
***Facultad de Ciencias***

# Evaluación de Data Lakes en Machine Learning y Big Data

## (Evaluation of Data Lakes in Machine Learning and Big Data)

**Trabajo de Fin de Grado**
**para acceder al**

**GRADO EN INGENIERÍA INFORMÁTICA**

**Autor: Marina López Murcia**

**Director: Diego García Saiz**

**Co-Director: Julio García Fernández**

**Junio – 2020**

# Resumen

Un *Data Lake* es un repositorio de datos en bruto provenientes de diversas fuentes que permite el uso de herramientas de *Big Data* y *Machine Learning*. Es un sistema pensado para procesos de inteligencia de negocio, ya que permite cruzar grandes volúmenes de información de distintos entornos haciendo uso de técnicas de paralelización masiva.
El problema es que estos sistemas tienen una estructura muy compleja y carecen de una estandarización, lo que los convierte en proyectos de alto riesgo con altas tasas de fracaso.

Este trabajo de fin de grado se desarrolla en colaboración con la empresa *CIC consulting informático*, una consultoría de desarrollo de proyectos de informática. La empresa quiere realizar la integración de tres productos desarrollados internamente, *IDboxRT*, *SgrWin* y *Fieldeas* que, aunque se desarrollen por separado, en ocasiones son usados de forma conjunta

El objetivo de este proyecto es evaluar la tecnología de los *Data Lake* para la integración de estos tres productos, comprobando su capacidad de integración con distintas herramientas de *Big Data* y *Machine Learning* mediante la implementación de distintas variantes del sistema.



**Palabras clave:** *Data Lake, Big Data, Machine Learning.*

# Abstract

A Data Lake is a repository of raw data from various sources that allows the use of Big Data and Machine Learning tools. It is a system designed for Business Intelligence processes, since it allows to cross large volumes of information from different environments by using massive parallelization techniques. The problem is that these systems have a very complex structure and also lack of standarization, making them high-risk projects with high failure rates.

This final degree project is developed in collaboration with the company *CIC consulting informático,* a computer consulting company. The company wants to integrate three internally developed products, IDboxRT, SgrWin y Fieldeas that, although developed separately, are sometimes used together.

The objective of this project is to evaluate the Data Lake technologies for the integration of these three products, verifying their integration capacity with different Big Data and Machine Learning tools through the implementation of different system variants.



**Key words:** *Data Lake, Big Data, Machine Learning.*

# Índice

# 1 Introducción

## 1.1 Contexto

Las empresas siempre han tenido la necesidad de almacenar toda la información necesaria para las operaciones y toma de decisiones dentro de las mismas, lo que se conoce como inteligencia de negocio, siendo las Bases de Datos Relacionales el método principal usado para obtener, y almacenar los datos. Estas Bases de Datos ofrecen la posibilidad de almacenar información muy estructurada y normalmente desnormalizada, algo que es suficiente para cubrir ciertas necesidades empresas.

Sin embargo, actualmente los sistemas de gestión de la información suelen ser múltiples dentro de una empresa e incluso se integra con fuentes externas, encontrándonos en un escenario de gran dispersión de la información con volúmenes muy grandes de datos, estructurados y no estructurados, que no pueden ser tratados por los sistemas relacionales.

Para poder gestionar toda esta información, en los últimos años surgieron los Data Lakes. Un Data Lake es un repositorio centralizado de los datos en bruto a gran escala, que permite utilizar herramientas de *Big Data* y *Machine Learning* sobre los datos almacenados. Los datos pueden ser estructurados, semiestructurados y no estructurados, lo que aporta una gran flexibilidad al sistema en relación al formato de los datos.

Al contrario que los sistemas tradicionales de inteligencia de negocio, los *Data Lakes* permiten realizar transformaciones bajo demanda, sustituyendo el proceso ETL tradicional (Extraer, Transformar, cargar/Load), donde primero se realizan las transformaciones y posteriormente se cargan los datos, por un proceso de tipo ELT, donde las transformación se hacen después de cargar los datos. Esto no excluye a que ciertas transformaciones de interés permanente no se sigan realizando de forma periódica.

El proceso de desarrollo de los Data Lakes es una tarea no estandarizada, pero que suele tener una serie de características comunes:

- Almacenamiento de datos distribuidos. Los datos se encuentran fragmentados y con múltiples copias.
- El formato de los datos de entrada suele ser mediante ficheros en bruto con, como mucho, transformaciones simples de cara a su almacenamiento inicial.
- Las transformaciones generalmente se realizan bajo demanda, pero existe la posibilidad de realizar ciertas transformaciones preprogramadas de forma periódica.
- La información está organizada por su nivel de refinamiento. Habitualmente la nomenclatura para designar las zonas es Bronce, Plata y Oro:
    - En la zona Bronce se encuentran los datos en bruto.
    - En la zona Plata se encuentran los resultados de las analíticas que no han sido pensadas para el consumo externo.
    - En la zona Oro se encuentran los datos de las analíticas en formato estructurado con los datos que van a ser expuestos a consumidores externos.

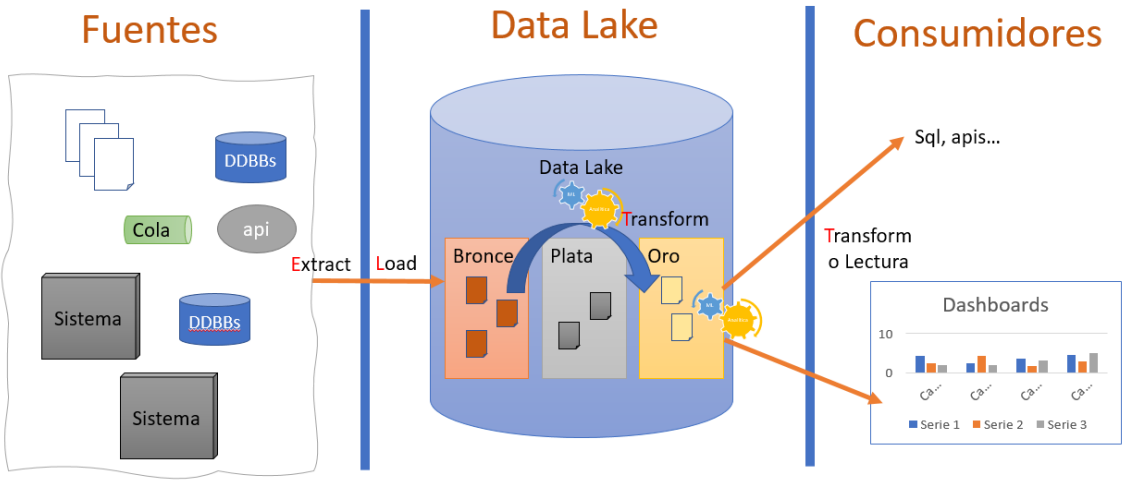


*Figura 1: Estructura ejemplo Data Lake*

Es precisamente la falta de estandarización la que hace que habitualmente los Data Lakes se ofrezcan como proyectos "ad hoc" para cada empresa o cliente concreto, apareciendo solo en los últimos años ofrecidos como un producto empaquetado, siendo este último, un enfoque muy reciente y todavía poco desarrollado.

Unido a esto, la complejidad de las estructuras que se contemplan, con la gestión de los accesos o la seguridad de los datos, convierte a los Data Lakes en un desarrollo de alto riesgo con elevadas tasas de fracaso.

## 1.2 La empresa

Este proyecto ha sido desarrollado para la empresa CIC consulting informático, una consultoría de ingeniería de desarrollo y proyectos de informática y comunicaciones. CIC dispone de una serie de productos desarrollados internamente, que se pueden instalar o consumir bien de forma individual o en ocasiones colectivamente. Unido a esto, los clientes potenciales disponen de otros sistemas de información. Por ello se presenta una situación ideal para hacer uso de un Data Lake.

De los productos existentes en CIC, los tres sobre los que se va a realizar la integración son:

- *IDboxRT*: Software de inteligencia operativa basado en la tecnología *Big Data* que ofrece información en tiempo real para la supervisión y el control de los distintos entornos y procesos de un negocio.
- *SgrWin*: Sistema de gestión para redes de transmisión digital formadas por múltiples fabricantes y tecnologías.
- *Fieldeas*: Soluciones de Movilidad Empresarial que permiten optimizar los procesos de negocio de las compañías que operan con equipos de trabajadores en campo

Si bien estos productos pueden trabajar por separado, puntalmente se realizan integraciones entre los mismos, los que produce una situación ideal para el uso de los Data Lakes.

# 2 Objetivos

El objetivo de este proyecto de fin de grado es comprobar la viabilidad de la tecnología de los Data Lakes para realizar la integración de los productos de CIC, *IDboxRT*, *SgrWin* y *Fieldeas*. Concretamente, las tareas a desarrollar para la consecución del objetivos son las siguientes:

- Determinar el servicio en la nube más adecuado para el Data Lake.
- Evaluar las ventajas e inconvenientes de los modelos de Data Lake.
- Evaluar las capacidades de integración con las distintas herramientas de Machine Learning y *Big Data*.
- Implementar y parametrizar variantes del sistema.
- Desarrollar simuladores para realizar la ingesta de datos y evaluar la capacidad de las herramientas usadas.

# 3 Herramientas y servicios empleados

Dentro de este apartado se diferencian en dos grupos: los que se ofrecen como servicio desde la nube, y los que se usan de forma local.

## 3.1 Servicios en la nube

Para la realización de este proyecto se escogió Azure como el servicio en la nube más adecuado. Al seleccionar cual sería el mejor servicio se planteó entre cualquiera de los tres principales AWS, Azure y Google Cloud, todas ellas en funcionalidad son muy parecidas ya que han ido implementando los mismos servicios durante los últimos años. Pero finalmente se escogió Azure debido a que, de los tres productos de CIC que se quieren integrar, dos están desarrollados en *Microsoft .NET Framework* por lo que la gente que en un futuro tenga que realizar la integración va a estar más acostumbrada a estar este lenguaje que es nativo en Azure.

### 3.1.1 Azure

Microsoft Azure es un servicio de computación en la nube desarrollado por Microsoft que fue publicado en Febrero de 2010. Actualmente Azure ofrece soporte a más de 60 regiones y 140 países y es el segundo proveedor de la nube, solo detrás de AWS.

El objetivo de Azure es ofrecer una serie de servicios y herramientas en la nube a las empresas para que puedan cumplir sus objetivos. Es compatible con tecnologías de código abierto. Lo que permite que los clientes tengan la libertad de seguir usando otras tecnologías que no pertenecen a Azure. Así mismo desde el propio portal de Azure se puede acceder a software ofrecidos por terceros, lo que aporta una gran flexibilidad de uso a los clientes.

Una de las principales ventajas que tiene Azure es que evita el proceso de montar, gestionar y escalar la infraestructura necesaria para las aplicaciones, evitando todos los gastos que ello supone. Azure tiene permite ajustar el número de servidores de las aplicaciones de forma dinámica dependiendo de la carga de trabajo que haya en cada momento, lo que junto a la política de pago de *pay-as-you-go*, con la que los clientes solo pagan por lo que usan, los gastos pueden ser ajustados a lo únicamente necesario.

