**PROYECTO A DESARROLLAR: SODIMAC**

1. **TABLERO SCRUM** (Detalle de actividades)

**Nombre Historia:** Compra por internet

**Descripción de la funcionalidad:** Caso que consiste en que uno o más clientes desde un pc, laptop u otro dispositivo realice una compra por Internet en la página Web de la Empresa Sodimac.

|  |
| --- |
| **Actividades:**   1. Solicitud de Registro por correo electrónico      1. Elegir opciones de despacho – Despacho a Domicilio y Despacho en Tienda          1. De la lista mostrada se selecciona una tienda para el despacho.      1. Persona que retira el Producto en Tienda      1. Pago del Producto – Elegir medio de pago |

1. **CASO DE USO**

Actualmente los clientes realizan gran volumen de transacciones online para realizar compra de diversos productos de acorde a las necesidades, las cuales pueden encontrarlas a través de la plataforma SODIMAC.

<https://www.sodimac.com.pe/sodimac-pe/>

Por tal motivo, este proyecto califica como proyecto de Big data al contar con las siguientes características:

* Base de datos a explotar de los diversos productos que ofrece SODIMAC.
* La información de los productos es visualizada por los diversos clientes.
* Los clientes pueden realizar compras online de estos productos
* Se puede predecir los productos que se necesitan como stock para cubrir la demanda en el SODIMAC.



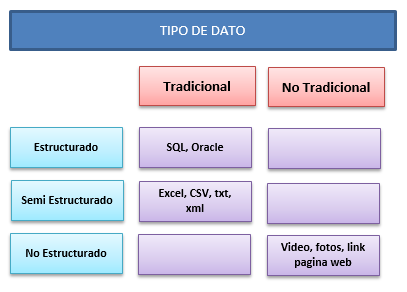
1. **NECESIDAD DE USAR EL BIG DATA**

El uso del Big Data en el caso de uso en estudio justifica mediante la definición de las 3V’s del Big Data.

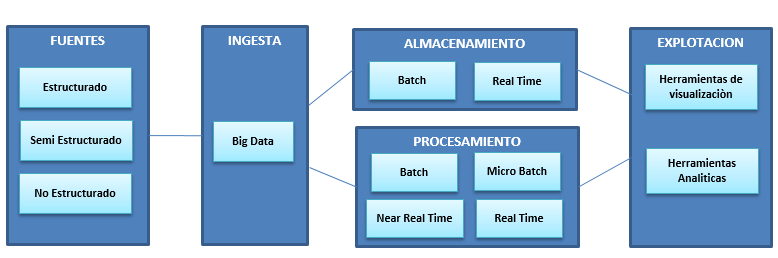
* 1. **Volumen:**
  + SODIMAC maneja gran volumen de información de transacciones que realizan sus clientes al momento de realizar sus compras online.
  + SODIMAC maneja gran volumen de productos, características y/o especificaciones técnicas
  1. **Velocidad:**
  + Los datos se generan en línea (web) dado que las transacciones se realizan de manera muy rápida, ya sea a nivel de stock de productos, tanto para la compra de productos como el pago respectivo.
  + Validación de datos del cliente en línea (usuario y crédito disponible) del medio de pago a usar
  1. **Variedad:**
  + SODIMAC maneja diversas fuentes de datos estructurada (base de datos relacionales SQL, Oracle), semi estructurado (archivos excel, csv, txt y xml) y no estructurada (videos y fotos)



1. **FUENTES EXTERNAS O INTERNAS**
   1. Fuente Externa: Medios de pago a través de tarjetas de crédito o débito.
   2. Fuente Interna: Base de Datos de productos
2. **SIZING DE LA GENERACIÓN DE DATA POR MINUTO, DÍA, HORA Y MENSUAL.**
3. **FUENTES DE INFORMACIÓN**



