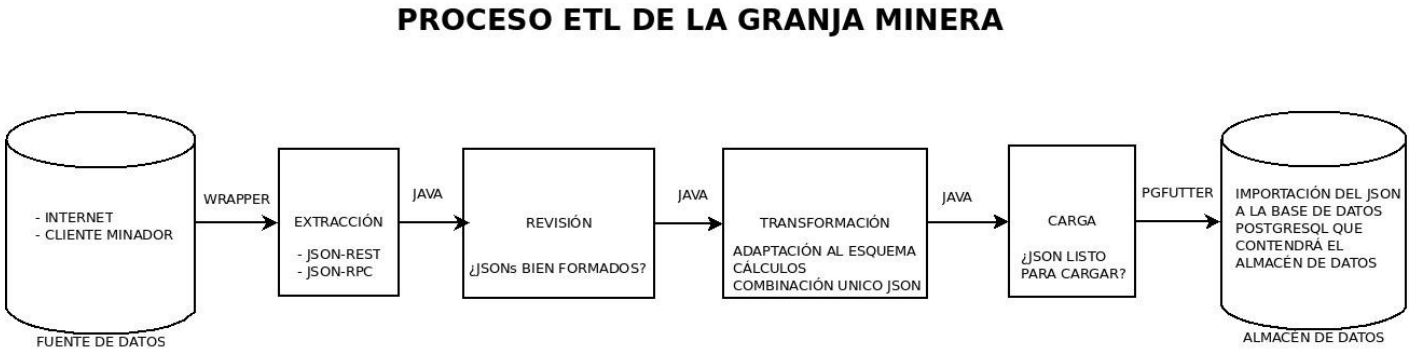
INTELIGENCIA DE NEGOCIO

ACTIVIDAD C3:
ETL



GRANJA MINERA DE CRIPTOMONEDA

MÁSTER EN BIG DATA

MOISÉS FRUTOS PLAZA
48488132-S
moises.frutos@um.es

PROCESO ETL DE LA GRANJA MINERA

A continuación se va a explicar de forma breve como sería el proceso ETL del dominio de la granja minera que hemos visto en las dos actividades anteriores. El proceso contaría con cuatro etapas principales:

1. EXTRACCIÓN DE DATOS

Los datos se deben de extraer de tres fuentes principales:

- El Mercado de las criptomonedas que está en internet mediante JSON-REST
  - https://poloniex.com/exchange#btc_zec
  - https://m.poloniex.com/support/api/
- El Pool donde está minando nuestro minero en internet mediante JSON-REST
  - https://zec.nanopool.org/account/t1MyUc7pGMfpbXZ8CvAX9N1bTbCLqpEYDTR
  - https://zec.nanopool.org/api
- El cliente minador que usa el minero para minar mediante JSON-RPC
  - https://forum.z.cash/t/miner-claymores-zcash-miner/7114
  - http://www.jsonrpc.org/specification

En el caso tanto del mercado como del pool existen por internet ya diversos wrappers para extraer la información, por ejemplo, en Python. Para Claymore GPU que es el cliente minador que por ejemplo utiliza la criptomoneda Zcash, no existe un wrapper propio, pero sí que existe un wrapper JSON-RPC para el primer cliente minador del que tengo constancia que exista, el cual es cgminer (https://github.com/ckolivas/cgminer). Si Claymore GPU sigue el estándar JSON-RPC los datos deberían de extraerse de igual manera. Esto ya no lo extraemos desde internet sino del propio equipo minador, de modo que sepamos sobre todo su temperatura, si está activo, etc. En cualquier caso, el resultado final debería de ser obtener tres ficheros JSON con los datos del mercado, del pool, y del minero, que coinciden con las tres dimensiones vistas en el diagrama multidimensional.

2. REVISIÓN DE LOS DATOS EXTRAÍDOS

Los ficheros JSON extraídos no dejan de ser cadenas de texto, por lo que antes de proceder a la transformación de los datos tenemos que revisar, o validar, que los JSON están bien formados. Esto lo podemos hacer mediante un pequeño código en Java que nos devolverá una excepción si la cadena pasada como JSON no es válida:

```
public boolean isJSONValid(String test) {
    try {
        new JSONObject(test);
    } catch (JSONException ex) {
        try {
            new JSONArray(test);
        } catch (JSONException ex1) {
            return false;
        }
    }
    return true;
}
```

Un objeto JSONArray corresponde con una colección de JSON, que no será nuestro caso ya que siempre lo que le pasaremos serán JSON simples que intentará convertir a JSONObjetc y en el caso de que no pueda lanzará la excepción indicando que no está validado, debiendo de esperar un nuevo JSON del wrapper que sea válido para pasarlo a la siguiente etapa.

3. TRANSFORMACIÓN DE LOS DATOS ACORDE AL ESQUEMA DEL ALMACÉN DE DATOS

En esta etapa es donde se reciben los objetos JSON en Java devueltos por el validador. En ellos ya tenemos los atributos con los datos que necesitamos y los podemos preprocesar de manera que extraigamos aquellos que nos interesan para nuestro esquema de datos del almacén final. Podremos hacer los calculos que sean necesarios (sobre todo aquellos correspondientes a los indicarores que ya definimos). Y finalmente con todos los datos ya preprocesados, realizamos la combinación a un único JSON que será lo que le pasaremos al cargador, para que lo introduzca en la base de datos.

4. CARGA DE LOS DATOS TRANSFORMADOS AL ALMACÉN DE DATOS

Investigando por la red, he encontrado esta herramienta que la he considerado bastante interesante y que se llama Pgfutter (https://github.com/lukasmartinelli/pgfutter). Esta herramienta nos permite cargar en PostgreSQL nuestro fichero JSON transformado mediante una importación. Por suerte, PostgreSQL cuenta con soporte para JSON y una vez importados los datos dentro de PostgreSQL los podemos tratar con operadores específicos de JSON (https://www.postgresql.org/docs/9.3/static/functions-json.html). Una vez ya tenemos los datos dentro de PostgreSQL lo único que nos restaría por hacer sería insertarlos en nuestro almacén de datos según nuestro esquema.

Algunos de los operadores más interesantes de PostgreSQL para JSON podrían ser los siguientes:

->	int	Get JSON array element	'[1,2,3]'::json->2
->	text	Get JSON object field	'{"a":1,"b":2}'::json->'b'
->>	int	Get JSON array element as text	'[1,2,3]'::json->>2
->>	text	Get JSON object field as text	'{"a":1,"b":2}'::json->>'b'
#>	array of text	Get JSON object at specified path	'{"a":[1,2,3],"b":[4,5,6]}'::json#>'{a,2}'
#>>	array of text	Get JSON object at specified path as text	'{"a":[1,2,3],"b":[4,5,6]}'::json#>>'{a,2}'

DIAGRAMA DEL PROCESO ETL SEGUIDO PARA LA GRANJA MINERA

Finalmente, podemos resumir todo el proceso ETL mediente el siguiente Diagrama ETL: