Integración de sistemas de información

Trabajo Práctico Especial

Alumno: Alejandro Bezdjian, 52108

Introducción

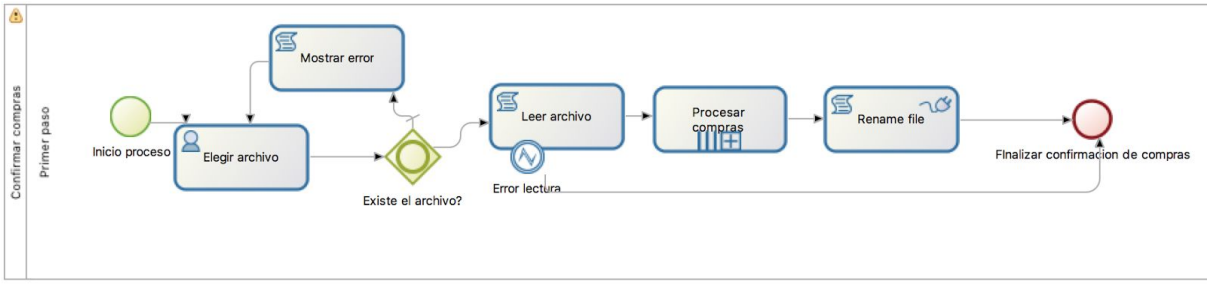
El objetivo de este trabajo práctico es la implementación de un proceso BPM en BonitaBPM, el cual debe integrarse con una base de datos, un servidor externo y debe poder leer y escribir archivos.

Descripción del proceso

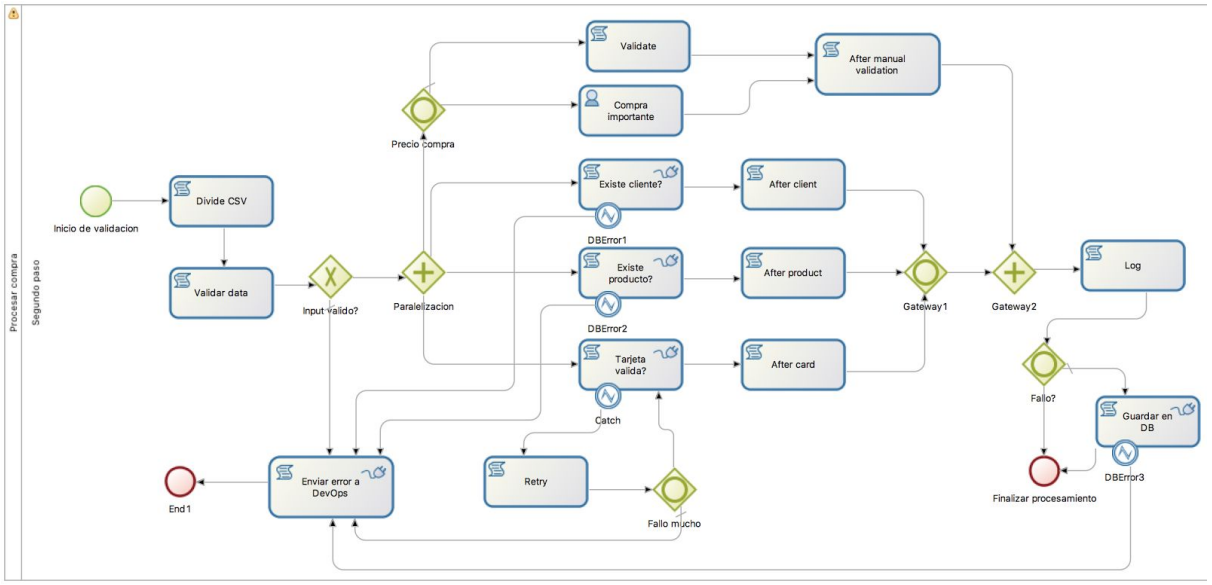
Se diseñaron e implementaron dos procesos que en su conjunto logran confirmar y validar ventas de algún producto. No se definió un producto particular ya que el proceso podría ser igual para un gran rango de los mismos en cualquier tipo de negocios.

El primer proceso, llamado “Confirmar compras” es el encargado de leer de un archivo CSV sin header una lista de ventas realizadas previamente. La forma con la que se genera este archivo no es parte del scope del trabajo y asume el siguiente formato: checkout_id, client_id, product_id, price (centavos), card_number, card_provider. La lectura del archivo transforma los datos en una lista de strings que representan cada fila del archivo. Si el archivo no existe, se le vuelve a preguntar al empleado que ingrese un archivo existente, mostrándole un mensaje de error.

Luego de leer el archivo, se procede a una tarea llamada “Procesar compras”, la cual es la encargada de llamar al segundo proceso creando múltiples threads en paralelo. A cada thread le envía la data de la fila y la guarda en una variable del segundo proceso. Al terminar, le agrega el prefijo “PROCESSED-” al nombre del archivo para que los empleados puedan distinguirlos de otros archivos no procesados aún. Si el archivo no existe se le vuelve a preguntar al usuario y se le muestra un mensaje de error.



