma rápida un gran volumen de datos, tanto actuales como históricos. Lo que nos permite obtener análisis muy precisos, que son un requisito fundamental para este tipo de tareas.

La incorporación de la herramienta Apache Spark para el procesamiento de datos, ha sido una buena elección. Ya que se trata de una herramienta novedosa que es realmente potente en el procesamiento y almacenamiento de datos. A través de este proyecto he tenido la oportunidad de conocer más afondo esta herramienta, que me ha sorprendido en muchos aspectos. A parte de proporcionarlos una capacidad de procesamiento asombrosa, también nos proporciona descarga de datos en tiempo real, tolerancia a fallos y muchas facilidades a la hora de manipular y almacenar los datos. Además comentar que

es una herramienta que continuamente está incorporando nuevas funcionalidades y mejoras.

En cuanto a los sistemas de gestión de bases de datos, todos ellos son bastante buenos y permiten el almacenamiento de grandes volúmenes de datos. Pero para el propósito de este proyecto, el que mejor rendimiento ofrece, sin duda, es MongoDB. A parte del rendimiento que aporta, al tratarse de un sistema de almacenamiento orientado documentos, nos ofrece mucha flexibilidad a la hora de realizar el almacenamiento de datos. Estos documentos no poseen un esquema predefinido, lo que permite importar datos sin tener que preocuparnos por su esquema, ni tipo de datos. Adicionalmente Spark posee componentes muy buenos para establecer la conexión con MongoDB, así como realizar operaciones de almacenamiento y extracción de datos.

A nivel personal, la realización de este proyecto me ha servido para adquirir muchos conocimientos nuevos. He aprendido a utilizar e interactuar con herramientas muy potentes y pioneras en la actualidad. Por otro lado, también me ha servido para profundizar y afianzar conocimientos adquiridos durante la carrera.

Durante el desarrollo del proyecto han ocurrido algunos problemas que me han llevado mucho tiempo, debido a que no hay disponible mucha información sobre los errores que pueden ocasionar las herramientas que se han utilizado. Pero a pesar que la información en este aspecto era escasa, se han podido solventar todos los problemas ocasionados.

Finalmente concluir con que considero que la experiencia ha sido muy gratificante y me ha hecho crecer como profesional. Con el aliciente de que el tema tratado en el proyecto, es un tema innovador y muy solicitado en el panorama laboral actual.

9.2 Trabajos Futuros

El sistema desarrollado cumple con los objetivos propuestos, pero se podrían realizar mejoras o añadir nuevas funcionalidades, para convertirlo en una herramienta más completa.

Una de las mejoras que se podría aplicar, es aumentar las fuentes de datos e incluso incluir nuevas fuentes de otras ciudades. De manera que tengamos una herramienta que nos permita analizar el tráfico de múltiples ciudades. Además se podrían añadir más datos referentes a otros temas que no sean el tráfico, para poder realizar análisis sobre otros aspectos de las ciudades.

Dado que Smart City es un campo muy amplio hay muchas funcionalidades que se podrían incluir, como por ejemplo un algoritmo que introduciendo dos puntos de la ciudad, nos indicara la ruta más rápida. O un algoritmo que nos indicara la ruta donde podemos encontrar más monumentos y paisajes, para fomentar el turismo de la ciudad.

Un problema bastante común en las ciudades es el de encontrar aparcamiento. Por lo que también podría ser interesante incluir un algoritmo que analizara las zonas de aparcamiento, para indicarnos donde hay mayor probabilidad de encontrar alguno.

10 Bibliografía

[1] Big Data Wikipedia. https://es.wikipedia.org/wiki/Big_data

[2] Apache Spark documentación oficial. http://spark.apache.org/docs/1.6.0/

[3] Java (lenguaje de programación) Wikipedia. https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)

[4] Netbeans Wikipedia. https://es.wikipedia.org/wiki/NetBeans

[5] MongoDB documentación oficial. https://docs.mongodb.com/?_ga=1.242520834.1614521116.1466082194

[6] Apache Cassandra documentación oficial. http://cassandra.apache.org/doc/latest/

[7] Apache Hive documentación oficial. https://cwiki.apache.org/confluence/display/Hive/Home

[8] Wordpress Wikipedia. https://es.wikipedia.org/wiki/WordPress

[9] Desarrollo en cascada Wikipedia. https://es.wikipedia.org/wiki/Desarrollo_en_cascada

[10] Metodología iterativa e incremental Wikipedia. https://es.wikipedia.org/wiki/Desarrollo_iterativo_y_creciente