A la hora de programar, Azure admite varios lenguajes de programación como C#, Python o Java entre otros. Azure también dispone de una integración con distintos IDEs como, por ejemplo, con Visual Studio haciendo uso de Azure SDK, lo que permite a los desarrolladores crear las aplicaciones desde estos entornos en vez de tener que desarrollarlo desde la plataforma de la nube

Azure garantiza la seguridad de la información, proporcionando confidencialidad, disponibilidad e integridad de los datos mediante copias redundantes de los datos para la recuperación ante errores inesperados, distintos modos de identificación para acceder a la información y mediante la protección de sus centros de datos.

Para el desarrollo de este proyecto se han utilizado los siguientes servicios de Azure: *Data Lake Storage Gen 1, Data Lake Analytics, Blob Storage, Data Factory Gen 2, Event Hubs, Azure Synapse, Azure Functions* y *Azure Data Bricks.*

### 3.1.2 Data Lake Storage

Azure Data Lake Storage es un repositorio de datos a nivel empresariales a gran escala para realizar análisis y transformaciones de tipo *Big Data*. Es un sistema de archivos de Hadoop, lo que permite que se pueda integrar fácilmente con sus aplicaciones.

Dispone de un almacenamiento ilimitado de ficheros, los cuales pueden estar en cualquier formato y no hay ninguna limitación desde el punto de vista del tamaño del fichero ni del tiempo de tiempo que un dato puede estar almacenado, realizando copias de seguridad para evitar la pérdida de datos. Tampoco precisa de la definición de un esquema de los datos, sino que este se define al realizar el análisis.

Esta tecnología está diseñada para realizar análisis a gran escala por lo que, para mejorar la lectura, los ficheros están fragmentados y distribuidos en varios servidores permitiendo la lectura en paralelo de los datos.

En relación con la protección de los datos, ADLS usa tres sistemas distintos:

- Autenticación mediante el *Azure Active Directory*: esto proporciona a los usuarios una identidad, y mediante su uso permite a los usuarios y a las aplicaciones autenticarse para acceder a los datos, así como mantener un control de acceso basado en roles.
- Control de acceso: permite proporcionar permisos a nivel de fichero o de carpeta a las distintas personas y aplicaciones que vayan a tratar los datos.
- Cifrado: ADLS proporciona la posibilidad de cifrar o no los datos que almacena.

Los precios por uso de este software se muestran en la Tabla 1:

*Tabla 1: precios ADLS*

| Cantidad en TB | Precio mensual por Giga |
|---|---|
| Menos de 100 | 0.0329€ |
| Entre 100 y 1.000 | 0.0321€ |
| Más de 1.000 | 0.0313€ |

Se puede ver que el coste de almacenamiento haciendo uso de esta herramienta es realmente barato.

### 3.1.3 Azure Data Lake Analytics

Azure Data Lake Analytics es un servicio que permite realizar trabajos a demanda para transformar y analizar *Big Data*.

Este servicio utiliza U-SQL como lenguaje para realizar las transformaciones. U-SQL mezcla la sencillez de SQL, pero da la posibilidad de usar C# para ciertas operaciones más complejas que no se pueden realizar con SQL.
Los trabajos se pueden lanzar desde el portal web de Azure, pero también cuentan con una integración con Visual Studio, lo que permite desarrollar los trabajos y lanzarlos desde este entorno de desarrollo, lo que lo hace más cómodo.

A los trabajos se les puede asignar de forma dinámica el número de unidades de análisis que se van a utilizar, lo que permite que se pueda adaptar a las necesidades dependiendo del volumen de la carga.

Tras lanzar el trabajo Data Lake Analytics muestra un gráfico donde se puede ir observando el avance de la operación, y en caso de fallo ver en qué punto ha sucedido.

También muestra una gráfica con las unidades de análisis usadas en cada instante de la ejecución y una predicción de cuál sería el mejor número de unidades dependiendo del coste o de la velocidad. Se muestra un ejemplo en la Figura 2

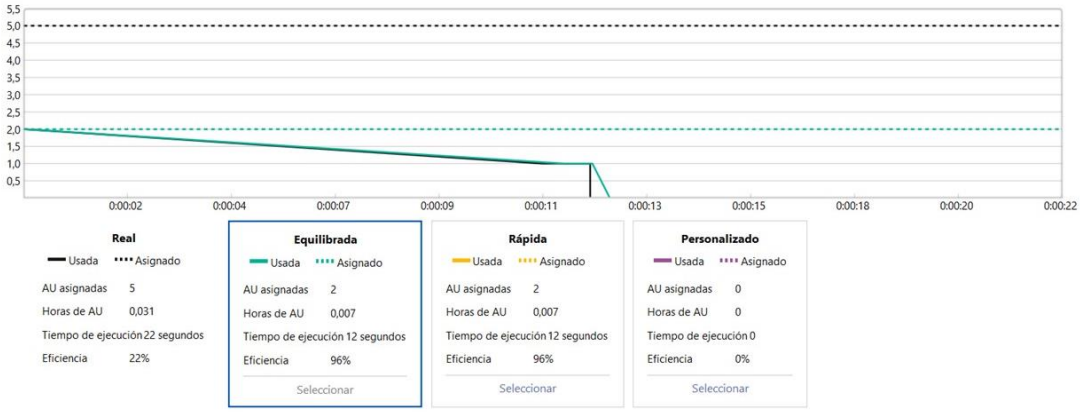


*Figura 2: Predicción UA*

El coste de este servicio es de 1,687€/hora por cada unidad de análisis.

### 3.1.4 Blob Storage

Blob Storage es una herramienta de almacenamiento de un gran volumen de datos no estructurados en la nube.

Los datos están estructurados de la siguiente forma:

- Cuenta de almacenamiento: puede contener un número ilimitado de contenedores. Proporciona un espacio de nombres único en Azure con el que se puede establecer un punto de conexión.
- Contenedor: se encuentra dentro de la cuenta de almacenamiento, sería el equivalente a una carpeta. Puede contener un número ilimitado de blobs.
- Blob: se encuentran dentro de los contenedores, son los archivos almacenados.

Los precios por uso de este software se indicar en la Tabla 2:

*Tabla 2: Costes Blob Storage*

| Cantidad en TB | Precio mensual por Giga |
|---|---|
| Menos de 50 | 0.0166 € |
| Entre 50 y 450 | 0.0159€ |
| Más de 500 | 0.0153€ |

### 3.1.5 Data Factory

Es una herramienta que permite la transformación y transferencia de datos de forma sencilla dentro de Azure, para ello dispone de una interfaz donde se pueden realizar las transformaciones de los datos de forma visual y de una serie de plantillas ya creadas para la transferencia.

Dentro de este proyecto se ha hecho uso de su función de transferencia para copiar datos almacenados en el Blob Storage al Data Lake Storage. Estos dos servicios aparecen ya precompilados como fuentes de entrada y salida, por lo que únicamente hay que hacer un enlace a las entidades del servicio para que, posteriormente se pueda hacer la transferencia de datos.

Las transferencias se pueden realizar bajo demanda o de forma programada siguiendo un horario escogido. Y de nuevo nos encontramos ante una herramienta sin servidor, por lo que solo se va a pagar el tiempo en el que el servicio está realizando alguna operación.

### 3.1.6 Event Hubs

Event Hubs es un servicio que realiza la ingesta de datos en tiempo real. Puede recibir y procesar millones de datos por segundo. Un servicio de Event Hubs internamente puede tener varios Event Hub que reciben y tratan la información de forma independiente entre ellos.

Este servicio dispone de una opción llamada “Capture” que permite periódicamente ir guardando los datos procesados por los Event Hub en el servicio de ADLS o de Blob Storage. Esta opción se ha utilizado en el proyecto para poder guardar en ficheros todos los datos que surgían como datos de tiempo real.

### 3.1.7 Azure Functions

Las Azure Functions son un servicio de computación sin servidor en Azure que están basadas en la respuesta a eventos.

La ventaja que aporta este servicio es la posibilidad de realizar una computación puntual, desencadenada por algún motivo, sin tener que hacerse cargo del pago de un servidor dedicado a esta tarea. Solo se paga por el tiempo de ejecución.

Hay una gran variedad de lenguajes de programación para desarrollar estas funciones y también de desencadenadores, para la realización de este proyecto se ha hecho uso del desencadenador basado en un temporizador.

### 3.1.8 Azure Synapse

Azure Synapse es una evolución de los Data Warehouses para dar soporte a las herramientas de Data Lake, el objetivo de esta tecnología dentro de este proyecto es poder comparar nuestra solución ad-hoc con una solución ya empaquetada.

Los Data Warehouse se emplean para almacenar datos provenientes de procesos ETL (Extraer, Transformar y cargar/Load), mientras que los Data Lakes forman parte de un proceso de tipo ELT, por lo que no tiene la capacidad de los Data Lakes para realizar transformaciones bajo demanda que no estuviesen planteadas desde un inicio, ya que al desnormalizar la información es imposible hacerlo con todas las combinaciones. Pero a cambio tiene unos patrones de trabajo, unos permisos ya creados y una estructura de los datos ya definida, algo que consciente o inconscientemente, se estaba copiando al desarrollar las soluciones ad-hoc de los Data Lakes.

Azure Synapse se ha suado para realizar la integración entre las estructuras y patrones de un Data Warehouse con las aplicaciones de *Big Data*, como es Spark que tienen disponibles los sistemas de Data Lake.

### 3.1.9 Azure Databricks

Es una colaboración entre Azure y Apache Spark que permite realizar análisis y transformaciones de tipo *Big Data*. Mediante el uso de este servicio se cuenta con un entorno de Spark ya configurado.

Las operaciones se programan mediante el uso de *Notebooks* y admite el uso de Python, Scala, R, Java y SQL. Para este proyecto se decidió usar Python por la familiaridad con el lenguaje.

Este servicio permite acceder a los datos almacenados en el ADLS y realizar transformaciones con una escalabilidad automática.

### 3.1.10 Servicios Cognitivos

Son un conjunto de herramientas de *Machine Learning* que permiten acceder de forma sencilla a herramientas de inteligencia artificial. Dentro de estos servicios se ha utilizado la categoría de *Speech* para poder transcribir audios de voz a texto.

## 3.2 Otros software utilizados

Estas aplicaciones se han empleado como apoyo a la evaluación de las otras tecnologías para generar, almacenar y visualizar los datos. Pero no forman parte de forma directa en el desarrollo del proyecto, a excepción de Visual Studio que por su integración con Azure se ha empleado para lanzar las consultas al Data Lake.

### 3.2.1 Kafka

Kafka es un sistema de colas de mensajería que permite realizar un gran escalado de los datos. Esta herramienta clasifica la información en distintos grupos, denominados *topics* con distintas particiones por *topic*, lo que permite tener un rendimiento muy alto y asegurar que la información vaya al destino adecuado sin mezclarse con otros *topics*.

