Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA (SEGOVIA)

Grado en Ingeniería Informática de Servicios y Aplicaciones

Definición y evaluación de métricas de eficiencia de vuelo para trayectorias reconstruidas a partir de datos (masivos) de vigilancia aérea

---

Alumna: Laura Hernán Muñoz

Tutor: Fernando Díaz Gómez

Agradecimientos

Quiero dar las gracias a todos los profesores que durante estos años me han enseñado tanto, ya que este trabajo es un poco parte de todos ellos, y especialmente a mi tutor Fernando, por su inestimable ayuda.

También quiero dar las gracias especialmente a mi madre y a mi padre, ya que sin ellos nada de esto hubiera sido posible, y también a mis abuelos y a toda mi familia.

Resumen

Este proyecto se enmarca en el contexto del *Proyecto AIRPORTS: Construcción de un prototipo de plataforma "big data" para el análisis de la eficiencia operacional de vuelos basada en trayectorias*, que persigue explotar datos capturados por los sistemas de vigilancia aérea para reconstruir trayectorias de vuelos de las aeronaves y evaluar su eficiencia operacional. En concreto, el trabajo plantea definir nuevos métodos de análisis de datos (métricas de eficiencia) cuyo diseño está condicionado por los nuevos paradigmas de computación soportados por los sistemas de Big Data actuales y se establecen para un dominio complejo, donde los datos capturados/reconstruidos (trayectorias de vuelo) son datos de carácter espacio-temporal.

Abstract

This project is a part of the *AIRPORTS Project: Construction of a prototype "big data" platform for the analysis of operational efficiency of flights based on trajectories*, which aims to exploit data captured by aerial surveillance systems to reconstruct flight trajectories of aircraft and to evaluate their operational efficiency. Specifically, the project proposes to define new methods of data analysis (efficiency metrics) whose design is conditioned by the new paradigms of computation supported by the current Big Data systems and are established for a complex domain, where the captured / reconstructed data (flight paths) are space-time data.

Índice

Índice de ilustraciones

Índice de tablas

SECCIÓN I: MEMORIA DEL PROYECTO

1. Introducción

El mundo de los negocios está experimentando una revolución impulsada por el uso y análisis de datos para guiar la toma de decisiones. Durante los últimos años, los datos disponibles para ser analizados han aumentado enormemente, y por ello ha surgido la necesidad de tratar y almacenar estos grandes volúmenes de datos, que conocemos como Big Data.

Uno de los sectores en los que está teniendo un gran impacto el Big Data es el sector aéreo, ya que cada vez agrupa más datos procedentes de distintas fuentes. Los grandes métodos de análisis de datos han revolucionado la forma en que, tanto los investigadores gubernamentales como los comerciales, pueden analizar bases de datos de aviación masivas que anteriormente eran demasiado incómodas, inconsistentes o irregulares para generar resultados de alta calidad. El objetivo del análisis realizado sobre los datos es utilizar la información obtenida para mejorar el tráfico aéreo o los sistemas de navegación actuales.

Los conjuntos de datos sobre aviación son publicados por diversas fuentes y no tienen los controles de estandarización y uniformidad requeridos para la investigación y el análisis de los datos. Por ello, la investigación de grandes volúmenes de datos de aviación (aerolíneas, aeronaves, vuelos, radar, etc.) requiere un trabajo previo de limpieza y transformación en una estructura uniforme, antes de pasar a analizar los datos. Los conjuntos de datos generan inmensos retos técnicos y humanos en la recolección, limpieza y clasificación.

Una vez limpios y estructurados los datos, el siguiente paso es analizarlos. La aplicación de métodos analíticos de grandes datos, almacenamiento de datos y soluciones software de rápida respuesta dan solución a estos problemas.

1.1. Identificación del Trabajo Fin de Grado

Título: Definición y evaluación de métricas de eficiencia de vuelo para trayectorias reconstruidas a partir de datos (masivos) de vigilancia aérea.

Autora: Laura Hernán Muñoz.

Fecha de entrega: 17 de julio de 2017.

Tutor: Fernando Díaz Gómez.

Universidad: Universidad de Valladolid (Escuela de Ingeniería Informática de Segovia).

Departamento: Informática.

1.2. Contenido del CD-ROM

El CD-ROM que acompaña a este Trabajo Fin de Grado contiene los siguientes directorios:

- **Memoria:** en este directorio se encuentra la memoria del Trabajo Fin de Grado en formato pdf.
- **Scripts:** en este directorio se localizan los scripts desarrollados en este proyecto.
- **Fuentes de datos:** en este directorio está el fichero que contiene los datos de los sectores ATC de España y el fichero con los datos de los FIRs.

1.3. Organización del documento

Este documento se divide en tres secciones, y las secciones en apartados y subapartados:

- **Sección I: Memoria del proyecto.** En esta sección se incluye la información relacionada con la introducción y contextualización del proyecto.
- **Sección II: Documentación técnica.** Esta sección está dedicada a la parte de documentación técnica, incluyendo el análisis, diseño, implementación y pruebas.
- **Sección III: Manual de usuario.** En esta sección se incluye el manual de usuario del sistema.
- **Sección IV: Conclusiones del trabajo.** En esta sección se incluyen las conclusiones, el trabajo futuro y las referencias.

2. Dominio de aplicación

2.1. Gestión del tráfico aéreo

La navegación aérea necesita de un **sistema de gestión del tráfico** (ATM - *Air Traffic Management*), que organice las trayectorias de las múltiples aeronaves que surcan el espacio aéreo al mismo tiempo. El objetivo principal de la gestión del tráfico aéreo es evitar que dos aeronaves ocupen el mismo lugar al mismo tiempo.

La gestión del tráfico aéreo es un término de la aviación que engloba todos los sistemas que asisten a las aeronaves a despegar desde un aeródromo, a transitar por el espacio aéreo y a aterrizar en el aeródromo de destino. ATM incluye:

- **Control del Tráfico Aéreo (ATC – *Air Traffic Control*):** es un servicio proporcionado por los controladores terrestres que dirigen las aeronave en tierra

y por medio del espacio aéreo controlado, y pueden prestar servicios de asesoramiento a aeronaves en el espacio aéreo no controlado.

- **Personal de Electrónica de Seguridad del Tránsito Aéreo (ATSEP – *Air Traffic Safety Electronics Personnel*):** es el personal técnico que participa en la creación y el apoyo de los sistemas electrónicos de hardware y software utilizados para apoyar la navegación aérea y la gestión del tráfico aéreo.

- **Meteorología Aeronáutica.**

- **Sistemas de navegación aérea** (ayudas a la navegación).

- **Servicios del Tráfico Aéreo (ATS – *Air Traffic Services*):** es un servicio que regula y asiste a las aeronaves en tiempo real para garantizar la seguridad de sus operaciones.

- **Gestión del Flujo del Tránsito Aéreo (ATFM – *Air Traffic Flow Management*):** es la regulación del tráfico aéreo con el fin de evitar exceder la capacidad de control de tráfico aeroportuario o aéreo para manejar el tráfico y asegurar que la capacidad disponible se utiliza de manera eficiente.

Dentro de la gestión del tráfico se incluyen también los sistemas de vigilancia aérea, que se detallan en el siguiente apartado “2.2. Vigilancia aérea”.

2.2. Vigilancia aérea

Los sistemas de vigilancia aérea se encargan de seguir e identificar una aeronave a lo largo de toda su trayectoria para mantener así la seguridad del tráfico aéreo, es decir, evitar que unas aeronaves irrumpan en la ruta de otras.

Este proyecto persigue explotar datos capturados por los sistemas de vigilancia aérea para reconstruir trayectorias de vuelos de las aeronaves y evaluar así su eficiencia operacional, es decir, lograr costes más bajos y una calidad superior.

2.3. Técnicas de vigilancia aérea

Actualmente, la vigilancia aérea se realiza utilizando dos técnicas diferentes: la técnica de radar y la técnica ADS.

2.3.1. Técnica de radar

La técnica de radar se basa en principios ya descubiertos a finales del siglo XIX, que fueron desarrollados para aplicaciones aéreas durante la Segunda Guerra Mundial.

Los radares determinan la distancia a la que se encuentran las aeronaves. Además algunos radares hacen uso de transpondedores instalados en las aeronaves, interrogando a la aeronave y respondiendo la aeronave enviando una contestación en la que además

se indica su altura e identificación, pudiendo así determinar la posición de la aeronave en tres dimensiones.

El funcionamiento del radar es el siguiente:

1. El radar emite una onda electromagnética.
2. El avión refleja la onda.
3. Se mide el tiempo transcurrido desde que la onda salió de la estación radar hasta que retorna.

La técnica de radar presenta cierta limitaciones geográficas y operativas, como por ejemplo las zonas oceánicas, que constituyen una limitación geográfica para los radares.

2.3.2. Técnica ADS

La Vigilancia Dependiente Automática (ADS - *Automatic Dependant Surveillance*) es una técnica de vigilancia en la que una aeronave transmite, vía enlace de datos, una serie de parámetros extraídos de los sistemas de navegación y posicionamiento de a bordo. La técnica ADS proporciona la identificación de la aeronave, la posición de la aeronave e información adicional, como la intención del vuelo.

La ADS tiene dos características fundamentales: es automática, es decir, no necesita la intervención del piloto para que los datos de la aeronave sean enviados, y es dependiente, porque la información que envía depende de los sistemas de a bordo.

Este nuevo sistema es especialmente útil para complementar la vigilancia en zonas oceánicas o en las que prácticamente no hay cobertura de los radares, así como para mejorar la vigilancia en zonas actualmente cubiertas con radar.

Existen dos técnicas de ADS: ADS-B (*Broadcast* – "difusión") y ADS-C (*Contract* – "contrato").

- La ADS-B consiste en la radiodifusión de ciertos datos de a bordo, en la que cualquier estación o aeronave dentro de la cobertura puede recibir la señal. Su funcionamiento es el siguiente:

    1. La aeronave emite una señal ADS-B.
    2. Las estaciones terrestres reciben la señal.
    3. Otras aeronaves también reciben la señal.

- La ADS-C es una técnica en la que la transmisión de la información se realiza punto a punto, entre la aeronave y una estación en tierra. La transmisión de los datos va dirigida a una estación con la que se ha establecido un acuerdo previo. Su funcionamiento es el siguiente:

    1. Se establece un contrato entre la aeronave y la estación de tierra.
    2. La aeronave transmite, a intervalos regulares, los datos a la estación con la que mantiene el contrato.

Todos los mensajes e información generados a través de las distintas técnicas de vigilancia aérea suponen gran cantidad de información que necesita ser almacenada y procesada. Al tratarse de grandes volúmenes de datos, se hace necesaria la utilización de sistemas Big Data (más información sobre Big Data en el apartado “3. Big Data”.

Además, se pretende obtener información y conocimiento a partir de los datos almacenados, así que se realizarán técnicas de análisis sobre los datos. De este tipo de análisis en grandes volúmenes de datos se encarga el campo de la ciencia de datos (data science) (más información sobre la ciencia de datos en el apartado “3.1. Data Science”.

3. Big Data

Se denomina Big Data a la gestión y análisis de enormes volúmenes de datos que no pueden ser tratados de manera convencional, ya que superan los límites y capacidades de las herramientas de software habitualmente utilizadas para la captura, gestión y procesamiento de datos. Este concepto se encuentra en continuo cambio ya que los avances tecnológicos permiten tratar volúmenes de datos cada vez mayores.

Big Data representa un avance de la tecnología que ha permitido crear un nuevo enfoque de entendimiento y toma de decisiones, la cual es utilizada para obtener información y conocimiento a partir de grandes volúmenes de datos.

Las 'Vs' del Big Data

Para definir este nuevo término (o concepto) informático, los expertos emplean las llamadas “tres Vs”: Volumen, Velocidad y Variedad. Sin embargo, en base a la experiencia adquirida por las empresas pioneras en Big Data, se ha ampliado la definición original añadiendo nuevas características, como son la Veracidad y Valor del dato (“cinco Vs”):

- **Volumen:** indica que hablamos de tantos datos que no caben en un disco normal, ni siquiera en uno realmente grande. Hacen falta multitud de ordenadores conectados entre sí, formando lo que se conoce como clúster.

- **Velocidad:** se refiere a la rapidez con la que los datos se reciben, se procesan y se toman decisiones a partir de ellos. A la mayoría de los sistemas tradicionales les es imposible analizar de forma inmediata los grandes volúmenes de datos que les llegan. Sin embargo, incorporar el concepto de tiempo real es imprescindible, por ejemplo, para sistemas de detección del fraude o la realización de ofertas personalizadas a los clientes.

- **Variedad:** hace referencia a la inclusión de nuevas fuentes con tipos de datos diferentes a los que se utilizan de forma tradicional. El nivel de estructura de los datos hace referencia a la forma en que se organizan los datos para facilitar su procesamiento utilizando un computador. Cuanto mayor sea el nivel de

estructura de los datos, su procesamiento será más sencillo; así, los datos menos estructurados requieren un procesamiento más complejo. Los tipos de datos según su nivel de estructura son:

- **Datos estructurados:** son todos aquellos datos que tienen bien definidos su longitud y su formato, tienen una estructura bien definida y se les aplican unas normas estrictas. Un ejemplo son las bases de datos relacionales, en las que la información se almacena en tablas.

- **Datos semiestructurados:** estos datos se almacenan conforme a un conjunto de reglas menos estrictas, es decir, que no se limitan a unos campos determinados, pero contienen una estructura o marcadores que permiten diferenciar y separar los distintos elementos. Algunos ejemplos de formatos semiestructurados más usados son XML, JSON y CSV.

- **Datos no estructurados:** estos datos no tienen ninguna estructura definida de forma explícita. Es posible que tengan algún tipo de estructura implícita, pero para un computador puede llegar a ser muy difícil de interpretar. Ejemplos de estos tipos de datos son los textos en lenguaje natural, vídeos, audios, imágenes, etc.

El procesamiento de información estructurada es el más sencillo y, desde hace tiempo, se consigue realizar de forma eficiente para grandes volúmenes de datos. Sin embargo, en Big Data, la mayoría de las fuentes externas de las que disponemos son de las consideradas semiestructuradas o no estructuradas.

- **Veracidad:** este concepto hace referencia a la confianza en los datos, a extraer datos de calidad eliminando la imprevisibilidad inherente de algunos para, de esta forma, llegar a una correcta toma de decisiones.

- **Valor:** los datos son importantes para las empresas y negocios, por ello es fundamental saber qué datos son los que se deben analizar. De hecho, ya se empieza a hablar del científico de datos (Data Scientist), un profesional con perfil científico, tecnológico y visión de negocio.

El Big Data requiere infraestructuras, tecnologías y servicios específicos que han sido creados para dar solución al procesamiento de grandes conjuntos de datos. El proyecto AIRPORTS, explicado en el apartado “4. Contexto: Proyecto AIRPORTS” propone una solución Big Data e implementa una plataforma con las tecnologías necesarias, denominada ADAPT.

3.1. Data Science

El objetivo principal de Big Data, al igual que los sistemas analíticos convencionales, es convertir los datos en información y conocimiento que facilite la toma de decisiones.

La ciencia de datos (*data science*) es un campo interdisciplinario que involucra métodos científicos, procesos y sistemas para extraer conocimiento de grandes volúmenes de datos. La ciencia de datos es una continuación de algunos campos de análisis de datos como la estadística, la minería de datos, el aprendizaje automático y la analítica predictiva.

La ciencia de datos trabaja con datos incompletos y desordenados, analizando grandes volúmenes de datos para extraer conocimiento. Los hallazgos en los datos aportan valor a una determinada organización.

En relación con la ciencia de datos, está la figura del científico de datos (*data scientist*), una nueva profesión que hoy es considerada clave en el mundo de las tecnologías.

4. Contexto: Proyecto AIRPORTS

Este Trabajo Fin de Grado es un trabajo de investigación que se enmarca en el contexto del *Proyecto AIRPORTS: Construcción de un prototipo de plataforma "big data" para el análisis de la eficiencia operacional de vuelos basada en trayectorias*. El trabajo realizado en este Trabajo Fin de Grado es una parte del Proyecto AIRPORTS. En los siguientes apartados se explican las principales características del Proyecto AIRPORTS.

4.1. Descripción general del proyecto AIRPORTS

El proyecto AIRPORTS es un proyecto de investigación en el que participa la Universidad de Valladolid, en colaboración con BR&TE, el departamento de tecnología e investigación de la empresa Boeing España. Este proyecto se inició el 1 de septiembre de 2015 y finalizará el 31 de diciembre de 2018.

El proyecto AIRPORTS tiene como objetivo principal mejorar la eficiencia de los futuros sistemas de transporte aéreo, desarrollando soluciones tecnológicas que contribuyan a modernizar y mejorar el sistema de transporte aéreo desde una perspectiva multidisciplinar, considerando tanto las operaciones aéreas como terrestres. Para conseguir su objetivo se basa en soluciones tecnológicas innovadoras basadas en Big Data y en la ciencia de datos (*data science*).

4.2. Fuentes de datos del proyecto AIRPORTS

En el proyecto AIRPORTS se trabaja con distintos conjuntos de datos (*datasets*) que proceden de distintas fuentes de datos, como son los datos obtenidos utilizando la técnica de radar, los datos obtenidos utilizando la técnica ADS-B, y otros datos que son combinaciones de datos obtenidos a partir de las dos técnicas anteriores:

- Conjuntos de datos procedentes de los radares:
  - Trazas de radar

- Conjuntos de datos obtenidos utilizando la técnica ADS-B:

- Frambuesa
- ADSBHub
- OpenSky

- Combinaciones de las fuentes anteriores:
  - FlightAware
  - Fr24

4.3. Arquitectura Lambda

Los problemas del Big Data son complejos de analizar y solucionar. El gran volumen, velocidad y variedad de los datos hacen que sea difícil extraer información y conocimiento. El problema ante el que nos solemos encontrar al tratar con grandes volúmenes de datos es que no existe una técnica predefinida para hacerlo.

Un buen primer paso es clasificar el problema de Big Data según el formato de los datos que se van a procesar, el tipo de análisis que se les va a aplicar, las técnicas de procesamiento a emplear y los orígenes de los datos. Es posible almacenar, adquirir, procesar y analizar Big Data de muchas formas. Por ello, es importante elegir una arquitectura apropiada que se ajuste a las necesidades de Big Data específicas.

La arquitectura Big Data propuesta para el proyecto AIRPORTS se basa en la arquitectura Lambda. La arquitectura Lambda propone un enfoque con tres capas: la capa por lotes (*batch layer*), la capa de servicio (*serving layer*) y la capa de velocidad (*speed layer*).

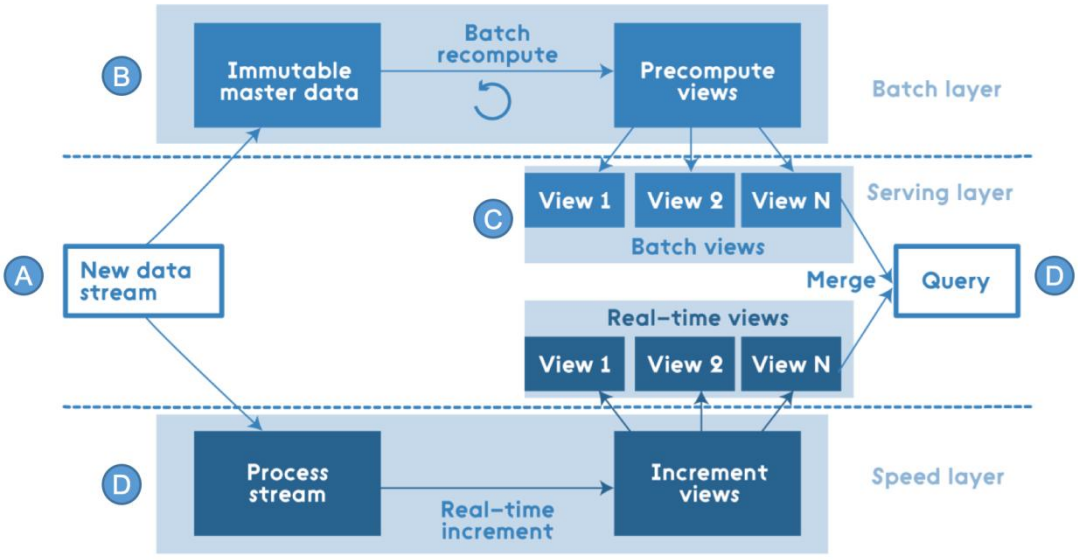


Ilustración 1. Arquitectura Lambda

- **Capa por lotes (*batch layer*):** esta capa tiene dos funciones:

    - Gestionar el conjunto de datos maestro (*master dataset*), que es un conjunto de datos inmutable y en constante crecimiento en el que se almacenan todos los datos.

    - Precalcular las consultas, es decir, se calculan las consultas por lotes y se obtienen unos resultados de la consulta. Estos resultados se denominan vistas por lotes (*batch views*).

- **Capa de servicio (*service layer*):** en esta capa se almacenan las vistas por lotes (*batch views*) calculadas en la capa por lotes (*batch layer*) y se indexan para que puedan ser consultadas rápida y eficientemente con posterioridad.

- **Capa de velocidad (*speed layer*):** esta capa se encarga de procesar los nuevos datos en tiempo real. Para ello, también genera vistas, denominadas vistas en tiempo real (*realtime views*), pero estas vistas no se generan por lotes como en la capa por lotes, sino que se van actualizando según se van recibiendo nuevos datos. El objetivo principal de esta capa es compensar la latencia que genera en la capa por lotes. Por ello, las vistas en tiempo real sólo se crean a partir de los datos recientes, de los datos los que aún no se han actualizado en las vistas por lotes. Una vez que el procesamiento por lotes haya terminado de precalcular las consultas y se hayan actualizado las vistas por lotes, los resultados correspondientes en las vistas en tiempo real ya no serán necesarios, por lo que se descartan.

Los pasos que se siguen en la arquitectura Lambda son los siguientes:

A) Todos los nuevos datos se envían tanto a la capa por lotes como a la capa de velocidad.

B) En la capa por lotes, los nuevos datos se añaden al conjunto de datos maestro, y se precalculan las consultas, generando las vistas por lotes.

C) Las vistas por lotes se almacenan en la capa de servicio y se indexan para que puedan ser consultadas con posterioridad.

D) En la capa de velocidad, los nuevos datos van actualizando las vistas en tiempo real.

E) Cuando se realiza una consulta, esta se resuelve obteniendo los resultados tanto de las vistas por lotes como de las vistas en tiempo real, y se unen o combinan sus resultados.

4.4. Arquitectura del proyecto AIRPORTS

La siguiente figura muestra la arquitectura general propuesta en el proyecto AIRPORTS. Esta arquitectura incluye los bloques de construcción básicos de la arquitectura Lambda, aplicándolos a los problemas particulares del proyecto AIRPORTS.

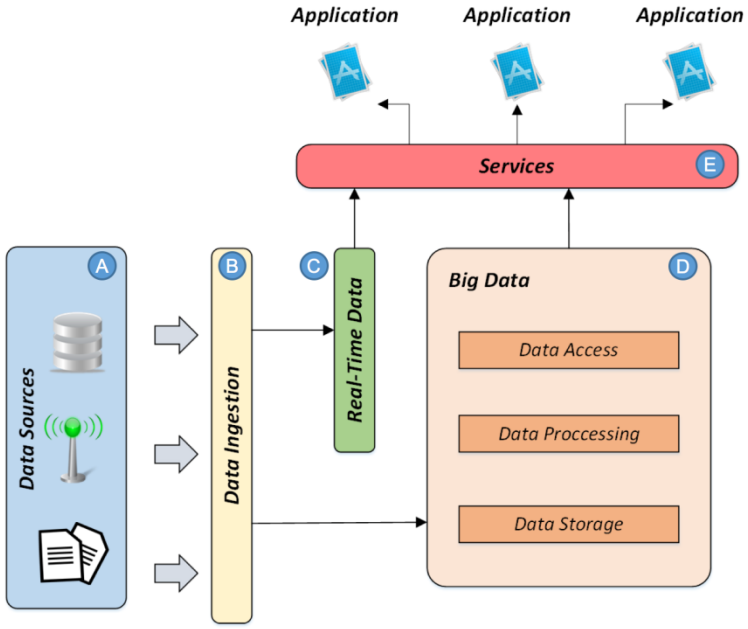


Ilustración 2. Arquitectura propuesta en el proyecto AIRPORTS

A) En primer lugar, se tienen las fuentes de datos (*Data Sources*) que representan todo tipo de fuentes de información disponibles, como pueden ser las fuentes de datos ADS-B, las fuentes de datos de radares, la información meteorológica, los planes de vuelo, etc.

