Examen Mercadolibre

Los carriers notifican los movimientos del envío indicando únicamente un estado y un subestado, el cual es enviado en el siguiente formato:

```
{
   “status”: “value”,
   “substatus”: “value”
}
```

En ocasiones el carrier no manda los datos de forma ordenada e inclusive deja de notificar eventos intermedios(*handling, ready to ship y shipped*).

Los posibles valores en orden que son informados por el carrier son:

- **Handling**
  - *Null*
  - *Manufacturing*
- **Ready To Ship**
  - *Ready To Print*
  - *Printed*
- **Shipped**
  - *Null*
  - *Soon Deliver*
  - *Waiting For Withdrawal*
- **Delivered**
  - *Null*
- **Not Delivered**
  - *Lost*
  - *Stolen*

Se te ha contratado para que desarrolles un proyecto que sea capaz de recibir los diferentes cambios de estado, en el orden en que llegaron e indicar cual es el estado actual del envío.

Para eso se te ha pedido crear un programa con un método o función con la siguiente firma (En alguno de los siguiente lenguajes: Java / Golang / C-C++ / Javascript (node) / Python / Ruby):

String package(List<Object> inputs); // Ejemplo Java

En donde recibirás como parámetro una lista de objetos e indicarás el siguiente texto dependiendo del estado actual del envío:

- Handling
    - Null (**“Le notificamos al vendedor sobre tu compra**”)
    - Manufacturing (“**El vendedor tendrá listo tu producto pronto y comenzará el envío**”).
- Ready To Ship
    - Ready To Print (“**El vendedor está preparando tu paquete**”)
    - Printed (“**El vendedor debe despachar tu paquete**”)
- Shipped
    - Null (“**En Camino**”)
    - Soon Deliver (“**Pronto a ser entregado**”)
    - Waiting For Withdrawal (“**En agencia**”)
- Delivered
    - Null (“**Entregado**”)
- Not Delivered
    - Lost (“**Perdido**”)
    - Stolen (“**Robado**”)

Desafíos:

Nivel 1:

Programa (en cualquier lenguaje de programación) el método o función que cumpla con la solicitud.

Nivel 2:

Crear una API REST, hostear esa API en un cloud computing libre (Google App Engine, Amazon AWS, etc), crear el servicio “/package/” en donde se pueda enviar en formato json los datos enviados por el carrier mediante un HTTP POST en formato Json, que irá en el siguiente formato:

```
POST → /package/
{
  “id”: “28123B”,
  “inputs”: [
    {
      “status”: “ready_to_ship”,
      “substatus”: “printed”
    },
    {
      “status”: “shipped”,
      “substatus”: null
    },
    {
      “status”: “delivered”,
```

```
      “substatus”: null
    }
  ]
}
```

En este caso se, debería devolver un HTTP 200-OK y la respuesta

```
{
“package” : “Entregado”
}
```

Nivel 3:

Implementar un mecanismo de persistencia de datos cada vez que sea consultada la API.

Exponer un servicio de estadísticas el cual utilizará como base de información los datos previamente almacenados. Tener en cuenta que la API puede recibir fluctuaciones agresivas de tráfico (Entre 100 y 1 millón de peticiones por segundo).

Exponer servicios para el monitoreo de la salud del aplicativo.

Generar Test-Automáticos con un code coverage mayor al 80%.

Nivel 4:

Escribir dentro del README de github:

- ¿Qué necesita el sistema para poder rastrear los paquetes en tiempo real?
- ¿Cómo implementarías la geolocalización de los paquetes?

Entregar:

- Código Fuente (Para Nivel 1, 2 y 3: En repositorio github).
- Instrucciones de cómo ejecutar el programa o la API. (Para Nivel 2 y 3: En README de github).
- URL de la API (Nivel 2 y 3).