### 3.2.2 Power BI Desktop

Power BI es una herramienta que permite analizar y visualizar cantidades muy grandes de datos de tipo inteligencia empresarial. A partir de esta aplicación se pueden cruzar distintas fuentes de datos y analizarlos para posteriormente a través de sus herramientas visualizarlos de forma fácil.

### 3.2.3 Eclipse

Este entorno de desarrollo se utilizó para desarrollar los programas en java, en concreto los simuladores de datos sobre los que desarrollar las transformaciones.

### 3.2.4 Visual Studio

Visual Studio es un entorno de desarrollo que, entre otras opciones, permite desarrollar aplicaciones compatibles con la plataforma .NET. Este entorno esta integrado con Azure, por lo que desde él es posible acceder a la cuenta de Azure e interactuar con los servicios registrados.

# 4 Diagrama de trabajo

Este trabajo de fin de grado, al no corresponder con el desarrollo de una aplicación, sino ser un trabajo de evaluación de una tecnología, fue dividido en varias fases donde se fueron estudiando y evaluando distintas tecnologías que componen un Data Lake. A pesar de que el objetivo del proyecto es el estudio, también se realizaron implementaciones como apoyo para comprobar de forma práctica las tecnologías.

El desarrollo del proyecto dividido por semanas fue aproximadamente el siguiente, desde su comienzo el día 3 de febrero, hasta su fin el 12 de junio, con aproximadamente 495 horas de trabajo en total.

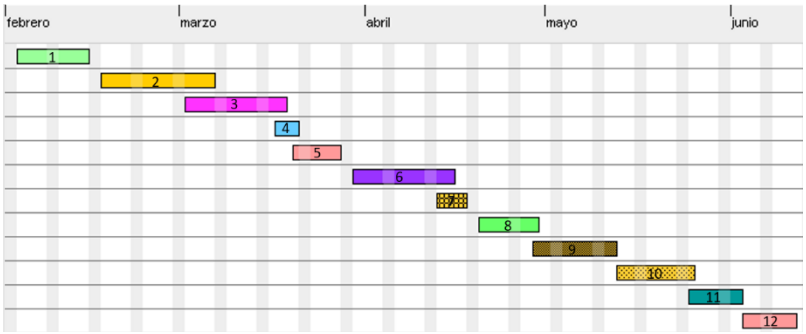


*Figura 3: Grafo Trabajo*

1. Aprendizaje sobre las tecnologías a emplear: las dos primeras semanas de este trabajo se destinaron a aprender sobre las tecnologías básicas que se iban a emplear.
2. Implementación de los dos primeros casos de uso: a esta tarea se le dedicaron alrededor de tres semanas. Estos dos casos de uso sirvieron como introducción al uso de estas tecnologías.
3. Automatización de los servicios: el objetivo de esta tarea era automatizar los casos de uso hechos hasta ese momento. Inicialmente se planteó destinar una semana para realizar la automatización de los primeros casos de uso, pero finalmente se destinaron alrededor de dos semanas y media.
4. Evaluación de los logs de las operaciones: se le dedicó media semana.
5. Primera prueba de carga para valorar el tamaño ideal de los ficheros del Data Lake y el escalado de la herramienta. Se le destinó una semana.
6. Desarrollo de un simulador de datos en relación con los tres productos de CIC para realizar futuras evaluaciones. Se le dedico en torno a tres semanas.
7. Desarrollar una primera implementación del Data Lake haciendo uso de los datos del nuevo simulador. Debido a la similitud de esta tarea con el primer caso de uso se le destinó alrededor de una semana.
8. Familiarizarse con la tecnología de Spark. Llevó en torno a una semana y media.
9. Realizar la implementación del Data Lake con Data Bricks. Se desarrolló en unas dos semanas.
10. Intento de implementación haciendo uso de Azure Synapse. Se le dedicó aproximadamente dos semanas.
11. Implementación de herramientas de Machine Learning con el Data Lake, se desarrolló en unas dos semanas.
12. Prueba de carga final. Esta fue la última fase, y llevó una semana y media.

# 5 Desarrollo del proyecto

Este apartado explica cómo se ha ido desarrollando el proyecto en función a las fases definidas en el apartado anterior.

## 5.1 Implementación de los dos primeros casos de uso

Esta primera fase fue una prueba de concepto para familiarizarse con el uso de estas tecnologías, durante la cual se construyó la estructura básica del sistema y se realizaron los dos primeros flujos de datos completos, desde la simulación de los datos hasta la visualización de las analíticas.

### 5.1.1 Primera implementación

En esta primera implementación se construyó la estructura básica de nuestro Data Lake, y se realizó un flujo completo de datos, desde los datos en local hasta la visualización de los resultados de las transformaciones realizadas en el Data Lake.

#### 5.1.1.1 Los datos

Estas dos primeras implementaciones utilizan un generado de datos de una red telefónica ya existente en la empresa. Este generador creaba tres tipos de ficheros que contenían información sobre:

- Antenas: estos ficheros contenían el identificador único de las antenas, su nombre y su localización en coordenadas, todas situadas en Cantabria.
- Terminales: son los dispositivos que se conectan a las antenas. De ellos se almacena su identificador único y su marca.
- Registros: guardan el registro de las conexiones entre las antenas y los terminales, específicamente la fecha donde se produjo la conexión, si la conexión tuvo éxito y los id de cada una de las entidades.

Estos datos se generan en ficheros de tipo CSV, en este caso deparados por “;” en un directorio local. A su vez este directorio estaba dividido en otros tres: antena, terminal y registro en función al tipo de fichero que fuesen.

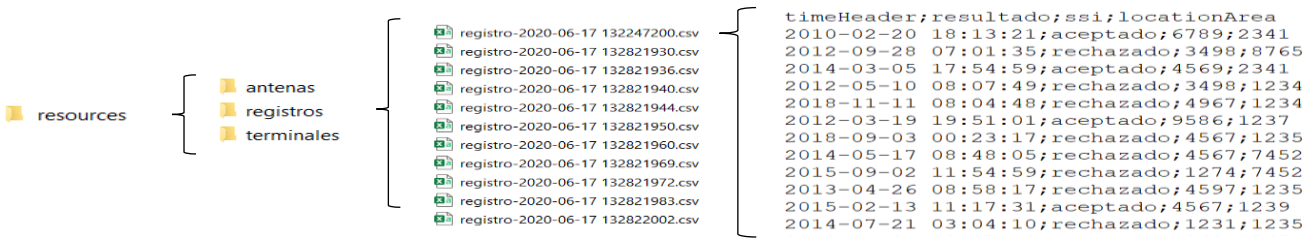


*Figura 4: Organización de los datos*

#### 5.1.1.2 Estructura:

En esta implementación encontramos dos zonas diferenciadas:

- Zona de intercambio: esta zona sirve como separación entre la subida de datos en local y el propio Data Lake. Esta zona no es común a todas las implementaciones de los Data Lakes, y no es imprescindible, se podría subir directamente los datos de local al ADLS. Pero en esta primera implementación sirve para disminuir los accesos al ADLS aumentando así la seguridad del sistema.
- Data Lake: en esta zona se almacenan los datos y se realizar las transformaciones y analíticas haciendo uso de Azure Data Lake Analytics con U-SQL. Los datos procedentes de la zona de intercambio de guardan en la zona Bronce, y tras realizar las analíticas los datos finales se guardarán en la zona Oro.

La zona de intercambio está compuesta por un Blob Storage, un servicio que permite almacenar ficheros clasificados dentro de contenedores.

El Data Lake está compuesto por dos servicios distintos, el Azure Data Lake Storage (ADLS), en el que se almacenan los datos; y el Azure Data Lake Analytics (ADLA), el servicio que permite hacer las transformaciones sobre los datos almacenados en el ADLS.

Estas dos zonas están unidas por el servicio *Data Factory*, el cual va a realizar las copias entre los datos de la zona de intercambio y el Data Lake.

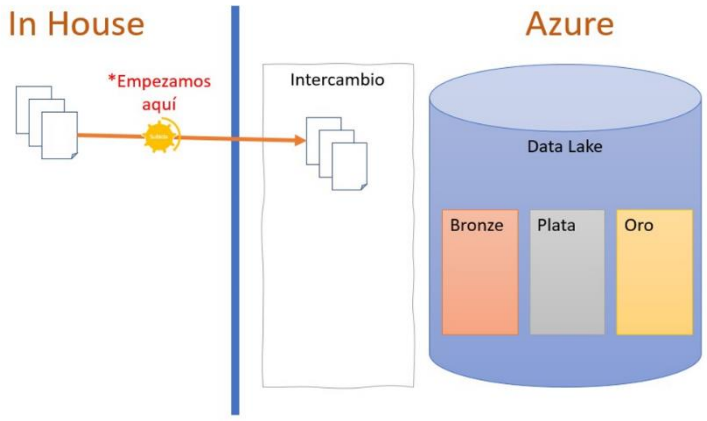


*Figura 5: Estructura primera implementación*

#### 5.1.1.3 Flujo de datos:

Los datos comienzan almacenados en un directorio local y se suben a la nube en la zona de intercambio. Posteriormente estos datos son copiados de la zona de intercambio a la zona Bronce del Data Lake. Se realizará una transformación sobre estos datos y el resultado se guardará en la zona Oro y se visualizarán usando Power BI.

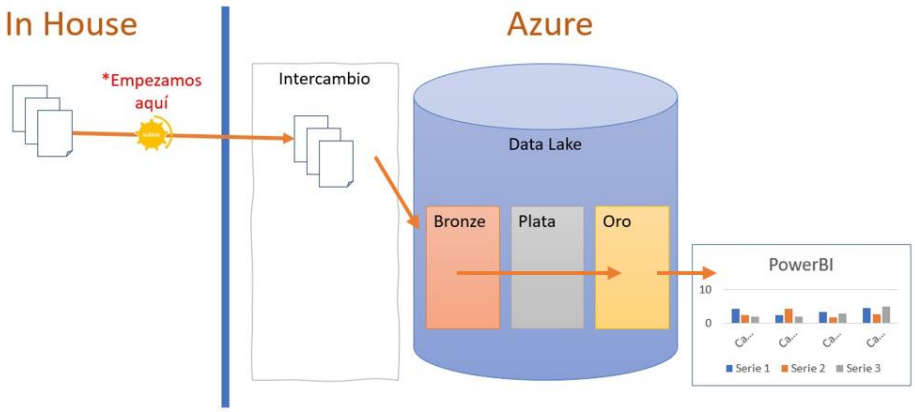


*Figura 6: Flujo datos primera implementación*

#### 5.1.1.4 Desarrollo:

**Creación de los servicios:**

Para esta implementación se usan cuatro servicios distintos: Blob Storage, Data Factory, Data Lake Storage y Data Lake Analytics. Así que el primer paso fue crearlos.

Todos estos servicios se han creado desde el portal de Azure, en la página principal del portal se encuentra la opción de "crear un recurso" que permite crear los recursos de forma guiada y muy sencilla.