B) Estos datos pasan al bloque de ingestión de datos (*Data Ingestion Block*), en el que los datos se almacenan en una cola y se realizan tareas de limpieza si fuera necesario. Después, los datos se envían tanto al bloque de datos en tiempo real (*Real-Time Data Block*), que se corresponde con la capa de velocidad (*speed layer*) de la arquitectura Lambda, como al bloque Big Data, que se corresponde con la capa por lotes (*batch layer*) de la arquitectura Lambda.

C) El bloque de datos en tiempo real se encarga de actualizar continuamente las vistas en tiempo real (*realtime views*) con los datos más recientes. Aunque aún no está claro si el proyecto AIRPORTS va a utilizar las capacidades de la capa de datos en tiempo real, se considera que, desde el punto de vista arquitectónico, es importante incluir una descripción completa en la que esta capa pueda ser añadida fácilmente sin modificar las otras capas.

D) El bloque de Big Data se divide a su vez en tres capas:

- La capa de almacenamiento de datos (*Data Storage Layer*), en la que se almacenan todos los datos. Este conjunto de datos se denomina conjunto de datos maestro (*master dataset*), que es inmutable y sólo permite añadir datos.

- La capa de procesamiento de datos (*Data Processsing Layer*), que itera continuamente recalculando las vistas por lotes (*batch views*).

- La capa de acceso a datos (*Data Access Layer*), que permite el acceso, recuperación y análisis rápido de la información invocando a las vistas por lotes.

E) El bloque de servicios (*Service Block*) utiliza los resultados los bloques de tiempo real y de Big Data para combinarlos. Este bloque se corresponde con la capa de servicios (*service layer*) de la arquitectura Lambda. La salida del boque de servicios puede ser utilizada por diferentes aplicaciones. Este bloque es básicamente una base de datos distribuida para almacenar las vistas procesadas y permitir que se pueda acceder a las mismas para consulta y visualización.

4.5. Plataforma Big Data ADAPT

Una vez definida una arquitectura Big Data, se necesita una plataforma capaz de implementar las herramientas necesarias que se ajusten a la arquitectura conceptual descrita en del apartado anterior.

Como se ha explicado con anterioridad, el proyecto AIRPORTS se desarrolla en colaboración con Boeing España, que utiliza para todos sus proyectos de investigación sobre analítica de gestión del tráfico aéreo el framework ADAPT, una distribución Hadoop de Cloudera (CDH – *Cloudera Hadoop Distribution*) (más detalles sobre Cloudera y Hadoop en los apartados "4.5.1. Cloudera" y "4.5.2. Hadoop", respectivamente). Por ello, el proyecto AIRPORTS se implementa sobre la plataforma ADAPT.

ADAPT (*Aviation Data Analytics Platform Testbed*) es como su nombre indica, una plataforma de prueba para análisis de datos de aviación. ADAPT es el entorno de desarrollo de una plataforma de análisis de datos (*Data Analytics*) diseñada para el crecimiento y soporte de muchas fuentes de datos relacionadas con ATM (*Air Traffic Management*). ADAPT permite a los investigadores acceder de una forma potente y oportuna a conjuntos de datos grandes y complejos.

4.5.1. Cloudera

Ilustración 3. Logotipo de Cloudera

Cloudera es una plataforma de código abierto (*open source*) basada en Apache Hadoop y se conoce con las siglas CDH (*Cloudera Distribution including Hadoop*). CDH incluye toda la funcionalidad de Apache Hadoop en cuanto a almacenamiento distribuido y procesamiento distribuido. Además, incluye componentes adicionales, como una interfaz de usuario y funciones de seguridad necesarias para las empresas. La siguiente figura muestra la arquitectura de Cloudera.

4.5.2. Hadoop

Ilustración 4. Logotipo de Hadoop

Apache Hadoop es un framework de código abierto (*open source*) desarrollado para una computación distribuida, confiable y escalable de grandes volúmenes de datos (Big Data).

Las principales funcionalidades de Hadoop son el procesamiento distribuido y el almacenamiento distribuido:

- El **procesamiento distribuido** permite procesar grandes conjuntos de datos a través de clústers de computadoras, permitiendo a las aplicaciones trabajar con miles de nodos y petabytes de datos, utilizando para ello modelos de programación simples. Apache Hadoop implementa un paradigma computacional denominado MapReduce, donde la aplicación se divide en muchos pequeños fragmentos de trabajo, y cada uno de los cuales se puede ejecutar en cualquier nodo del clúster.

- El **almacenamiento distribuido** consiste en un sistema de archivos distribuido (HDFS - Hadoop Distributed File System) en el que los ficheros de datos no se almacenan en una única máquina, sino que su información se distribuye en distintos nodos del clúster.

Apache Hadoop está diseñado para escalar desde servidores individuales a miles de máquinas, proporcionando cada una de ellas procesamiento y almacenamiento.

Apache Hadoop incluye los siguientes módulos básicos:

- **Hadoop Common:** es el núcleo principal, y contiene utilidades comunes para soportar los otros módulos de Hadoop.

- **Hadoop Distributed File System (HDFS):** sistema de ficheros distribuido que proporciona acceso a los datos de aplicación con un alto rendimiento.

- **Hadoop YARN:** un framework para la programación de tareas y gestión de recursos del clúster.

- **Hadoop MapReduce:** un sistema basado en YARN para el procesamiento paralelo de grandes conjuntos de datos.

Estos son los módulos básicos de Apache Hadoop, pero con el tiempo, se han desarrollado muchos otros módulos y componentes también de código abierto que utilizan o aprovechan la tecnología de Apache Hadoop. Al conjunto de Apache Hadoop y los paquetes de software adicionales se les conoce como Ecosistema Hadoop. Algunos de los módulos y componentes que se añaden a Apache Hadoop son HBase, Hive, Spark, Flume, Sqoop, etc.

4.6. Arquitectura detallada del proyecto AIRPORTS

Una vez propuesta una arquitectura Big Data (véase apartado “4.4. Arquitectura del proyecto AIRPORTS” y definida la plataforma en la que se va a implementar (véase apartado “4.5. Plataforma Big Data ADAPT”), a continuación se detallan las herramientas que se utilizarán en cada capa de la arquitectura.

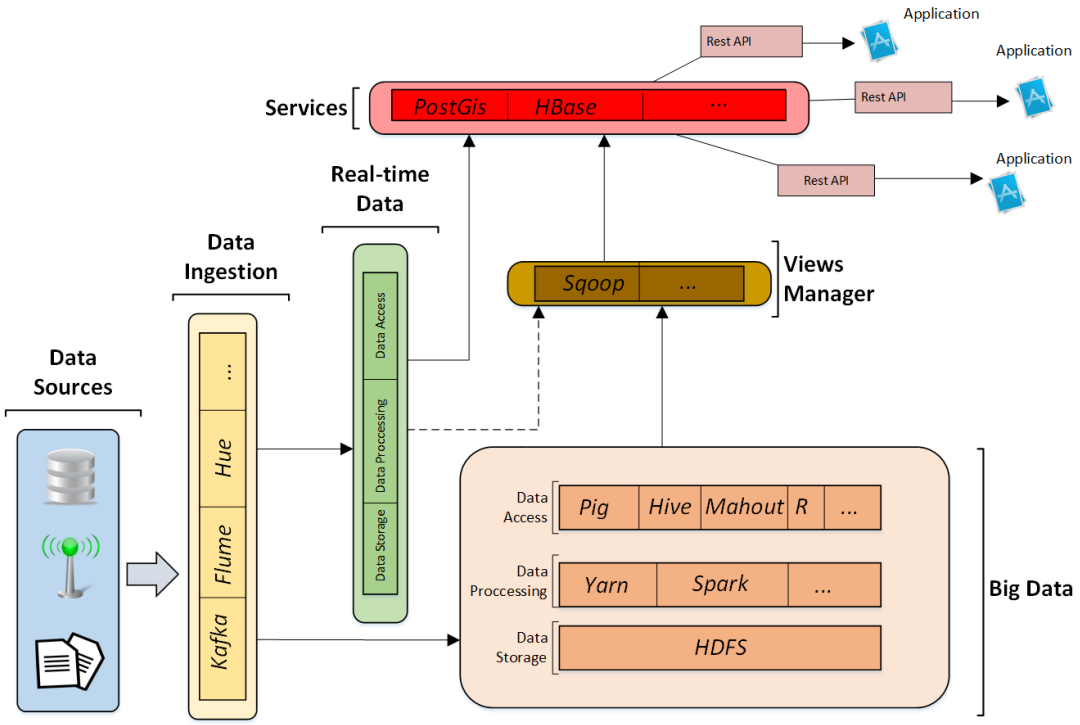


Ilustración 5. Arquitectura detallada del proyecto AIRPORTS

- Para el bloque de ingestión de datos (*Data Ingestion Block*) se utilizan las tecnologías Kafka, Flume y Hue:

    - La cola de Kafka recibe y almacena temporalmente la información recibida de las fuentes de datos.

    - El agente de Flume lee de la cola de Kafka y realiza procesos para recoger, agregar y mover eficientemente grandes cantidades de datos.

    - Hue se puede utilizar para insertar manualmente datos estáticos en HDFS.

- En el bloque de datos en tiempo real (*Real-Time Data Block*) aún no se ha propuesto el uso de ninguna tecnología en concreto, debido a que los datos en tiempo real no son el objetivo principal del proyecto AIRPORTS.

- En el bloque de Big Data, se utilizan las siguientes tecnologías:
    - En la capa de almacenamiento de datos (*Data Storage Layer*), se emplea el sistema de ficheros distribuido HDFS para almacenar todos los datos.
    - En la capa de procesamiento de datos (*Data Processing Layer*), se usa YARN y Spark.
        - YARN es uno de los módulos básicos de Apache Hadoop. Es un framework para la programación de tareas y gestión de recursos.
        - Spark, un motor rápido y general para procesamiento de datos a gran escala.
    - En la capa de acceso de datos (*Data Access Layer*) se utiliza:

        - Pig, una plataforma de alto nivel para simplificar el desarrollo de los algoritmos de MapReduce usando el lenguaje de scripting Pig Latin.
        - Hive, una infraestructura para proporcionar agrupación, consulta y análisis de datos utilizando un lenguaje del tipo SQL llamado HiveQL.
        - Mahout, para implementar algoritmos de aprendizaje de máquina distribuidos y escalables.
        - R, un lenguaje de programación para análisis estadístico.

- En el bloque de servicios (*Services Block*) las tecnologías aún no están muy definidas, ya que dependen del uso final de la plataforma. Sin embargo, se considera que serán necesarios PostgreSQL con su módulo PostGIS, y HBase:
    - PostgreSQL es un sistema de gestión de bases de datos relacional, orientado a objetos y libre, y el módulo PostGIS añade soporte de objetos geográficos.
    - HBase es una base de datos distribuida y escalable que admite el almacenamiento estructurado de datos para grandes tablas.

- También se ha propuesto un bloque adicional, denominado gestor de vistas (*Views Manager*), para transferir fácilmente la información entre el bloque de Big Data y el bloque de servicios. En este bloque se ha propuesto incluir Sqoop, que se utiliza para transferir datos entre Hadoop y bases de datos relacionales, como PostgreSQL, etc.

5. Proyecto Métricas

El título de este Trabajo Fin de Grado es *Definición y evaluación de métricas de eficiencia de vuelo para trayectorias reconstruidas a partir de datos (masivos) de vigilancia aérea*, pero en lo que sigue en este documento se hará referencia a este título denominándolo únicamente "Proyecto Métricas". El Proyecto Métricas se encuadra dentro del Proyecto AIRPORTS, explicado en el apartado "4. Contexto: Proyecto AIRPORTS".

5.1. Motivación

La capacidad de evaluar con precisión el impacto de los nuevos conceptos operativos sobre el consumo de combustible y otros aspectos de la eficiencia del vuelo, así como su impacto medioambiental es crucial para los interesados en el aprovechamiento del espacio aéreo, ya que aportaría información de gran valor para analizar el impacto económico de esos nuevos conceptos sobre los usuarios y sobre el sistema de Gestión de Tráfico Aéreo (ATM – *Air Traffic Mangement*), en general.

En la actualidad, se está trabajando en el desarrollo de herramientas avanzadas para el análisis de la eficiencia y el impacto medioambiental de los nuevos conceptos operativos para mejorar el actual sistema ATM. Por otra parte, la capacidad de estas herramientas para evaluar el consumo de combustible y el correspondiente impacto medioambiental de las operaciones actuales a partir de datos de vigilancia es limitada. Estas herramientas necesitan definir un escenario de referencia o *baseline* que

represente las operaciones actuales en un contexto dado, es decir, un escenario cuyo rendimiento medido en términos de consumo de combustible (y de las otras métricas a evaluar) constituya una referencia con respecto a la cual medir el impacto de las nuevas operaciones.

Desde el punto de vista de los usuarios del espacio aéreo, además del impacto sobre el consumo de combustible y el medio ambiente de los nuevos conceptos operativos, sería de gran interés conocer con exactitud el impacto que dichas operaciones tendrían sobre el consumo real de combustible asociado a sus operaciones actuales en los escenario de interés. Para ello, es necesario establecer una referencia que incluya el consumo de combustible real de cada vuelo en el escenario así como el peso inicial real de cada aeronave en el momento inicial de la operación, para poder estimar así qué cantidad de combustible consumiría esa aeronave si en lugar de volar la trayectoria real volara, a partir de las mismas condiciones iniciales y en las mismas condiciones atmosféricas, una trayectoria alternativa resultante de aplicar un nuevo concepto operativo.

5.2. Alcance y objetivos

El proyecto Métricas persigue explotar datos capturados por los sistemas de vigilancia aérea para reconstruir trayectorias de vuelos de las aeronaves y evaluar su eficiencia operacional mediante métricas de eficiencia de vuelo.

El proyecto Métricas es un proyecto de investigación novedoso en el dominio de aplicación de la gestión aérea, que plantea definir nuevos métodos de análisis de datos (métricas de eficiencia), cuyo diseño está condicionado por los nuevos paradigmas de computación soportados por los sistemas de Big Data actuales, dentro de un dominio complejo, en el que los datos capturados (trayectorias de vuelo) son datos de carácter espacio-temporal. Este planteamiento se traduce en dos objetivos principales de investigación:

- Planteada una arquitectura Big Data inicial a utilizar en el proyecto AIRPORTS y modelados los datos operacionales históricos disponibles a partir de las diferentes fuentes (ADSBHub, FlightAware, Fr24, Frambuesa, etc.), identificar qué datos adicionales se necesitan para implementar las métricas que se definan.

- Definir una lista de métricas de eficiencia de vuelo basadas en trayectorias, comenzando por métricas sencillas, para pasar después a métricas más complejas.

5.3. ¿Qué es una métrica?

Una métrica o KPI (*Key Performance Indicator*) es la medida o conjunto de medidas establecidas en un contexto para la comprensión de las tendencias en los datos en el tiempo. Es una medida cuantitativa sobre un sistema, componente o proceso. La métrica representa una extrapolación o un cálculo matemático de las mediciones resultantes de un valor determinado.

Concretamente, en el contexto de este proyecto, una métrica representa una fórmula que agrega **trayectorias** en un **volumen** y un tiempo definidos para obtener una medida representativa de interés, según ciertos factores. Estos factores dependerán del tipo de métrica del que se trate. Por ello, en los siguientes apartados se explica qué es una trayectoria y distintos volúmenes que se pueden considerar dentro del espacio aéreo.

5.4. Trayectoria

Una trayectoria es el recorrido que describe un objeto al desplazarse por el espacio. En cinemática, la trayectoria es el lugar geométrico de las posiciones sucesivas por las que pasa un cuerpo en su movimiento.

En este proyecto y como su título indica: *Definición y evaluación de métricas de eficiencia de vuelo para* ***trayectorias reconstruidas a partir de datos (masivos)*** *de vigilancia aérea*, las métricas se basan en trayectorias reconstruidas. Los datos masivos de vigilancia aérea contienen información sobre los puntos por los que han pasado las distintas aeronaves. Las técnicas de vigilancia aérea obtienen información acerca de la posición de las aeronaves en distintos instantes de tiempo, es decir, de forma discreta. Para realizar el cálculo de las métricas, estos puntos discretos se deben agrupar por trayectorias.

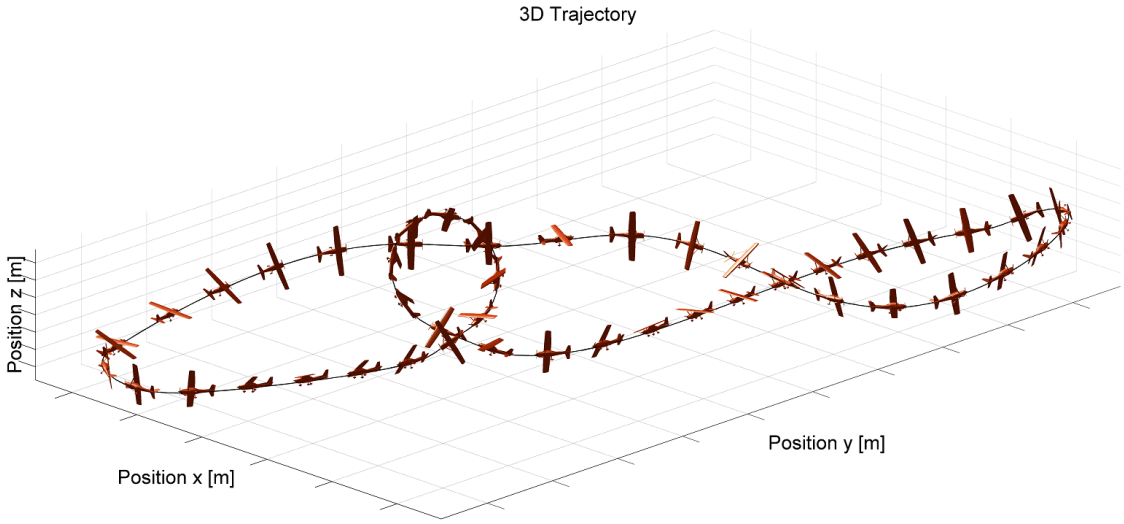


Ilustración 6. Dibujo de la trayectoria de un avión

5.5. Espacio aéreo

El espacio aéreo tiene divisiones y subdivisiones.

5.5.1. FIR

FIR (*Flight Information Region*) son regiones de información de vuelo. En España hay 3 FIRs: FIR Madrid, FIR Barcelona y FIR Canarias. En la siguiente imagen se muestran los FIRs de España.

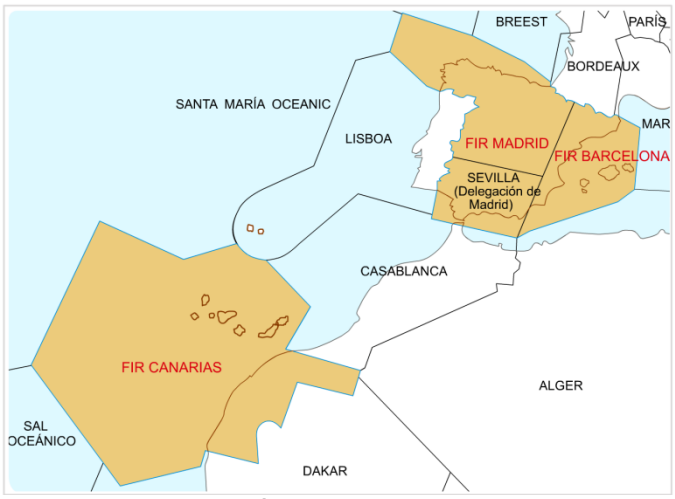


Ilustración 7. FIRs de España

5.5.2. Regiones y sectores

En España existen cinco **direcciones regionales**, que son Centro-Norte, Este, Canaria, Sur y Balear. Las regiones están divididas, a su vez, en **sectores**, como se puede observar en las siguientes imágenes.

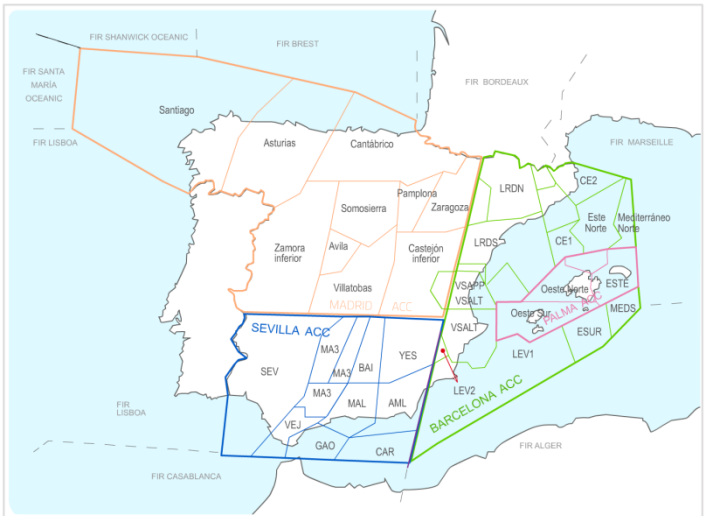


Ilustración 8. Regiones y sectores de la península ibérica

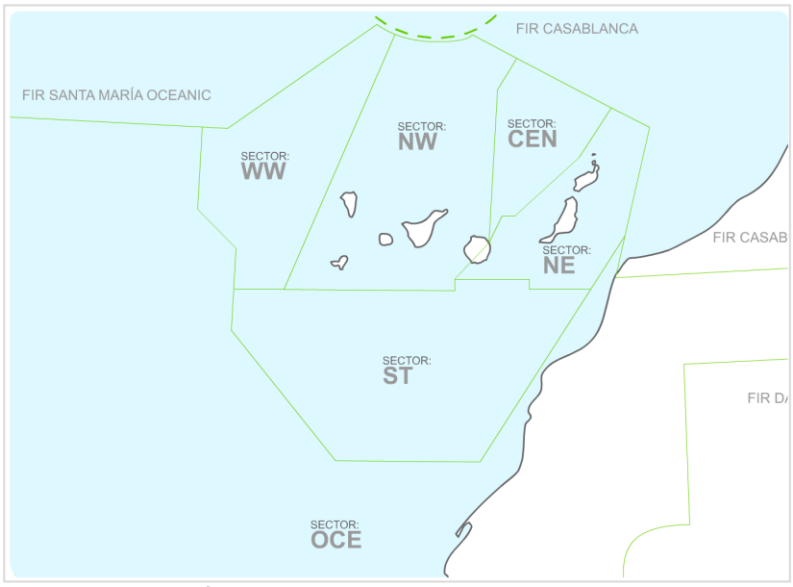


Ilustración 9. Regiones y sectores Islas Canarias

5.5.3. Áreas en torno a los aeropuertos

El área de secuenciación y medición de llegada (ASMA – *Arrival Sequencing and Metering Area*) se define como un cilindro virtual de un radio determinado alrededor de un aeropuerto. Este radio es de 40 nm (millas náuticas), aunque en algunos casos pueden llegar a considerarse hasta 100 nm.

Esta área también es interesante a la hora de calcular las métricas.

5.6. Métricas propuestas

Tras realizar el estudio y análisis de las métricas, las métricas consideradas y propuestas para implementar son las detalladas a continuación.

5.6.1. Densidad de tráfico (Traffic Density)

Esta métrica se define como el número promedio de aeronaves presentes en un volumen específico del espacio aéreo durante un intervalo de tiempo definido. Dado un sector S definido como un volumen tridimensional en el espacio, la métrica de densidad de tráfico se define así:

$$KPI_{td}^{S} = \frac{N_T^S}{T}$$

donde N_T^S es el número de aeronaves que están volando a través de un sector S durante un intervalo de tiempo de referencia T.