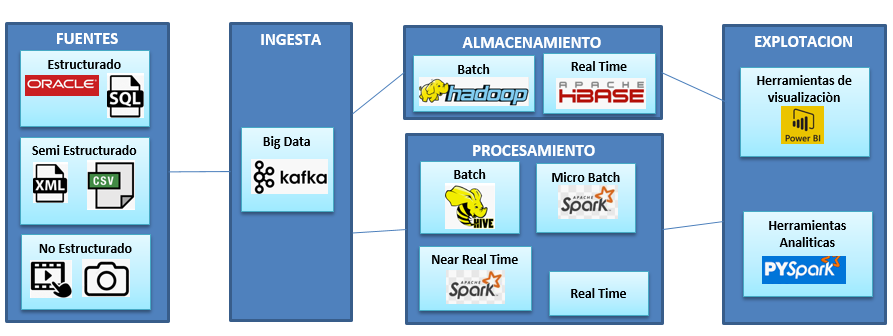
1. **ARQUITECTURA CONCEPTUAL**



1. **ARQUITECTURA TECNOLOGICA**

Respecto a la Arquitectura utilizada en el proyecto se ha tomado en cuenta la Arquitectura Lambda que realiza procesamiento en modo batch y en modo Stream.

La siguiente figura muestra la Arquitectura Tecnológica que se ha diseñado para el proyecto.



1. **PERFILES DEL BIG DATA**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PERFIL** | **TECNICO** | **NEGOCIO** | **ETAPA** | **FASE** |
| **DATA ENGINEER** | Alto: Programación | Alto: Encargado de la data de visualización | Captura | Producción |
| Bajo: Conocimiento de arquitectura y cloud | Bajo: Conocimiento del negocio | Almacenamiento |  |
|  | Apoyo con análisis de información | Análisis |  |
|  |  | Visualización |  |
| **DATA SCIENTIST** | Alta: Capacidad análisis programación,  estadística | Medio: conocimiento del negocio | Análisis | Exploratorio |
|  | Debe saber contar la historia sin términos técnicos. | Visualización | Desarrollo |
|  | Ve el modelo como un servicio | Realiza, transformaciones, visualizaciones,  análisis con el objetivo exploratorio y de desarrollo |  |
| **DATA QUALITY** | Bajo: Programación (verificación de datos de origen y destino) | Bajo: Conocimiento negocio | Esta afuera y dentro con tecnología antes del almacenamiento |  |
| Control (Algoritmos de control) |  |  |  |
| **DATA ARCHITECT** | Bajo: Programación | Bajo: Conocimiento del negocio | Por fuera de las fases |  |
| Alto: Tecnología | Diversidad de áreas |  |  |
| **DATA EXPERT** | Nada técnico | Alto: Conocimiento de negocio | Conoce los datos y variables que tiene el negocio |  |
| **DATA ANALYST** | Bajo: Programación y reportes | Medio: Conocimiento de negocio | Análisis |  |
|  | Medio: Presentación de la historia (story telling) | Visualización |  |
| **CHIEF DATA OFFICER** | Medio: Tecnología | Alto: Conocimiento del negocio |  |  |
| Nada: Programa | Alto: Presentación de la historia (story telling) |  |  |
|  | Rol de gerente de data |  |  |
| **HADOOP ADMIN** |  | DBA de Big Data | Almacenamiento |  |
| **CLOUD ARCHITECT** | Trabaja de la mano con Data Architect |  |  |  |