Lo primero es crear un grupo de recursos, que son un contenedor que permite agrupar varios servicios para tratarlos como una unidad, dando la opción de asignar permisos, gestionar logs o eliminar todos los servicios al tiempo. Al crear el grupo de recursos se nos va a pedir un nombre único y una región, la región que se ha usado en todos los servicios de este proyecto es el oeste de Europa, ya que es la más cercana, por lo que los costes y latencia son lo más bajos posible.

Posteriormente creamos el Blob Storage, Data Factory y el ADLS, estos tres servicios nos piden los mismos campos, un grupo de recursos, en el que se asigna el que se acaba de crear, un nombre único para el espacio de nombres y una región.

Dentro del ADLS, con el explorador de archivos se crean tres directorios nuevos: Bronce, Plata y Oro.

A continuación se crea el ADLA, donde se vuelven a solicitar los mismos tres campos pero además también se tiene que especificar el nombre de la cuenta de ADLS, por lo que se le asigna el creado en el párrafo anterior.

## Subida de datos a la nube:

En este punto se tiene los datos generados y la estructura ya está creada, por lo que se prosigue a subir los datos a la nube.

La zona de intercambio en esta implementación está compuesta por un Blob Storage, en el cual se tiene que crear un container donde almacenar los datos para posteriormente subirlos.

Esto se realiza a través de un programa en Java, usando la biblioteca Azure Blob Storage. Esta librería permite gestionar los Blob Storages dando la opción de crear o eliminar contenedores, carpetas, ficheros, etc.

El programa hacia uso de un fichero de configuración, donde se especificaba:

- La cadena de conexión del Blob Storage: esto es una cadena de texto que contiene la clave para acceder a la cuenta de almacenamiento, se puede localizar en el apartado de claves de acceso desde el portal de Azure.
- El nombre del container que vamos a crear en el Blob Storage y donde se guardaran todos los ficheros, en este caso siembre se identificó como intercambio.
- El tipo de subida que se iba a realizar, de un directorio completo o de un fichero, y en función de esto la ruta y el nombre del elemento a subir.

El programa carga los datos introducidos en el fichero de configuración y en función a ellos realiza los siguientes pasos:

- Crea una referencia al Blob Storage haciendo uso del comando CloudStorageAccount.parse(), que recibe como parámetro la cadena de conexión de la cuenta de almacenamiento y retorna un objeto del tipo CloudStorageAccount.
- Crea un cliente del BS con el comando storageAccount.createCloudBlobClient().
- Comprueba si existe un contenedor con el nombre especificado en el fichero de configuración. Si existe crea una referencia a él, sino existe lo crea.
- Posteriormente si realiza la subida de los elementos seleccionados:
    - Si se trata de un único fichero se realiza la subida del mismo con el comando CloudBlockBlob.upload(), que recibe como parámetros un FileInputStream con la referencia a la ruta del fichero y el tamaño del fichero.
    - Si se trata de una carpeta completa, se recorre la carpeta de forma recursiva y haciendo uso del mismo comando que en el caso anterior se va subiendo cada uno de los ficheros uno a uno.

De esta forma después de ejecutar el programa, se crea un container en el Blob Storage y en su interior se guardan los tres directorios con los ficheros que teníamos en local

## Trasladar los datos de la zona de intercambio al Data Lake:

Para transferir los datos desde la zona de intercambio hasta el Data Lake se hizo uso del servicio Data Factory.

**Permisos:**

Para poder acceder al Data Lake Storage hay que tener permisos de tipo ACL (listas de control de acceso) y de IAM (administrador de identidad de Azure), por lo que el primer paso es dar estos permisos al Data Factory.

Se comienza por el ACL, para ello desde el portal de Azure se accede al explorador de archivos del DLS, ahí se encuentra un apartado de "Accesos" desde el que se asignan los nuevos ACL. Para poder copiar los datos desde el Blob hasta el DLS el Data Factory deberá de tener permisos sobre la carpeta "Bronce" y sus carpetas recursivas con acceso de entrada y permisos por defecto.

Posteriormente se asignan los permisos de IAM, estos son permisos basados en roles por lo que desde el portal de Azure en el DLS accedemos al apartado de "control de acceso IAM", buscamos el servicio empleado de Data Factory y se le asigna el rol de propietario.

**Transferencia:**

Con la opción de "Copy data" del Data Factory se puede transferir la información de forma muy sencilla entre el Blob Storage y el Data Lake haciendo uso de un formulario guiado dividido en varios apartados.

Desde este formulario se especifica cuales van a ser los servicios de origen y de destino, los datos que se quieren transferir y la ruta donde se van a almacenar en el servicio destino. En este caso se establece el servicio de origen como el Blob Storage de intercambio y el destino como la carpeta Bronce del ADLS.

También se especifica si la transferencia se va a realizar una única vez o si se va a ejecutar periódicamente. En este caso, para la prueba, se ejecuta una única vez.

**Transformaciones de los datos:**

El objetivo de esta primera implementación era familiarizarse con el uso de las herramientas, por lo que esta primera transformación fue una transformación sencilla donde se pasó directamente de Bronce a Oro.

Partimos de los datos en Bronce que son los datos en bruto que se crearon en el generador. La transformación que se planteó fue para poder mostrar en un mapa la posición de las antenas con el número de conexiones que se habían producido en cada una, para ello se debía de guardar en un solo fichero el id y nombre de las antenas, con el número de conexiones y las coordenadas.

Las transformaciones se desarrollan en lo que se denomina trabajos de Azure Data Lake Analytics, estos se pueden desarrollar desde el portal, pero por comodidad se hizo desde Visual Studio. Este entorno de desarrollo tiene una implementación con Azure que permite al usuario identificarse con su suscripción de Azure y poder acceder a sus servicios e interaccionar con ellos, en este caso en específico, permite lanzar los trabajos de ADLA.

El lenguaje que utiliza ADLA es el U-SQL, que es una combinación de SQL, que permite realizar consultas de forma sencilla y C# para operaciones más complejas. Como esta transformación era muy sencilla se empleó únicamente SQL.

Para extraer la información de los ficheros de registros y antenas se empleó *Extractors.Text()*, posteriormente se agruparon los registros en función del id de las antenas, se realizó un contador del número de conexiones y se le añadieron los campos de los nombres de las antenas y sus

coordenadas. Finalmente esos datos se escribieron en un fichero de la zona Oro haciendo uso de *Outputters.Csv()*.

#### 5.1.1.5 Visualización de los resultados.

Para la visualización de los resultados se utilizó Power BI Desktop, esta aplicación tiene la opción de acceder a los datos almacenados en el Data Lake Storage directamente introduciendo la dirección del fichero (dentro del espacio de nombres del Data Lake Storage) e introduciendo las credenciales de la cuenta de Azure para identificarse.

Al leer el fichero se crea una consulta en Power BI, y en este caso la mostramos visualmente haciendo uso de un mapa, obteniendo el siguiente resultado:

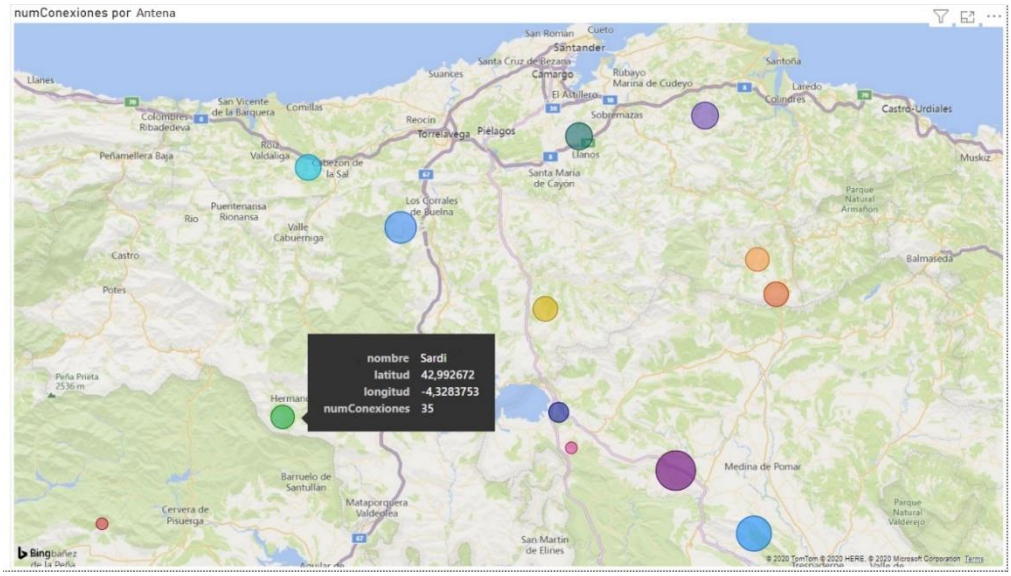


*Figura 7: Visualización resultados*

### 5.1.2 Segunda implementación

Esta segunda implementación se basa en la estructura generada anteriormente, mientras que para los datos de registro se realizó una simulación en tiempo real. Para ello los datos en local han sido almacenados en un Kafka y a la zona de intercambio de le ha añadido un nuevo servicio, los Event Hubs que permiten tratar estos datos en tiempo real.

También se ha añadido un flujo de datos hacia la izquierda, de la zona Oro a la zona de Intercambio.

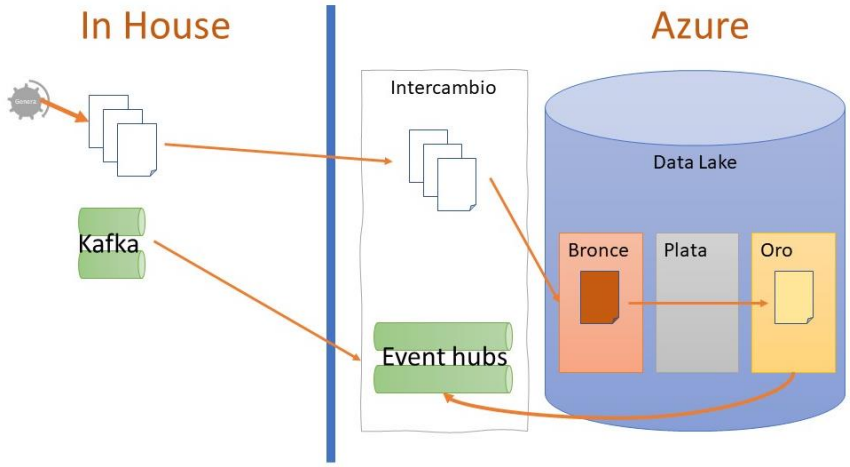


*Figura 8: Segunda implementación*

#### 5.1.2.1 Los datos

Se mantienen los datos empleados en la primera implementación, pero los datos de los registros se simulará que llegan en tiempo real. Por lo tanto los datos de las antenas y de los terminales seguirán la estructura anterior, pero los datos de los registros serán guardados en un Kafka.

Para guardar los datos de los ficheros de los registros en Kafka se implementó un programa en C# usando la librería *Confluent.Kafka*. El programa lee los ficheros de registro y mediante un productor de Kafka se añaden línea a línea los datos a la herramienta.

