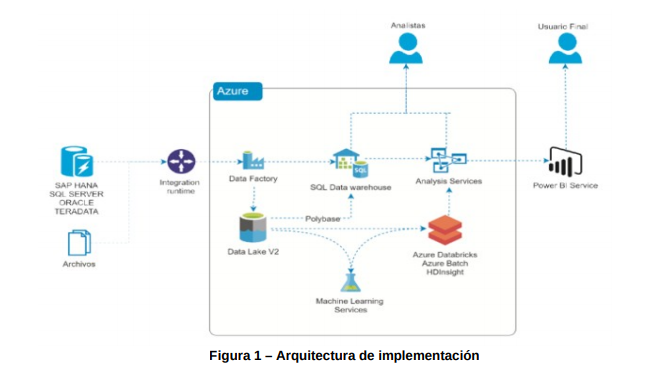
Ejercicio 3

1 INTRODUCCIÓN El documento tiene como objetivo presentar la arquitectura creada en Microsoft Azure on-cloud para la solución de Laguna de Datos de la compañía Té-dé-pastor. El documento contiene una descripción general de arquitectura y componentes con un anexo de metodología de trabajo en la empresa.



2 ARQUITECTURA LAGUNA DE DATOS AZURE

Componentes

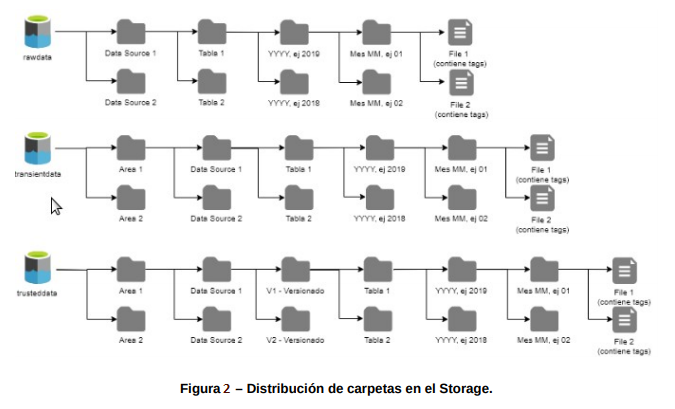
**Integration Runtime**: Componente que se utiliza de gateway entre los orígenes de datos, bases de datos, archivos, etc y la plataforma de solución de Laguna de datos en la nube de Microsoft Azure.

**Azure Data Factory**: Componente de diseño de ELT (Extract Load and Transform), este componente nos permite diseñar en modalidad 4GL, (similar al Integration services de MS) los caminos de extracción, JOIN, MERGE y transformación de los datos para ser almacenados en nuestro Storage Data Lake V2.

Acá se realizarán los pipelines de ingesta de datos. Para tablas con campos de auditoria se creará un pipeline delta que lea los datos diariamente y serán procesados en una notebook Databricks en donde se hará el upsert correspondiente.

Estos pipelines tendrán manejo de errores en donde se insertará en una tabla log toda la información de la corrida de los mismos. Hora de comienzo y fin, nombre del pipeline, trazabilidad de las actividades que conforman el pipeline para que en el momento de que ocurra algún error nos indique en que parte del pipeline ocurre el error. Como anexo estos pipelines tendrán asociada una Azure function que enviara un mail en caso de error o éxito en la ejecución de los mismos.

**Azure Data Lake V2**: Servicio de Storage de Microsoft Azure. Este servicio es similar al conocido AWS S3, es un servicio de Storage que nos permite almacenar en Carpetas la información en nuestro DataLake, la distribución de carpetas y el acceso a las mismas es totalmente parametrizable, pudiendo otorgar permisos independientes de acceso por Áreas. Un típico caso de distribución de carpetas es el que se presenta en la siguiente imagen:



En esta imagen pueden observarse diferentes Blobs o discos (rawdata, transientdata y trusteddata).

En **rawdata (Golden info)** se almacenan aquellos “datos transaccionales sin procesar”, aquellos datos que son extraídos desde la fuente de datos y almacenados en el Storage, los datos pueden ser extraídos por procesos batch o por consumo de una queue como Kafka, MQ o el propio Azure Storage Queue Service.

En **transientdata** se define un espacio de trabajo temporal y permanente para las áreas de la empresa, muchas veces las áreas de las empresas obtienen información de diversos sistemas y fuentes y realizan el cruce con herramientas como Excel. Estos repositorios son creados a tal fin, de manera de permitir el almacenamiento en el Storage. Cada espacio de trabajo es independiente y con permisos propios. Esto evita que las áreas internas utilicen recursos externos como Google drive u otro medio de almacenamiento.

En **trusteddata (Silver Data)** se almacenan los datos consolidados, aquellos datos definidos en estructura prediseñados que pueden generar historial y alimentar un informe o tablero de Frontend.

