

**Tugas I IF4052 Komputasi Layanan**  
***Business Process Modeling Notation Orchestration***



Disusun oleh:

Kelompok 5

Manuella Ivana Uli Sianipar 13521051

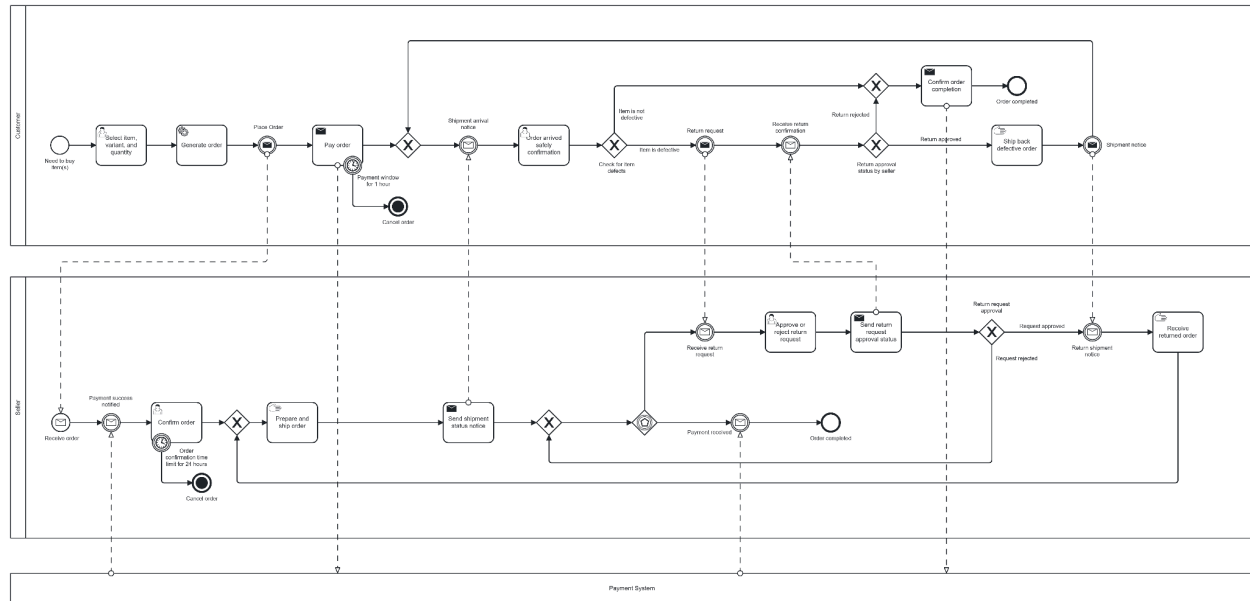
Arleen Chrysantha Gunardi 13521059

Jericho Russel Sebastian 13521107

**TEKNIK INFORMATIKA**  
**INSTITUT TEKNOLOGI BANDUNG**  
**2024**

## 1. Model BPMN

Berikut merupakan model diagram BPMN sederhana untuk proses *e-commerce*.



Gambar 1.1 Model BPMN

Untuk gambar yang lebih jelas dapat dilihat pada tautan berikut:

<https://drive.google.com/file/d/1fEvjWg7KmxSF3NluWfY8G2wnJe5F2x48/view?usp=sharing>

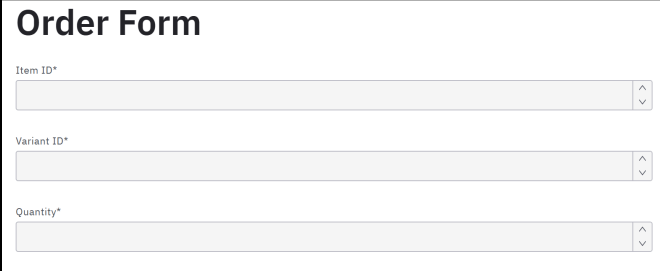
## 2. Implementasi

Terdapat dua *pool* yang masing-masing menggambarkan *role* pembeli (*customer*) dan penjual (*seller*), dan juga satu *blackbox pool* untuk sistem pembayaran. Pemodelan dilakukan menggunakan Camunda dan diimplementasikan dengan Client API Zeebe.