5.6.2. Pico de tráfico (Peak Load)

Esta métrica se define como el número máximo de aeronaves que están volando simultáneamente dentro de un volumen específico del espacio aéreo durante un intervalo de tiempo de referencia. Dado un sector S, la métrica pico de tráfico se define así:

$$KPI_{pl}^{S} = \max_{t \in T}\left(N_t^S(t)\right)$$

donde $N_t^S(t)$ es el número de aeronaves presentes en el sector S en el instante de tiempo t, y T es el intervalo de tiempo de referencia sobre el que se mide el pico de tráfico.

5.6.3. Número de conflictos (Number of Conflicts)

Esta métrica representa el número de violaciones de los mínimos de separación especificados dentro de un volumen del espacio aéreo durante un intervalo de tiempo definido. Para evaluar esta métrica, se consideran sólo los vuelos durante la fase "en ruta", y los mínimos de separación considerados son:

- Separación lateral: 5 nm (millas náuticas - nautical miles).
- Separación vertical: 1000 pies.

La siguiente figura muestra un ejemplo de ocurrencia de conflictos. Los segmentos de una trayectoria donde la separación con otras aeronaves es inferior a 5 nm están marcados en rojo.

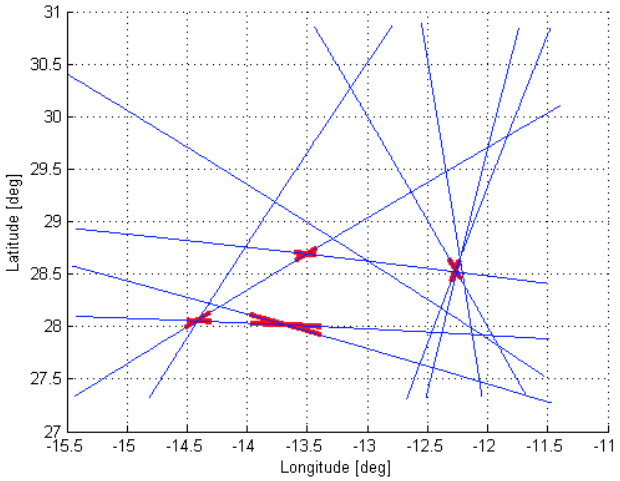


Ilustración 10. Ejemplo de conflictos

5.7. ¿Por qué precalcular las métricas?

A medida que se ha ido avanzando en este proyecto de investigación y una vez que se han implementado las métricas, se ha observado que el cálculo de una determinada métrica para unos valores concretos requiere un gran consumo de tiempo y de recursos debido a la gran cantidad de datos que hay que mover de un almacén a otro para poder calcular la métrica. Por ello, se ha propuesto una solución, que consiste en precalcular los resultados de las métricas, es decir, calcular las métricas para ciertos valores e intervalos predefinidos y almacenar los resultados. Estos cálculos son procesamientos por lotes (*batch processing*), que encajan perfectamente dentro de la capa por lotes (*batch layer*) de la arquitectura Lambda (descrita en el apartado "4.3. Arquitectura Lambda"). Los resultados del precálculo de las métricas se corresponden con las vistas por lotes (*batch views*).

Así, cuando se quiera obtener un resultado concreto de una métrica, sólo se tendrán que consultar los resultados precalculados (vistas por lotes) y ordenar o agrupar estos resultados. Además, estas vistas por lotes son útiles para ser analizadas posteriormente utilizando gráficos o técnicas estadísticas, permitiendo así obtener información y conocimiento a partir de la gran cantidad de datos.

6. Entorno de aplicación del proyecto Métricas

En este apartado se explican con más detalle las herramientas que se han utilizado en el proyecto Métricas.

6.1. HDFS

HDFS (Hadoop Distributed File System) es el sistema de ficheros distribuido, escalable y portátil de Hadoop. Está escrito en Java, es altamente tolerante a fallos y está diseñado para ser desplegado en hardware de bajo coste. HDFS proporciona un alto rendimiento de acceso a datos y es adecuado para manejar grandes conjuntos de datos.

Una de sus principales características es un tamaño de bloque muy superior al habitual (64 MB), para no perder tiempo en los accesos de lectura.

Los ficheros se dividen en bloques y se distribuyen entre los nodos que forman el clúster, por lo que bloques de un mismo fichero pueden estar almacenados en nodos distintos. Además, los bloques se replican 3 veces en el clúster.

El clúster HDFS tiene dos tipos de nodos, que siguen una arquitectura maestro-esclavo (*master-slave*):

- **Namenode**: Este tipo de nodo es el más importante y sólo hay uno por clúster. Es el responsable de almacenar los metadatos de los ficheros, es decir, la

información acerca de qué bloques forman un archivo y su localización en los nodos del clúster.

- **Datanodes**: Este tipo de nodos, de los que normalmente van a existir varios, se encargan de almacenar los bloques de información y de recuperarlos bajo demanda.

6.2. YARN

YARN (Yet Another Resource Negotiator) por sus siglas en inglés de "otro negociador de recuros" es una tecnología de administración de clústeres de Apache Hadoop. Básicamente, es una versión mejorada de MapReduce y está disponible a partir de la segunda generación de Hadoop.

YARN es una reescritura de software que desacopla las capacidades de gestión de recursos y planificación de MapReduce del componente de procesamiento de datos, permitiendo a Hadoop soportar enfoques más variados de procesamiento, y una gama más amplia de aplicaciones.

YARN combina un administrador central de recursos, denominado **ResourceManager**, que arbitra todos los recursos disponibles del clúster, y los agente de administración de cada nodo, los **NodeManager**, que monitorean las operaciones de procesamiento de nodos individuales.

YARN también cuenta con un elemento por aplicación, denominado **ApplicationMaster**, que se encarga de negociar los recursos con el ResourceManager y trabajar con los NodeManager para poder ejecutar y controlar las tareas, solicitándoles recursos para poder trabajar.

6.3. Spark

Apache Spark es un framework de computación de clúster de código abierto. Originalmente fue desarrollado en la Universidad de California, pero más tarde fue donado a la fundación de software de Apache, quien lo ha mantenido desde entonces.

Spark está escrito en el lenguaje de programación Scala y se ejecuta en el entorno JVM (*Java Virtual Machine*). Proporciona una API de programación para Scala, Java, Python y R.

Spark es un motor rápido y general para procesamiento de datos a gran escala. Spark proporciona una interfaz para programar clústeres enteros con paralelismo implícito y tolerancia a fallos.

Las principales características de Spark son las siguientes:

- **Procesamiento en memoria:** la característica principal de Spark es que tiene la capacidad de realizar procesamiento en memoria dentro de un clúster, aumentando la velocidad de procesamiento. La computación en memoria significa almacenar los datos en RAM y procesarlos en paralelo. Además, Spark tiene un motor de ejecución avanzado que facilita el cálculo en memoria.

- **Velocidad:** ejecuta programas hasta 100 veces más rápido que MapReduce en memoria y 10 veces más rápido en disco. Spark es capaz de mantener los resultados intermedios en memoria, en lugar de escribirlos en disco, proporcionando así un gran rendimiento. Spark intenta almacenar la mayor cantidad de datos posibles en memoria, mientras que los datos restantes permanecen en disco.

- **Tolerancia a fallos:** Spark está diseñado para manejar de forma transparente los fallos de cualquier nodo del trabajo en el clúster, evitando así la pérdida de datos e información.

- **Procesamiento de flujos de datos en tiempo real:** Spark es capaz de trabajar con datos en tiempo real.

- **Compatible con Hadoop:** Spark puede ejecutarse dentro en la primera generación de Hadoop sobre MapReduce, o sobre YARN en la segunda generación de Hadoop.

- **Evaluación perezosa (*lazy evaluation*):** es otra característica excepcional de Spark. Espera instrucciones antes de proporcionar un resultado final, ahorrando tiempo. Cuando se llama a una operación, no se ejecuta inmediatamente. Spark mantiene el registro de esa operación, y no la ejecutará hasta que no sea necesaria.

- **Fácil de usar:** permite escribir aplicaciones en Java, Scala, Python y R.

- **Librerías:** permite combinar SQL, streaming y analíticas complejas. Proporciona una pila de librerías, que se pueden combinar de forma transparente en la misma aplicación. Estas librerías son:

    - Spark SQL: permite la consulta de datos estructurados utilizando el lenguaje SQL.

    - MLib: para el aprendizaje automático (machine learning).

    - GraphX: para el procesamiento gráfico.

    - Spark Streaming: para el procesamiento de flujos de datos en tiempo real.

    - SparkR: para utilizar Spark desde R.

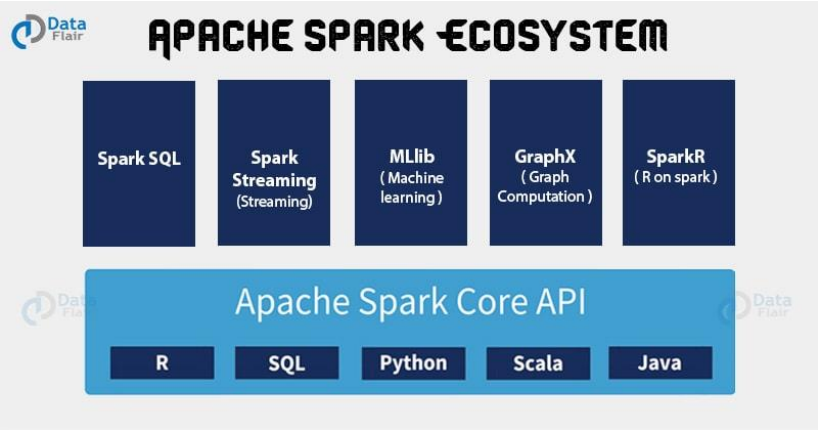


Ilustración 11. Librerías de Spark

6.4. Hive

Apache Hive es una infraestructura de almacenamiento de datos construida sobre Apache Hadoop para administrar grandes conjuntos de datos que se encuentran almacenados en un entorno distribuido, como HDFS.

Hive ofrece un lenguaje de consultas basado en SQL llamado HiveQL para leer y convertir consultas de forma transparente en MapReduce.

6.5. R

R es un entorno y un lenguaje de programación enfocado al análisis estadístico muy utilizado en el campo de la minería de datos.

Este lenguaje forma parte de GNU y se distribuye bajo licencia GNU GLP.

Características y ventajas de la programación en R:

- El lenguaje de programación R es un proyecto colaborativo y abierto, los desarrolladores pueden descargar el código de forma gratuita y modificarlo para incluir mejoras.

- Es un lenguaje interpretado, funciona mediante comandos.

- R proporciona una amplia gama de herramientas estadísticas que incluyen análisis de datos y generación de gráficos. Este lenguaje tiene capacidad de generar gráficos de alta calidad. Estas características lo convierten en una potente herramienta de cálculo.

- Gracias a este lenguaje de programación, los Data Scientists pueden manejar grandes volúmenes de datos.

- Puede integrarse con distintas bases de datos. Una de las ventajas más importantes de R es que funciona con diferentes tipos de hardware y software (Windows, Unix, Linux…)

- El lenguaje R ofrece la posibilidad de cargar bibliotecas y paquetes con diversas funcionalidades lo que permite a los usuarios extender su configuración básica.

- La comunidad en torno a R es muy activa por lo que es sencillo encontrar soluciones rápidamente a los problemas que los usuarios se puedan encontrar.

6.6. PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacional, orientado a objetos y libre, publicado bajo la licencia PostgreSQL.

PostgreSQL fue desarrollado en la Universidad de California.

PostgreSQL está diseñado para funcionar en plataformas tipo UNIX pero, como PostgreSQL es portátil, también podría funcionar en otras plataformas como Mac OS X, Solaris y Windows.

6.6.1. PostGIS

PostGIS es un módulo añade soporte de objetos geográfico a la base de datos relacional PostgreSQL, convirtiéndola en una base de datos espacial para su utilización en Sistemas de Información Geográfica. Se publica bajo la Licencia Pública General de GNU.

PostGIS permite a PostgreSQL almacenar, manipular y realizar consultas de ubicación que se ejecutan en SQL. Además, PostGIS ofrece muchas características que raramente se encuentran en otras bases de datos espaciales como Oracle Locator/Spatial y SQL Server.

7. Gestión del proyecto

El proyecto Métricas es un proyecto de investigación, por lo que la gestión este proyecto va a tener un enfoque diferente al de una aplicación software convencional.

7.1. Planificación

En este apartado se incluyen las metodologías propuestas, la planificación temporal y el presupuesto, es decir, lo que se ha realizado previo al desarrollo del proyecto.

7.1.1. Metodología propuesta

Inicialmente, la metodología propuesta para realizar el proyecto sigue el modelo en cascada. El modelo en cascada, también llamado lineal o secuencial, es el enfoque metodológico que ordena rigurosamente las etapas del proceso para el desarrollo del software, de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior. Las principales etapas que constituyen este modelo son las siguientes:

- **Análisis:** en esta fase, se identifican las características que debe tener el sistema y se recopilan los requisitos, que deben determinar qué objetivos y funcionalidades debe tener el sistema a desarrollar. Esta fase se centra en **el qué** se debe desarrollar.

- **Diseño:** en esta etapa se identifican y describen las abstracciones del software, describiendo así la estructura interna del producto, de forma que se cumpla con los objetivos y requisitos de la fase de análisis. En el proceso de diseño se traducen los requisitos en una representación del software con la calidad requerida antes de que comience la implementación. Esta fase se centra en **el cómo** se va a desarrollar el sistema.

- **Implementación:** en esta fase se traduce el diseño en una forma legible para la máquina, implementando el código fuente.

- **Pruebas:** los elementos, ya programados, se ensamblan para componer el sistema y se comprueba que funcionan correctamente y que cumplen con los requisitos, antes de entregar el software al usuario final.

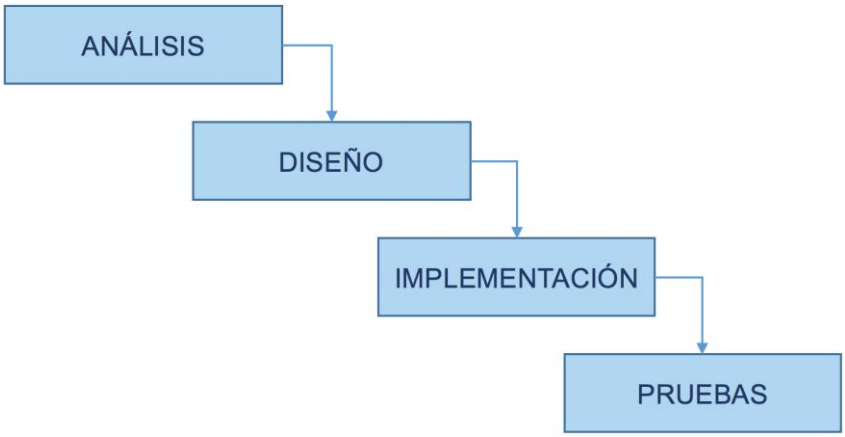


Ilustración 12. Etapas del modelo en cascada

7.1.2. Planificación temporal

El proyecto Métricas se encuadra en un marco temporal de cuatro meses de duración del proyecto. La carga de trabajo de este Trabajo Fin de Grado se calcula como 24 ECTS, ya que se ha realizado de forma conjunta con la asignatura Prácticas en Empresa. Si 1 ECTS supone aproximadamente 25 horas de trabajo, se estima la duración del proyecto en:

$$24\ ECTS \cdot \frac{25\ horas}{1\ ECTS} = 600\ horas$$

Debido a que la carga de trabajo repartirá entre cuatro meses:

$$\frac{600\ horas}{4\ meses} = 150\ horas/mes$$

Para calcular el número de horas al día que se deben dedicar al proyecto, considerando que se trabajan 30 días al mes:

$$\frac{150\ horas}{1\ mes} \cdot \frac{1\ mes}{30\ días} = 5\ horas/día$$

Resumiendo, se establece como horizonte temporal cuatro meses de duración del proyecto con una carga de trabajo total de 600 horas, que supondrían trabajar 5 horas al día, trabajando 30 días al mes.

Plan previsto de actividades

En el plan de tareas previstas para la realización de este proyecto se han incluido dos etapas más a las etapas principales del modelo en cascada descrito anteriormente: "1. Formación inicial" y "5. Documentación". Además, las etapas Análisis y Diseño se agrupan en la etapa "2. Definición de las métricas de eficiencia de vuelo e identificación de datos adicionales necesarios". El plan de tareas es el siguiente:

1. **Formación inicial (1 mes):** durante esta etapa se pretende adquirir la formación adecuada y necesaria para abordar y resolver el problema propuesto en este proyecto. La formación inicial se divide en dos subtareas:

    a. Familiarización con Hadoop y otras tecnologías relacionadas con el Big Data (15 días).

    b. Revisar el estado actual del proyecto AIRPORTS, especialmente en relación con la arquitectura Big Data propuesta, las fuentes de datos disponibles y el modelo de datos definido actualmente en AIRPORTS (15 días).

2. **Definición de las métricas de eficiencia de vuelo e identificación de datos adicionales necesarios (1 mes):** durante esta segunda actividad se propondrán posibles métricas de evaluación de eficiencia de vuelo basadas en sus trayectorias.

3. **Implementación de las métricas propuestas (1 mes):** durante esta etapa se implementarán las métricas propuestas en la plataforma Big Data del proyecto AIRPORTS.

4. **Evaluación de las métricas implementadas (15 días):** durante esta cuarta

actividad se testeará el correcto funcionamiento de las métricas implementadas y se realizará una evaluación preliminar de la adecuación de las mismas desde el punto de vista de la eficiencia de vuelo, sobre una muestra de trayectorias representativa.

5. **Documentación (15 días):** durante esta última etapa se profundizará en la documentación del proyecto, aunque deberá realizarse a lo largo de todo el proyecto. En la documentación se incluirán las descripciones conceptual y técnica de las técnicas implementadas en las actividades 2 y 3, así como una explicación del proceso de evaluación realizado y las conclusiones extraídas al respecto.

Tarea	**Duración**	**Inicio**	**Fin**
Proyecto Métricas	**120 días**	**1/2/2017**	**31/5/2017**
1. Formación inicial	30 días	1/2/2017	2/3/2017
Familiarización con Hadoop	15 días	1/2/2017	15/2/2017
Revisar el proyecto AIRPORTS	15 días	16/2/2017	2/3/2017
2. Definición de las métricas	30 días	3/3/2017	1/4/2017
1. Implementación de las métricas	30 días	2/4/2017	1/5/2017
2. Evaluación de las métricas	15 días	2/5/2017	16/5/2017
3. Documentación	15 días	17/5/2017	31/5/2017

Tabla 1. Plan de actividades

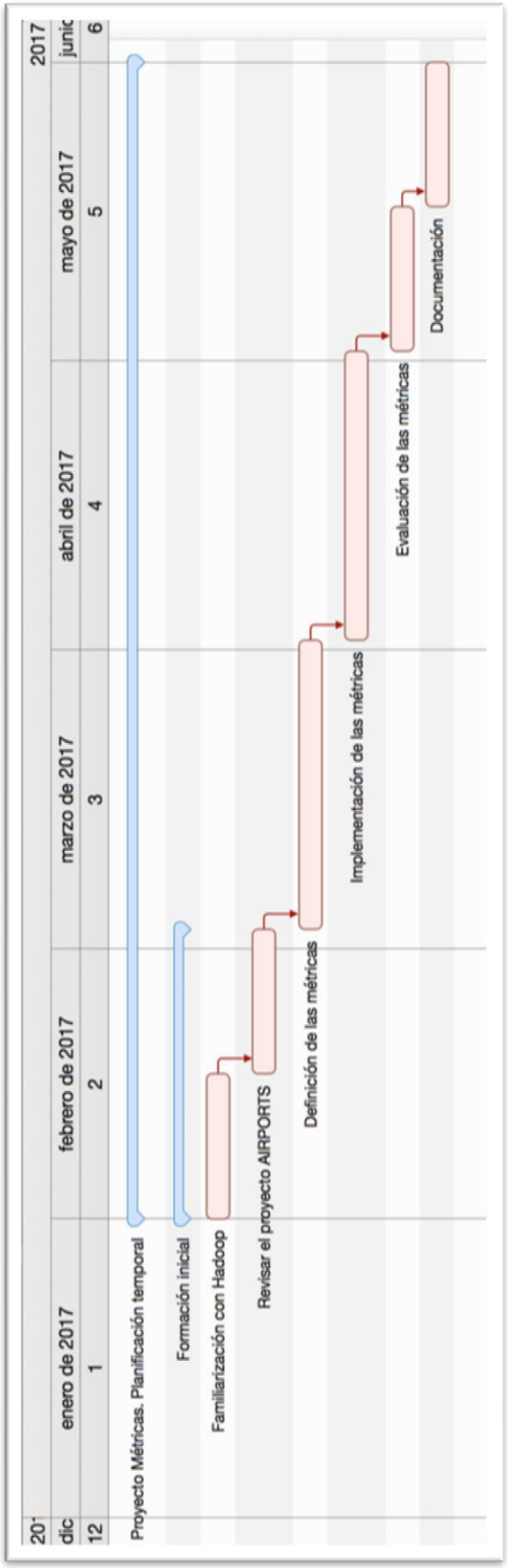


Ilustración 13. Diagrama de Gantt planificación

7.1.3. Presupuesto

Presupuesto hardware

Los costes relacionados con la plataforma ADAPT no se consideran debido a que es la empresa Boeing la que se encarga de sufragar estos costes. A continuación se detallan el presupuesto para el hardware:

Ordenador personal: MacBook Pro (Retina, 15 pulgadas), procesador 2,2 GHz Intel Core i7, memoria 16 GB 1600 MHz DDR3, gráficos Intel Iris Pro 1536 MB, sistema operativo OS X El Capitan.

- Uso: se estima el tiempo de vida útil del ordenador en 5 años ($5 \cdot 12 = 60\ meses$), por lo que si se va a utilizar durante 4 meses:

$$60\ meses \rightarrow 100\ \%$$
$$4\ meses \rightarrow 6.67\ \%$$

Conexión a internet: fibra óptica de 50 MB con movistar fusión, con un coste de 80 € al mes ($4\ meses \cdot 80 = 320$ €).

- Uso: debido a que el servicio contratado de movistar fusión incluye además servicios de telefonía fija, móvil y televisión, se estima el coste de la conexión a internet en un 65 % del coste total.

Costes de impresión: se estiman los costes de impresión y encuadernación en 20 €.

Presupuesto hardware	**Coste (€)**	**Uso (%)**	**Total (€)**
Ordenador personal	2200	6.67	146.74
Conexión a internet	320	65	208
Costes de impresión	20	100	20
		TOTAL	**374.74**

Tabla 2. Presupuesto hardware

Presupuesto software

Se han utilizado distintas herramientas software, pero las que son gratuitas no se han incluido en el presupuesto.

Microsoft Office Hogar y Estudiantes 2016 para Mac.

- Uso: se estima el uso total de este software en 4 años ($4 \cdot 12 = 48\ meses$), y

como se va a utilizar durante 4 meses:

$$48\ meses \rightarrow 100\ \%$$
$$4\ meses \rightarrow 8.33\ \%$$

Presupuesto software	Coste (€)	Uso (%)	Total (€)
Microsoft Office Hogar y Estudiantes 2016 para Mac	149.95	8.33	12.49
		TOTAL	**12.49**

Tabla 3. Presupuesto software

Presupuesto de personal

El trabajo realizado en este Trabajo Fin de Grado es un trabajo de investigación, relacionado con el Big Data y el Data Science, por lo que se han consultado el salario mínimo de un Data Scientist, que es de unos 30.000 € al año. Si consultamos el salario mínimo de un analista, estos cobran unos 25.000 € al año. Debido a que el personal es un Data Scientist en formación, se ha considerado como salario de referencia 25.000 € al año.

Considerando que al año se trabajan 12 meses, 20 días al mes y 8 horas al día, el coste por hora es de 13,02 €/h.

Presupuesto de personal	Tiempo (h)	Coste (€/h)	Total (€)
Data Scientist	600	13.02	7812
		TOTAL	**7812**

Tabla 4. Presupuesto de personal

Presupuesto total

A continuación se calcula el presupuesto total como la suma de los presupuestos hardware, software y de personal:

PRESUPUESTO	Total (€)
Presupuesto hardware	374.74
Presupuesto software	12.49
Presupuesto de personal	7812
TOTAL	**8199.23**

Tabla 5. Presupuesto total

7.2. Real

En este apartado se incluyen la metodología aplicada en realidad, el coste temporal real y el coste económico real, es decir, lo calculado después de la realización del proyecto.