Se establecieron las configuraciones del Kafka, indicando el puerto sobre el que está montado el servicio y el *topic* en el que se quieren almacenar los datos. Posteriormente se inicializó el productor de Kafka haciendo uso de *ProducerBuilder* con las configuraciones anteriores y por cada línea en los ficheros se fueron añadiendo al Kafka con el método *ProduceAsync*.

## Estructura

La estructura empleada en esta implementación es la misma que en la anterior, pero se amplia la zona de intercambio añadiendo los Event Hubs para el tratamiento de datos en tiempo real.

En la implementación anterior se explicó que la zona de intercambio servía para limitar los accesos al Data Lake, pero en esta implementación tiene más funciones que únicamente esa. En los Data Lake solo se pueden tratar datos provenientes de ficheros, por lo que los datos que llegan por tramas en tiempo real no pueden ser tratados. La zona de intercambio sirve también como una zona donde pueden subirse datos en distintos formatos sin preocupaciones de que este formato no sea compatible con el Data Lake.

### 5.1.2.2 Flujo de datos

Se encuentran dos flujos de datos distintos:

- Hacia la derecha: este es el flujo de datos que va desde local, pasando por la zona de intercambio hasta el Data Lake donde se realiza la transformación de los datos.
- Hacia la izquierda: los datos van desde la zona Oro del Data Lake a la zona de intercambio donde podrían ser descargados.

### 5.1.2.3 Desarrollo:

#### 1- Creación de la estructura:

Se partió de la estructura usada en la implementación anterior, por lo que solo hubo que crear los Event Hubs y un nuevo *container* del Blob Storage para el nuevo flujo hacia la izquierda.

Al igual que en la primera implementación, la creación de estos componentes se hizo desde el portal de Azure.

Para crear el servicio de Event Hubs, al igual que en la mayoría de los servicios, solicitan un grupo de recursos, un nombre único y una región, pero sumado a esto también se puede escoger la tarifa que se va a utilizar el número de unidades de procesamiento, ya que nos encontramos ante una prueba de concepto, se hizo con los requisitos más bajos para mantener los costes en lo mínimo posible.

Dentro del servicio de Event Hubs, se pueden crear varios Event Hub, por lo que se creó uno llamado “intercambio” en el que se subirán los datos. El Event Hub cuenta con un servicio denominado Capture, con esto los datos que le llegan al Event Hubs se pueden guardar directamente en ficheros en un Blob Storage o en Data Lake Storage, por lo que se configuró esta herramienta para que guardase estos datos en la zona Bronce del Data Lake.

También se crea un segundo Event Hub llamado “Oro” para realizar el flujo de datos hacia la izquierda, pero en este caso no se activó la función *Capture*.

## 2- Flujo hacia la derecha:

En este flujo se deben de subir los datos a la zona de intercambio, realizar las transformaciones y visualizar los resultados.

**Subir los datos a la zona de intercambio:**

La subida de datos al blob de la zona de intercambio se realiza igual que en la implementación anterior, pero también hay que subir los datos guardados en el Kafka al Event Hubs.

Al igual que se hizo para guardar los datos en el Kafka, se desarrolló otro programa en C#, pero esta vez con un consumidor de Kafka y un productor de Event Hubs usando las librerías *KafkaNet* y *Microsoft.ServiceBus.Messaging.*

El programa crea un consumidor de Kafka en el puerto 9092 sobre el *topic* "registro", al igual que el productor. También se crea un cliente de Event Hubs para el que se necesita la cadena de conexión, la cual se puede obtener desde el portal, y el nombre del Event Hub que se vaya a usar, "intercambio" en este caso.
Posteriormente por cada dato que haya en ese *topic* en el Kafka se envía al Event Hub de la zona de intercambio.

**Transformaciones en el ADLA:**

La transformación que se realizó en este caso fue la misma que en la implementación anterior, pero los datos que se guardan haciendo uso del capture en el Event Hub, los datos no quedan guardados en formato CSV como se tenían anteriormente, sino que quedan organizados en directorios según la fecha y en ficheros de tipo AVRO.

ADL Analytics tiene soporte para estos ficheros, se utiliza el método *Microsoft.Analytics.Samples.Formats.ApacheAvro.AvroExtractor* para extraer la información de los ficheros, pero se debe de crear una estructura *json* con el esquema de los datos del fichero AVRO.

**Visualización de los datos:**

Lo mismo que en la implementación anterior.

## 3- Flujo hacia la izquierda:

Este flujo de datos consiste en la extracción de datos de la zona Oro para guardarlas en la zona de intercambio para, posteriormente, descargarlas en local.

Dentro de este flujo de datos encontramos dos distintos:

- En forma de ficheros: este flujo se compone por la copia de ciertos ficheros de la zona Oro al Blob Storage de la zona de Intercambio mediante el uso del Data Factory, y de la descarga de estos datos a ficheros en un directorio local.
- Datos en tiempo real: este es el flujo de datos que va desde la zona Oro hasta uno de los Event Hub de la zona de Intercambio haciendo uso de una Azure Function, y después de la copia de estos datos del Event Hub a un Kafka.

**En forma de ficheros:**

En este flujo se comenzó realizando la copia de los ficheros desde la zona Oro del Data Lake al *container* Oro del Blob Storage. Esto se hizo mediante el servicio de Data Factory, de la misma forma que se realiza el flujo hacia la derecha, pero cambiando los servicios origen y destino.

Posteriormente se creó un programa en C# que permitía descargar estos ficheros a un directorio local. El programa hacía uso de la librería *Azure.Storage.Blobs* que permite listar y descargar los ficheros almacenados en un contenedor.

**Datos en tiempo real:**

Para la realización de este flujo se comenzó creando una Azure Function que recorría algunos ficheros de la zona Oro del Data Lake y escribía estos datos en el Event Hub Oro.

Una Azure Function necesita permisos para acceder a los datos guardados en el Data Lake, para ello se debe de crear una aplicación desde el *Azure Active Directory*, al que se puede acceder desde el portal. De esta forma se le da permisos a la aplicación para acceder a los datos del Data Lake y se usa su identificación desde la Azure Function.

Una vez asignados los permisos, se descargan los ficheros del Data Lake en forma de Stream de datos y, línea a línea se fueron guardando en el Event Hub Oro. Para ello se tuvo que crear un *Sender* del Event Hub usando de nuevo la librería *Microsoft.ServiceBus.Messaging*.

Siguiendo estos pasos, la información que se quiere descargar ya está en la zona de Intercambio, por lo que ahora tendría que descargarse en local y añadirse al Kafka. Para ello se creó otro programa en C#.

Usando las mismas librerías que en los programas anteriores se creo un *Receiver* de Event Hub para poder acceder a sus datos, y cada dato leído en el Event Hub se iba introduciendo al Kafka.

### 5.1.3 Conclusiones

En apenas unos días de uso de la herramienta de capture de los Event Hubs esta herramienta consumió casi al completo el crédito del que disponía en Azure debido a que, a pesar de no enviarse casi datos, comprueba periódicamente si hay algún mensaje que no se haya procesado. El capture podía ajustarse en un tiempo de entre cinco y quince minutos, un periodo muy pequeño, para este proyecto interesaría un periodo de unas seis horas.

Por lo tanto se descartó seguir usando esta herramienta en las pruebas de concepto, pero no en el sistema final. Ya que, a pesar de que para la prueba los costes eran muy elevados, estos costes si podrían ser asumibles a nivel empresarial.

## 5.2 Automatización

Dado que esta tecnología podría llegar a ofrecerse como un producto final, no es conveniente que la inicialización de los sistemas y de los flujos tenga que realizarse desde el portal de Azure.Por lo que se evaluó también la capacidad de automatización de estos servicios.

Azure cuenta con una interfaz de la línea de comandos llamada **Azure CLI**. Mediante estos comandos se pueden lanzar operaciones sobre Azure desde la PowerShell de Windows.

El objetivo era crear una serie de scripts de la PowerShell de Windows que permitiesen automatizar la creación de los servicios empleados en Azure, de los flujos de datos y de los permisos necesarios para la realización de los flujos.

### 5.2.1 Identificación

Para comenzar, el usuario debe de poder identificar su cuenta de Azure donde se van a crear los servicios. Azure CLI cuenta con un comando *az login*, el cual abre una pestaña en el explorador y permite al usuario identificarse.

Posteriormente mediante el comando *az account set* el usuario podría escoger cual de las suscripciones disponibles en su cuenta va a utilizar (en caso de tener más de una).

### 5.2.2 Creación de los servicios

Posteriormente hay que crear los recursos que componen la estructura de las implementaciones anteriores, que son: el grupo de recursos, el Blob Storage, el ADLS, ADLA. Descartando ya el uso de los Event Hubs.

#### 5.2.2.1 Grupo de recursos

Para crear este servicio se dispone del comando *az group create*, donde hay que especificar un nombre y una localización. Este nombre debe de ser único, para comprobar si existe Azure CLI tiene también un comando *az group exists*, que retorna un booleano en función de si el nombre especificado existe o no.

#### 5.2.2.2 Blob Storage

Este servicio se puede crear haciendo uso del comando *az storage account create*, al que hay que especificarle el nombre del servicio, el grupo de recursos donde se va a crear y la localización. El problema con este servicio (y con los que vienen a continuación) es que el nombre debe de seguir siendo único, pero no existe ninguna forma de comprobarlo.

En el Blob Storage también hay que crear el contenedor donde se van a almacenar los ficheros. Esto se crea mediante el comando *az storage container create* donde hay que especificar el nombre del contenedor y el Blob Storage donde se va a crear.

#### 5.2.2.3 Azure Data Lake Storage

Este servicio puede crease con el comando *az dls account create*, el cual requiere especificar el nombre del servicio y el grupo de recursos.

Además de crear el servicio, también hay que crear los directorios Bronce, Plata y Oro donde se va a ir clasificando la información. Esto se puede hacer mediante el comando *az dls fs create*, donde hay que especificar el nombre del ADLS y la dirección de la carpeta.

#### 5.2.2.4 Azure Data Lake Analytics

El ADLA puede crease haciendo uso del comando *az dla account créate*, que requiere indicar el nombre del servicio, el grupo de recursos y el nombre del ADLS sobre el que se van a realizar las transformaciones.

#### 5.2.2.5 Data Factory

Al contrario que con el resto de servicios, Data Factory no dispone de unos comandos propios para su creación y gestión. Se consiguió resolver este problema haciendo uso de un comando más genérico *az group deployment create*, el cual como parámetro recibía el nombre de un fichero de tipo *json* donde se especificaba el servicio que se quería crear y sus características.

#### 5.2.2.6 Problemas encontrados

El problema que surgió al automatizar la creación de los recursos era que no se podía asegurar que los nombres que se asignaban a los recursos fuesen a ser únicos, al no poder comprobarlo. Como solo se podía comprobar el nombre del grupo de recursos, se decidió utilizar este nombre como sufijo al resto de los recursos. Así, por ejemplo, el nombre que se utilizó para el grupo de recursos en las implementaciones era *cic001*. Por lo que el nombre empleado en el ADLS fue *adlscic001*.