Dua entitas yang ada dalam model ini adalah `Order`, yang terdiri dari properti `Id` dan `Item` (berisi `Id`, `VariantId`, dan `Quantity`) yang merepresentasikan informasi terkait pesanan yang dibuat oleh pembeli, serta `ReturnRequest` yang terdiri atas properti `approved` (*boolean*) untuk menangani logika permintaan pengembalian barang.

Implementasi tiap proses adalah sebagai berikut:

### 1. Pemilihan barang, varian, dan kuantitas oleh pembeli



The image shows a web form titled "Order Form". It contains three input fields, each with a label and an asterisk indicating it is required: "Item ID\*", "Variant ID\*", and "Quantity\*". Each field has a light gray border and a small dropdown arrow on the right side, suggesting they might be dropdown menus or have validation feedback.

**Gambar 2.1.1** Formulir Pesanan

Tahapan ini direpresentasikan sebagai *user task* (berupa formulir) yang berisi *field* untuk mengisi ID barang, ID varian barang, dan kuantitas barang yang ingin dibeli oleh pembeli. Submisi formulir ini akan diisikan dalam `Order` untuk atribut `Item` (atribut `Id` masih bernilai *null*).

### 2. Pembuatan pesanan

Tahapan ini direpresentasikan sebagai *service task* dan ditangani oleh prosedur `GenerateOrderHandler` yang akan membuat sebuah pesanan dengan ID acak, sesuai dengan informasi yang diberikan dalam variabel proses `Order`.

```
private static async Task GenerateOrderHandler(
    IJobClient jobClient, IJob job) {
    var vars = JsonConvert
        .DeserializeObject<CustomerProcVariables>(job.Variables);
    var order = vars.Order;

    order.Id = random.Next(0, int.MaxValue);
    logger.LogInformation("Generated order ID {OrderId} for order
        with item ID {ItemId}, Variant ID {VariantId}, Qty
        {Quantity}",
        order.Id, order.Item.Id, order.Item.VariantId,
        order.Item.Quantity
    );

    await jobClient.NewCompleteJobCommand(job)
        .Variables(JsonConvert.SerializeObject(vars)).Send();
}
```

### 3. Penempatan pesanan

Tahapan ini berupa *message intermediate throw event*. Ketika pembeli melakukan konfirmasi, *message* dengan nama `msgCustomerPlaceOrder` dikirim untuk melanjutkan tahapan berikutnya dan menjadi *trigger* untuk memulai proses *seller*.

Variabel ID pesanan dan detail item digunakan untuk mengorelasikan *message* ini dengan pesanan yang relevan.

```
private static void RegisterWorkers() {  
    ...  
    workers.Add(  
        MessageThrowJob<CustomerProcVariables>("PlaceOrder",  
        "msgCustomerPlaceOrder",  
        correlationKey: vars => vars.Order.Id!.ToString()!,  
        sentVariables: JsonConvert.SerializeObject,  
        preCall: vars => logger.LogInformation(  
            "Place order for item ID: {ItemId}, Variant ID:  
            {VariantId}, Qty: {Quantity}",  
            vars.Order.Item.Id, vars.Order.Item.VariantId,  
            vars.Order.Item.Quantity  
        )  
    ));  
    ...  
}
```

#### 4. Pembayaran pesanan

Tahap ini direpresentasikan sebagai *send task* yang mengirimkan *message* ke *blackbox pool* sistem pembayaran. Setelah pembayaran dilakukan, *message* msgCustomerPayOrder dikirim. Variabel ID pesanan digunakan untuk mengaitkan pembayaran dengan pesanan.

```
private static void RegisterWorkers() {  
    ...  
    workers.Add(MessageThrowJob<CustomerProcVariables>(  
        "PayOrder", "msgCustomerPayOrder",  
        correlationKey: vars => vars.Order.Id!.ToString()!,  
        preCall: vars => logger.LogInformation("Payment for order  
        ID: {OrderId}", vars.Order.Id)  
    ));  
    ...  
}
```

### 5. Konfirmasi pesanan oleh penjual

Setelah pembayaran, penjual harus mengonfirmasi pesanan. Tahap ini direpresentasikan sebagai *user task* berupa formulir kosong.