1. **INGESTAR UNA O MÁS FUENTES A UN DATALAKE (Google cloud) UTILIZANDO COMANDOS HDFS**

|  |
| --- |
| 1. **CREAMOS EL DIRECTORIO HDFS:**   hdfs dfs -mkdir /proyecto  hdfs dfs -mkdir /proyecto/producto  hdfs dfs -mkdir /proyecto/cliente  hdfs dfs -mkdir /proyecto/tienda  hdfs dfs -mkdir /proyecto/distrito  hdfs dfs -mkdir /proyecto/comprobante  hdfs dfs -mkdir /proyecto/transacción  hdfs dfs -mkdir /proyecto/transacción\_particion\_dinamica  hdfs dfs -mkdir /proyecto/transacción\_particion\_estatica  **Visualizar el directorio creado**  hdfs dfs -ls /proyecto     1. **COPIAMOS LOS ARCHIVOS PLANOS AL DIRECTORIO HDFS**   hdfs dfs -put /home/dataproc/proyecto/producto.txt /proyecto/producto  hdfs dfs -put /home/dataproc/proyecto/cliente.txt /proyecto/cliente  hdfs dfs -put /home/dataproc/proyecto/tienda.txt /proyecto/tienda  hdfs dfs -put /home/dataproc/proyecto/distrito.txt /proyecto/distrito  hdfs dfs -put /home/dataproc/proyecto/comprobante.txt /proyecto/comprobante  hdfs dfs -put /home/dataproc/proyecto/transaccion.txt /proyecto/transaccion  **Visualizar archivos copiados en el directorio Proyecto**  hdfs dfs -ls /proyecto/             1. **CREAMOS LAS TABLAS:**   beeline -u jdbc:hive2://localhost:10000/proyecto -f /home/dataproc/proyecto/producto.hql  beeline -u jdbc:hive2://localhost:10000/proyecto -f /home/dataproc/proyecto/cliente.hql  beeline -u jdbc:hive2://localhost:10000/proyecto -f /home/dataproc/proyecto/tienda.hql  beeline -u jdbc:hive2://localhost:10000/proyecto -f /home/dataproc/proyecto/distrito.hql  beeline -u jdbc:hive2://localhost:10000/proyecto -f /home/dataproc/proyecto/comprobante.hql  beeline -u jdbc:hive2://localhost:10000/proyecto -f /home/dataproc/proyecto/transaccion.hql  beeline -u jdbc:hive2://localhost:10000/proyecto -f /home/dataproc/proyecto/tienda\_buckets.hql  beeline -u jdbc:hive2://localhost:10000/proyecto -e /home/dataproc/proyecto/load\_tienda\_bucketing.hql   * **Creación de tabla Producto**      * **Creación de tabla Cliente**      * **Creación de tabla Tienda**      * **Creación de tabla Distrito**      * **Creación de tabla Comprobante** |

1. **CREAR AL MENOS 4 TABLAS EN HIVE CON LA DATA CARGADA, UNA SIMPLE, CON PARTICIÓN DINÁMICA, OTRA CON PARTICIÓN ESTÁTICA Y BUCKETING**

