# EJERCICIO

Desde una empresa B2C compuesta por tiendas físicas y un e-commerce, se quiere hacer el ciclo de reparto a tiendas del día, y para ello, necesitan saber cuánto stock (y de qué zonas del almacén) deben coger.

Partimos de un fichero con los datos de la propuesta de reparto: "Prereparto_bruto.json". Este fichero tiene guardados qué artículos (y qué cantidad) deben ser repartidos a tiendas concretas en ciclos de reparto concretos. Los campos relevantes para el ejercicio son:

- key: identifica el artículo
- propuesta: Cantidad que debe ser repartida
- tiendaId: identifica la tienda a la que se debe hacer el reparto
- grupoLocalizacionDesc: identifica el ciclo de reparto (los repartos se hacen de forma escalada en el tiempo, y por eso se organizan en ciclos)
- esEcommerce: identifica si la tienda destino es online o tienda física. Un 1 indica que la tienda destino es online. 0 para lo contrario

Tenemos un segundo fichero con los datos del stock del almacén, del cual se tirará para hacer el reparto: "Stock_unificado.json". Los campos relevantes para el ejercicio son:

- key: identifica el artículo
- tipoStockDesc: zona del almacén donde está guardado el stock. Hay tres: "ZAR" (zona de alta rotación), "MSR" (suelo) y "SILO" (silos)
- stockEm05: stock en estado 5, exclusivo para servir las peticiones de tiendas online. Las tiendas físicas no pueden "tirar" de ese stock.
- stockEm01: stock en estado 1, para servir tiendas físicas, pero las tiendas online, en caso de agotarse su stock en estado 5, pueden tirar de él adicionalmente.
- posicioncompleta: identificador de la posición donde se encuentra el stock para el artículo indicado en key.

La lógica de funcionamiento aplicable al ejercicio es la siguiente:

- Para unas unidades de propuesta de reparto dadas, el sistema debe coger el stock del almacén, yendo a las diferentes zonas a "buscar" las unidades que se necesitan siguiendo el siguiente orden: ZAR -> MSR -> SILO. Es decir, para una petición de unidades de un artículo concreto (propuesta), el sistema debe ir a buscar el stock del artículo al ZAR primero, y si no se satisfacen las unidades con lo que hay en el ZAR, ir a la zona MSR para coger las adicionales necesarias, y si no fuese suficiente, iría al SILO. En el caso de que vaya a las tres zonas, el sistema devolverá un resultado con la cantidad de unidades y las posiciones que moverá de cada una de las 3 zonas. Si usa 2, pues 2, y si usa 1, pues 1.
- Para la operativa descrita arriba debe tenerse en cuenta que, si la línea de propuesta especificada en el "Prereparto_bruto.json" es ecommerce (es decir un destino de tienda online), el sistema deber ir a stock en estado 5 en primer lugar (el primer stock en estado 5 que encuentre en las 3 zonas, siguiente la prioridad descrita en el punto 4a), y si no tuviese suficiente debería ir a buscar el stock en estado 1.
- Si la tienda destino de la propuesta fuese NO ecommerce (esEcommerce = 0), debe coger solo el stock en estado 1 (stockEm01).

Se pide:

1. Sabiendo que desde almacén, necesitan saber las posiciones de stock de almacén que deben mover para satisfacer la propuesta de reparto para todas las tiendas y artículos que cumplan estos parámetros:
    - grupoLocalizacionDesc = “CICLO 2 GRUPO A2” y “CICLO 1 GRUPO B” y “CICLO 1 GRUPO A2”
    - esEcommerce = 1

    Devolver una tabla con los siguientes campos:
        1. Key (artículo a repartir)
        2. idTienda
        3. propuesta (unidades a repartir)
        4. tipoStockDesc (zona del almacén de la que sale)
        5. EstadoStock (nuevo campo que tendrá valores 1 o 5 según el estado del stock que ha cogido [stockEm05 o stockEM01])
        6. posicioncompleta (id de la posición en el almacén)

    Por supuesto, esta tabla, debería tener tantas líneas por artículo como fuese necesario si ese artículo debiese ser repartido desde varias zonas del almacén, coger stock de diferentes estados, etc.

    Implementa tu solución en javascript, utiliza las librerías que consideres oportunas siempre y cuando no incurran en problemas de licenciamiento. Detalla las decisiones de tu solución que consideres pertinentes.

2. Describe cómo implementarías esta solución si tuvieras que acabar mostrando el resultado en un sistema low code.
    a. Consideraciones de rendimiento.
    b. Requisitos que necesitarías del API.

3. Si el json real del prerepato ocupase 20Gb, explica si el problema de forma distitna y por qué.

4. Si tuvieras que de forma visual presentar en una pantalla desde que partes de un almacen se rellena un pedido, que propuesta de visualizacion plantearías teniendo en cuenta que se quiere implementar con una herramienta low code.