### 6. Pengemasan dan pengiriman pesanan

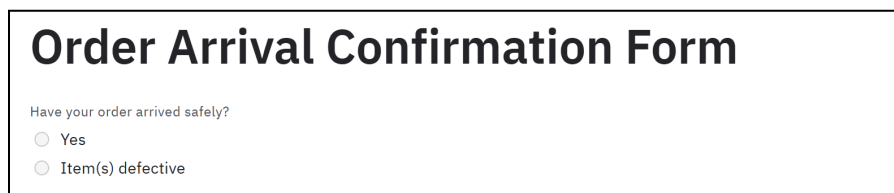
Setelah mengonfirmasi pesanan, penjual dapat mengirimkan pesanan. Tahap ini direpresentasikan sebagai *manual task*.

### 7. Pemberitahuan pengiriman pesanan

Setelah *manual task* untuk mengemas dan mengirim pesanan, terdapat *send task* untuk memberitahukan pengiriman pesanan kepada pembeli (yang kemudian akan diterima oleh *message intermediate catch event* pada proses *customer*). Penjual mengirimkan *message* `msgSellerShipOrder` untuk memberi tahu sistem bahwa pesanan telah dikirim.

```
private static void RegisterWorkers() {  
    ...  
    workers.Add(MessageThrowJob<SellerProcVariables>(  
        "ShipOrder", "msgSellerShipOrder",  
        correlationKey: vars => vars.Order.Id!.ToString()!,  
        preCall: vars => logger.LogInformation("Sent shipment  
        status for order ID: {OrderId}", vars.Order.Id)  
    ));  
    ...  
}
```

### 8. Konfirmasi penerimaan pesanan oleh pembeli



The form is titled "Order Arrival Confirmation Form". Below the title, it asks "Have your order arrived safely?". There are two radio button options: "Yes" and "Item(s) defective".

**Gambar 2.8.1** Formulir Konfirmasi Pesanan Sampai

Setelah pembeli menerima pesanan, pembeli perlu mengisi konfirmasi yang menyatakan apakah pesanan telah diterima dengan baik atau terdapat cacat / kerusakan pada barang. Tahap ini adalah *user task* berupa formulir pertanyaan apakah barang aman atau rusak. Jika barang aman, maka tahapan selanjutnya adalah pesanan selesai. Jika barang rusak, maka tahap selanjutnya adalah pengajuan permintaan pengembalian barang.

## 9. Pengajuan permintaan pengembalian

Tahap ini dilakukan jika pembeli mengisi konfirmasi bahwa barang rusak. Pembeli mengirimkan pesan permintaan pengembalian barang. Tahap ini berupa *message intermediate throw event* yang kemudian diterima oleh proses penjual. *Message* `msgCustomerRequestReturn` dikirim oleh pembeli untuk mengajukan permintaan pengembalian.

```
private static void RegisterWorkers() {  
    ...  
    workers.Add(MessageThrowJob<CustomerProcVariables>(  
        "ReturnRequest", "msgCustomerRequestReturn",  
        correlationKey: vars => vars.Order.Id!.ToString()!,  
        preCall: vars => logger.LogInformation("Return requested  
        for order ID: {OrderId}", vars.Order.Id)  
    ));  
    ...  
}
```

## 10. Persetujuan pengembalian oleh penjual

**The customer wants to return the defective item(s)...**

☐ Approve the return request

**Gambar 2.10.1** Formulir Persetujuan Pengembalian Pesanan

Setelah menerima pesan pengajuan (*catch event*), penjual memutuskan apakah akan menyetujui atau menolak permintaan pengembalian melalui *user task* berupa formulir *checkbox*. Hasil formulir tersebut (variabel `returnRequest` dalam *boolean*) akan dikirimkan melalui *send task* kepada proses pembeli (diterima melalui *catch event*). Jika disetujui, pembeli dapat mengirim barang kembali. Jika ditolak, maka pesanan diselesaikan. *Message* `msgSellerReturnApproval` dikirim oleh penjual dengan variabel yang menyatakan status permintaan pengembalian disetujui atau tidak.