|  |
| --- |
| **TABLA SIMPLE**  #Tabla Cliente  CREATE EXTERNAL TABLE IF NOT EXISTS proyecto.cliente(  id\_cliente string COMMENT 'id\_cliente',  cliente string COMMENT 'cliente',  ubigeo string COMMENT 'ubigeo',  direccion string COMMENT 'direccion',  correo string COMMENT 'correo',  telefono string COMMENT 'telefono'  )  COMMENT 'Tabla cliente'  ROW FORMAT DELIMITED  FIELDS TERMINATED BY '|'  LINES TERMINATED BY '\n'  STORED AS TEXTFILE  LOCATION '/proyecto/cliente'  tblproperties("skip.header.line.count" = "1");  #Tabla Comprobante  CREATE EXTERNAL TABLE IF NOT EXISTS proyecto.comprobante(  IdComprobante string,  TipoComprobante string  )  COMMENT 'Tabla comprobante'  ROW FORMAT DELIMITED  FIELDS TERMINATED BY '|'  COLLECTION ITEMS TERMINATED BY ','  LOCATION '/proyecto/comprobante'  tblproperties("skip.header.line.count" = "1");  **TABLA CON PARTICIÓN DINÁMICA**  set hive.exec.dynamic.partition=true;  set hive.exec.dynamic.partition.mode=nonstrict;  CREATE TABLE proyecto.transaccion\_particion\_dinamica  (  cliente String,  tipotransaccion String  )  PARTITIONED BY (tipocliente STRING)  STORED AS PARQUET  LOCATION '/proyecto/transaccion\_particion\_dinamica';    **Carga de datos a la tabla particionada dinamica;**  set hive.exec.dynamic.partition.mode=nonstrict;  insert into table proyecto.transaccion\_particion\_dinamica partition (tipocliente)  SELECT distinct UPPER(cliente) AS Cliente, Case when tipocomprobante = 'BOLETA DE VENTA' then 'Boleta' when tipocomprobante = 'NOTA DE CREDITO' then 'Nota\_credito' else 'Factura' end as tipotransaccion, (CASE WHEN cast(venta\_mn as decimal(19, 2)) < '100' THEN 'BAJA' WHEN cast(venta\_mn as decimal(19, 2)) BETWEEN '100' AND '2000' THEN 'MEDIA' ELSE 'ALTA' END) AS tipocliente FROM proyecto.transaccion;    **Comprobando la partición dinámica;**    **TABLA CON PARTICIÓN ESTÁTICA**  CREATE TABLE proyecto.transaccion\_particion\_estatica  (  Fecha String,  Tipo\_cliente String,  Tipo\_Transaccion String,  Monto String  )  PARTITIONED BY (Mensual string)  STORED AS PARQUET  LOCATION '/proyecto/transaccion\_particion\_estatica';    **Carga de datos a la tabla particionada estatica;**  insert into proyecto.transaccion\_particion\_estatica partition(mensual ='201804')  select  from\_unixtime (unix\_timestamp(Concat(substring(fecha,1,4),SUBSTRING(Fecha,5,2), SUBSTRING(Fecha,7,2)), 'yyyyMMdd'), 'yyyy-MM-dd') as Fecha,  (CASE WHEN cast(venta\_mn as decimal(19, 2)) < '100' THEN 'BAJA' WHEN cast(venta\_mn as decimal(19, 2)) BETWEEN '100' AND '2000' THEN 'MEDIA' ELSE 'ALTA' END) AS tipocliente,  Case when tipocomprobante = 'BOLETA DE VENTA' then 'Boleta' when tipocomprobante = 'NOTA DE CREDITO' then 'Nota\_credito' else 'Factura' end as Tipo\_Transaccion,  cast(Venta\_mn as decimal(19, 2)) as Monto from proyecto.transaccion;    **Comprobando la partición estatica;**    **TABLA CON BUCKETING**  CREATE EXTERNAL TABLE IF NOT EXISTS proyecto.tienda\_buckets (   id\_tienda string,   tienda string,   distrito string   )   CLUSTERED BY (tienda) INTO 2 BUCKETS   ROW FORMAT DELIMITED   FIELDS TERMINATED BY '|'   COLLECTION ITEMS TERMINATED BY ','   MAP KEYS TERMINATED BY ':'  LOCATION '/proyecto/tienda\_bucketing';  set map.reduce.tasks = 2;  set hive.enforce.bucketing = true;  INSERT OVERWRITE TABLE proyecto.tienda\_buckets SELECT id\_tienda, tienda,distrito FROM proyecto.tienda; |
|  |

1. **DOS DE LAS TABLAS DEBEN TENER AL MENOS UNA ESTRUCTURA COMPLEJA (ARRAY, STRUCT O MAP)**

|  |
| --- |
| 1. **MAP 🡪 Tabla: Producto**   **#Tabla producto**  use proyecto;  CREATE EXTERNAL TABLE IF NOT EXISTS proyecto.producto(  id\_articulo string COMMENT 'Id\_articulo',  rubro string COMMENT 'rubro',  familia string COMMENT 'Familia',  marca string COMMENT 'marca',  articulo string COMMENT 'articulo',  familiamarca **map**<string,string> COMMENT '**Familiamarca**'  )  COMMENT 'Tabla producto'  ROW FORMAT DELIMITED  FIELDS TERMINATED BY '|'  COLLECTION ITEMS TERMINATED BY ','  MAP KEYS TERMINATED BY ':'  STORED AS TEXTFILE  LOCATION '/proyecto/producto'  tblproperties("skip.header.line.count" = "1");  **En spark-shell 🡪 Vemos el contenido de la tabla**  spark.sql("select \* from proyecto.producto").show()     1. **MAP 🡪 Tabla: Distrito**   **#Tabla distrito**  CREATE EXTERNAL TABLE IF NOT EXISTS proyecto.distrito(  Id\_Ubicacion string COMMENT 'Id\_Ubicacion',  Ubigeo string COMMENT 'Ubigeo',  Distrito string COMMENT 'Distrito',  Provincia string COMMENT 'Provincia',  Departamento string COMMENT 'Departamento',  Zona string COMMENT 'Zona',  departamentozona **map**<string,string> COMMENT '**departamentozona**'  )  COMMENT 'Tabla distrito'  ROW FORMAT DELIMITED  FIELDS TERMINATED BY '|'  COLLECTION ITEMS TERMINATED BY ','  MAP KEYS TERMINATED BY ':'  STORED AS TEXTFILE  LOCATION '/proyecto/distrito'  tblproperties("skip.header.line.count" = "1");  **En spark-shell 🡪 Vemos el contenido de la tabla**  spark.sql("select \* from proyecto.distrito").show()     1. **STRUCT 🡪 Tabla: Tienda**   CREATE EXTERNAL TABLE IF NOT EXISTS proyecto.tienda(  id\_tienda string,  tienda string,  figura string,  id\_ubicacion string,  direccion string,  distrito string,  distritotienda **struct**<distrito:STRING,tienda:string>  )  COMMENT 'Tabla tienda'  ROW FORMAT DELIMITED  FIELDS TERMINATED BY '|'  COLLECTION ITEMS TERMINATED BY ','  MAP KEYS TERMINATED BY ':'  LOCATION '/proyecto/tienda'  tblproperties("skip.header.line.count" = "1");  **En spark-shell 🡪 Vemos el contenido de la tabla**  spark.sql("select \* from proyecto.tienda").show() |

