GeSoc

Gestión de Proyectos Sociales

Trabajo Práctico Anual Integrador

-2020-

Entrega 2: presupuestos y validador

Objetivo de la entrega

- Implementar, de manera incremental, el diseño de la entrega anterior
- Incorporar nociones de comunicación asincrónica, tomando las decisiones de diseño implicadas

Alcance

- Presupuestos
- Validador de transparencia

Requerimientos detallados

1. Se debe llevar registro de los **presupuestos**, tanto el total, como sus detalles y los documentos comerciales (manteniendo los datos requeridos para las operaciones de egresos).
2. Cada ítem de un presupuesto debe estar asociado obligatoriamente a un **egreso** (compras). No es posible cargar presupuestos si previamente no se cargó el egreso correspondiente (esto se define para evitar que el sistema incorpore presupuestos de productos o servicios que luego no se adquieren y que no se convierta en un repositorio de presupuestos, lo que lo alejaría de su objetivo)
3. Se desea verificar las siguientes condiciones en los egresos y los presupuestos:
    a. Si la compra requiere presupuestos (para lo cual se debe tener previamente configurados cuántos presupuestos son requeridos para una compra[1]) se debe verificar que efectivamente se encuentran cargada la cantidad indicada
    b. Si la compra requiere presupuestos se debe verificar que la compra haya sido realizada en base a alguno de esos presupuestos
    c. Si el criterio de selección del proveedor fuera el de menor valor, debe validarse que, entre los presupuestos cargados en el sistema, se haya seleccionado el de menor valor.
4. Sin embargo, no se desea evitar la creación de egresos que incumplan las reglas anteriores, sino dar un tiempo prudencial para cargarlos y ajustarlos a estas reglas. Luego, en un momento posterior, se ejecutarán estas validaciones. Por el momento se solicita que este proceso se pueda **ejecutar de forma manual**, por consola, **para todos los egresos pendientes de validación**.
5. Los resultados de la validación deberán ser vistos por el sistema por parte de algunos usuarios (en una **bandeja de mensajes**). Para que un usuario pueda ver los resultados de la validación, debe haberse dado de alta como revisor de una compra, en la propia compra
6. En la iteración anterior ya se planteó que cada operación de egreso de fondos debe conocer el proveedor o prestador de servicios, y éste a su vez conoce su **dirección postal**. En esta nueva entrega toda dirección postal debe estar conformada por: dirección propiamente dicha (calle, altura, piso, etc.), ciudad, provincia, país.
    a. La información de **países, provincias y ciudades** se deben obtener mediante la utilización

1 Como valor único que será usado a lo largo de todo el sistema, para todas las compras, al menos por el momento (luego se prevé su modificación)

de la API de Mercado Libre[2].

b. De igual forma, se busca incorporar el concepto de **moneda** a todo valor monetario; la información de las monedas se debe obtener también mediante el API antedicha.

Entregables requeridos

1. Modelo de Objetos (Diagrama de clases)
2. Implementación en código Java de los requerimientos enumerados y sus pruebas unitarias.
3. Alternativas pensadas para diferentes requerimientos según corresponda y su comparación

Anexo 1: recomendaciones sobre la implementación de un proceso batch

Para implementar un proceso batch necesitaremos exponer de alguna forma nuestro modelo de objetos al mundo exterior, fuera de la JVM. Para ello, una técnica sencilla es implementar un programa de tipo main:

```
package ar.edu.frba.dds.gesoc.Main;
public class Main {
    public static void main(String[] args) {
        //... utilizá acá tus objetos dominio...
    }
}
```

Sin embargo, para que este código sea ejecutable por fuera de nuestro entorno de desarrollo, deberemos *empaquetarlo*, es decir, crear un archivo ejecutable. Este archivo será reconocido por el sistema operativo, el cual creará una máquina virtual de Java, y lo ejecutará dentro de ella.

Para crear dicho paquete (conocido como Jar en Java), deberemos configurar el assembly plugin de maven:

```
<build>
  <plugins>
    ...
    <plugin>
      <artifactId>maven-assembly-plugin</artifactId>
      <executions>
        <execution>
          <phase>package</phase>
          <goals>
            <goal>single</goal>
```

2 https://developers.mercadolibre.com.ar/es_ar/ubicacion-y-monedas

```
          </goals>
        </execution>
      </executions>
      <configuration>
        <descriptorRefs>
          <descriptorRef>jar-with-dependencies</descriptorRef>
        </descriptorRefs>
        <archive>
          <manifest>
            <mainClass>ar.edu.frba.dds.gesoc.Main</mainClass>
          </manifest>
        </archive>
      </configuration>
    </plugin>
  </plugins>
</build>
```

Anexo 2: conexión con API de MercadoLibre

Mercado Libre nos ofrece una interfaz para conectarnos a sus sistemas y obtener información. Sin embargo, en esta ocasión dicha interfaz no se trata de un conjunto de objetos en un paquete que podamos fácilmente importar en nuestro código, sino de un protocolo de comunicación de red (HTTP) aplicando un conjunto de *buenas prácticas* (REST)[3]

Si bien HTTP y REST son ciertamente complejos, afortunadamente no es necesario conocer en detalle qué son ni cómo funcionan para realizar algunos primeros programas, dado que existen cientos de bibliotecas y frameworks que nos abstraen de sus complejidad. Por ejemplo, utilizando la biblioteca Jersey, acceder al recurso `classified_locations/countries` mediante el método GET es fácil:

```
Client.create()
      .resource("https://api.mercadolibre.com/")
      .path("classified_locations/countries")
      .accept(MediaType.APPLICATION_JSON)
      .get(ClientResponse.class);
```

Más información:

- Ejemplos de uso de Jersey: Jersey es una tecnología que permite, desde Java, consumir una interfaz HTTP.

3 Tanto HTTP como REST serán objeto de estudio en el segundo cuatrimestre.

- cURL / Postman: Son herramientas que permiten acceder a una interfaz HTTP sin tener que construir un programa. El primero está basado en texto, el segundo viene con una interfaz gráfica.