7.2.1. Metodología real

A pesar de que la metodología propuesta seguía el modelo en cascada, a partir de que se han propuesto y seleccionado las métricas a implementar, se ha seguido un **modelo iterativo e incremental** en el proceso de desarrollo del software. Este modelo se utiliza cuando es necesario gestionar objetivos poco definidos o de una alta complejidad y permite reducir las posibles diferencias que pueden surgir entre los resultados esperados por el cliente y las características del producto final, permitiendo al equipo de desarrollo incorporar retroalimentación durante el proyecto.

El modelo de desarrollo iterativo e incremental consiste en la iteración de varios ciclos de vida en cascada, es decir, un conjunto de etapas (análisis, diseño, implementación y pruebas) agrupadas en pequeñas iteraciones. Al final de cada iteración se obtiene una versión del producto mejorada o con mayores funcionalidades. El producto final es la acumulación de mejoras y funcionalidades añadidas en cada iteración.

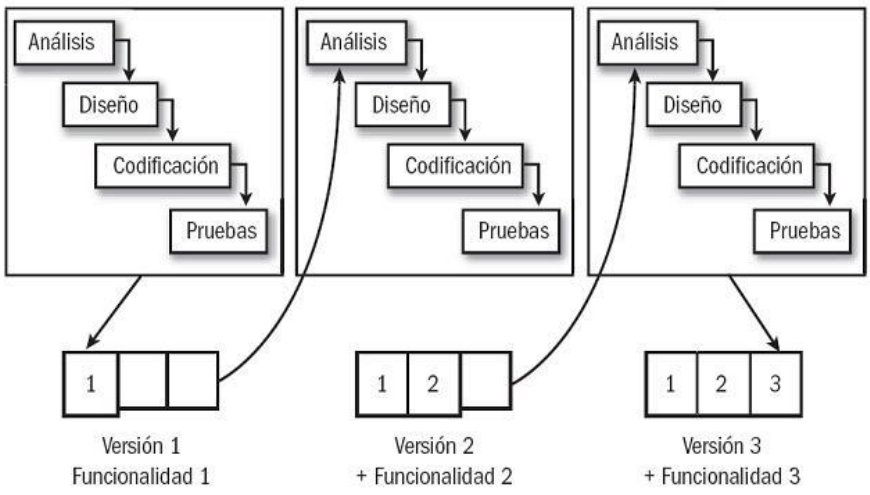


Ilustración 14. Iteraciones del modelo iterativo e incremental

7.2.2. Coste temporal real

Debido a que finalmente la metodología utilizada durante el desarrollo del proyecto ha sido el modelo iterativo e incremental, las actividades realizadas también se han visto modificadas, incluyendo las iteraciones características del modelo aplicado.

Informe de actividades realizadas

A continuación se detallan las actividades que se han realizado en el desarrollo de este proyecto:

1. **Formación inicial (30 días):** durante esta etapa se han realizado las siguientes actividades:

    - Revisión del estado actual del *Proyecto AIRPORTS*.
    - Instalación y familiarización con PostgreSQL y PostGIS.
    - Familiarización con el lenguaje R.
    - Familiarización con la plataforma Big Data ADAPT de Boeing.
    - Aprender sobre las herramientas del ecosistema Hadoop: sistema de ficheros HDFS, YARN, Spark, Hive, etc.

2. **Estudio y análisis de las métricas de eficiencia de vuelo (20 días):** durante esta etapa se han estudiado las métricas de eficiencia de vuelo existentes y se han definido y analizado las métricas de eficiencia de vuelo consideradas.

3. **ITERACIÓN 1. Métricas en local (16 días):** durante esta primera iteración, las métricas se implementan en local, es decir, fuera de la plataforma ADAPT. Para ello, se han realizado las siguientes tareas:

    - **Analizar** las métricas propuestas para su posterior implementación en local.

    - **Diseñar** las métricas propuestas para su posterior implementación en local.

    - **Implementar** las métricas propuestas en local, utilizando unos datos de prueba de las trazas de radar.

    - **Realizar pruebas** de la implementación realizada.

4. **ITERACIÓN 2. Métricas en la plataforma ADAPT. Datos trazas de radar desde Hive (23 días):** durante esta segunda iteración, las métricas se implementan ya en la plataforma ADAPT, obteniendo los datos de las trazas de radar directamente desde Hive. Para ello, se han realizado las siguientes tareas:

    - **Analizar** el proceso de integración del cálculo de las métricas dentro de la plataforma ADAPT.

    - **Diseñar** las métricas para que obtengan los datos de las trazas de radar directamente de la propia plataforma ADAPT, utilizando Hive para ello, y almacenen esos datos en PostgreSQL.

    - **Implementar** las métricas de forma que obtengan los datos de las trazas de radar directamente de la plataforma ADAPT, utilizando Hive para ello, y almacenen esos datos en PostgreSQL.

    - **Realizar pruebas** de la implementación realizada.

5. **ITERACIÓN 3. Métricas en la plataforma ADAPT. Procesar ficheros con información geográfica sobre las áreas (16 días):** durante esta iteración, se han procesado los ficheros con la información geográfica de las distintas áreas de interés consideradas (aeropuertos, sectores ATC y FIRs) y se han almacenado los datos en PostgreSQL. Para ello, se han realizado las siguientes tareas:

    - **Analizar** los ficheros con los datos sobre las áreas.

    - **Diseñar** una solución para procesar los ficheros y almacenar los resultados en PostgreSQL.

    - En la etapa de **implementación** se incluyen las siguientes tareas:

        - Cargar un fichero *json* con los datos de los aeropuertos desde HDFS a una tabla en PostgreSQL.

        - Procesar un fichero almacenado en local con los datos de los sectores ATC de España y cargar los datos procesados en una tabla en PostgreSQL.

        - Procesar un fichero almacenado en local con los datos de los FIRs y cargar los datos procesados en una tabla en PostgreSQL.

    - **Realizar pruebas** de la implementación realizada.

6. **ITERACIÓN 4. Métricas en la plataforma ADAPT. Calcular métricas para un mes (29 días):** en esta última iteración se implementado las funciones necesarias para calcular las métricas para un mes entero, a modo de procesamiento por lotes (*batch processing*). Para ello, se han realizado las siguientes tareas:

    - **Analizar** el problema y estudiar las funciones vectoriales de R para su uso en el cálculo de las métricas de un mes. Estudiar también la escritura en HDFS y en Hive de unos resultados de métricas de prueba.

    - **Diseñar** una solución para calcular las métricas de un mes empleando la vectorización en R.

    - **Implementar** las funciones necesarias empleando la vectorización en R para el cálculo de las métricas de un mes. También se han realizado las siguientes tareas de implementación:

        - Crear una tabla en Hive para almacenar los resultados del cálculo de las métricas.

        - Crear una consulta para ejecutar en PostgreSQL con el objetivo de filtrar los FIRs, seleccionando únicamente los FIRs para los que haya datos en las trazas de radar.

- **Probar** el correcto funcionamiento de todas las funciones implementadas y calcular los resultados de las métricas de un mes entero y almacenarlos en HDFS y en Hive.

7. **Documentación (30 días):** En esta última etapa se ha realizado la documentación necesaria, describiendo el trabajo realizado y las técnicas implementadas, así como una explicación detallada del proceso de evaluación realizado y las conclusiones extraídas al respecto.

Tarea	Duración	Inicio	Fin
Proyecto Métricas	**164 días**	**1/2/2017**	**14/7/2017**
1. Formación inicial	30 días	1/2/2017	2/3/2017
Revisar el proyecto AIRPORTS	10 días	1/2/2017	10/2/2017
Familiarización con Hadoop, R, etc.	20 días	11/2/2017	2/3/2017
2. Estudio y análisis de las métricas de eficiencia de vuelo	20 días	3/3/2017	22/3/2017
8. ITERACIÓN 1	16 días	23/3/2017	7/4/2017
Análisis	3 días	23/3/2017	25/3/2017
Diseño	4 días	26/3/2017	29/3/2017
Implementación	7 días	30/3/2017	5/4/2017
Pruebas	2 días	6/4/2017	7/4/2017
9. ITERACIÓN 2	23 días	8/4/2017	30/4/2017
Análisis	4 días	8/4/2017	11/4/2017
Diseño	7 días	12/4/2017	18/4/2017
Implementación	10 días	19/4/2017	28/4/2017
Pruebas	2 días	29/4/2017	30/4/2017
10. ITERACIÓN 3	16 días	1/5/2017	16/5/2017
Análisis	3 días	1/5/2017	3/5/2017
Diseño	4 días	4/5/2017	7/5/2017
Implementación	7 días	8/5/2017	14/5/2017
Pruebas	2 días	15/5/2017	16/5/2017
11. ITERACIÓN 4	29 días	17/5/2017	14/6/2017
Análisis	4 días	17/5/2017	20/5/2017
Diseño	6 días	21/5/2017	26/5/2017
Implementación	15 días	27/5/2017	10/6/2017
Pruebas	4 días	11/6/2017	14/6/2017
12. Documentación	30 días	15/6/2017	14/7/2017

Tabla 6. Actividades realizadas

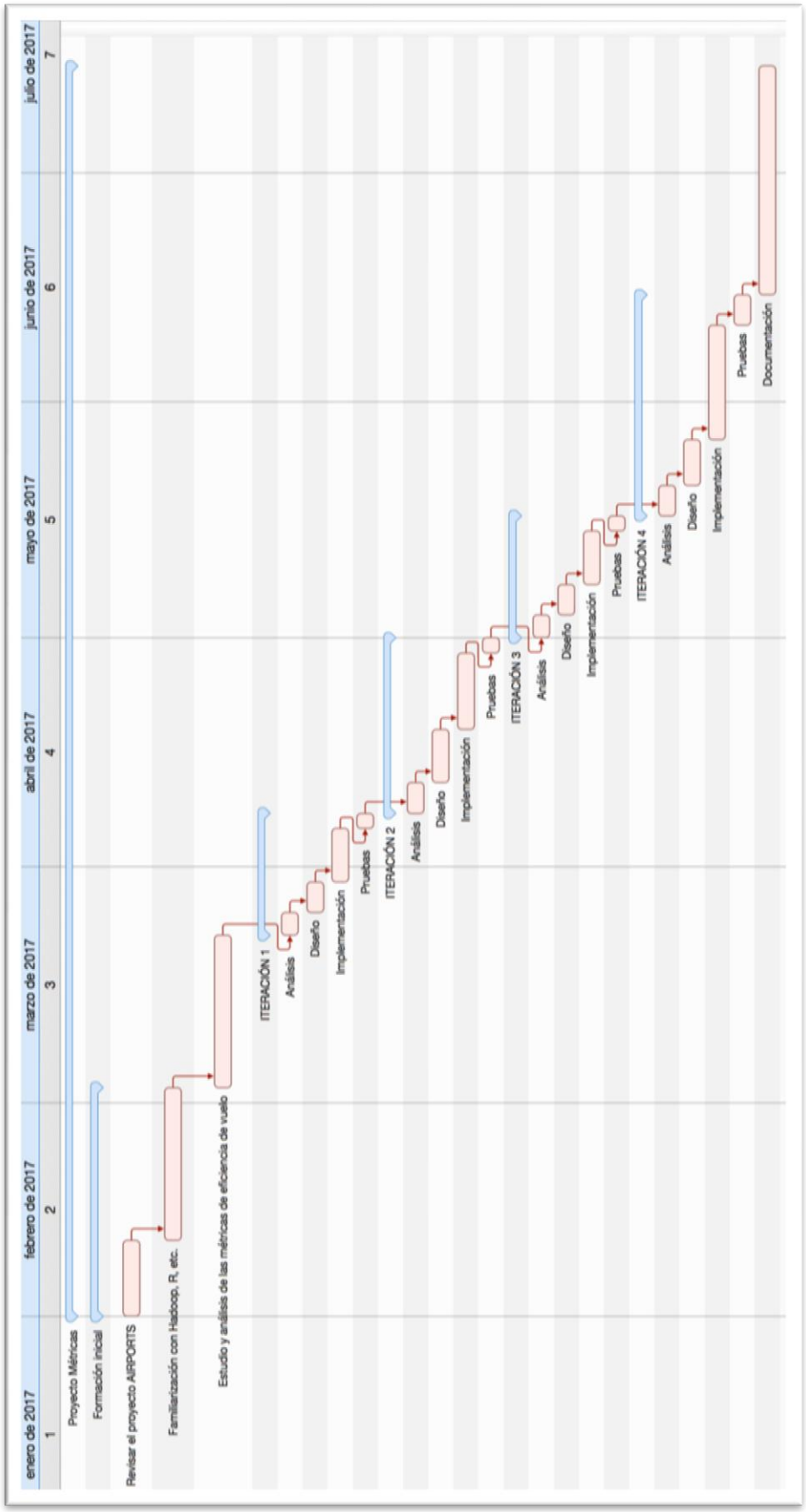


Ilustración 15. Diagrama de Gantt real

Coste temporal real total:

$$164\ días \cdot \frac{5\ horas}{día} = 820\ horas\ en\ total$$

$$164\ días \cdot \frac{30\ días}{mes} = 5.47\ meses\ en\ total$$

7.2.3. Coste económico real

Costes reales del hardware

Ordenador personal.

- Uso: se estima el tiempo de vida útil del ordenador en 5 años ($5 \cdot 12 = 60\ meses$), por lo que como se ha utilizado durante 5.47 meses:

$$60\ meses \rightarrow 100\ \%$$
$$5.47\ meses \rightarrow 9.11\ \%$$

Conexión a internet: con un coste de 80 € al mes ($5.47\ meses \cdot 80 = 437.60$ €).

- Uso: debido a que el servicio contratado de movistar fusión incluye además servicios de telefonía fija, móvil y televisión, se considera el coste de la conexión a internet en un 65 % del coste total.

Costes de impresión: los costes de impresión y encuadernación son de 20 €.

Costes hardware	Coste (€)	Uso (%)	Total (€)
Ordenador personal	2200	9.11	200.42
Conexión a internet	437.60	65	307.84
Costes de impresión	20	100	20
		TOTAL	**528,26**

Tabla 7. Costes hardware

Costes reales del software

Microsoft Office Hogar y Estudiantes 2016 para Mac.

- Uso: se estima el uso total de este software en 4 años ($4 \cdot 12 = 48\ meses$), y como se ha utilizado durante 5.47 meses:

$$48\ meses \rightarrow 100\ \%$$
$$5.47\ meses \rightarrow 11.40\ \%$$

Costes software	Coste (€)	Uso (%)	Total (€)
Microsoft Office Hogar y Estudiantes 2016 para Mac	149.95	11.40	17.09
		TOTAL	**17.09**

Tabla 8. Costes software

Coste real de personal

El salario se considera igual al considerado en la estimación, es decir, con un coste por hora de 13,02 €/h.

Coste de personal	Tiempo (h)	Coste (€/h)	Total (€)
Data Scientist	820	13.02	10676.40
		TOTAL	**10676.40**

Tabla 9. Coste de personal

Coste económico real total

A continuación se calcula el coste real total como la suma de los costes reales del hardware, del software y de personal:

Coste económico real	Total (€)
Coste real del hardware	528.26
Coste real del software	17.09
Coste real de personal	10676.40
TOTAL	**11221.75**

Tabla 10. Coste económico real

7.3. Comparación

Costes temporales

	Estimado	Real	Diferencia
Meses	4	5.47	1.47
Horas	600	820	220

Tabla 11. Comparación costes temporales

El desarrollo del proyecto ha requerido de 220 horas más de lo planificado, debido, en gran parte a la naturaleza del proyecto, con requisitos poco definidos inicialmente y cambiantes en el tiempo.

Costes económicos

	Estimado (€)	Real (€)	Diferencia
Hardware	374.74	528.26	153.52
Software	12.49	17.09	4.60
Personal	7812	10676.40	2864.40
TOTAL	**8199.23**	**11221.75**	**3022.52**

Tabla 12. Comparación costes económicos

El coste económico real también ha sido mayor al presupuesto, ya que ha sido afectado por la variación en el tiempo de desarrollo del proyecto.

SECCIÓN II: DOCUMENTACIÓN TÉCNICA

1. Análisis

Debido a que este proyecto es un proyecto de investigación, tanto en el apartado de análisis como en el de diseño, no se han considerado ni los apartados ni los diagramas típicos de una aplicación software convencional, sino que se han seleccionado distintos apartados y diagramas con el objetivo de que se ajusten mejor a este proyecto.

1.1. Diagrama de contexto

Los diagramas de flujos de datos representan gráficamente el sistema e ilustran cómo fluyen los datos a través de los distintos procesos del sistema. Los diagramas de flujo de datos se realizan a distintos niveles de abstracción. Por ejemplo, el diagrama de flujo de datos de nivel 0 (diagrama de contexto) modela al más alto nivel los flujos de datos del sistema. El diagrama de contexto representa visualmente el alcance, es decir, el límite entre el nuevo sistema y el resto del universo.

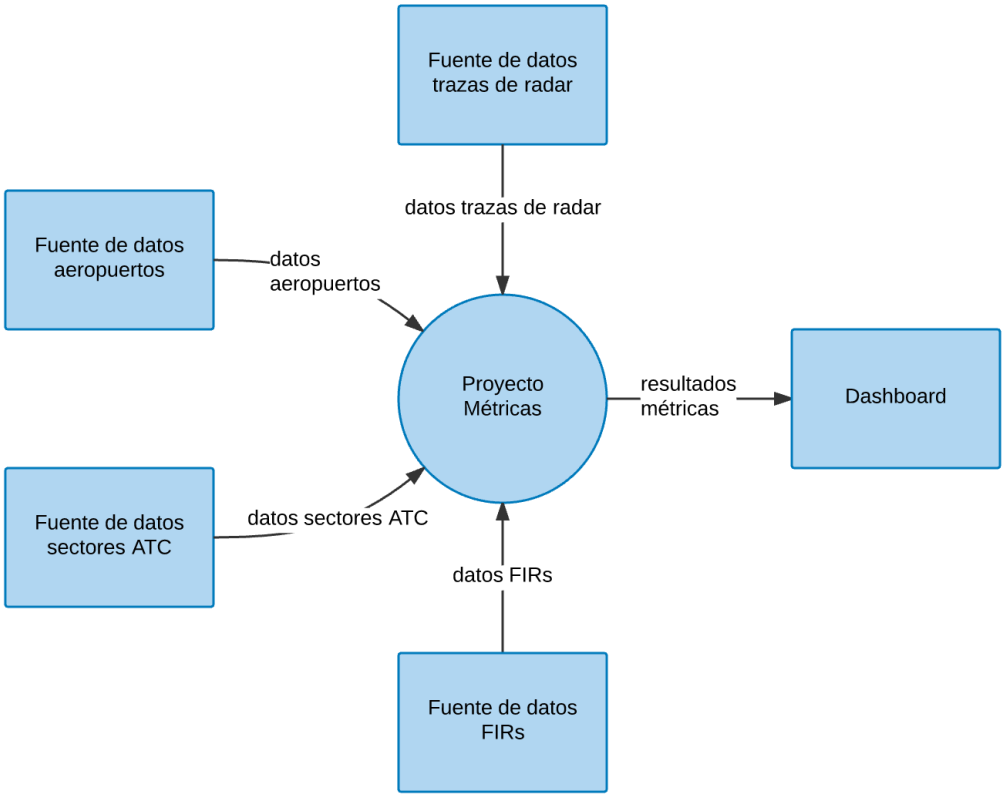


Ilustración 16. Diagrama de contexto

La notación utilizada en los diagramas de flujo de datos es la siguiente:

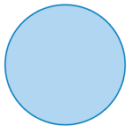
Descripción	Símbolo
Entidad externa	
Proceso	
Almacén de datos	
Flujo de datos	

Tabla 13. Notación de los diagramas de flujo de datos

1.2. Diagrama de flujo de datos

Los sucesivos niveles de diagramas de flujo de datos ofrecen más nivel de detalle. El siguiente diagrama muestra el diagrama de flujo de datos de nivel 1 del sistema:

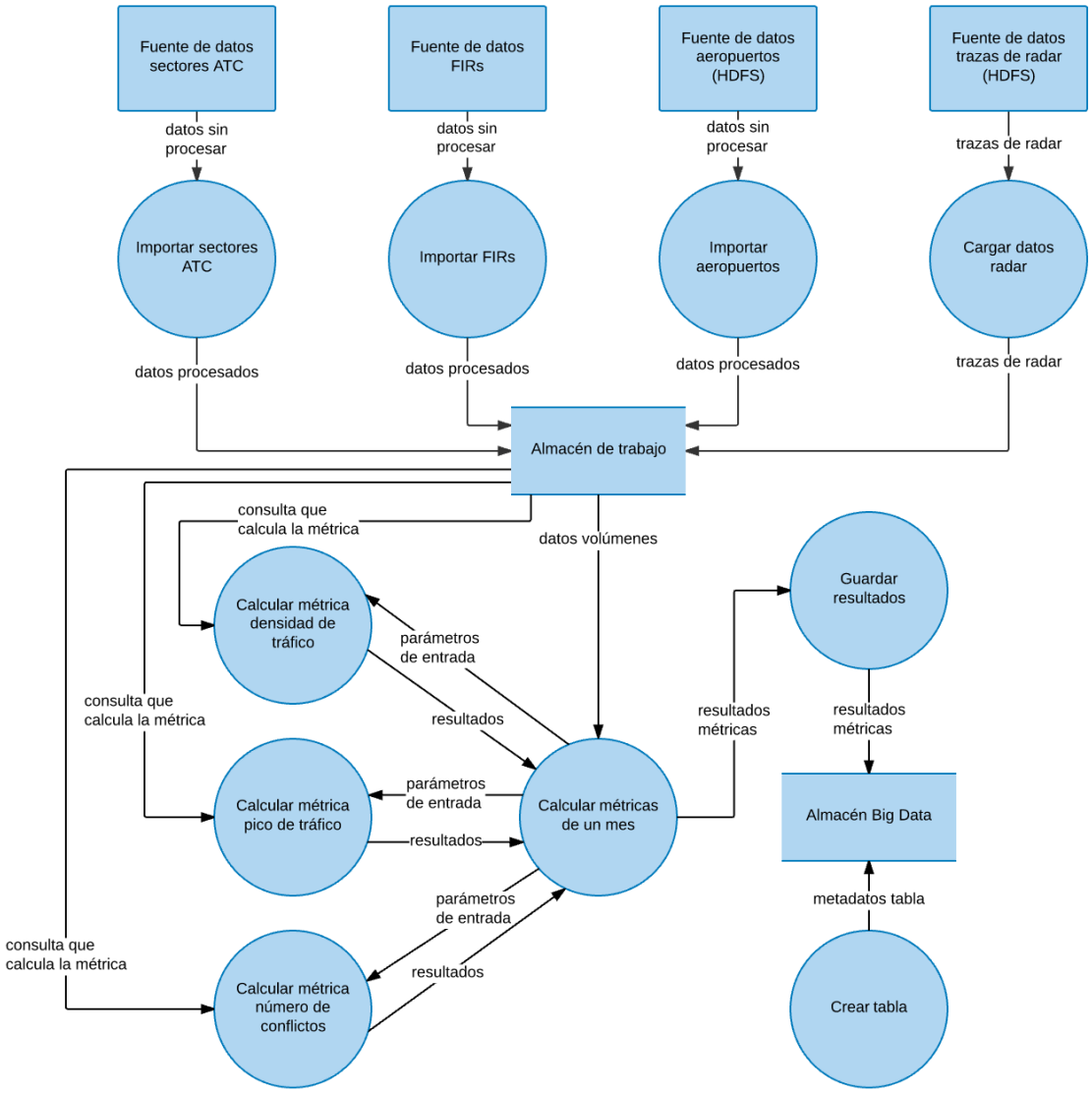


Ilustración 17. Diagrama de flujo de datos (nivel 1)

1.3. Reglas de negocio

Las reglas de negocio describen las políticas, normas, operaciones, definiciones y restricciones presentes en una organización. A lo largo del desarrollo del proyecto, se han ido identificando las siguientes reglas de negocio:

- **RN-01:** Se trabajará con los datos de vigilancia aérea obtenidos a partir de la técnica del radar procedentes de la técnica de radar.