### 5.2.3 Flujo de datos

En estas primeras implementaciones se producían tres flujos de datos distintos en Azure:

1. La subida a la zona de intercambio: a pesar de que esto se había realizado mediante un programa en Java, mediante el uso de estos comandos de la CLI de Azure, se encontró una forma mucho más sencilla.
2. La copia de datos de la zona de intercambio a el Data Lake mediante el uso de Data Factory.
3. Las transformaciones mediante ADLA, este proceso se realizaba a demanda mediante Visual Studio, por lo que no es un proceso que pueda ser automatizado.

#### 5.2.3.1 Subida de los datos a la zona de intercambio

Para subir los datos a la zona de intercambio, es decir, a un Blob Storage se puede usar el comando *az storage blob upload-batch*, que permite, especificando el contenedor, el Blob Storage y la dirección en el sistema de ficheros local, subir un directorio entero, incluso con los directorios que pueda contener. Por lo que a partir de este momento se empezó a utilizar esta forma de subida de los ficheros sobre el uso del programa anterior.

#### 5.2.3.2 De la zona de intercambio al Data Lake:

Para poder automatizar este flujo de datos, el primer paso es dar el Data Factory los permisos para poder acceder y copiar los datos al ADLS.

- Para asignar el ACL se utiliza el comando *az dls fs access set-entry* donde se especifica el id del servicio al que se va a dar permiso y el directorio interno del ADLS al que se le da acceso.
- Para asignar los permisos de tipo IAM existe el comando *az role assignment create*, donde se especifica el id de los servicios implicados y el tipo de rol asignado.

Posteriormente había que automatizar el proceso de copia del Data Factory de los ficheros del Blob Storage al ADLS, pero no se encontró forma de hacerlo. Al no haber un grupo de comandos propio para el Data Factory como si tienen el resto de servicios no logré realizar este flujo y tras varios días sin saber que más intentar se dejo de lado este apartado.

### 5.2.4 Conclusiones

Mediante el uso del CLI de Azure se pudo automatizar todo de forma bastante sencilla, el problema apareció a la hora de automatizar el Data Factory. Esta parte se acabó abandonando por la falta de documentación sobre el tema, por lo que hay que asumir que, o bien algunas partes van a tener que seguir haciéndose desde la interfaz, o que con el tiempo estos problemas se puedan solucionar. Pero en esta prueba inicial no se pudo hacer.

## 5.3 Logs de las operaciones

Esta es una herramienta muy efectiva para detectar problemas en el sistema y determinar por qué están surgiendo. Por lo que se evaluó cómo funcionan estos registros en Azure.

En Azure el registro de la actividad viene activado por defecto, se guardan estos recursos por cada servicio de forma individual, y también en un conjunto desde el grupo de recursos. Se puede acceder a ello desde el portal de Azure en el apartado "Registro de actividad".

Los eventos vienen clasificados según su gravedad en cuatro niveles distintos: informativo, advertencia, error y crítico.

Dentro de cada registro se guarda un resumen del evento, donde muestra la información básica del evento, la fecha, el nombre de la actividad y un pequeño mensaje donde se explica brevemente el evento. También hay un *JSON* con mucha más información detallada de la operación y del error.

Esta herramienta viene inicializada directamente desde Azure en cada servicio y de forma muy completa, por lo que su uso es muy sencillo.

## 5.4 Generador de datos

A pesar de que ya se contaba con un generador de datos, este era muy sencillo, y las transformaciones que se podían realizar con él eran muy limitadas. Podrían haberse utilizado también datos reales, pero dado a que esto era una prueba, no era conveniente subir datos a la nube sin llegar a garantizar su seguridad. Por lo que se decidió crear un generador de datos más complejo con datos relacionados con los tres productos de CIC que se querían integrar.

Este generador está programado en Java y crea datos relacionados con parques eólicos en España y Portugal. El objetivo de este simulador no es ser especialmente complejo o exacto, sino ofrecer una serie de datos relacionados entre ellos para poder extraer analíticas más complejas.

Las fuentes de datos que se simularon en este generador fueron los tres productos de CIC, IDboxRT, SGRwin y Fieldeas, y una fuente de datos externa, en este caso previsiones meteorológicas.

- IDboxRT: es un producto de CIC que trata con un flujo continuo de datos en tiempo real. En este simulador sería el encargado de tratar las señales continuas de los sensores (potencia, velocidad de rotación, porcentaje y nivel de servicio de los molinos y velocidad del viento).
- SGRwin: es un producto que realiza la organización de la estructura en relaciones jerárquicas. En este caso relaciona los molinos con el parque al que pertenecen y mediante esta relación calcula el estado del parque en función del estado de los molinos que lo componen.
- Fieldeas: realiza los procesos del día a día. En caso de que se produzca alguna avería genera una incidencia y posteriormente una orden de trabajo.
- Meteorología: se crea una predicción de la velocidad del viento en los próximos siete días.

En el generador encontramos cinco entidades distintas cada una de ellas con sus propias señales:

- Parque eólico: de los parques eólicos se guarda su identificador, la empresa que gestiona el parque y sus coordenadas. A su vez en el parque se generan dos señales distintas:
    - La velocidad del viento actual del parque.
    - El nivel de servicio: este valor depende del estado de los molinos del parque.

- Molino: de cada molino se guarda su identificador, su marca, el identificador del parque al que pertenece y sus coordenadas. De los molinos también se generan las siguientes señales:
    - Potencia que el molino está generando en el momento.
    - Porcentaje servicio: muestra en un porcentaje el estado del molino.
    - Nivel de servicio: es un rango de valores (del 1 al 5) el estado del molino.
    - Velocidad de rotación del molino en el momento.

- Previsión del viento: Se genera una previsión del viento que va sucederse en la próxima semana, con el rango de vientos (velocidad máxima y mínima del día).

- Incidencia: cuando se produce una avería en un molino o el nivel de servicio de un parque es muy bajo se genera una incidencia para solucionar estos problemas.

- Parte trabajo: se genera para solventar una incidencia.

Todas las señales están relacionadas entre ellas y van evolucionando con el tiempo.

### 5.4.1 Parque eólico

#### 5.4.1.1 Entidad

Como se ha dicho antes, de los parques como entidad se guarda el identificador, la empresa y las coordenadas.

El identificador es un valor numérico que comienza en 1 y por cada parque nuevo se va aumentando este valor.

Para la empresa, se realizó una búsqueda de cuales las mayores empresas eólicas de España y se introdujeron los nombres de las empresas en un fichero de configuración. Por cada parque generado se escoge aleatoriamente una de estas empresas y se le asigna.

Para establecer las coordenadas de forma sencilla se "dibujó" un cuadrado sobre el mapa de la Península y aleatoriamente se escogían valores de coordenadas (latitud y longitud) que estuviesen contenidos en el rango de valores. Además, para evitar que dos parques estuviesen en la misma posición o muy próximos, se comprobaba que no hubiese ningún parque a menos de 10km. Esto se hizo convirtiendo, aproximadamente, la distancia entre coordenadas a metros. Si había algún parque a una distancia menor a la asignada, se repetía el cálculo.

#### 5.4.1.2 Señales

Velocidad del viento

Este valor depende de la previsión que se haya hecho para ese día. Al comenzar el día se realiza una previsión de la velocidad del viento que va a producirse en el parque a cada hora.

Para calcular la velocidad del viento en cada momento, cada vez que se cambia de hora, se calcula la ecuación de la recta entre el valor del viento actual y el que se prevé en la próxima hora. Una vez calculada, se calcula la velocidad del viento en función de la pendiente de la recta, del valor del viento actual y de un porcentaje de variación (para que los valores no sigan únicamente una recta). Y para el próximo valor se volverá a calcular la pendiente entre el punto actual calculado y el valor al que se debe llegar al final de la recta.

Nivel de servicio

Es un valor calculado que depende del porcentaje de servicio de los molinos e indica el estado del parque.

Se calcula la media de los porcentajes de servicio de los molinos que componen el parque, pero si el porcentaje del molino es superior al 80%, se cuenta este valor como si fuese al 100%, y si es inferior al 20% se cuenta como si estuviese al 0%.

### 5.4.2 Molino

#### 5.4.2.1 Entidad

Del molino se genera su identificador, su marca, el identificador del parque del que forma parte y sus coordenadas.

Al igual que con los parques, el identificador es un valor que comienza en 1 y por cada molino generado este valor va aumentando.

Para la marca también se buscaron las marcas más comunes de molinos y se introdujeron en el fichero de configuración para, posteriormente, ser asignadas a cada molino de forma aleatoria.

Las coordenadas se calculan a partir de las coordenadas el parque. Se buscó cuales eran las distancias mínimas que se pueden dar entre los molinos (ocho veces el diámetro de las aspas entre las columnas y dos veces el diámetro de las aspas entre las filas). El primer molino se coloca en las coordenadas del parque, y a partir de ahí se van calculando las distancias entre los molinos y asignando las coordenadas para mantener las distancias.

#### 5.4.2.2 Señales

Velocidad de rotación

Según la información encontrada, los molinos de viento pueden funcionar cuando la velocidad del viento se encuentra entre 3 y 25 km/h. Por lo que si la velocidad del viento no se encontraba entre estos valores, la velocidad de rotación era de 0 rpm.

Por el contrario, si la velocidad se encontraba en el rango adecuado se calculaban las revoluciones en función de la velocidad del viento.

Nivel de servicio

Esta señal indica en un rango del 1 al 5 en qué estado se encuentra el molino, siendo 5 el mejor y 1 el peor nivel. Este valor se calcula en función del estado en el que se encuentra, aleatoriamente este valor se mantiene o aumenta/disminuye en un nivel al no ser que llegue al nivel 1, en este caso se entiende que el molino esta averiado y solo puede cambiar arreglando el molino.

Porcentaje de servicio

Esta señal indica cual es el porcentaje de potencia que puede obtener el molino respecto al ideal, es decir, si el molino está al 100% va a obtener toda la potencia para la que el modelo está diseñado, pero en un molino real pueden surgir problemas que hacen que el molino no funcione a todo su potencial.

Este porcentaje va variando periódicamente, ascendiendo o descendiendo, pero siempre dentro del rango del nivel de servicio en el que se encuentre.

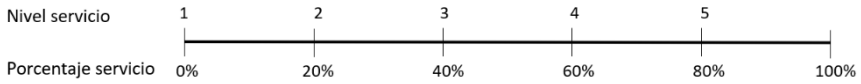


*Figura 9: Relación nivel, porcentaje estado*

Potencia

La potencia generada por los molinos fue calculada haciendo uso de la siguiente fórmula:

$$P = \mu \cdot p \cdot A \cdot v^3$$

Donde:

- P es la potencia calculada en W/m$^2$.
- μ es la eficiencia del molino. Este factor depende de dos elementos distintos.
    - Por un lado de la eficiencia del modelo, es un factor que no varía, generalmente los molinos pueden obtener el 40% de la potencia ideal
    - Del porcentaje de servicio del molino.
- $p$ es la densidad del aire.
- A el área que recorren las aspas.
- v la velocidad del viento.