1. **REALIZAR TRANSFORMACIONES SIMPLES CON FUNCIONES UDFS NATIVAS APACHE HIVE (USO CONCAT, CAST, SUBSTRING)**

|  |
| --- |
| Detalle de UDFs nativas en Hive;  SELECT distinct CASE WHEN cast(venta\_mn as decimal(19, 2)) < '100' THEN 'BAJA' WHEN cast(venta\_mn as decimal(19, 2)) BETWEEN '100' AND '2000' THEN 'MEDIA' ELSE 'ALTA' END AS tipocliente, UPPER(cliente) AS Cliente, Case when tipocomprobante = 'BOLETA DE VENTA' then 'Boleta' when tipocomprobante = 'NOTA DE CREDITO' then 'Nota\_credito' else 'Factura' end as tipotransaccion FROM proyecto.transaccion |

1. **REALIZAR TRANSFORMACIONES COMPLEJAS (JOINS, AGGREGATES, ETC) CON APACHE SPARK**

|  |
| --- |
| spark.sql("select a.id\_tienda,b.tienda, sum(cast(venta\_mn as decimal(19, 2))) total\_venta from proyecto.transaccion a inner join proyecto.tienda b on a.id\_tienda=b.id\_tienda group by a.id\_tienda, b.tienda").show(); |

1. **GENERAR AL MENOS DOS TABLAS CON APACHE HBASE, NO NECESARIAMENTE TIENE QUE GUARDAR RELACIÓN DE NEGOCIO CON TODO EL CASO DE USO (monica)**

|  |
| --- |
| <https://www.tutorialspoint.com/es/hbase/hbase_installation.htm> |

1. **EXPLICAR TEÓRICAMENTE CÓMO FUNCIONARÍA EL FLUJO EN STREAMING USANDO APACHE KAFKA**

|  |
| --- |
| Kafka es el sistema de intermediación de mensajes basado en el modelo publicador/subscriptor que se encarga de dividir el flujo streaming en temas o topics (mensaje) en particiones. Cada partición es una secuencia ordenada de mensajes y cada partición es consumida por un único consumidor. A cada topic se le puede definir un número de particiones, en función del número de servidores y de conexiones que vayamos a tener. Esto aumenta considerablemente la disponibilidad.  Cada topic o tema tiene un offset para que cada consumidor indique que mensaje quiere que se le devuelva. A mayor número de particiones más tardará el productor (escribe mensaje) en guardar el mensaje, pero tardará menos el consumidor (lee mensajes) en recuperarlo. La idea está pensada para procesamiento en paralelo.  Cada mensaje publicado en un topic se entrega a una instancia de consumidor dentro de cada grupo de consumidores suscriptores.  Apache Kafka se distribuye junto con Zookeeper este está compuesto por brokers que actúan como líder de una o más particiones. |

1. **EXPLICAR TEÓRICAMENTE CÓMO FUNCIONARÍA EL FLUJO EN STREAMING USANDO APACHE FLUME**

|  |
| --- |
| Flume es un**servicio distribuido** que mueve de forma fiable y eficiente grandes cantidades de datos, especialmente **logs, es** sencilla y flexible; además funciona de tal forma que los flujos de datos en streaming permiten construir flujos múltiples por donde viajan los eventos a través de diferentes agentes hasta que alcanzan el destino final. |