- **RN-02:** Los volúmenes de referencia de interés considerados para el cálculo de las métricas son:

    - FIRs.

    - Sectores ATC de España.
    - Áreas en torno a los aeropuertos.

- **RN-03:** Las métricas deberán poder calcularse para un mes entero, considerando como intervalo temporal una hora.

1.4. Requisitos funcionales

Los requisitos funcionales definen funciones o funcionalidades del sistema:

- **RF-01:** El sistema permitirá cargar los datos de los aeropuertos desde una fuente de datos externa a una base de datos espacial.
- **RF-02:** El sistema permitirá procesar y cargar los datos de los sectores ATC de España desde una fuente de datos externa a una base de datos espacial.
- **RF-03:** El sistema permitirá procesar y cargar los datos de los FIRs desde una fuente de datos externa a una base de datos espacial.
- **RF-04:** El sistema será capaz de calcular la métrica densidad de tráfico.
- **RF-05:** El sistema será capaz de calcular la métrica pico de tráfico.
- **RF-06:** El sistema será capaz de calcular la métrica número de conflictos.
- **RF-07:** El sistema permitirá calcular los resultados de los tres tipos de métricas para cada hora, de cada día, de un mes entero.
- **RF-08:** El sistema almacenará los resultados de las métricas en un almacenamiento estable.

1.5. Requisitos de información

Se han identificado los siguientes requisitos de información:

- **RI-01:** El sistema deberá almacenar información sobre los tiempos de referencia utilizados para el cálculo de las métricas.
- **RI-02:** El sistema deberá almacenar información sobre los volúmenes de referencia utilizados para el cálculo de las métricas.
- **RI-03:** El sistema deberá almacenar información sobre las trayectorias de las aeronaves.

- **RI-04:** El sistema deberá almacenar información sobre los valores resultantes del cálculo de las métricas.

1.5.1. Diagrama Entidad-Relación

El proyecto métricas integra conjuntos de datos procedentes de distintas fuentes, pero están interrelacionadas entre sí. Por ello, a nivel conceptual, se pueden modelar utilizando un diagrama entidad-relación, ya que permite representar las entidades de un sistema, así como sus interrelaciones y propiedades.

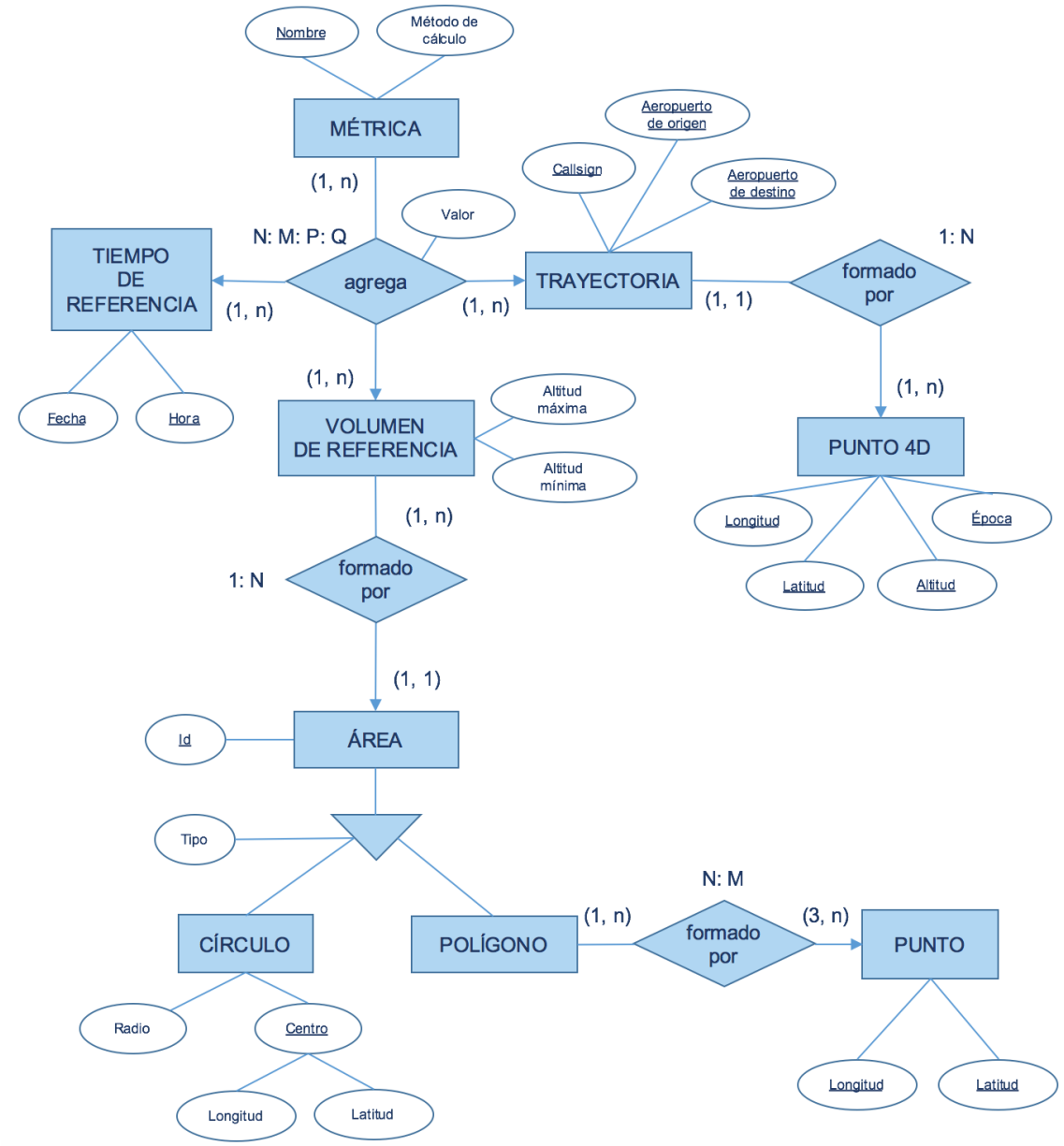


Ilustración 18. Diagrama Entidad-Relación

1.5.2. Diccionario de datos

Entidades

MÉTRICA		
Atributo	**Descripción**	**Tipo de dato**
Nombre	Nombre de la métrica	`string`
Método de cálculo	Instrucciones a seguir para calcular la métrica	`string`

Tabla 14. Entidad MÉTRICA

TIEMPO DE REFERENCIA		
Atributo	**Descripción**	**Tipo de dato**
Fecha	Fecha indicando el año, mes y día.	`timestamp`
Hora	Hora.	`timestamp`

Tabla 15. Entidad TIEMPO DE REFERENCIA

VOLUMEN DE REFERENCIA		
Atributo	**Descripción**	**Tipo de dato**
Altitud máxima	Altitud máxima del volumen de referencia.	`float`
Altitud mínima	Altitud mínima del volumen de referencia.	`float`

Tabla 16. Entidad VOLUMEN DE REFERENCIA

ÁREA		
Atributo	**Descripción**	**Tipo de dato**
Id	Identificador del área.	`string`
Tipo	Tipo de área. Puede ser círculo o polígono.	`string`

Tabla 17. Entidad ÁREA

CÍRCULO		
Atributo	**Descripción**	**Tipo de dato**
Radio	Radio del círculo.	`float`
Centro	Punto que representa el centro del círculo.	`float` (Atributo compuesto).

Tabla 18. Entidad CÍRCULO

PUNTO		
Atributo	**Descripción**	**Tipo de dato**
Longitud	Longitud.	`float`
Latitud	Latitud.	`float`

Tabla 19. Entidad PUNTO

TRAYECTORIA		
Atributo	**Descripción**	**Tipo de dato**
Callsign	Callsign.	`string`
Aeropuerto de origen	Aeropuerto de origen.	`string`
Aeropuerto de destino	Aeropuerto de destino.	`string`

Tabla 20. Entidad TRAYECTORIA

PUNTO 4D		
Atributo	**Descripción**	**Tipo de dato**
Longitud	Longitud.	`float`
Latitud	Latitud.	`float`
Altitud	Altitud.	`float`
Época	Cantidad de segundos transcurridos desde la medianoche UTC del 1 de enero de 1970.	`time`

Tabla 21. Entidad PUNTO 4D

Relaciones

AGREGA		
Descripción	Relación cuaternaria entre las entidades MÉTRICA, TIEMPO DE REFERENCIA, VOLUMEN DE REFERENCIA y TRAYECTORIA.	
Cardinalidad	N: M: P: Q	
Atributo	**Descripción**	**Tipo de dato**
Valor	Resultado del cálculo de la métrica.	`float`

Tabla 22. Relación AGREGA

FORMADO POR (Entre VOLUMEN DE REFERENCIA y ÁREA)	
Descripción	Relación entre las entidades VOLUMEN DE REFERENCIA y ÁREA. Asocia un volumen de referencia con el área que lo forma.
Cardinalidad	1: N

Tabla 23. Relación FORMADO POR

FORMADO POR (Entre POLÍGONO y PUNTO)	
Descripción	Relación entre las entidades POLÍGONO y PUNTO. Asocia un polígono con los puntos que lo forman.
Cardinalidad	N: M

Tabla 24. Relación FORMADO POR

FORMADO POR (Entre TRAYECTORIA y PUNTO 4D)	
Descripción	Relación entre las entidades TRAYECTORIA y PUNTO 4D. Asocia una trayectoria con los puntos que la forman.
Cardinalidad	1: N

Tabla 25. Relación FORMADO POR

2. Diseño

Debido a que este proyecto es un proyecto de investigación, tanto en el apartado de análisis como en el de diseño, no se han considerado ni los apartados ni los diagramas típicos de una aplicación software convencional, sino que se han seleccionado distintos apartados y diagramas con el objetivo de que se ajusten mejor a este proyecto.

2.1. Descripción de las fuentes de datos externas

Datos de las trazas de radar

Esta información se encuentra en el almacén Big Data, en la base de datos “dart” en la tabla “radar” y se accede a ella a través de Hive. Esta tabla se representan los atributos más relevantes:

Tabla “radar”		
Atributo	**Descripción**	**Tipo de dato**
callsign	Callsign.	`string`
adep	Aeropuerto de origen.	`string`
ades	Aeropuerto de destino.	`string`
aircraft	Tipo de aeronave.	`string`
date_t	Fecha.	`string`
time_t	Hora.	`string`
lat	Latitud.	`float`
lng	Longitud.	`float`
modo_c	Flight level.	`smallint`
vel_mod	Módulo de la velocidad.	`float`
vel_x	Componente x de la velocidad.	`float`
vel_y	Componente y de la velocidad.	`float`
vel_z	Componente z de la velocidad.	`float`

Tabla 26. Tabla "radar"

Datos de los aeropuertos

Los datos de los aeropuertos están originalmente en un fichero en formato *json* en el sistema de ficheros HDFS de la plataforma Big Data ADAPT. Un fragmento de este fichero se muestra a continuación.


```
[{"code":"AAA","name":"Anaa","country_code":"PF",
"updated":"2015-10-07T16:59:28.000Z","icao":"NTGA","city_code":"AAA",
"lat":-17.05,"lng":-145.41667,"alternatenames":"Anaa,Anaa,Анаа,อานา,阿纳机场
","timezone":"Pacific/Tahiti","gmt":-10,"is_rail_road":0,"is_bus_station":0,
"tch_code":null,"popularity":0,"phone":"","website":"","geoname_id":6947726,
"routes":0,"wiki":""},{"code":"AAB","name":"Arrabury","country_code":"AU",
```


Ilustración 19. Fragmento del fichero de los aeropuertos

Este fichero json continene muchos campos con información sobre los aeropuertos, pero no todos son de interés para el cálculo de las métricas. En la siguiente tabla se detallan los campos más relevantes:

Datos de los aeropuertos		
Atributo	Descripción	Tipo de dato
country_code	Código del país donde se sitúa el aeropuerto.	`text`
icao	Código en formato ICAO del aeropuerto.	`text`
lat	Latitud a la que se sitúa el aeropuerto.	`double precision`
lng	Longitud a la que se sitúa el aeropuerto.	`double precision`
name	Nombre del aeropuerto.	`text`

Tabla 27. Datos de los aeropuertos

Datos de los sectores ATC de España

La información acerca de los sectores ATC de España se extrae de un fichero externo proporcionado por la empresa Boeing. Este fichero sigue la estructura TAAM (*Total Airspace and Airport Modeller*), en la que los datos de los sectores vienen en forma jerárquica o de árbol. A continuación se muestra un fragmento del fichero en el aparece la información sobre dos sectores ATC.

```
1 12 "ATC Sector Boundary" <GCCC_AAC1>[0.0, 12500.0]xyzf(set color)
-13.7000000000 29.6166666667
-13.6666666667 29.5277777778
-12.9491666667 29.5277777778
-13.3250000000 28.6725000000
-13.8333333333 27.8333333333
-14.6666666667 27.8333333333
-14.6666666667 28.0083333333
-14.9166666667 28.6500000000
-13.7602777778 29.5277777778
-13.7000000000 29.6166666667
1 12 "ATC Sector Boundary" <GCCC_ACW1>[0.0, 24500.0]xyzf(set color)
-19.0000000000 27.4833333333
-17.7308333333 29.9952777778
-16.0000000000 30.0000000000
-16.6666666667 29.0000000000
-16.6650000000 28.8627777778
-17.0000000000 28.8000000000
-17.0000000000 28.2466666667
-17.3666666667 28.1666666667
-17.2500000000 27.5000000000
-19.0000000000 27.4833333333
```

Ilustración 20. Fragmento del fichero de los sectores ATC

La siguiente imagen refleja la estructura jerárquica del fichero de los sectores, incluyendo los campos de datos de que se van a extraer y a procesar.

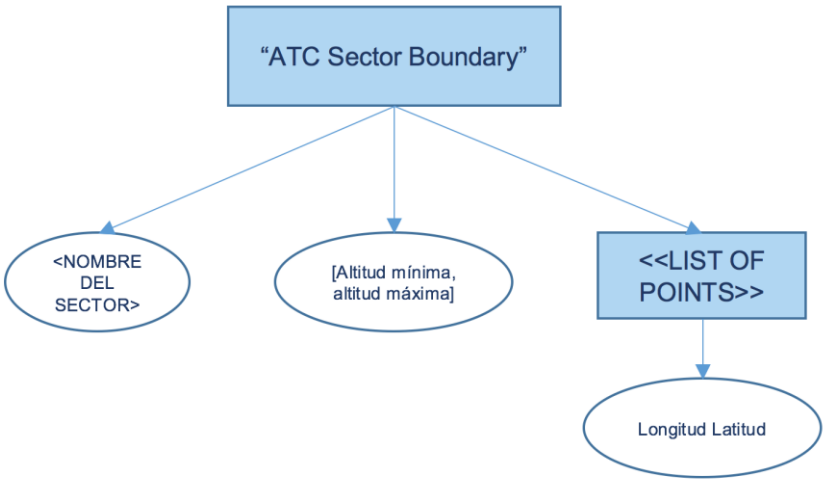


Ilustración 21. Estructura fichero sectores ATCÇ

Datos de los FIRs

La información acerca de los FIRs de España se extrae de un fichero externo proporcionado por la empresa Boeing. Este fichero sigue la estructura TAAM (*Total Airspace and Airport Modeller*), en la que los datos de los FIRs vienen en forma jerárquica o de árbol. A continuación se muestra un fragmento del fichero.

```
1 15 "FIR Boundary" <EB>[0.0, 46000.0]xyzf(set color)
1.998690 51.466267
3.364461 51.333021
3.364461 51.316365
3.381117 51.299710
3.364461 51.283054
3.364461 51.266398
3.364461 51.249742
3.397773 51.233087
3.414429 51.216431
3.447740 51.199775
3.481052 51.216431
3.514363 51.216431
3.514363 51.233087
3.514363 51.249742
3.547675 51.249742
3.564330 51.266398
3.580986 51.266398
3.614298 51.266398
3.630953 51.249742
3.664265 51.249742
3.697576 51.233087
```

Ilustración 22. Fragmento del fichero de los FIRs

La siguiente imagen refleja la estructura jerárquica del fichero de los FIRs, incluyendo los campos de datos de que se van a extraer y a procesar.

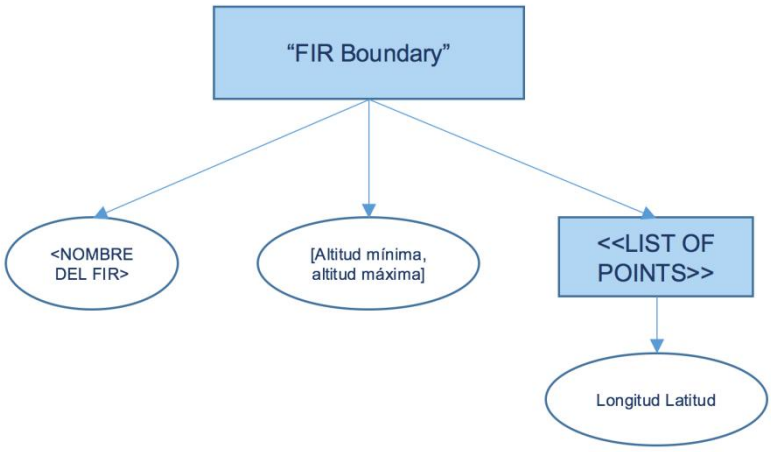


Ilustración 23. Estructura del fichero de los FIRs

2.2. Tablas internas

Tabla aeropuertos

El fichero con la información de los aeropuertos detallado en el apartado anterior se carga en R, se procesa y se almacena en una tabla en PostgreSQL, denominada "test_airports". Esta tabla contiene, además de toda la información contenida en el fichero json, una columna denominada "point", en la que se almacena la información geográfica del aeropuerto representada mediante un punto creado a partir de los valores de latitud y longitud de las columnas "lat" y "lng", respectivamente.

Tabla sectores ATC de España

Tras procesar el fichero de los sectores ATC, se crea en PostgreSQL una tabla denominada "test_atc_sectors_spain", con los siguientes atributos:

Tabla "test_atc_sectors_spain"		
Atributo	**Descripción**	**Tipo de dato**
sector_name	Nombre del sector ATC.	`text`
alt_min	Altitud mínima (en pies).	`numeric`
alt_max	Altitud máxima (en pies).	`numeric`
sector_polygon	Polígono del sector ATC.	`geography`

Tabla 28. Tabla "test_atc_sectors_spain"

Tabla FIRs

Tras procesar el fichero de los FIRs, se crea en PostgreSQL una tabla denominada "test_firs", con los siguientes atributos:

Tabla "test_firs"		
Atributo	**Descripción**	**Tipo de dato**
fir_name	Nombre del FIR.	`text`
alt_min	Altitud mínima (en pies).	`numeric`
alt_max	Altitud máxima (en pies).	`numeric`
fir_polygon	Polígono del FIR.	`geography`

Tabla 29. Tabla "test_firs"

Tabla con las trazas de radar creada para el cálculo de las métricas

Las métricas se traducen en consultas en PostgreSQL. Para calcular las métricas, una parte de todos los datos de las trazas de radar se carga en PostgreSQL para aprovechar las capacidades de consulta sobre datos geográficos que ofrece PostGIS. La tabla en la que se cargan las distintas porciones de trazas de radar tiene los siguientes atributos:

Tabla "test_radar_tracks"		
Atributo	**Descripción**	**Tipo de dato**
callsign	Callsign.	`text`
adep	Aeropuerto de origen.	`text`
ades	Aeropuerto de destino.	`text`
time_s	Fecha y hora en formato texto originalmente registrado en Hive.	`text`
lng	Longitud a la que se encuentra la aeronave en un determinado punto.	`double precision`
lat	Latitud a la que se encuentra la aeronave en un determinado punto.	`double precision`
alt	Altitud a la que se encuentra la aeronave en un determinado punto.	`double precision`
time	Fecha y hora alternativas en formato timestamp.	`timestamp without time zone`
point	Punto geográfico (lng, lat).	`geography`
point4d	Punto 4D (lng, lat, alt, época)	`geometry`

Tabla 30. Tabla "test_radar_tracks"

Tabla para almacenar los resultados de las métricas

Los resultados de las métricas se almacenan en una tabla en Hive denominada "metrics_results", con los siguientes atributos:

Tabla "metrics_results"		
Atributo	**Descripción**	**Tipo de dato**
type_of_metric	Tipo de métrica.	`string`
initial_time	Tiempo inicial.	`string`
ref_time_int	Intervalo de tiempo de referencia.	`float`
area_type	Tipo de área: `circle` o `poligon`.	`string`
area_name	Nombre del área.	`string`
radius	Radio si el tipo de área es `circle`. En otro caso, `NA`.	`float`
flight_level_min	Nivel de vuelo mínimo del volumen de referencia.	`float`
flight_level_max	Nivel de vuelo máximo del volumen de referencia.	`float`
result	Resultado de la métrica.	`float`

Tabla 31. Tabla "metrics_results"

2.3. Diagrama de estructura

El objetivo de este diagrama es representar la estructura modular del sistema, definiendo los parámetros de entrada y salida de cada uno de los módulos. A continuación se muestran dos diagramas, el segundo es continuación del primero (a partir del módulo "Calcular métricas de un mes".