### 5.4.3 Previsión de la velocidad del viento

Para simular la previsión del viento, primero se clasificó el viento según sus velocidades y a cada uno se le asignó una probabilidad distinta, quedando de la siguiente forma:

*Tabla 3: Niveles de viento*

| Nombre | V. mínima (km/h) | Probabilidad (%) |
|---|---|---|
| Tranquilo | 0 | 5 |
| Suave | 6 | 7.5 |
| Moderado | 13 | 45 |
| Vivo | 30 | 25 |
| Fuerte | 40 | 10 |
| Muy fuerte | 62 | 5 |
| Masivo | 88 | 2 |
| Huracanado | 117 | 0.5 |

Inicialmente se crea una previsión para los siete días siguientes donde se asigna uno de los tipos de vientos teniendo en cuenta la probabilidad y después se calcula el rango de velocidades del viento en ese día dentro de la franja de valores establecidos y la previsión de viento a cada hora.

Posteriormente, a medida que se cambia de día en el simulador, estas predicciones pueden ir modificándose, como ocurre en la realidad. Pero estos valores tendrán una menor probabilidad de cambiar cuanto más cercano sea el día.

*Tabla 4: Margen cambio predicción*

| Margen(días) | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| Probabilidad cambio | 5% | 10% | 15% | 20% | 25% | 30% | 30% |

Para simular el valor del viento a cada hora y que los datos no sean demasiado cambiantes y que se aproximen a las gráficas del viento normales se favoreció la tendencia que seguían estos niveles. Es decir, si el viento en la última hora estaba en aumento, si prioriza que siga aumentando

y lo mismo si está disminuyendo. Posteriormente se calcula el nuevo valor de la velocidad del viento con mayor prioridad de que el cambio no sea muy brusco.

Además de esto, para hacer el viento algo continuo entre los días, si el viento con el que se termina un día no esta dentro del rango de velocidades del día siguiente se establece un periodo de ajuste, donde en las próximas horas se cambia este valor para que progresivamente se ajuste al rango.

### 5.4.4 Incidencia

Una incidencia se puede producir por dos causas, o bien un molino está averiado, o el nivel de servicio de un parque está muy bajo. La incidencia guarda la información de cual ha sido el problema, la fecha en la que se ha producido y la gravedad.

La gravedad es un rango entre el 1 y el 5 calculado de forma aleatoria.

### 5.4.5 Parte de trabajo

Cuando se gestiona una incidencia se crea un parte de trabajo para arreglar el problema. Un parte de trabajo indica que es lo que se ha arreglado, el tiempo que ha llevado y el identificador de la incidencia.

El tiempo que lleva el arreglo depende de la gravedad que se indique en la incidencia, y del tipo de problema que sea. La reparación de un molino lleva menos que la de una incidencia del parque en general.

Si el parte de trabajo era sobre un molino, el nivel de servicio de este se pone al máximo tras la reparación y el porcentaje de servicio se establece entre el 80-100%.

Si el parte es sobre un parque eólico, se realiza esto mismo con todos los molinos cuyo porcentaje de servicio sea menor al 60%.

### 5.4.6 Puesta en común

Como bien se indicó anteriormente, todas estas señales están conectadas entre ellas. El simulador comienza creando treinta parques eólicos con un número inicial de diez molinos en cada uno de ellos.

Cada una de las señales y entidades va cambiando con una periodicidad distinta y cada vez que estos se modifican se escriben los nuevos valores en ficheros CSV, creando ficheros distintos cada día simulado.

Por cada día simulado de calcula y actualizan los nuevos valores de las predicciones del viento por cada parque eólico. Además hay una probabilidad de que vayan apareciendo más parques eólicos y que más molinos se vayan construyendo en los parques ya existentes hasta llegar a un máximo.

Cada 10 horas se calcula el nuevo valor del nivel de servicio de los molinos, y cada 5 minutos el porcentaje de servicio de los molinos y del parque. Si por alguna de estas modificaciones se produjese una incidencia esta se inicializa y se calcula el tiempo que tardaría en gestionarse.

Además cada treinta segundos se simula el nuevo valor de la velocidad actual del viento en cada uno de los parques, y junto a ello la potencia y velocidad de rotación de cada uno de los molinos. También se comprueba si hay alguna incidencia o parte de trabajo que gestionar y si es el caso, se gestiona.

## 5.5 Transformaciones con ADLA con los nuevos datos

Para realizar estas transformaciones se volvió a montar una estructura de Data Lake como en la primera implementación. Con una zona de intercambio compuesta únicamente por un Blob Storage, el Data Lake con las tres zonas, Bronce, Plata y Oro, y el Data Factory haciendo la transferencia de los datos.

En esta fase se plantearon cuatro transformaciones distintas:

- Selección de las averías de molinos dentro de las incidencias. Es una transformación que coge los datos de las incidencias de la Zona Bronce y guarda los resultados en la zona Plata.
- Transformación para poder mostrar los molinos averiados en un mapa. Extrae los datos de las zonas Bronce y Plata y el resultado, ya final, se guarda en la zona Oro para realizar la visualización de los resultados.
- Selección de las potencias generadas por un parque eólico en particular. Extrae los datos de la zona Bronce, los transforma y guarda en la zona Plata.
- Cálculo de la potencia media por cada hora en un parque eólico. Extrae los datos de las potencias de los parques de la zona Plata y guarda el resultado en la zona Oro.

### 5.5.1 Selección averías molino

Haciendo uso de ADLA se puede escoger el número de unidades de análisis para la realización de una transformación, pero no todo el proceso de la transformación puede aprovechar o necesita de el mismo número de unidades de análisis. Para ajustar al máximo los costes y el tiempo, estas transformaciones se pueden dividir en varias partes y asignar un número distinto de unidades a cada una de ellas.

En este caso el objetivo final es poder mostrar en un mapa los molinos averiados, y esta es la primera parte de la transformación, donde se extraen los datos de las incidencias que se encuentran en la zona Bronce, se seleccionan únicamente aquellas que indiquen averías en molinos y el resultado se guarda en la zona Plata para usarse posteriormente.

### 5.5.2 Transformación para mostrar averías en un mapa

Para realizar esta transformación se extrajeron los datos sobre los molinos en la zona Bronce, donde se indica el identificador del molino y sus coordenadas entre otros datos y los datos de la transformación anterior de la zona Plata.

Se pusieron en común los datos de tal forma que en los molinos donde se había producido alguna incidencia se guardaban las coordenadas del mismo, su identificador y la fecha de la avería. Estos datos se guardaron en la zona Oro ya listos para visualizarse.

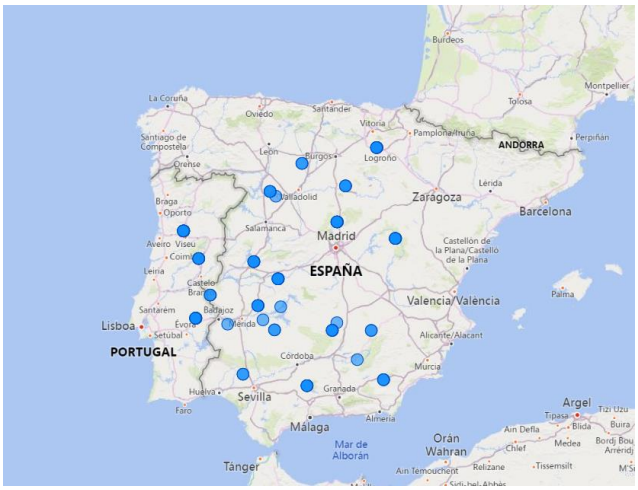

*Figura 10: Visualización averías*

### 5.5.3 Selección potencias parque

Los datos de las potencias se guardan en común identificando únicamente el identificador del molino, por lo que en esta transformación se seleccionan las potencias de un parque en específico.

Para ello se extrajeron los datos de los molinos y de las potencias en la zona Bronce y se seleccionaron únicamente los datos de las potencias cuyos molinos estuviesen en el parque indicado, guardando estos datos en la zona Plata.

### 5.5.4 Cálculo de la potencia media por hora

Para realizar esta transformación se partió de los datos generados en la transformación anterior, calculando la potencia media por cada hora en un parque en concreto.

Se comenzó cambiando el formato de la fecha, para poder clasificar los datos en una hora y una fecha en concreto, permitiendo ordenar y agrupar esta fecha de forma sencilla e ignorando los minutos y segundos mediante instrucciones de SQL. Esta transformación al ser algo más compleja, se usó una función en C# a la que se llamó desde la zona de código en C#.

Posteriormente se agruparon los datos en función de la fecha calculando la media de cada hora y se ordenaron los datos de forma ascendente según la fecha, guardando estos datos en la zona Oro.

## 5.6 Implementación mediante Azure Databricks

Azure Databricks es otra de las tecnologías que se evaluó para ser usada en el Data Lake.

### 5.6.1 Inicialización del servicio

Esta herramienta se ofrece desde el portal de Azure y para crearla hay que especificar un nombre, una localización y un plan de tarifa, para esta prueba se utilizó el plan estándar.

Para poder acceder a los datos almacenados en el ADLS hay que dar a este servicio los permisos ACL y IAM. Para ello se crea una aplicación de Azure y se le dan estos permisos a la aplicación. Posteriormente, se debe de montar un punto de acceso al ADLS, para ello se deben de aportar los credenciales de la aplicación al que se le asignaron los permisos e indicar la ruta al que se le da acceso dentro del ADLS.

### 5.6.2 Transformaciones

En Azure Databricks las transformaciones se programan en *Notebooks* y se pueden emplear distintos lenguajes, en este caso se escogió Python con el módulo *PySpark*.

Las transformaciones que se hicieron en esta implementación son las mismas que se realizaron en la implementación anterior con ADLA, como el objetivo es evaluar la tecnología al utilizar las mismas transformaciones es más fácil comprobar que la transformación se ha realizado de forma correcta.

Para poder realizar operaciones con Spark hay que añadir los datos de los ficheros a lo que se denomina el contexto de Spark, para ello se usó el comando *sc.textFile* donde se especifica la ruta de los ficheros. Posteriormente se crean los esquemas de los datos haciendo uso de la instrucción *sqlContext.createDataFrame* para poder identificar los distintos campos.

Las transformaciones se han realizado mediante programación funcional haciendo uso de las siguientes funciones:

- *map:* permite aplicar una función a todo el conjunto de datos.
- *filter:* filtra los datos respecto a una función.
- *join:* permite unir dos conjuntos de datos distintos.
- *lambda:* se usan en el interior de las funciones anteriores, permite inicializar una función sin tener que identificarla.

Finalmente al realizar las operaciones deseadas se guardan los datos mediante la instrucción *coalesce(1).write.format('com.databricks.spark.csv')*, que permite guardar los datos en formato CSV. Mediante el uso de esta instrucción no se especifica el número de ficheros finales o el tamaño de estos, se ajusta automáticamente al tamaño que más se ajuste a Spark.