Y se crea un nuevo repositorio **Gold Data** que contiene información que solo necesita el usuario final sin tener la necesidad de acceder a una tabla completa. Puede ser guardada en tablas tabulares.

**Azure SQL Data Warehouse**: Servicio de Data Warehouse SQL Server 2019, el cuál dispone de la característica de conexión Polybase el cual nos permite generar tablas las cuáles apuntan a carpetas definidas en el Blob Storage de Trusteddata. Esto es, si en las Carpetas definidas en trusteddata agregamos los .csv, .json o .parquet, con una estructura pre-definida, a los registros o archivos sucesivos que se agreguen al folder se verán como registros nuevos en las tablas creadas.

**Azure Analysis Services**: Servicio de disponibilidad de datos en memoria. Acceso de gran velocidad y disponibilidad.

**Azure DataBricks:** Es un servicio de análisis rápido, sencillo y de colaboración basado en Apache Spark, con este componente tenemos la posibilidad de empezar y no descuidar la rama de la analítica avanzada, utilizando cluster y notebooks, a través de una interfaz Jupyter podemos interactuar con información del Storage para realizar analítica. El resultado de ese análisis puede ser presentado en Azure Analysis Services para conectar desde un Excel hasta un PowerBI y poner en disponibilidad la información analizada al negocio.

Este servicio será utilizado para manipular la información que tiene como origen por ejemplo un archivo .parquet y como destino una delta table particionada por fecha preferentemente, lo cual ayuda tanto a la velocidad de procesamiento como también a la búsqueda de un conjunto de registros en particular.

3 ANEXO – METODOLOGÍA DE IMPLEMENTACIÓN

Los siguientes pasos pretenden definir el proceso de desarrollo e implementación de proyectos en Azure DataLake, Teniendo en cuenta modelos ágiles de implementación en continuo delivery y continuo Integration (CI-CD). Debe considerarse:

3.1 Determinar los actores:

PM (Scrum master)

Arq. (Arquitecto de datos)

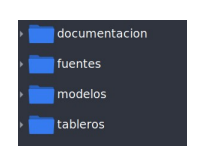
AFU (Analista Funcional, QA tester)

DI (Ingeniero de datos)

3.2 Metodología general (Workflow de trabajo)

• Crear proyecto en JIRA. (PM)

• Crear proyecto en GITLAB. (Arq.) Se define el proyecto en GITLAB con la siguiente estructura de directorios.



• Realizar el análisis funcional, relevando el requerimiento del cliente o interno, identificar los orígenes de datos (general, bases de datos, por ejemplo), relevar el requerimiento de proceso de transformación requerido (general) y primer bosquejo de prototipo de Tablero o archivo o salida. Disparador -> JIRA generado por cliente, pedido o requerimiento interno. (AFU o analista)

• Crear documento con el objetivo del proyecto, alcance, requerimientos funcionales identificados y requerimientos no funcionales identificados. Versionar en GITLAB. (AFU o analista)

• En caso de un proyecto de cliente interno, aprobar por el cliente el documento de solicitud de requerimientos y prototipo anexado, solicitar aprobación por mail. (AFU o analista)

• Identificar los orígenes de datos (Bases de datos y tablas identificadas en el proceso de extracción), en caso de servicios REST o SOAP o colas MQ o Kafka (relevar información de conexión, key, usuarios, pwd, IP, etc.).

En el proceso de identificación de orígenes de datos se debe relevar el significado de las columnas más significativas. Versionar en GITLAB un Excel resultante (AFU o analista)

• Crear primer bosquejo de diagrama de ELT. Diagrama descriptivo del proceso de extracción de información de los orígenes de datos, carga en destino y transformación. Versionar en GITLAB. (Arq.)

• Realizar análisis de la documentación, códigos fuentes actuales, modelos actuales, requerimientos funcionales y no funcionales. (DI)

• Realizar la implementación en Data Factory. Realizar calendarización establecida en el diseño de ELT si fuese necesario. (DI)

• Realizar la implementación en DL, crear estructura de directorios. (DI)

• Realizar la implementación en DW, implementar el modelo físico a partir del modelo lógico si fuese necesario. (DI)

• Realizar las pruebas unitarias. (DI)

• Versionar en GITLAB los procedimientos almacenados creados si fuese necesario. (DI)

• QA: Realizar las pruebas en casos de prueba, regresión e integridad si fuese necesario. (AFU o analista)

• Si el proyecto requiere Analysis Services (Datamarts) y tableros. Solicitar a frontend, el desarrollo. (PM)

• Implementar en producción los cambios de backend. (DI)

• Tareas cross al proyecto: (PM)

• Seguimiento del proyecto, revisión de la documentación (formato, prototipo, aprobación del cliente),

• Control de comentarios diarios en JIRA por parte de los ingenieros de datos.

• Realizar dos checkpoints semanales de avance de las tareas del backlog con todo el equipo.

• Realizar planning con todo el equipo los siguientes lunes luego de la finalización del sprint anterior.

• Realizar retrospectiva.