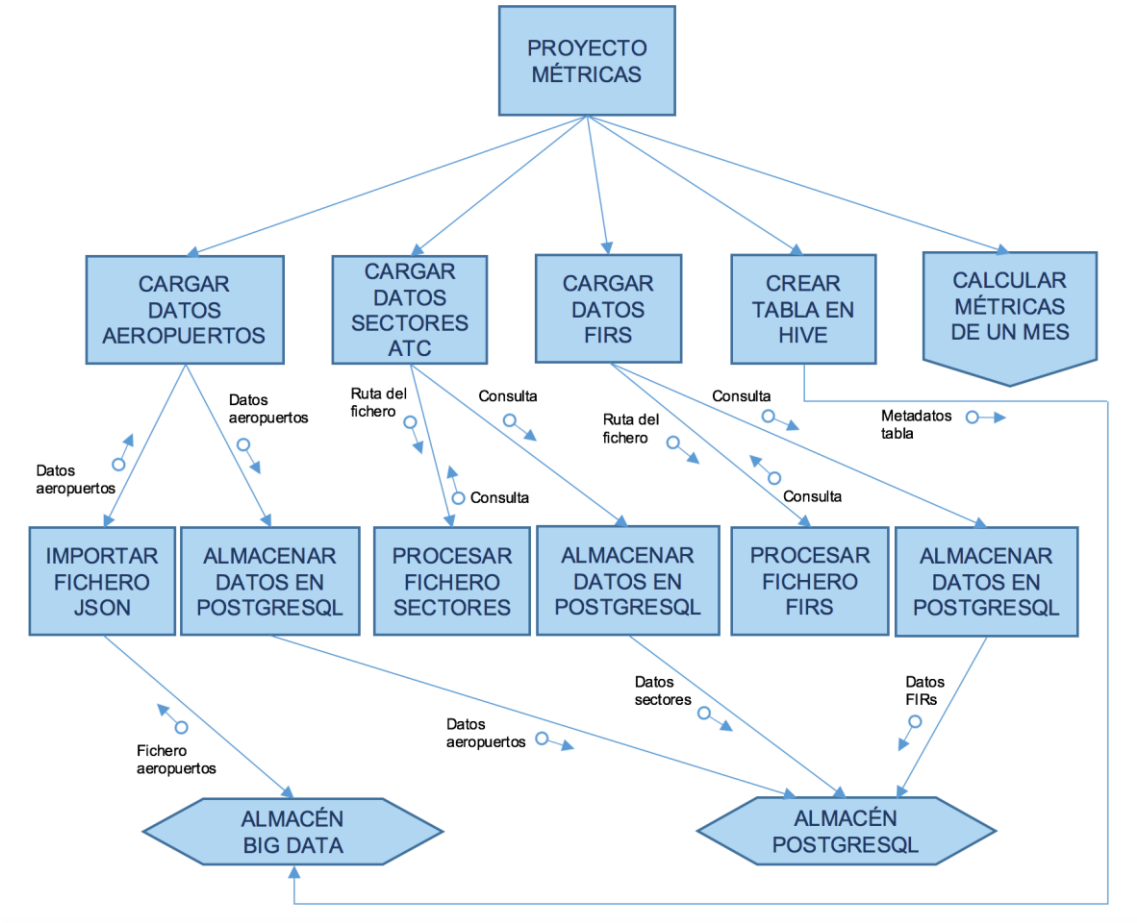


Ilustración 24. Diagrama de estructura (primera parte)

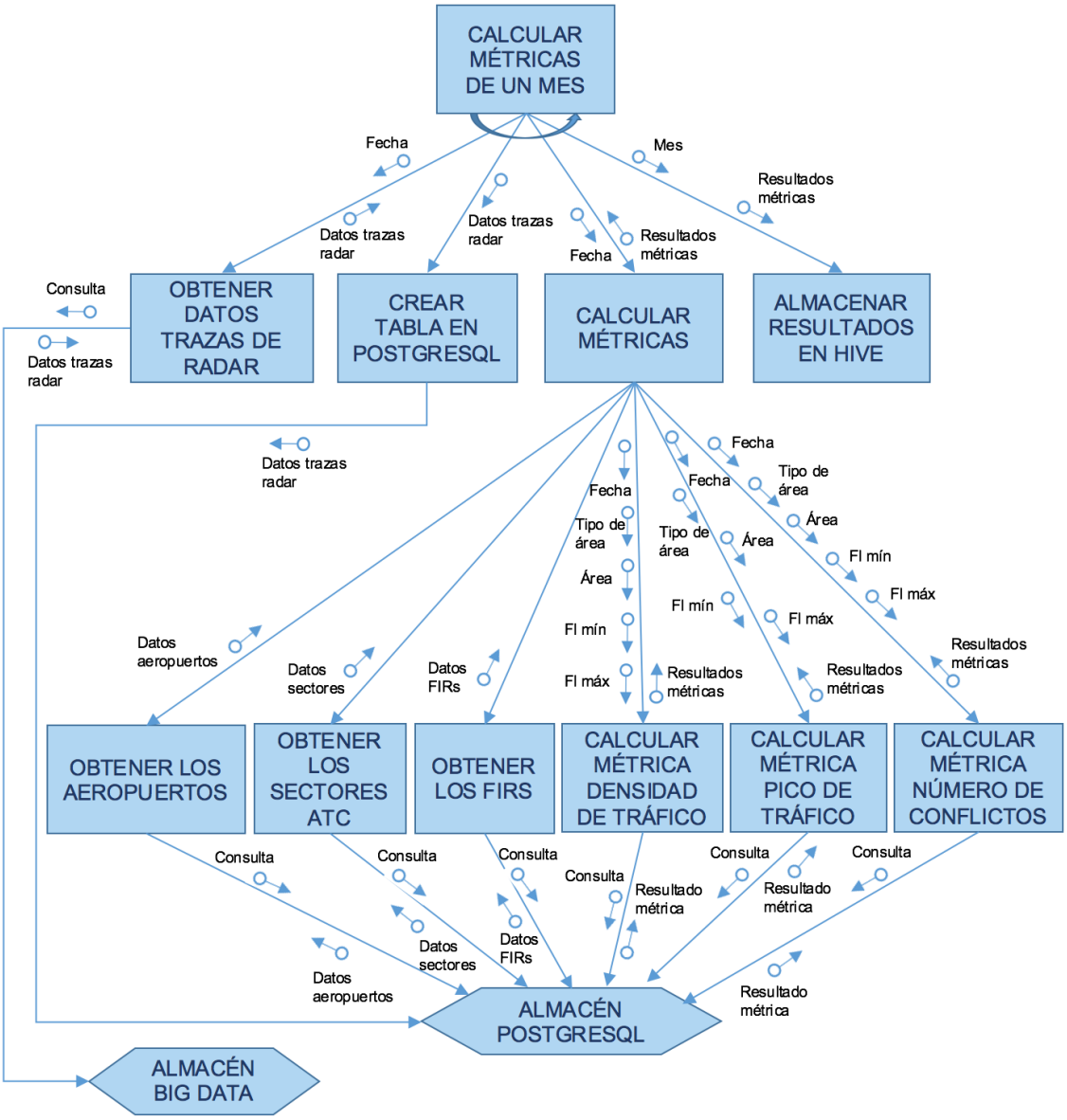


Ilustración 25. Diagrama de estructura (segunda parte)

La notación utilizada en el diagrama de estructura es la siguiente:

Descripción	Símbolo
Módulo	
Conexión	
Almacén de datos	
Parámetro de datos	

Tabla 32. Notación del diagrama de estructura

2.4. Flujo de trabajo (workflow)

En el siguiente diagrama o flujo de trabajo se representan de forma visual los ficheros del sistema y cómo interaccionan entre ellos y los distintos almacenes de datos. Este diagrama se basa en el diagrama de flujo de datos, agrupando algunos de los procesos en el mismo script, como ocurre con los procesos “Cargar datos radar”, “Guardar resultados” y “Calcular métricas de un mes”, que se agrupan todos ellos en un mismo script, denominado “Process_Radar_Tracks”.

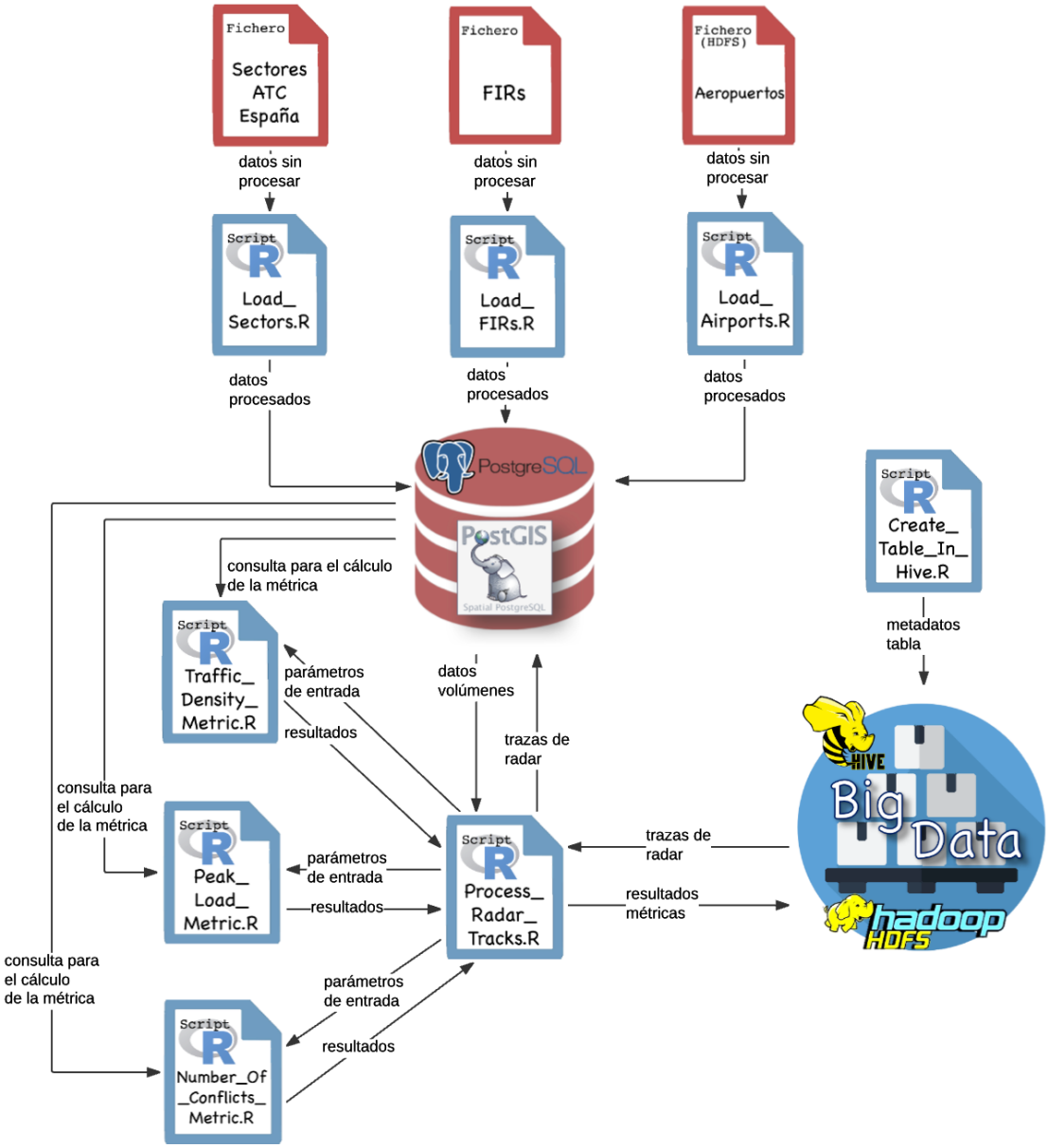


Ilustración 26. Flujo de trabajo

3. Implementación

La implementación se ha realizado en diversos scripts escritos en el lenguaje R. En este apartado se reflejan algunos de los aspectos más destacados de los scripts.

Librerías

Las librerías que se utilizan en los scripts son la librería RPostgreSQL y la librería SparkR:

```
library(RPostgreSQL)

Sys.setenv(SPARK_HOME = "/opt/cloudera/parcels/SPARK2/lib/spark2")
library(SparkR, lib.loc = c(file.path(Sys.getenv("SPARK_HOME"), "R", "lib")))
```

Ilustración 27. Código para importar las librerías

Inicialización de la sesión de Spark

Para trabajar con Spark es necesario inicializar la sesión de Spark, donde se crean el resto de variables que maneja Spark. En el siguiente fragmento se inicia la sesión de Spark, con el parámetro master = "yarn", para que utilice YARN como gestor de los recursos del clúster, utilizando así las capacidades de computación de todos los nodos del clúster. También se configuran otros parámetros relacionados con Hive y la conexión con PostgreSQL a través del conector jdbc.

```
# Initialize a Spark Session.
SparkR::sparkR.session(master = "yarn",
                       enableHiveSupport = TRUE,
                       sparkConfig=list(spark.driver.memory = "8g",
                                        spark.driver.extraClassPath = "postgresql-9.3-1104.jdbc41.jar",
                                        spark.sql.warehouse.dir="/user/hive/warehouse",
                                        hive.metastore.uris="thrift://filisteo5:9083"
                       ))
```

Ilustración 28. Código para inicializar la sesión de Spark

Parar la sesión de Spark

Después de terminar de trabajar con Spark, se para la sesión de Spark.

```
# Stop the Spark Session.
SparkR::sparkR.session.stop()
```

Ilustración 29. Código para parar la sesión de Spark

Conexión con PostgreSQL

Es necesario crear una conexión con PostgreSQL, indicando el host con el que se va a realizar la conexión, el puerto, usuario, contraseña y nombre de la base de datos.

```
# Connection with PostgreSQL.
con <- RPostgreSQL::dbConnect(RPostgreSQL::PostgreSQL(), host = "job",
                              port = "5432", user= "palvarez",
                              password = "****", dbname = "adsb")
```

Ilustración 30. Código para conectarse con PostgreSQL

Desconexión de PostgreSQL

Después de terminar de trabajar con PostgreSQL hay que desconectarse de la base de datos.

```
# PostgreSQL disconnection.
RPostgreSQL::dbDisconnect(con)
```

Ilustración 31. Código para desconectarse de PostgreSQL

Vectorización en R

Una de las características del lenguaje R es el uso de la vectorización, que permite realizar cálculos sobre un vector completo en vez de sobre cada uno de sus componentes secuencialmente. Supone una mejora respecto al uso de bucles. En el siguiente fragmento se utiliza la función vectorial mapply.

```
## Calculate the metric for airports.
results_airports_traffic_density <- unname(mapply(densityTrafficMetric, t0=grid_airports$hours, aType="circle",
                                  aParam=grid_airports$points))
```

Ilustración 32. Código que utiliza la vectorización en R

Métrica densidad de tráfico

Código que crea la consulta para calcular la métrica densidad de tráfico:

```
#Construct a query with all the input params.
query <- paste0("SELECT DISTINCT count(callsign)/double precision '", refTimeInt,"' AS TrafficDensity FROM (")
query <- paste0(query, "SELECT callsign, adep, ades FROM public.test_radar_tracks AS rt ")
query <- paste0(query, "WHERE rt.time BETWEEN '", t0, "' AND (timestamp '", t0, "' ")
query <- paste0(query, "+ double precision '",refTimeInt,"' * interval '1 hour') AND ")
query <- paste0(query, "rt.alt BETWEEN ", flMin, " AND ", flMax, " AND ")
query <- paste0(query, ifelse(outer,"NOT ", ""), "ST_CoveredBy(rt.point, ", refAreaQuery, ") ")
query <- paste0(query, "GROUP BY callsign, adep, ades) AS dt;")
```

Ilustración 33. Código de la consulta de la métrica densidad de tráfico

Métrica pico de tráfico

Código que crea la consulta para calcular la métrica pico de tráfico:

```
#Construct a query with all the input params.
query <- paste0("SELECT max(N_t) AS PeakLoad FROM (")
query <- paste0(query, "SELECT time, count(callsign) AS N_t FROM public.test_radar_tracks AS rt ")
query <- paste0(query, "WHERE rt.time BETWEEN '", t0, "' AND (timestamp '", t0, "' ")
query <- paste0(query, "+ double precision '",refTimeInt,"' * interval '1 hour') AND ")
query <- paste0(query, "rt.alt BETWEEN ", flMin, " AND ", flMax, " AND ")
query <- paste0(query, ifelse(outer,"NOT ", ""), "ST_CoveredBy(rt.point, ", refAreaQuery, ") ")
query <- paste0(query, "GROUP BY time) AS pl;")
```

Ilustración 34. Código de la consulta de la métrica pico de tráfico

Métrica número de conflictos

Código que crea la consulta para calcular la métrica número de conflictos:

```
#Construct a query with all the input params.
query <- paste0("SELECT count(*) AS NumberOfConflicts FROM (")
query <- paste0(query, "SELECT DISTINCT rt1.callsign, rt2.callsign FROM (")
query <- paste0(query, "SELECT * FROM public.test_radar_tracks AS rt ")
query <- paste0(query, "WHERE rt.time BETWEEN '", t0, "' AND (timestamp '", t0, "' ")
query <- paste0(query, "+ double precision '", refTimeInt,"' * interval '1 hour')) AS rt1 ")
query <- paste0(query, "JOIN (SELECT * FROM public.test_radar_tracks AS rt ")
query <- paste0(query, "WHERE rt.time BETWEEN '", t0, "' AND (timestamp '", t0, "' ")
query <- paste0(query, "+ double precision '", refTimeInt,"' * interval '1 hour')) AS rt2 ")
query <- paste0(query, "ON rt1.callsign > rt2.callsign ")
query <- paste0(query, "WHERE ST_distance(ST_Force_2D(rt1.point4D)::geography, ")
query <- paste0(query, "ST_Force_2D(rt2.point4D)::geography) <= ", lsc*1852.0, " ")
query <- paste0(query, "AND rt1.time = rt2.time ")
query <- paste0(query, "AND rt1.alt BETWEEN ", flMin, " AND ", flMax, " ")
query <- paste0(query, "AND rt2.alt BETWEEN ", flMin, " AND ", flMax, " ")
query <- paste0(query, "AND ST_CoveredBy(rt1.point, ", refAreaQuery, ") ")
query <- paste0(query, "AND ST_CoveredBy(rt2.point, ", refAreaQuery, ") ")
query <- paste0(query, "AND abs(ST_Z(rt1.point4D)-ST_Z(rt2.point4D))::double precision/0.3048 <= ", vsc, " ")
query <- paste(query, ") AS nc;")
```

Ilustración 35. Código de la consulta de la métrica número de conflictos

Tabla “test_airports”

Creación de la tabla “test_airports” en PostgreSQL, incluyendo una nueva columna y añadiendo índices:

```
RPostgreSQL::dbGetQuery(con, statement="DROP TABLE IF EXISTS public.test_airports CASCADE;")

# Write the data in PostgreSQL.
SparkR::write.jdbc(sdf_json,
                   url = "jdbc:postgresql://job:5432/adsb?user=palvarez&password=af0Kaqg",
                   tableName = "public.test_airports")

# Create a new column called point.
RPostgreSQL::dbGetQuery(con,"ALTER TABLE public.test_airports ADD COLUMN point geography(POINT, 4326);
                        UPDATE public.test_airports
                        SET point = ST_MakePoint(lng,lat)::geography;")

# Create indexes.
RPostgreSQL::dbGetQuery(con,"CREATE INDEX idx_callsign_air ON public.test_airports (country_code);" )
RPostgreSQL::dbGetQuery(con,"CREATE INDEX idx_icao_air ON public.test_airports (icao);" )
RPostgreSQL::dbGetQuery(con,"CREATE INDEX idx_point_air ON public.test_airports USING gist(point);" )
```

Ilustración 36. Código para crear la tabla “test_airports”

Tabla “test_atc_sectors_spain”

Creación de la tabla “test_atc_sectors_spain” en PostgreSQL, añadiendo índices:

```
RPostgreSQL::dbGetQuery(con, statement="DROP TABLE IF EXISTS public.test_atc_sectors_spain;")

# Create the table.
RPostgreSQL::dbGetQuery(con, statement="CREATE TABLE public.test_atc_sectors_spain (
                                        sector_name TEXT,
                                        alt_min NUMERIC,
                                        alt_max NUMERIC,
                                        sector_polygon GEOGRAPHY(Polygon, 4326)
                                       );")

# Execute the query.
RPostgreSQL::dbSendQuery(con, statement=query)

# Create indexes.
RPostgreSQL::dbGetQuery(con,"CREATE INDEX idx_sector_name_atc ON public.test_atc_sectors_spain (sector_name);" )
RPostgreSQL::dbGetQuery(con,"CREATE INDEX idx_alt_min_atc ON public.test_atc_sectors_spain (alt_min);" )
RPostgreSQL::dbGetQuery(con,"CREATE INDEX idx_alt_max_atc ON public.test_atc_sectors_spain (alt_max);" )
RPostgreSQL::dbGetQuery(con,"CREATE INDEX idx_sector_polygon_atc ON public.test_atc_sectors_spain USING gist(sector_polygon);" )
```

Ilustración 37. Código para crear la tabla “test_atc_sectors_spain”

Tabla “test_firs”

Creación de la tabla “test_firs” en PostgreSQL, añadiendo índices:

```
RPostgreSQL::dbGetQuery(con, statement="DROP TABLE IF EXISTS public.test_firs;")

# Create the table.
RPostgreSQL::dbGetQuery(con, statement="CREATE TABLE public.test_firs (
                        fir_name TEXT,
                        alt_min NUMERIC,
                        alt_max NUMERIC,
                        fir_polygon GEOGRAPHY(Polygon, 4326)
                        );")

# Execute the query.
RPostgreSQL::dbSendQuery(con, statement=query)

# Create indexes.
RPostgreSQL::dbGetQuery(con,"CREATE INDEX idx_fir_name_firs ON public.test_firs (fir_name);" )
RPostgreSQL::dbGetQuery(con,"CREATE INDEX idx_alt_min_firs ON public.test_firs (alt_min);" )
RPostgreSQL::dbGetQuery(con,"CREATE INDEX idx_alt_max_firs ON public.test_firs (alt_max);" )
RPostgreSQL::dbGetQuery(con,"CREATE INDEX idx_fir_polygon_firs ON public.test_firs USING gist(fir_polygon);" )
```

Ilustración 38. Código para crear la tabla “test_firs”

Tabla “metrics_results”

Creación de la tabla “metrics_results” en la base de datos “airports” en Hive:

```
SparkR::sql("DROP TABLE IF EXISTS airports.metrics_results")

# Create the table.
SparkR::sql("CREATE EXTERNAL TABLE airports.metrics_results (
                    type_of_metric STRING,
                    initial_time STRING,
                    ref_time_int FLOAT,
                    area_type STRING,
                    area_name STRING,
                    radius FLOAT,
                    flight_level_min FLOAT,
                    flight_level_max FLOAT,
                    result FLOAT
                )
                PARTITIONED BY (
                    `date` STRING
                )
                ROW FORMAT DELIMITED
                FIELDS TERMINATED BY ','
                STORED AS TEXTFILE
                LOCATION '/user/fdiaz/metrics'")
```

Ilustración 39. Código para crear la tabla “metrics_results”

Tabla “test_radar_tracks”

Creación de la tabla “test_radar_tracks” en PostgreSQL, incluyendo una nueva columna y añadiendo índices:

```
RPostgreSQL::dbGetQuery(con, statement="DROP TABLE IF EXISTS public.test_radar_tracks CASCADE;")

# Write the data in PostgreSQL.
SparkR::write.jdbc(sdf,
                   url = "jdbc:postgresql://job:5432/adsb?user=palvarez&password=af0Kaqg",
                   tableName = "public.test_radar_tracks")

# Add column "time".
RPostgreSQL::dbGetQuery(con,"ALTER TABLE public.test_radar_tracks ADD COLUMN time TIMESTAMP;
                        UPDATE public.test_radar_tracks
                        SET time = to_timestamp(time_s,'YYYY-MM-DD HH24:MI:SS');")

# Alter column "time".
RPostgreSQL::dbGetQuery(con,"ALTER TABLE public.test_radar_tracks ALTER COLUMN time TYPE TIMESTAMP WITHOUT TIME ZONE;")

# Add column "point".
RPostgreSQL::dbGetQuery(con,"ALTER TABLE public.test_radar_tracks ADD COLUMN point geography(POINT, 4326);
                        UPDATE public.test_radar_tracks
                        SET point = ST_MakePoint(lng,lat)::geography;")

# Add column "point4D".
RPostgreSQL::dbGetQuery(con,"ALTER TABLE public.test_radar_tracks ADD COLUMN point4D geometry(POINTZM, 4326);
                        UPDATE public.test_radar_tracks
                        SET point4D = ST_SetSRID(ST_MakePoint(lng,lat,30.48*alt,extract(epoch from time)),4326);"
) # 30.48 to convert FL --> feet --> meters

# Create indexes.
RPostgreSQL::dbGetQuery(con,"CREATE INDEX idx_callsign_rt ON public.test_radar_tracks (callsign);" )
RPostgreSQL::dbGetQuery(con,"CREATE INDEX idx_alt_rt ON public.test_radar_tracks (alt);" )
RPostgreSQL::dbGetQuery(con,"CREATE INDEX idx_time_rt ON public.test_radar_tracks (time);" )
RPostgreSQL::dbGetQuery(con,"CREATE INDEX idx_points_rt ON public.test_radar_tracks USING gist(point);" )
RPostgreSQL::dbGetQuery(con,"CREATE INDEX idx_points4D_rt ON public.test_radar_tracks USING gist(point4D);" )
```

Ilustración 40. Código para crear la tabla “test_radar_tracks”

4. Pruebas

4.1. Pruebas de caja blanca

Las pruebas de caja blanca (también llamadas pruebas estructurales) se centran en los detalles procedimentales del software, por lo que su diseño está fuertemente ligado al código fuente. A continuación se muestran las pruebas de caja blanca realizadas:

PCB-01: Métrica densidad de tráfico	**Descripción**
`SELECT DISTINCT count(callsign)/double precision '<$refTimeInt>' AS TrafficDensity FROM (`	Se calcula el número medio de aeronaves
`SELECT callsign, adep, ades FROM public.test_radar_tracks AS rt`	Se obtienen los datos de las trazas de radar
`WHERE rt.time BETWEEN '<$t0>' AND (timestamp '<$t0>' + double precision '<$refTimeInt>' * interval '1 hour')`	Intervalo de tiempo de referencia
`AND rt.alt BETWEEN <$flMin> AND <$flMax>`	Altura máxima y mínima del volumen de referencia
`AND <$outer> ST_CoveredBy(rt.point, ( SELECT ST_Buffer(ST_GeographyFromText('SRID =4326;<$aParam>'),<$radius>*1852.0)) )`	Área de referencia
`GROUP BY callsign, adep, ades) AS dt;`	Se agrupa por trayectorias