A pesar de que estas transformaciones funcionaban correctamente, Spark esta optimizado para operar con ficheros de tipo *Parquet*. Estos ficheros están organizados por columnas lo que permite ahorrar tiempo a la hora de procesar datos de tipo *Big Data* ya que permite no realizar lecturas completas del archivo, escogiendo solo la información necesaria para la operación. Por lo que en las transformaciones intermedias (de Bronce a Plata) se realizó la implementación con este tipo de ficheros, pero se mantuvieron los CSV en el resto, ya que los datos estaban generados en ese formato y la visualización de los datos estaba planteada para leer ficheros de tipo CSV.

## 5.7 Azure Synapse

Una vez implementado lo que sería una solución *ad-hoc* de un Data Lake, con transformaciones mediante el uso de ADLA y Databricks, se quería comparar el rendimiento y facilidad de uso con Azure Synapse.

### 5.7.1 Almacenamiento

El almacenamiento en el Azure Synapse se almacena en tablas, para poder insertar datos a este servicio se realizó una copia de los datos almacenados en el ADLS. Pero estos datos no tienen un esquema que defina su estructura, puesto que este se genera al iniciar las transformaciones, por lo que lo primero fue crear este esquema.

Este esquema se denomina tabla externa, para crearlo se hizo uso de Azure Data Studio. Esta aplicación permite acceder a los servicios de Azure y realizar operaciones a partir de *Notebooks.* Se creó un esquema con una tabla externa para cada uno de los distintos tipos de datos del ADLS, definiendo los campos que tiene cada uno.

Una vez definidas las tablas externas se crearon las tablas en el Azure Synapse y se realizó la copia de los datos entre una y otra.

### 5.7.2 Transformaciones

Para comenzar se realizaron transformaciones desde el portal de Azure haciendo uso de la herramienta de Azure Synapse llamada *Editor de Consultas.* Esta herramienta permite visualizar y transformar los datos mediante transformaciones de tipo SQL, se utilizó para comprobar que los datos estaban bien cargados.

Las transformaciones anteriores funcionaban correctamente, pero esas transformaciones no son de tipo *Big Data*. Azure Synapse está directamente integrado con herramientas de tipo *Big Data* como con Azure Databricks, pero finalmente se vio que esta herramienta solo estaba en versión preliminar y que, en verdad, solo contaba con el almacenamiento de Data Warehouse y ningún servicio más, por lo que la evaluación de esta parte tuvo que apartarse.

### 5.7.3 Conclusiones

Desde el comienzo del proyecto se planteó comparar la solución *ad-hoc* desarrollada durante el proyecto con Azure Synapse, y desde Microsoft se vendía como un servicio nuevo ya prácticamente completo, pero solo se puede acceder a una versión previa, la cual no ofrece los servicios que interesaba probar.

El servicio que se ofrece actualmente con Azure Synapse es el antiguo Data Warehouse de Azure, en el que no se pueden realizar transformaciones de tipo *Big Data*, por lo que no es algo que interesase en esta evaluación.

## 5.8 Prueba de carga

Se quería evaluar la capacidad de paralelización de las herramientas empleadas, Azure anunciaba un escalado prácticamente lineal, por lo que se quiso comprobar.

Para esta prueba de carga se generó un total de 110GB de datos, en términos de *Big Data* no estamos ante un gran volumen, pero se consideró suficiente para comprobar el rendimiento de los servicios.

### 5.8.1 Azure Data Lake Analytics

Este servicio permite escoger el número de unidades de análisis a emplear en una transformación en un rango de entre 1 y 32 unidades, por lo que se realizó la prueba haciendo uso de una única unidad, de diez y de treinta para tener una visión de como van modificando los tiempos.

Se obtuvieron de media los siguientes resultados:

*Tabla 5: Prueba de carga de ADLA*

| Unidades | Tiempo ejecución (min) | Tiempo efectivo (min) |
|---|---|---|
| 1 | 162 | 162 |
| 10 | 16,7 | 167 |
| 30 | 7 | 210 |

Se puede observar en la Tabla 5 que, aunque los datos no son realmente lineales, si que la paralelización es bastante efectiva, reduciendo de forma drástica los tiempos de ejecución.

### 5.8.2 Azure Databricks

Con este servicio se crean *clusters* donde se establece el rango de unidades de análisis a emplear, para que fuesen similares a la prueba hecha con el ADLA se realizaron las siguientes pruebas:

*Tabla 6: Prueba carga de Databricks*

| Unidades | Tiempo1(minutos) | Tiempo2 |
|---|---|---|
| 1-2 | 31 | 31 |
| 10-11 | 6,5 | 65 |
| 30-31 | 2,6 | 78 |

Con este servicio se observa que la capacidad de paralelización no es tan lineal como en el servicio anterior, pero los tiempos obtenidos son mucho menores que usando ADLA.

## 5.9 Machine learning

Además de evaluar la capacidad de integración del Data Lake con las herramientas de *Big Data*, también se quería comprobar esta capacidad con las herramientas de Machine Learning, de entre las tecnologías ofrecidas por Azure se decidieron usar los Servicios Cognitivos de Microsoft.

Para integrar el uso de estas tecnologías en el sistema, se simuló que los operadores del parque puedan describir con un audio los arreglos hechos en el parque eólico y que, mediante los servicios cognitivos, este audio pueda ser convertido a texto y guardado en el ADLS.

El audio comenzaría almacenado en local, este se sube a la zona de intercambio y mediante el uso de una función de Azure se transforma el audio a texto y se guarda el resultado en el ADLS.

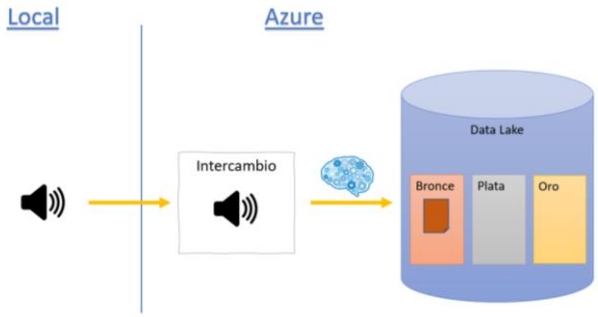


*Figura 11: Flujo Datos Servicios Cognitivos*

### 5.9.1 Inicialización de los Servicios Cognitivos

El servicio que se va a utilizar es el *Speech*, el cual se puede inicializar desde el propio portal. Este servicio ofrece varias tarifas, pero para esta prueba se escogió la opción gratuita ya que cubría lo que se estaba buscando.

### 5.9.2 Desarrollo

Mediante los *script* creados en el apartado de Automatización se añadió al Blob Storage de la zona de intercambio un nuevo contenedor al que subir los datos y se realizó la subida de los mismos.

Para realizar la transformación de los datos con los Servicios Cognitivos se usó una Azure Function que se disparaba cada vez que un fichero era subido al contenedor del Blob Storage. En esta función se lanza una petición http haciendo uso de la API del servicio cognitivo y se realiza la transformación de los datos. Finalmente el texto transcrito se añade a los ficheros del Data Lake.

El problema es que no se consiguió que la petición http funcionara de forma correcta accediendo a los datos del Blob Storage. Se intentó hacer el mismo flujo pero accediendo a los datos desde local y esto si que se pudo realizar correctamente.

### 5.9.3 Conclusiones

No se logró conectar correctamente los servicios cognitivos con la estructura diseñada del Data Lake, a pesar de que debería de ser una implementación sencilla no se supo resolver el problema encontrado, y la falta de documentación no aportó otras formas de solucionarlo.

En teoría si que se ofrece la posibilidad de usar este servicio dentro de la estructura planteada, por lo que no se da por descartado la implementación con herramientas de Machine Learning. Pero la documentación es prácticamente inexistente y lo que se pudo encontrar hasta el momento no fue suficiente.

A pesar de que no se consiguió que este servicio funcionase, el flujo de datos usando las *Azure Functions* funcionó correctamente usando el disparador que detecta si nuevos ficheros han sido subidos al blob, lo que ofrece otra forma de analizar los ficheros que no se había probado anteriormente.

# 6 Conclusiones

El objetivo del proyecto era evaluar las tecnologías de Data Lake y comprobar si su uso era viable dentro de CIC. Tras la realización de las distintas pruebas e implementaciones se ha llegado a la conclusión de que es un servicio que podría funcionar correctamente y que aporta múltiples ventajas.

Es una tecnología que permite analizar un gran volumen de datos con herramientas con una gran escalabilidad y un rendimiento muy alto. Además, al usarse dentro de Azure, nos evitamos la tarea de tener que montar y mantener los sistemas *in-house* donde se va a realizar el almacenamiento y las transformaciones de los datos, una tarea muy complicada que hacía que estas tecnologías solo estuviesen disponibles para las grandes empresas.

Siendo este proyecto la primera ocasión en la que he trabajado con tecnologías en la nube, y en especial con los Data Lakes, me he tenido que enfrentar a muchos problemas que no sabía cómo solucionar y de los que no pude encontrar mucha documentación, pero que buscando soluciones alternativas y junto a toda la ayuda proporcionada desde la empresa me han hecho aprender mucho sobre estas tecnologías y también a buscar otros enfoques a los problemas encontrados.

# 7 Trabajos futuros

Durante este proyecto ha habido ciertos puntos que no se han podido resolver correctamente, como la automatización de todo el sistema o la implementación con las herramientas de *Machine Learning*. Por lo que este proyecto continuará tratando de resolver estos elementos que no se han podido llevar a cabo.

Además, Azure está ofreciendo lo denominado Nube Híbrida, que permite utilizar las instancias locales de las que disponga el usuario y, una vez superadas sus capacidades, seguir operando con los servicios en la nube para aumentar la capacidad.

CIC dispone de varios servidores locales, por lo que el próximo paso sería comprobar el uso de estas tecnologías haciendo uso de la Nube Híbrida. De esta forma se podrían operar datos más sensibles ya que el almacenamiento se haría de forma local y, además se aprovechan los recursos con los que ya cuenta la empresa.

# 8 Bibliografía


1) Mohanty, S., & Jagadeesh, M., & Srivatsa, H. (2013). *Big Data Imperatives.* Recuperado de: https://link-springer-com.unican.idm.oclc.org/book/10.1007%2F978-1-4302-4873-6

*2)* Introducción a U-SQL en Azure Data Lake Analytics. Disponible en: https://docs.microsoft.com/es-es/azure/data-lake-analytics/data-lake-analytics-u-sql-get-started

*3)* Funciones Lambda en Python. Disponible en: https://www.analyticslane.com/2019/06/26/funciones-lambda-de-python/

*4)* PySpark package. Disponible en: https://spark.apache.org/docs/latest/api/python/pyspark.html

*5)* Introducción a Azure CLI. Disponible en: https://docs.microsoft.com/es-es/cli/azure/?view=azure-cli-latest

*6)* *Azure Data Lake Storage Gen1- Databricks Documentation.* Disponible en: https://docs.databricks.com/data/data-sources/azure/azure-datalake.html

*7)* Reconocimiento de voz almacenada en Blob Storage. Disponible en: https://docs.microsoft.com/es-es/azure/cognitive-services/speech-service/quickstarts/from-blob?pivots=programming-language-csharp

*8)* Documentación de Azure Functions. Disponible en: https://docs.microsoft.com/es-es/azure/azure-functions/