El segundo proceso, llamado “Procesar compra” es el mas complejo de los dos y es el encargado de procesar cada fila del archivo CSV. Los datos son recibidos en un string con los datos separados por “,”. La tarea “Divide CSV” se encarga de separar los datos y guardarlos en variables de proceso para facilitar el manejo de los mismos.



Luego de obtener los datos, se procede a validar los mismos. Si los datos son nulos o no tienen sentido se decide terminar con el proceso escribiendo en un archivo de error que luego será analizado por un empleado DevOps. Como veremos mas adelante, todos los errores técnicos terminan de la misma manera, escribiendo en el archivo de error en la tarea “Enviar error a DevOps”.

Si pasa la validación técnica, se realiza la validación contra los servicios y bases de datos. Se decidió realizar esto de forma paralela y las validaciones son las siguientes:

1. Si la compra supera un monto de $20.000, la validación se realiza de forma manual por un empleado de la empresa (Walter Bates).
2. Se verifica que el cliente exista en la base de datos, haciendo una query: “SELECT COUNT(*) FROM clients WHERE id = ?;”.
3. Se verifica que el producto exista en la base de datos, haciendo una query: “SELECT COUNT(*) FROM products WHERE id = ?;”.
4. Se verifica que los datos de la tarjeta de crédito sean válidos. Para esto se hace un request a una API REST (creada especialmente para este trabajo).

Una vez terminadas las validaciones se guarda en variables de proceso la información obtenida de las mismas, como por ejemplo si hubo algún error.

El anteúltimo paso es el de Log, el cual escribirá en un archivo (seleccionado por el proceso anterior) el resultado de cada venta procesada. El resultado puede ser positivo, lo que hará que escriba “SUCCESS” y el número de compra (checkout_id). En caso de que sea negativo escribirá el motivo por el cual falló (obtenido en las validaciones).

Por último, si las validaciones fueron exitosas, se procede a guardar en la base de datos la compra con la siguiente query: “INSERT INTO purchases (product_id, client_id, price) VALUES(?, ?, ?);”.

Manejo de errores

Dividiremos los errores del proceso en técnicos y de negocio, ya que la solución de los mismos dependen de personas con diferentes roles.

Errores de negocio

- Archivo inexistente: si el archivo ingresado por el usuario no existe, se le vuelve a presentar el formulario para que lo cambie antes de que se produzca un error de lectura. Consideramos que es un error de negocio ya que lo más probable es que el usuario haya introducido un error al ingresar el nombre del archivo.
- Producto o cliente inexistente: estos errores son logueados en el archivo al final del proceso, consideramos que son de negocio porque pudo haber un cambio en las bases de datos del momento de la creación al momento del proceso de la compra.

Errores técnicos

Todos los errores técnicos del proceso llevan a una tarea la cual es la encargada de escribir en un archivo los errores ocurridos. El procesamiento de dichos errores no está en el scope de este proyecto.

- Web service: se decidió realizar un retry de 3 veces antes de considerarlo como un error. Consideramos que es un error técnico ya que podría ser que el usuario no tenga internet, o que la URL este mal ingresada en el sistema o que el servicio no esté disponible. Ninguna de estas alternativas pueden ser arregladas por el empleado encargado de procesar las ventas.
- Base de datos: en este caso se decidió que si ocurre un error instantáneamente se proceda a enviarselo a un empleado de DevOps para que verifique el problema. Esto se debe a que, como la base de datos es local y que es muy raro que ocurra un error debido a la base de datos en si, lo mas probable es que haya algún error de conexión con la misma como puede ser un error de autenticación o que la url de la base esté mal, por lo tanto no es probable que lo pueda arreglar el empleado de ventas.
- Datos ingresados como input en el CSV: si existe un error en el CSV, como asumimos que dicho archivo proviene de otra parte del sistema, lo más probable es que esa parte del sistema esté funcionando mal y por lo tanto no sea un error manejable por el vendedor.

Anexo

Tablas de la base de datos

```
CREATE TABLE clients (
    id SERIAL PRIMARY KEY,
    email VARCHAR(255) NOT NULL
);

CREATE UNIQUE INDEX clients_id_index ON clients(id);

CREATE TABLE credit_cards (
    id SERIAL PRIMARY KEY,
    card_number VARCHAR(32) NOT NULL,
    card_provider VARCHAR(32) NOT NULL,
    client_id INT NOT NULL,
    FOREIGN KEY (client_id) REFERENCES clients(id)
);

CREATE UNIQUE INDEX credit_cards_id_index ON credit_cards(id);

CREATE TABLE products (
    id SERIAL PRIMARY KEY,
    price INT NOT NULL,
    description VARCHAR(255) NOT NULL
);

CREATE UNIQUE INDEX products_id_index ON products(id);

CREATE TABLE purchases (
    id SERIAL PRIMARY KEY,
    price INT,
    product_id INT,
    client_id INT,
    FOREIGN KEY (product_id) REFERENCES products(id),
    FOREIGN KEY (client_id) REFERENCES clients(id)
);

CREATE UNIQUE INDEX purchases_id_index ON purchases(id);
```

Repositorio

https://github.com/alebianitba/bpm