Facturación de MercadoLibre

En MercadoLibre estamos interesados en desarrollar un nuevo sistema de cobranza para los usuarios que usan los servicios brindados por el sitio, entiéndase publicar, vender, publicitar, realizar envíos o comprar con créditos a través de nuestra plataforma. Para ello, lo contactamos a usted con el fin de evaluar una propuesta de software que cumpla con todas las expectativas y exigencias que puede tener un desarrollo de tanta importancia. A continuación se indicará el flujo requerido y las particularidades que debe tener en cuenta el mismo.

- MercadoLibre genera **cargos** a cada **usuario** por distintos **eventos** que ocurren dentro de nuestra plataforma.
- Los mismos están relacionados a las distintas actividades que realizan nuestros usuarios dentro del sitio (como se mencionó antes, publicar, vender, etc).
- Estos cargos deben agruparse en **facturas** por **usuario**.
- Las facturas agrupan cargos por mes, por ejemplo, un cargo relacionado a un evento del mes de mayo 2019, queda en la factura de mayo 2019.
- Una vez concluido el mes, la factura no puede añadir más cargos.
- También contamos con los **pagos**, que es el dinero ingresado por los usuarios para pagar los cargos que le generamos.
- Los mismos se **asocian** a los cargos, para poder reflejar qué pago está pagando cada cargo.
- Un mismo pago puede estar asociado a más de un cargo y viceversa.
- Todo cargo que no posea pagos asociados por el total de su monto se lo podrá considerar **deuda**.
- Los eventos y pagos pueden llegar en dolares o pesos argentinos, pero dentro de nuestro sistema sólo manejaremos pesos argentinos.
- *(1)* Los pagos sólo pueden realizarse por montos iguales o inferiores a la deuda que el usuario tenga, en caso de que se realice un pago superior, el mismo será rechazado.

En la empresa ocurren **8 tipos de eventos**, a continuación dejamos un ejemplo de evento:

```
{
  "event_id": 1, (Indica el id del evento)
  "amount": 150.10, (Indica el monto a cobrar, con dos decimales)
  "currency": "USD", (USD o ARS)
  "user_id": 1, (Indica el número de usuario)
   "event_type": "CLASIFICADO", (Tipo de evento -> Puede tener 8
valores diferentes)
   "date": "2019-05-16T00:00:00" (Fecha y hora en que ocurrió el
evento)
}
```

Los 8 valores posibles del event_type son

- CLASIFICADO
- VENTA
- PUBLICIDAD
- ENVÍO
- CRÉDITO
- MERCADOPAGO
- MERCADOSHOP
- FIDELIDAD

Los cargos se dividen 3 categorías diferentes:

- **MARKETPLACE**
  - CLASIFICADO
  - VENTA
  - ENVÍO
- **SERVICIOS**
  - CRÉDITO
  - FIDELIDAD
  - PUBLICIDAD
- **EXTERNO**
  - MERCADOPAGO
  - MERCADOSHOP

En cuanto a los pagos, podemos basarnos en el siguiente ejemplo:

```
{
  "amount": 150.10, (Indica el monto abonado con dos decimales)
  "currency": "USD", (USD o ARS)
  "user_id": 1 (Indica el número de usuario)
}
```

Se requiere que el software desarrollado genere los cargos indicados, los incluya en facturas, y asocie con pagos. Para ello, tendrá que contar con las siguientes apis

- post de evento para creación de cargo (se enviará un evento a la vez por request)
- post de pago (se enviará un pago a la vez por request)

Y a su vez, para luego poder obtener la información procesada, necesitaremos:

- get de cargos
- get de facturas
- get de pagos
- get de status del usuario

A tener en cuenta que:

- **Los posts de pagos puede recibir miles de request por minuto** (ejemplo 15mil RPM)
- Dada la importancia del proyecto y la velocidad que queremos darle a nuestros usuarios en sus procesos, **necesitamos que los tiempos de respuesta sean lo más bajo posibles**.
- Los datos se deben persistir en una base de datos. (Usar el motor de su preferencia)

Se valorará la originalidad y el análisis que realice el desarrollador sobre **qué parámetros podrá recibir cada uno de estos request (GETs) y qué información devolverán**. A su vez, solicitaremos documentación sobre las distintas APIs.
Por último, **esperamos que el software tenga los test necesarios** como para poder hacer las modificaciones pertinentes cuando haga falta, con la tranquilidad de que no se van perder las funcionalidades requeridas o que se generen errores, o al menos, que haya que modificarlos si la funcionalidad debe cambiar.

Entregar:

- El código del software subido en un repositorio GIT público.
- Instrucciones para levantar local la aplicación (pueden estar dentro del repositorio).
- La documentación requerida (puede estar dentro del repositorio).
- Consideraciones que hayan surgido con el ejercicio (puede estar dentro del repositorio).

Bonus 1:

En el punto (1) del enunciado se nos indica que los pagos no pueden ser mayores a la deuda del usuario. Desarrollar aparte (puede ser en otro branch) los cambios necesarios dentro del software para que los pagos puedan superar la deuda del usuario, dejando al mismo con saldo a favor. Analizar bien esta solución, porque puede resultar más compleja de lo que parece y tener varias implicancias en distintas partes del código, incluso generar inconsistencias si no se toman en cuenta todas las ocurrencias posibles.

Bonus 2:

Disponibilizar las apis del ejercicio en una url pública para poder realizar pruebas sin la necesidad de levantarlo local. Se puede utilizar el servicio que el desarrollador guste (AWS, Windows Azure, Google Cloud o el que quieran).