1. **TODO EL CÓDIGO GENERADO DEBE ESTAR CORRECTAMENTE VERSIONADO EN GITHUB (JANET)**

|  |
| --- |
|  |

1. **ANALIZAR LA DATA CON PYSPARK (PYTHON CON SPARK)**

|  |
| --- |
|  |

1. **MOSTRAR GRÁFICAS Y/O INDICADORES EN POWER BI O EXCEL (MONICA)**

|  |
| --- |
|  |

1. **ANALIZAR LA DATA CON PYSPARK (PYTHON CON SPARK) (tema opcional lo va a desarrollar JESSICA**

|  |
| --- |
|  |

1. **CREAR EL CLUSTER CON UN NOTEBOOK DE JUPYTER INSTALADO**

|  |
| --- |
| 1. Jupyter      1. Instalar ipyparallel      1. Habilitar el IPython Clusters tab in Jupyter Notebook:      1. Instalar los usuarios en el JupyterHub como root:          1. Se observa el clúster creado. |

Otros

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Plus**   1. Ejecutar un modelo de ML utilizando SparkML 2. Desarrollar un flujo básico de streaming con Apache Kafka 3. Conectar Hive a Power BI   <https://docs.microsoft.com/es-es/azure/hdinsight/interactive-query/apache-hadoop-connect-hive-power-bi-directquery>  **Carga de datos desde HDInsight**  La tabla de Hive hivesampletable incluye todos los clústeres de HDInsight.   1. Inicie Power BI Desktop. 2. En la barra de menús, vaya a **Inicio** > **Obtener datos** > **Más...** .   Datos abiertos de Power BI en HDInsight   1. Desde la ventana **Obtener datos**, escriba **hdinsight** en el cuadro de búsqueda. 2. En los resultados de la búsqueda, seleccione **HDInsight Interactive Query** y, después, **Conectar**. Si no ve **HDInsight Interactive Query**, tiene que actualizar Power BI Desktop a la versión más reciente. 3. Seleccione **Continuar** para cerrar el cuadro de diálogo **Conectándose a un servicio externo**. 4. En la ventana **HDInsight Interactive Query**, escriba la información siguiente y luego seleccione **Aceptar**:  | **Propiedad** | **Valor** | | --- | --- | | Server | Escriba el nombre de clúster, por ejemplo,*myiqcluster.azurehdinsight.net*. | | Base de datos | Escriba **Predeterminada** para este artículo. | | Modo Conectividad de datos | Seleccione **DirectQuery** para este artículo. |  1. Conexión de HDInsight Interactive Query y Power BI con DirectQuery 2. Escriba las credenciales HTTP y, a continuación, seleccione **Conectar**. El nombre de usuario predeterminado es **admin**. 3. En el panel izquierdo, en la ventana del **Navegador** seleccione **hivesampletale**. 4. Seleccione **Carga** desde la ventana principal.   HDInsight Interactive Query y Power BI con hivesampletable  **Visualización de datos en un mapa**  Continúe a partir del último procedimiento.   1. En el panel Visualizaciones, seleccione **Mapa**, el icono del globo. Aparece un mapa genérico en la ventana principal.   HDInsight Power BI personaliza el informe   1. En el panel Campos, seleccione **country** y **devicemake**. Transcurridos unos instantes, un mapa del mundo con los puntos de datos aparece en la ventana principal. 2. Expanda el mapa. 3. Utilizar para las transformaciones de Spark el API de Scala   <https://github.com/ipython/ipyparallel>  **Requisitos no funcionales**   * Resumir todo el proyecto en un artículo del equipo publicado en Medium y que contenga el enlace al repositorio de Github (tablero scrum, código y el enlace a la presentación claramente indicado).   Puede utilizarse como alternativa para presentar su proyecto (expo)  <https://colab.research.google.com/notebooks/welcome.ipynb#scrollTo=xitplqMNk_Hc>  colab.research.google.com |
|  |

telef prof Andy Quiroz

980-888425