Tabla 33. PCB-01

PCB-02: Métrica pico de tráfico	**Descripción**
`SELECT max(N_t) AS PeakLoad FROM (`	Se calcula el número máximo de aeronaves
`SELECT time, count(callsign) AS N_t FROM public.test_radar_tracks AS rt`	Se cuenta el número de aeronaves. Se obtienen los datos de las trazas de radar
`WHERE rt.time BETWEEN '<$t0>' AND (timestamp '<$t0>' + double precision '<$refTimeInt>' * interval '1 hour')`	Intervalo de tiempo de referencia
`AND rt.alt BETWEEN <$flMin> AND <$flMax>`	Altura máxima y mínima del volumen de referencia
`AND <$outer> ST_CoveredBy(rt.point, ( SELECT`	Área de referencia

ST_Buffer(ST_GeographyFromText('SRID=4326;<$aParam>'),<$radius>*1852.0)))	
GROUP BY time) AS pl;	Se agrupan por instantes de tiempo

Tabla 34. PCB-02

PCB-03: Métrica número de conflictos	**Descripción**
SELECT count(*) AS NumberOfConflicts FROM (	Se cuenta el número de conflictos
SELECT DISTINCT rt1.callsign, rt2.callsign FROM (	Se seleccionan las filas resultantes, teniendo en cuenta que, aunque se violen los criterios de separación entre dos aeronaves en varios instantes de tiempo, sólo se contabilizará como 1 conflicto
SELECT * FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '<$t0>' AND (timestamp '<$t0>' + double precision '<$refTimeInt>' * interval '1 hour')) AS rt1 JOIN (SELECT * FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '<$t0>' AND (timestamp '<$t0>' + double precision '<$refTimeInt>' * interval '1 hour')) AS rt2	Se realiza un JOIN entre los datos obtenidos a través de una subconsulta de las trazas de radar y otra subconsulta idéntica con los mismos datos, para obtener todas las combinaciones (producto cartesiano). En la subconsulta de las trazas de radar ya se han seleccionado sólo los datos que están dentro del intervalo de tiempo de referencia
ON rt1.callsign > rt2.callsign	Se realiza el JOIN según este criterio para que evitar las combinaciones de una fila de una subconsulta con la misma fila de la otra subconsulta
WHERE ST_distance(ST_Force_2D(rt1.point4D)::geography, ST_Force_2D(rt2.point4D)::geography) <= <$lsc>*1852.0	Criterio de separación lateral
AND rt1.time = rt2.time	Para detectar conflictos, debe

	coincidir el instante de tiempo
`AND rt1.alt BETWEEN <$flMin> AND <$flMax>`	Altura máxima y mínima del volumen de referencia a aplicar sobre la primera subconsulta
`AND rt2.alt BETWEEN <$flMin> AND <$flMax>`	Altura máxima y mínima del volumen de referencia a aplicar sobre la segunda subconsulta
`AND <$outer> ST_CoveredBy(rt1.point, ( SELECT ST_Buffer(ST_GeographyFromText('SRID=4326;<$aParam>'),<$radius>*1852.0)))`	Área de referencia a aplicar sobre la primera subconsulta
`AND <$outer> ST_CoveredBy(rt2.point, ( SELECT ST_Buffer(ST_GeographyFromText('SRID=4326;<$aParam>'),<$radius>*1852.0)))`	Área de referencia a aplicar sobre la segunda subconsulta
`AND abs(ST_Z(rt1.point4D)-ST_Z(rt2.point4D))::double precision/0.3048 <= <$vsc> ) AS nc;`	Criterio de separación vertical

Tabla 35. PCB-03

4.2. Pruebas de caja negra

Las pruebas de caja negra pretenden probar el funcionamiento de las funciones, proporcionando unas entradas y estudiando si las salidas obtenidas concuerdan con las esperadas.

En este proyecto no se ha contado con resultados de las métricas fiables ni de prueba para verificar el funcionamiento de las funciones que calculan las métricas, por lo que se han realizado las pruebas de caja negra comparando las consultas generadas por las métricas con las consultas esperadas según los parámetros de entrada.

PCN-01	**Prueba 1 métrica densidad de tráfico**
Datos de entrada	t0 = as.POSIXct(strptime("2016/05/06 10:00:00",format="%Y/%m/%d %H:%M:%OS",tz="UTC"))
	refTimeInt = 2.0
	aType = "circle"
	aParam = "POINT(-3.570209 40.46515)"
	radius = 110
	flMin = 5
	flMax = 500
	outer = TRUE
Consulta	`SELECT DISTINCT count(callsign)/double`

esperada	precision '2' AS TrafficDensity FROM (SELECT callsign, adep, ades FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 10:00:00' AND (timestamp '2016-05-06 10:00:00' + double precision '2' * interval '1 hour') AND rt.alt BETWEEN 5 AND 500 AND NOT ST_CoveredBy(rt.point, (SELECT ST_Buffer(ST_GeographyFromText('SRID=4326;POINT (-3.570209 40.46515)'),203720))) GROUP BY callsign, adep, ades) AS dt;
Consulta obtenida	SELECT DISTINCT count(callsign)/double precision '2' AS TrafficDensity FROM (SELECT callsign, adep, ades FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 10:00:00' AND (timestamp '2016-05-06 10:00:00' + double precision '2' * interval '1 hour') AND rt.alt BETWEEN 5 AND 500 AND NOT ST_CoveredBy(rt.point, (SELECT ST_Buffer(ST_GeographyFromText('SRID=4326;POINT (-3.570209 40.46515)'),203720))) GROUP BY callsign, adep, ades) AS dt;
Resultado de la prueba	Correcto

Tabla 36. PCN-01

PCN-02	**Prueba 2 métrica densidad de tráfico**
Datos de entrada	t0 = as.POSIXct(strptime("2016/05/06 07:00:00",format="%Y/%m/%d %H:%M:%OS",tz="UTC"))
	aType = "circle"
	aParam = "POINT (-1.863333 38.948334)"
Consulta esperada	SELECT DISTINCT count(callsign)/double precision '1' AS TrafficDensity FROM (SELECT callsign, adep, ades FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 07:00:00' AND (timestamp '2016-05-06 07:00:00' + double precision '1' * interval '1 hour') AND rt.alt BETWEEN 0 AND 510 AND ST_CoveredBy(rt.point, (SELECT ST_Buffer(ST_GeographyFromText('SRID=4326;POINT (-1.863333 38.948334)'),185200))) GROUP BY callsign, adep, ades) AS dt;
Consulta obtenida	SELECT DISTINCT count(callsign)/double precision '1' AS TrafficDensity FROM (SELECT callsign, adep, ades FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 07:00:00' AND (timestamp '2016-05-06 07:00:00' + double precision '1' * interval '1 hour') AND rt.alt BETWEEN 0 AND 510

	AND ST_CoveredBy(rt.point, (SELECT ST_Buffer(ST_GeographyFromText('SRID=4326;POINT (-1.863333 38.948334)'),185200))) GROUP BY callsign, adep, ades) AS dt;
Resultado de la prueba	Correcto

Tabla 37. PCN-02

PCN-03	**Prueba 3 métrica densidad de tráfico**
Datos de entrada	t0 = as.POSIXct(strptime("2016/05/06 10:00:00",format="%Y/%m/%d %H:%M:%OS",tz="UTC"))
	refTimeInt = 2.0
	aType = "polygon"
	aParam = "POLYGON((-13.7 29.6166666667,-13.6666666667 29.5277777778,-12.9491666667 29.5277777778, -13.325 28.6725,-13.8333333333 27.8333333333,-14.6666666667 27.8333333333,-14.6666666667 28.0083333333, -14.9166666667 28.65,-13.7602777778 29.5277777778,-13.7 29.6166666667))”
	flMin = 5
	flMax = 125
	outer = FALSE
Consulta esperada	SELECT DISTINCT count(callsign)/double precision '2' AS TrafficDensity FROM (SELECT callsign, adep, ades FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 10:00:00' AND (timestamp '2016-05-06 10:00:00' + double precision '2' * interval '1 hour') AND rt.alt BETWEEN 5 AND 125 AND ST_CoveredBy(rt.point, (SELECT ST_GeographyFromText('SRID=4326;POLYGON((-13.7 29.6166666667,-13.6666666667 29.5277777778,-12.9491666667 29.5277777778, -13.325 28.6725,-13.8333333333 27.8333333333,-14.6666666667 27.8333333333,-14.6666666667 28.0083333333, -14.9166666667 28.65,-13.7602777778 29.5277777778,-13.7 29.6166666667))'))) GROUP BY callsign, adep, ades) AS dt;
Consulta obtenida	SELECT DISTINCT count(callsign)/double precision '2' AS TrafficDensity FROM (SELECT callsign, adep, ades FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 10:00:00' AND

	(timestamp '2016-05-06 10:00:00' + double precision '2' * interval '1 hour') AND rt.alt BETWEEN 5 AND 125 AND ST_CoveredBy(rt.point, (SELECT ST_GeographyFromText('SRID=4326;POLYGON((-13.7 29.6166666667,-13.6666666667 29.5277777778,-12.9491666667 29.5277777778, -13.325 28.6725,-13.8333333333 27.8333333333,-14.6666666667 27.8333333333,-14.6666666667 28.0083333333, -14.9166666667 28.65,-13.7602777778 29.5277777778,-13.7 29.6166666667))'))) GROUP BY callsign, adep, ades) AS dt;
Resultado de la prueba	Correcto

Tabla 38. PCN-03

PCN-04	**Prueba 4 métrica densidad de tráfico**
Datos de entrada	t0 = as.POSIXct(strptime("2016/05/06 07:00:00",format="%Y/%m/%d %H:%M:%OS",tz="UTC"))
	aType = "polygon"
	aParam = "POLYGON((-13.7 29.6166666667,-13.6666666667 29.5277777778,-12.9491666667 29.5277777778, -13.325 28.6725,-13.8333333333 27.8333333333,-14.6666666667 27.8333333333,-14.6666666667 28.0083333333, -14.9166666667 28.65,-13.7602777778 29.5277777778,-13.7 29.6166666667))"
Consulta esperada	SELECT DISTINCT count(callsign)/double precision '1' AS TrafficDensity FROM (SELECT callsign, adep, ades FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 07:00:00' AND (timestamp '2016-05-06 07:00:00' + double precision '1' * interval '1 hour') AND rt.alt BETWEEN 0 AND 510 AND ST_CoveredBy(rt.point, (SELECT ST_GeographyFromText('SRID=4326;POLYGON((-13.7 29.6166666667,-13.6666666667 29.5277777778,-12.9491666667 29.5277777778, -13.325 28.6725,-13.8333333333 27.8333333333,-14.6666666667 27.8333333333,-14.6666666667 28.0083333333, -14.9166666667 28.65,-13.7602777778 29.5277777778,-13.7 29.6166666667))'))) GROUP BY callsign, adep, ades) AS dt;
Consulta obtenida	SELECT DISTINCT count(callsign)/double

	precision '1' AS TrafficDensity FROM (SELECT callsign, adep, ades FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 07:00:00' AND (timestamp '2016-05-06 07:00:00' + double precision '1' * interval '1 hour') AND rt.alt BETWEEN 0 AND 510 AND ST_CoveredBy(rt.point, (SELECT ST_GeographyFromText('SRID=4326;POLYGON((-13.7 29.6166666667,-13.6666666667 29.5277777778,-12.9491666667 29.5277777778, -13.325 28.6725,-13.8333333333 27.8333333333,-14.6666666667 27.8333333333,-14.6666666667 28.0083333333, -14.9166666667 28.65,-13.7602777778 29.5277777778,-13.7 29.6166666667))'))) GROUP BY callsign, adep, ades) AS dt;
Resultado de la prueba	Correcto

Tabla 39. PCN-04

PCN-05	**Prueba 1 métrica pico de tráfico**
Datos de entrada	t0 = as.POSIXct(strptime("2016/05/06 10:00:00",format="%Y/%m/%d %H:%M:%OS",tz="UTC"))
	refTimeInt = 2.0
	aType = "circle"
	aParam = "POINT(-3.570209 40.46515)"
	radius = 110
	flMin = 5
	flMax = 500
	outer = TRUE
Consulta esperada	SELECT max(N_t) AS PeakLoad FROM (SELECT time, count(callsign) AS N_t FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 10:00:00' AND (timestamp '2016-05-06 10:00:00' + double precision '2' * interval '1 hour') AND rt.alt BETWEEN 5 AND 500 AND NOT ST_CoveredBy(rt.point, (SELECT ST_Buffer(ST_GeographyFromText('SRID=4326;POINT(-3.570209 40.46515)'),203720))) GROUP BY time) AS pl;
Consulta obtenida	SELECT max(N_t) AS PeakLoad FROM (SELECT time, count(callsign) AS N_t FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 10:00:00' AND (timestamp '2016-05-06 10:00:00' + double

	precision '2' * interval '1 hour') AND rt.alt BETWEEN 5 AND 500 AND NOT ST_CoveredBy(rt.point, (SELECT ST_Buffer(ST_GeographyFromText('SRID=4326; POINT(-3.570209 40.46515)'),203720))) GROUP BY time) AS pl;
Resultado de la prueba	Correcto

Tabla 40. PCN-05

PCN-06	**Prueba 2 métrica pico de tráfico**
Datos de entrada	t0 = as.POSIXct(strptime("2016/05/06 07:00:00",format="%Y/%m/%d %H:%M:%OS",tz="UTC"))
	aType = "circle"
	aParam = "POINT (-1.863333 38.948334)"
Consulta esperada	SELECT max(N_t) AS PeakLoad FROM (SELECT time, count(callsign) AS N_t FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 07:00:00' AND (timestamp '2016-05-06 07:00:00' + double precision '1' * interval '1 hour') AND rt.alt BETWEEN 0 AND 510 AND ST_CoveredBy(rt.point, (SELECT ST_Buffer(ST_GeographyFromText('SRID=4326;POINT (-1.863333 38.948334)'),185200))) GROUP BY time) AS pl;
Consulta obtenida	SELECT max(N_t) AS PeakLoad FROM (SELECT time, count(callsign) AS N_t FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 07:00:00' AND (timestamp '2016-05-06 07:00:00' + double precision '1' * interval '1 hour') AND rt.alt BETWEEN 0 AND 510 AND ST_CoveredBy(rt.point, (SELECT ST_Buffer(ST_GeographyFromText('SRID=4326;POINT (-1.863333 38.948334)'),185200))) GROUP BY time) AS pl;
Resultado de la prueba	Correcto

Tabla 41. PCN-06

PCN-07	**Prueba 3 métrica pico de tráfico**
Datos de entrada	t0 = as.POSIXct(strptime("2016/05/06 10:00:00",format="%Y/%m/%d %H:%M:%OS",tz="UTC"))

	refTimeInt = 2.0
	aType = "polygon"
	aParam = "POLYGON((-13.7 29.6166666667,-13.6666666667 29.5277777778,-12.9491666667 29.5277777778, -13.325 28.6725,-13.8333333333 27.8333333333,-14.6666666667 27.8333333333,-14.6666666667 28.0083333333, -14.9166666667 28.65,-13.7602777778 29.5277777778,-13.7 29.6166666667))"
	flMin = 5
	flMax = 125
	outer = FALSE
Consulta esperada	`SELECT max(N_t) AS PeakLoad FROM (SELECT time, count(callsign) AS N_t FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 10:00:00' AND (timestamp '2016-05-06 10:00:00' + double precision '2' * interval '1 hour') AND rt.alt BETWEEN 5 AND 125 AND ST_CoveredBy(rt.point, (SELECT ST_GeographyFromText('SRID=4326;POLYGON((-13.7 29.6166666667,-13.6666666667 29.5277777778,-12.9491666667 29.5277777778,-13.325 28.6725,-13.8333333333 27.8333333333,-14.6666666667 27.8333333333,-14.6666666667 28.0083333333,-14.9166666667 28.65,-13.7602777778 29.5277777778,-13.7 29.6166666667))'))) GROUP BY time) AS pl;`
Consulta obtenida	`SELECT max(N_t) AS PeakLoad FROM (SELECT time, count(callsign) AS N_t FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 10:00:00' AND (timestamp '2016-05-06 10:00:00' + double precision '2' * interval '1 hour') AND rt.alt BETWEEN 5 AND 125 AND ST_CoveredBy(rt.point, (SELECT ST_GeographyFromText('SRID=4326;POLYGON((-13.7 29.6166666667,-13.6666666667 29.5277777778,-12.9491666667 29.5277777778,-13.325 28.6725,-13.8333333333 27.8333333333,-14.6666666667 27.8333333333,-14.6666666667 28.0083333333,-14.9166666667 28.65,-13.7602777778 29.5277777778,-13.7 29.6166666667))'))) GROUP BY time) AS pl;`
Resultado de la prueba	Correcto

Tabla 42. PCN-07

PCN-08	Prueba 4 métrica pico de tráfico
Datos de entrada	t0 = as.POSIXct(strptime("2016/05/06 07:00:00",format="%Y/%m/%d %H:%M:%OS",tz="UTC"))
	aType = "polygon"
	aParam = “POLYGON((-13.7 29.6166666667,-13.6666666667 29.5277777778,-12.9491666667 29.5277777778, -13.325 28.6725,-13.8333333333 27.8333333333,-14.6666666667 27.8333333333,-14.6666666667 28.0083333333, -14.9166666667 28.65,-13.7602777778 29.5277777778,-13.7 29.6166666667))”
Consulta esperada	`SELECT max(N_t) AS PeakLoad FROM (SELECT time, count(callsign) AS N_t FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 07:00:00' AND (timestamp '2016-05-06 07:00:00' + double precision '1' * interval '1 hour') AND rt.alt BETWEEN 0 AND 510 AND ST_CoveredBy(rt.point, (SELECT ST_GeographyFromText('SRID=4326;POLYGON((-13.7 29.6166666667,-13.6666666667 29.5277777778,-12.9491666667 29.5277777778,-13.325 28.6725,-13.8333333333 27.8333333333,-14.6666666667 27.8333333333,-14.6666666667 28.0083333333,-14.9166666667 28.65,-13.7602777778 29.5277777778,-13.7 29.6166666667))'))) GROUP BY time) AS pl;`
Consulta obtenida	`SELECT max(N_t) AS PeakLoad FROM (SELECT time, count(callsign) AS N_t FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 07:00:00' AND (timestamp '2016-05-06 07:00:00' + double precision '1' * interval '1 hour') AND rt.alt BETWEEN 0 AND 510 AND ST_CoveredBy(rt.point, (SELECT ST_GeographyFromText('SRID=4326;POLYGON((-13.7 29.6166666667,-13.6666666667 29.5277777778,-12.9491666667 29.5277777778,-13.325 28.6725,-13.8333333333 27.8333333333,-14.6666666667 27.8333333333,-14.6666666667 28.0083333333,-14.9166666667 28.65,-13.7602777778 29.5277777778,-13.7 29.6166666667))'))) GROUP BY time) AS pl;`
Resultado de la prueba	Correcto

Tabla 43. PCN-08

PCN-09	Prueba 1 métrica número de conflictos
Datos de entrada	t0 = as.POSIXct(strptime("2016/05/06 10:00:00",format="%Y/%m/%d %H:%M:%OS",tz="UTC"))
	refTimeInt = 2.0
	aType = "circle"
	aParam = "POINT(-3.570209 40.46515)"
	radius = 110
	flMin = 5
	flMax = 500
	outer = TRUE
	lsc = 10
	vsc = 500
Consulta esperada	`SELECT count(*) AS NumberOfConflicts FROM (SELECT DISTINCT rt1.callsign, rt2.callsign FROM (SELECT * FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 10:00:00' AND (timestamp '2016-05-06 10:00:00' + double precision '2' * interval '1 hour')) AS rt1 JOIN (SELECT * FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 10:00:00' AND (timestamp '2016-05-06 10:00:00' + double precision '2' * interval '1 hour')) AS rt2 ON rt1.callsign > rt2.callsign WHERE ST_distance(ST_Force_2D(rt1.point4D)::geography, ST_Force_2D(rt2.point4D)::geography) <= 18520 AND rt1.time = rt2.time AND rt1.alt BETWEEN 5 AND 500 AND rt2.alt BETWEEN 5 AND 500 AND NOT ST_CoveredBy(rt1.point, (SELECT ST_Buffer(ST_GeographyFromText('SRID=4326;POINT(-3.570209 40.46515)'),203720))) AND NOT ST_CoveredBy(rt2.point, (SELECT ST_Buffer(ST_GeographyFromText('SRID=4326;POINT(-3.570209 40.46515)'),203720))) AND abs(ST_Z(rt1.point4D)-ST_Z(rt2.point4D))::double precision/0.3048 <= 500) AS nc;`
Consulta obtenida	`SELECT count(*) AS NumberOfConflicts FROM (SELECT DISTINCT rt1.callsign, rt2.callsign FROM (SELECT * FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 10:00:00' AND (timestamp '2016-05-06 10:00:00' + double precision '2' * interval '1 hour')) AS rt1 JOIN (SELECT * FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 10:00:00' AND (timestamp '2016-05-06 10:00:00' + double precision '2' * interval '1 hour')) AS rt2 ON rt1.callsign > rt2.callsign WHERE ST_distance(ST_Force_2D(rt1.point4D)::geography, ST_Force_2D(rt2.point4D)::geography) <= 18520`

	AND rt1.time = rt2.time AND rt1.alt BETWEEN 5 AND 500 AND rt2.alt BETWEEN 5 AND 500 AND NOT ST_CoveredBy(rt1.point, (SELECT ST_Buffer(ST_GeographyFromText('SRID=4326;POINT (-3.570209 40.46515)'),203720))) AND NOT ST_CoveredBy(rt2.point, (SELECT ST_Buffer(ST_GeographyFromText('SRID=4326;POINT (-3.570209 40.46515)'),203720))) AND abs(ST_Z(rt1.point4D)-ST_Z(rt2.point4D))::double precision/0.3048 <= 500) AS nc;
Resultado de la prueba	Correcto

Tabla 44. PCN-09

PCN-10	**Prueba 2 métrica número de conflictos**
Datos de entrada	t0 = as.POSIXct(strptime("2016/05/06 07:00:00",format="%Y/%m/%d %H:%M:%OS",tz="UTC"))
	aType = "circle"
	aParam = "POINT (-1.863333 38.948334)"
Consulta esperada	SELECT count(*) AS NumberOfConflicts FROM (SELECT DISTINCT rt1.callsign, rt2.callsign FROM (SELECT * FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 07:00:00' AND (timestamp '2016-05-06 07:00:00' + double precision '1' * interval '1 hour')) AS rt1 JOIN (SELECT * FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 07:00:00' AND (timestamp '2016-05-06 07:00:00' + double precision '1' * interval '1 hour')) AS rt2 ON rt1.callsign > rt2.callsign WHERE ST_distance(ST_Force_2D(rt1.point4D)::geography, ST_Force_2D(rt2.point4D)::geography) <= 9260 AND rt1.time = rt2.time AND rt1.alt BETWEEN 0 AND 510 AND rt2.alt BETWEEN 0 AND 510 AND ST_CoveredBy(rt1.point, (SELECT ST_Buffer(ST_GeographyFromText('SRID=4326;POINT (-1.863333 38.948334)'),185200))) AND ST_CoveredBy(rt2.point, (SELECT ST_Buffer(ST_GeographyFromText('SRID=4326;POINT (-1.863333 38.948334)'),185200))) AND abs(ST_Z(rt1.point4D)-ST_Z(rt2.point4D))::double precision/0.3048 <= 1000) AS nc;
Consulta obtenida	SELECT count(*) AS NumberOfConflicts FROM (SELECT DISTINCT rt1.callsign, rt2.callsign FROM (SELECT * FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 07:00:00'

	AND (timestamp '2016-05-06 07:00:00' + double precision '1' * interval '1 hour')) AS rt1 JOIN (SELECT * FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 07:00:00' AND (timestamp '2016-05-06 07:00:00' + double precision '1' * interval '1 hour')) AS rt2 ON rt1.callsign > rt2.callsign WHERE ST_distance(ST_Force_2D(rt1.point4D)::geography, ST_Force_2D(rt2.point4D)::geography) <= 9260 AND rt1.time = rt2.time AND rt1.alt BETWEEN 0 AND 510 AND rt2.alt BETWEEN 0 AND 510 AND ST_CoveredBy(rt1.point, (SELECT ST_Buffer(ST_GeographyFromText('SRID=4326;POINT(-1.863333 38.948334)'),185200))) AND ST_CoveredBy(rt2.point, (SELECT ST_Buffer(ST_GeographyFromText('SRID=4326;POINT(-1.863333 38.948334)'),185200))) AND abs(ST_Z(rt1.point4D)-ST_Z(rt2.point4D))::double precision/0.3048 <= 1000) AS nc;
Resultado de la prueba	Correcto

Tabla 45. PCN-10

PCN-11	**Prueba 3 métrica número de conflictos**
Datos de entrada	t0 = as.POSIXct(strptime("2016/05/06 10:00:00",format="%Y/%m/%d %H:%M:%OS",tz="UTC"))
	refTimeInt = 2.0
	aType = "polygon"
	aParam = "POLYGON((-13.7 29.6166666667,-13.6666666667 29.5277777778,-12.9491666667 29.5277777778, -13.325 28.6725,-13.8333333333 27.8333333333,-14.6666666667 27.8333333333,-14.6666666667 28.0083333333, -14.9166666667 28.65,-13.7602777778 29.5277777778,-13.7 29.6166666667))"
	flMin = 5
	flMax = 125
	outer = FALSE
	lsc = 20
	vsc = 900
Consulta esperada	SELECT count(*) AS NumberOfConflicts FROM (SELECT DISTINCT rt1.callsign, rt2.callsign FROM (SELECT * FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 10:00:00' AND (timestamp '2016-05-06 10:00:00' + double precision '2' * interval '1 hour')) AS rt1 JOIN (SELECT * FROM public.test_radar_tracks AS rt

	WHERE rt.time BETWEEN '2016-05-06 10:00:00' AND (timestamp '2016-05-06 10:00:00' + double precision '2' * interval '1 hour')) AS rt2 ON rt1.callsign > rt2.callsign WHERE ST_distance(ST_Force_2D(rt1.point4D)::geography, ST_Force_2D(rt2.point4D)::geography) <= 37040 AND rt1.time = rt2.time AND rt1.alt BETWEEN 5 AND 125 AND rt2.alt BETWEEN 5 AND 125 AND ST_CoveredBy(rt1.point, (SELECT ST_GeographyFromText('SRID=4326;POLYGON((-13.7 29.6166666667,-13.6666666667 29.5277777778,-12.9491666667 29.5277777778,-13.325 28.6725,-13.8333333333 27.8333333333,-14.6666666667 27.8333333333,-14.6666666667 28.0083333333,-14.9166666667 28.65,-13.7602777778 29.5277777778,-13.7 29.6166666667))'))) AND ST_CoveredBy(rt2.point, (SELECT ST_GeographyFromText('SRID=4326;POLYGON((-13.7 29.6166666667,-13.6666666667 29.5277777778,-12.9491666667 29.5277777778,-13.325 28.6725,-13.8333333333 27.8333333333,-14.6666666667 27.8333333333,-14.6666666667 28.0083333333,-14.9166666667 28.65,-13.7602777778 29.5277777778,-13.7 29.6166666667))'))) AND abs(ST_Z(rt1.point4D)-ST_Z(rt2.point4D))::double precision/0.3048 <= 900) AS nc;
Consulta obtenida	SELECT count(*) AS NumberOfConflicts FROM (SELECT DISTINCT rt1.callsign, rt2.callsign FROM (SELECT * FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 10:00:00' AND (timestamp '2016-05-06 10:00:00' + double precision '2' * interval '1 hour')) AS rt1 JOIN (SELECT * FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 10:00:00' AND (timestamp '2016-05-06 10:00:00' + double precision '2' * interval '1 hour')) AS rt2 ON rt1.callsign > rt2.callsign WHERE ST_distance(ST_Force_2D(rt1.point4D)::geography, ST_Force_2D(rt2.point4D)::geography) <= 37040 AND rt1.time = rt2.time AND rt1.alt BETWEEN 5 AND 125 AND rt2.alt BETWEEN 5 AND 125 AND ST_CoveredBy(rt1.point, (SELECT ST_GeographyFromText('SRID=4326;POLYGON((-13.7 29.6166666667,-13.6666666667 29.5277777778,-12.9491666667 29.5277777778,-13.325 28.6725,-13.8333333333 27.8333333333,-14.6666666667 27.8333333333,-14.6666666667 28.0083333333,-14.9166666667 28.65,-13.7602777778 29.5277777778,-13.7 29.6166666667))'))) AND ST_CoveredBy(rt2.point, (SELECT

	ST_GeographyFromText('SRID=4326;POLYGON((-13.7 29.6166666667,-13.6666666667 29.5277777778,-12.9491666667 29.5277777778,-13.325 28.6725,-13.8333333333 27.8333333333,-14.6666666667 27.8333333333,-14.6666666667 28.0083333333,-14.9166666667 28.65,-13.7602777778 29.5277777778,-13.7 29.6166666667))'))) AND abs(ST_Z(rt1.point4D)-ST_Z(rt2.point4D))::double precision/0.3048 <= 900) AS nc;
Resultado de la prueba	Correcto

Tabla 46. PCN-11

PCN-12	**Prueba 4 métrica número de conflictos**
Datos de entrada	t0 = as.POSIXct(strptime("2016/05/06 07:00:00",format="%Y/%m/%d %H:%M:%OS",tz="UTC"))
	aType = "polygon"
	aParam = "POLYGON((-13.7 29.6166666667,-13.6666666667 29.5277777778,-12.9491666667 29.5277777778, -13.325 28.6725,-13.8333333333 27.8333333333,-14.6666666667 27.8333333333,-14.6666666667 28.0083333333, -14.9166666667 28.65,-13.7602777778 29.5277777778,-13.7 29.6166666667))"
Consulta esperada	SELECT count(*) AS NumberOfConflicts FROM (SELECT DISTINCT rt1.callsign, rt2.callsign FROM (SELECT * FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 07:00:00' AND (timestamp '2016-05-06 07:00:00' + double precision '1' * interval '1 hour')) AS rt1 JOIN (SELECT * FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 07:00:00' AND (timestamp '2016-05-06 07:00:00' + double precision '1' * interval '1 hour')) AS rt2 ON rt1.callsign > rt2.callsign WHERE ST_distance(ST_Force_2D(rt1.point4D)::geography, ST_Force_2D(rt2.point4D)::geography) <= 9260 AND rt1.time = rt2.time AND rt1.alt BETWEEN 0 AND 510 AND rt2.alt BETWEEN 0 AND 510 AND ST_CoveredBy(rt1.point, (SELECT ST_GeographyFromText('SRID=4326;POLYGON((-13.7 29.6166666667,-13.6666666667 29.5277777778,-12.9491666667 29.5277777778,-13.325 28.6725,-13.8333333333 27.8333333333,-14.6666666667 27.8333333333,-14.6666666667 28.0083333333,-14.9166666667 28.65,-13.7602777778 29.5277777778,-13.7 29.6166666667))'))) AND ST_CoveredBy(rt2.point, (SELECT ST_GeographyFromText('SRID=4326;POLYGON((-13.7

	29.6166666667,-13.6666666667 29.5277777778,-12.9491666667 29.5277777778,-13.325 28.6725,-13.8333333333 27.8333333333,-14.6666666667 27.8333333333,-14.6666666667 28.0083333333,-14.9166666667 28.65,-13.7602777778 29.5277777778,-13.7 29.6166666667))')) AND abs(ST_Z(rt1.point4D)-ST_Z(rt2.point4D))::double precision/0.3048 <= 1000) AS nc;
Consulta obtenida	SELECT count(*) AS NumberOfConflicts FROM (SELECT DISTINCT rt1.callsign, rt2.callsign FROM (SELECT * FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 07:00:00' AND (timestamp '2016-05-06 07:00:00' + double precision '1' * interval '1 hour')) AS rt1 JOIN (SELECT * FROM public.test_radar_tracks AS rt WHERE rt.time BETWEEN '2016-05-06 07:00:00' AND (timestamp '2016-05-06 07:00:00' + double precision '1' * interval '1 hour')) AS rt2 ON rt1.callsign > rt2.callsign WHERE ST_distance(ST_Force_2D(rt1.point4D)::geography, ST_Force_2D(rt2.point4D)::geography) <= 9260 AND rt1.time = rt2.time AND rt1.alt BETWEEN 0 AND 510 AND rt2.alt BETWEEN 0 AND 510 AND ST_CoveredBy(rt1.point, (SELECT ST_GeographyFromText('SRID=4326;POLYGON((-13.7 29.6166666667,-13.6666666667 29.5277777778,-12.9491666667 29.5277777778,-13.325 28.6725,-13.8333333333 27.8333333333,-14.6666666667 27.8333333333,-14.6666666667 28.0083333333,-14.9166666667 28.65,-13.7602777778 29.5277777778,-13.7 29.6166666667))')) AND ST_CoveredBy(rt2.point, (SELECT ST_GeographyFromText('SRID=4326;POLYGON((-13.7 29.6166666667,-13.6666666667 29.5277777778,-12.9491666667 29.5277777778,-13.325 28.6725,-13.8333333333 27.8333333333,-14.6666666667 27.8333333333,-14.6666666667 28.0083333333,-14.9166666667 28.65,-13.7602777778 29.5277777778,-13.7 29.6166666667))')) AND abs(ST_Z(rt1.point4D)-ST_Z(rt2.point4D))::double precision/0.3048 <= 1000) AS nc;
Resultado de la prueba	Correcto

Tabla 47. PCN-12

SECCIÓN III: MANUAL DE USUARIO

1. Manual de usuario

Debido a que el usuario final de este proyecto va a ser un programador, el manual de usuario está enfocado a definir los scripts y las funciones a modo de API (Interfaz de Programación de Aplicaciones - Application Programming Interface), explicando qué hace cada script y función, así como los parámetros de entrada y salidas de las funciones.

Script "Load_Airports.R"

Script para cargar un fichero JSON con datos de los aeropuertos desde HDFS a PostgreSQL.

Funciones:

loadJSON2DF( )
Carga el fichero JSON desde HDFS a un Spark Data Frame.
Salida de la función
Spark Data Frame con la información de los aeropuertos.

Tabla 48. Función loadJSON2DF( )

createTablePostGisFromDF(sdf_json)	
Crea una tabla en PostgreSQL y escribe los datos de los aeropuertos desde el Spark Data Frame.	
Parámetro de entrada	**Descripción**
sdf_json	Spark Data Frame con los datos de los aeropuertos.

Tabla 49. Función createTablePostGISFromDF( )

Script "Load_Sectors.R"

Script para cargar un fichero con los datos de los Sectores ATC de España desde local a PostgreSQL.

Funciones:

readFileSectors(filepath)	
Lee el fichero con los datos de los Sectores ATC desde local y crea una consulta de la forma "INSERT INTO public.test_atc_sectors_spain VALUES […]".	
Parámetro de entrada	**Descripción**
filepath	String con la ruta del fichero con los datos de los Sectores ATC.
Salida de la función	

Consulta de la forma “INSERT INTO public.test_atc_sectors_spain VALUES […]” con los datos de todos los sectores ATC.

Tabla 50. Función readFileSectors( )

createTablePostGIS(query)	
Crea una tabla en PostgreSQL y escribe los datos de los Sectores ATC.	
Parámetro de entrada	**Descripción**
query	String con la consulta creada en la función “readFileSectors(filepath)” con los datos de los Sectores ATC.

Tabla 51. Función createTablePostGIS( )

Script “Load_FIRs.R”
Script para cargar un fichero con los datos de los FIRs desde local a PostgreSQL.

Funciones:

readFileFIRs(filepath)	
Lee el fichero con los datos de los FIRs desde local y crea una consulta de la forma “INSERT INTO public.test_firs VALUES […]”.	
Parámetro de entrada	**Descripción**
filepath	String con la ruta del fichero con los datos de los FIRs.
Salida de la función	
Consulta de la forma “INSERT INTO public.test_firs VALUES […]” con los datos de todos los FIRs.	

Tabla 52. Función readFileFIRs( )

createTablePostGIS(query)	
Crea una tabla en PostgreSQL y escribe los datos de los FIRs.	
Parámetro de entrada	**Descripción**
query	String con la consulta creada en la función “readFileFIRs(filepath)” con los datos de los FIRs.

Tabla 53. Función createTablePostGIS( )

Script “Create_Table_In_Hive.R”
Script para crear una tabla en Hive para almacenar posteriormente los resultados de las métricas.

Script “Process_Radar_Tracks.R”
Script para procesar las trazas de radar de la base de datos “dart” en Hive y calcular las métricas de un mes.

Funciones:

loadHive2DF(date)	
Carga una porción de las trazas de radar en la tabla “radar” de la base de datos “dart” desde Hive a un Spark Data Frame.	
Parámetro de entrada	**Descripción**
date	String con la fecha en el formato “yyyy-mm-dd”.
Salida de la función	
Spark Data Frame con los datos de las trazas de radar.	

Tabla 54. Función loadHive2DF( )

createTablePostGisFromDF(sdf)	
Crea una tabla en PostgreSQL y escribe los datos de las trazas de radar desde el Spark Data Frame.	
Parámetro de entrada	**Descripción**
sdf	Spark Data Frame con los datos de las trazas de radar.

Tabla 55. Función createTablePostGisFromDF( )

queryTheAirports( )
Consulta los aeropuertos españoles y selecciona los datos de los aeropuertos en los que las columnas “icao” y “point” no estén vacías.
Salida de la función
Data frame de R con la información de los aeropuertos.

Tabla 56. Función queryTheAirports( )

queryTheSectors( )
Consulta los sectores ATC de España.
Salida de la función
Data frame de R con la información de los sectores.

Tabla 57. Función queryTheSectors( )

queryTheFIRs( )
Consulta los FIRs, seleccionando sólo los que están dentro de (o intersectan con) un polígono formado con las longitudes y latitudes mínimas y máximas obtenidas de los datos de la tabla “test_radar_tracks”.
Salida de la función
Data frame de R con la información de los FIRs.

Tabla 58. Función queryTheFIRs( )

calculateMetrics(date)
Calcula los resultados de las métricas densidad de tráfico, pico de tráfico y número de

conflictos, para cada hora de un día y para cada aeropuerto, sector ATC y FIR.	
Parámetro de entrada	**Descripción**
date	String con la fecha en el formato "yyyy-mm-dd".
Salida de la función	
Data frame de R con los resultados de las métricas.	

Tabla 59. Función calculateMetrics( )

writeInHive(df_metrics_month, month)	
Escribe un data frame de R en un fichero csv en HDFS, y carga los datos en la tabla "metrics_results" de la base de datos "airports" en Hive.	
Parámetro de entrada	**Descripción**
df_metrics_month	Data frame de R con los resultados de las métricas de un mes.
month	String con el mes en el formato "yyyy-mm".

Tabla 60. Función writeInHive( )

Script "Traffic_Density_Metric.R"

Script para calcular la métrica densidad de tráfico.

Funciones:

densityTrafficMetric(t0, refTimeInt=1.0, aType, aParam, radius=100, flMin=0, flMax=510, outer=FALSE)	
Calcula el número medio de aeronaves presentes en un volumen específico del espacio aéreo durante un intervalo de tiempo definido.	
Parámetro de entrada	**Descripción**
t0	Instante inicial de tiempo en el formato POSIXct.
refTimeInt	Intervalo de tiempo de referencia en horas y mayor que 0. (Valor por defecto: 1)
aType	Tipo de área XY para el cálculo de la métrica. Puede tomar los valores "circle" o "polygon".
aParam	String con los parámetros que definen el área XY.
radius	Radio del área del círculo, en millas náuticas. Se utiliza sólo cuando el tipo de área sea "circle". (Valor por defecto: 1)
flMin	Nivel de vuelo (flight level) mínimo, en centenares de pies. (Valor por defecto: 0)
flMax	Nivel de vuelo (flight level) máximo, en centenares de pies. (Valor por defecto:

	510)
outer	TRUE si se utiliza para calcular la métrica el área externa del área definido como “aParam”. FALSE si se utiliza para calcular la métrica el área interna del área definido como “aParam”. (Valor por defecto: FALSE)
Salida de la función	
Resultado numérico de la métrica.	

Tabla 61. Función densityTrafficMetric( )

Script “Peak_Load_Metric.R”

Script para calcular la métrica pico de tráfico.

Funciones:

peakLoadMetric(t0, refTimeInt=1.0, aType, aParam, radius=100, flMin=0, flMax=510, outer=FALSE)	
Calcula el número máximo de aeronaves que están volando simultáneamente dentro de un volumen específico del espacio aéreo durante un intervalo de tiempo de referencia.	
Parámetro de entrada	**Descripción**
t0	Instante inicial de tiempo en el formato POSIXct.
refTimeInt	Intervalo de tiempo de referencia en horas y mayor que 0. (Valor por defecto: 1)
aType	Tipo de área XY para el cálculo de la métrica. Puede tomar los valores “circle” o “polygon”.
aParam	String con los parámetros que definen el área XY.
radius	Radio del área del círculo, en millas náuticas. Se utiliza sólo cuando el tipo de área sea “circle”. (Valor por defecto: 1)
flMin	Nivel de vuelo (flight level) mínimo, en centenares de pies. (Valor por defecto: 0)
flMax	Nivel de vuelo (flight level) máximo, en centenares de pies. (Valor por defecto: 510)
outer	TRUE si se utiliza para calcular la métrica el área externa del área definido como “aParam”. FALSE si se utiliza para calcular la métrica el área interna del área definido como “aParam”. (Valor por defecto: FALSE)

Salida de la función
Resultado numérico de la métrica.

Tabla 62. Función peakLoadMetric( )

Script "Number_Of_Conflicts_Metric.R"

Script para calcular la métrica número de conflictos.

Funciones:

numberOfConflictsMetric(t0, refTimeInt=1.0, aType, aParam, radius=100, flMin=0, flMax=510, outer=FALSE, lsc=5, vsc=1000)	
Calcula el número de violaciones de los mínimos de separación especificados dentro de un volumen del espacio aéreo durante un intervalo de tiempo definido.	
Parámetro de entrada	**Descripción**
t0	Instante inicial de tiempo en el formato POSIXct.
refTimeInt	Intervalo de tiempo de referencia en horas y mayor que 0. (Valor por defecto: 1)
aType	Tipo de área XY para el cálculo de la métrica. Puede tomar los valores "circle" o "polygon".
aParam	String con los parámetros que definen el área XY.
radius	Radio del área del círculo, en millas náuticas. Se utiliza sólo cuando el tipo de área sea "circle". (Valor por defecto: 1)
flMin	Nivel de vuelo (flight level) mínimo, en centenares de pies. (Valor por defecto: 0)
flMax	Nivel de vuelo (flight level) máximo, en centenares de pies. (Valor por defecto: 510)
outer	TRUE si se utiliza para calcular la métrica el área externa del área definido como "aParam". FALSE si se utiliza para calcular la métrica el área interna del área definido como "aParam". (Valor por defecto: FALSE)
lsc	Criterio de separación lateral (Lateral Separation Criterium), en millas náuticas. (Valor por defecto: 5 millas náuticas)
vsc	Criterio de separación vertical (Vertical Separation Criterium), en pies. (Valor por defecto: 1000 pies)
Salida de la función	
Resultado numérico de la métrica.	

Tabla 63. Función numberOfConflictsMetric( )

SECCIÓN IV: CONCLUSIONES DEL TRABAJO

1. Conclusiones y trabajo futuro

1.1. Conclusiones

El trabajo desarrollado durante este Trabajo Fin de Grado plantea un reto de investigación relacionado con un área emergente como es el Big Data y el uso y explotación de la información contenida en grandes volúmenes de datos, con aplicación específica al ámbito del Tráfico Aéreo, área con un gran impacto social, económico y medioambiental. Por ello, durante este proyecto se han adquirido conocimientos y habilidades que suponen una sólida formación en las nuevas herramientas y técnicas desarrolladas en el área del Big Data, ya que se ha trabajado en una plataforma Big Data y aprendido sobre su arquitectura, formada por distintos componentes y herramientas, entre los que destaca el ecosistema Hadoop, integrado por módulos como son el sistema de ficheros HDFS, YARN, Hive, y Spark. Además, sobre estos datos se han aplicado técnicas de estadística y de minería de datos, utilizando para realizar estos cálculos y operaciones PostgreSQL, y en especial, su módulo PostGIS, que permite almacenar y realizar consultas sobre datos espaciales. Como lenguaje de programación se ha empleado R, un lenguaje dedicado especialmente al análisis estadístico y de minería de datos.

Estos conocimientos y habilidades tienen un gran interés formativo, y este interés es aún mayor si se considera que, en la actualidad, existe una creciente demanda de este perfil profesional. Por lo tanto, este proyecto ha permitido adquirir competencias profesionales que podrán utilizarse tanto en el ámbito empresarial, como en otros más relacionados con la investigación aplicada.

Desde la perspectiva del plan de estudios del Grado en Ingeniería Informática de Servicios y Aplicaciones, este Trabajo Fin de Grado ha complementado algunas de las competencias generales adquiridas durante la titulación:

- G03. Capacidad de análisis y síntesis.
- G05. Comunicación oral y escrita en la lengua propia.
- G09. Resolución de problemas.
- G10. Toma de decisiones.
- G11. Capacidad crítica y autocrítica.
- G16. Capacidad de aplicar los conocimientos en la práctica.
- G17. Habilidades de investigación.
- G18. Capacidad de aprender.
- G19. Capacidad de adaptarse a nuevas situaciones.
- G20. Capacidad de generar nuevas ideas.
- G21. Habilidad para trabajar de forma autónoma.

A un nivel específico, las competencias adquiridas durante la realización del proyecto complementan las adquiridas en las siguientes asignaturas:

- Tratamiento Automático de la Información.
- Programación y Estructuras de Datos.
- Sistemas de Bases de Datos.
- Administración de Bases de Datos.

1.2. Trabajo futuro

Este Trabajo Fin de Grado se enmarca dentro del proyecto AIRPORTS en el que trabaja la Universidad de Valladolid junto con la empresa Boeing. Este proyecto aún no ha finalizado, por lo que están previstas nuevas funcionalidades y características a añadir al cálculo de las métricas. Estos son algunos de los posibles trabajos futuros relacionados con las métricas:

- Definir e implementar nuevas métricas, como por ejemplo, una relacionada con los retrasos que se producen en las aeronaves.

- Adaptar el cálculo de las métricas para otros tipos de conjuntos de datos, como pueden ser los datos obtenidos a partir de la técnica ADS.

- Utilizar la tecnología HBase para mejorar el rendimiento del proceso de cálculo de las métricas.

2. Referencias

- SESAR WP 16 Transversal Areas. SWP 16.3 Environmental Sustainability, http://www.sesarju.eu/programme/workpackages/wp-16-rd-transversal-areas--203

- Aviation Environmental Design Tool (AEDT). FAA's Office of Environment and Energy. https://aedt.faa.gov/

- Nathan Marz and James Warren. "Big Data: Principles and best practices of scalable realtime data systems". Manning Publications. ISBN: 9781617290343

- <<Air Traffic Management (ATM)Gestión del Trafico Aéreo>>, http://www.aloftaviationconsulting.com/es/air-traffic-management.
- <<¿Cómo se divide y organiza el espacio aéreo? FIR,CTR,TMA,ATZ,CTA… (Vídeo)>>, http://www.takeoffbriefing.com/como-se-divide-y-organiza-el-espacio-aereo-firctrtmaatzcta-video/

- <<PostGIS — Spatial and Geographic Objects for PostgreSQL>>, http://postgis.net/

- <<PostgreSQL: The world's most advanced open source database>>, https://www.postgresql.org/

- <<SparkR (R on Spark) - Spark 2.0.0 Documentation>>, http://spark.apache.org/docs/2.0.0/sparkr.html

- <<Apache Hive>>, https://es.wikipedia.org/w/index.php?title=Apache_Hive&oldid=97122022

- <<R (lenguaje de programación)>>, https://es.wikipedia.org/w/index.php?title=R_(lenguaje_de_programaci%C3%B3n)&oldid=99503010

- <<Apache Hadoop open source ecosystem>>, https://www.cloudera.com//content/www/en-us/products/open-source/apache-hadoop.html