## 11. Pengiriman pengembalian barang oleh pembeli

Jika `returnRequest.approved` bernilai *true* (disetujui), pembeli mengirim barang kembali melalui *manual task*. Setelah itu, sistem mengirimkan *message (throw event)* kepada proses penjual yang mengonfirmasi bahwa barang telah dikirim kembali. *Message* `msgCustomerShipReturn` dikirim oleh pembeli setelah mereka mengirimkan kembali barang ke penjual. Ketika proses penjual menerima *message*

tersebut (*catch event*), *manual task* menerima pesanan dilakukan dan memulai kembali ke proses mengemas dan mengirim kembali barang (*loop*).

```
private static void RegisterWorkers() {  
    ...  
    workers.Add(MessageThrowJob<CustomerProcVariables>(  
        "ShipReturn", "msgCustomerShipReturn",  
        correlationKey: vars => vars.Order.Id!.ToString()!,  
        preCall: vars => logger.LogInformation("Sent back returned  
        item for order ID: {OrderId}", vars.Order.Id)  
    ));  
    ...  
}
```

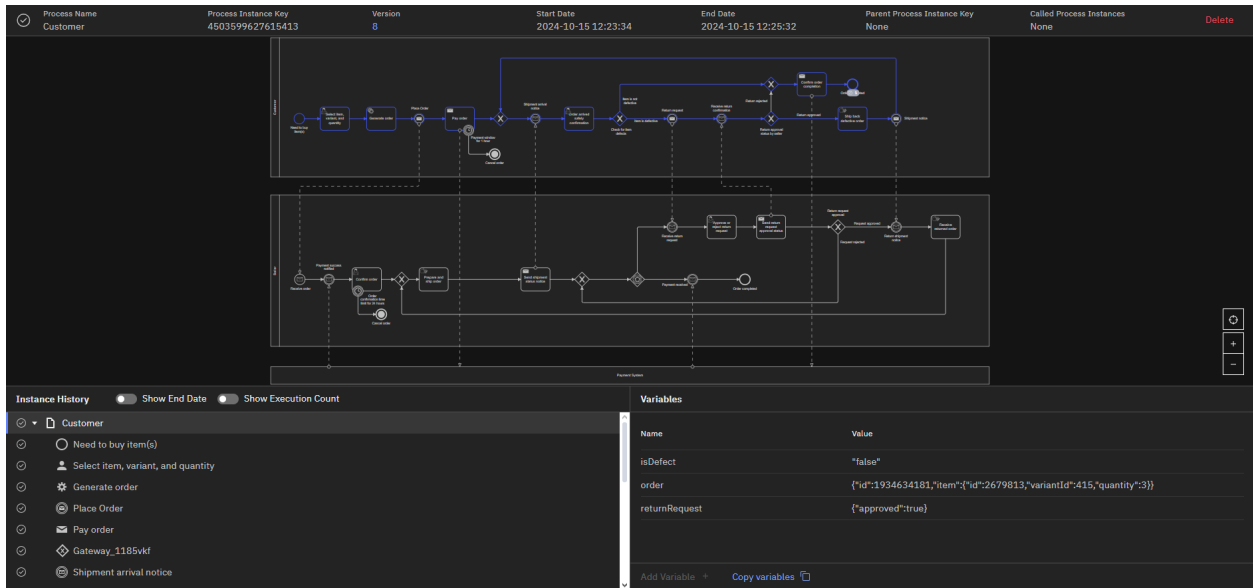
## 12. Penerimaan pembayaran oleh penjual dan pesanan selesai

Jika pembeli mengonfirmasi barangnya telah sampai dengan selamat (atau permintaan pengembalian ditolak), pembeli mengirim *message* `msgCustomerConfirmComplete` untuk memberi tahu sistem bahwa pesanan telah selesai (berupa *send task* ke sistem pembayaran *blackbox*). Setelah melalui sistem pembayaran, *message* ini kemudian diterima oleh proses penjual (*catch event*), serta semua proses selesai dan berhenti.

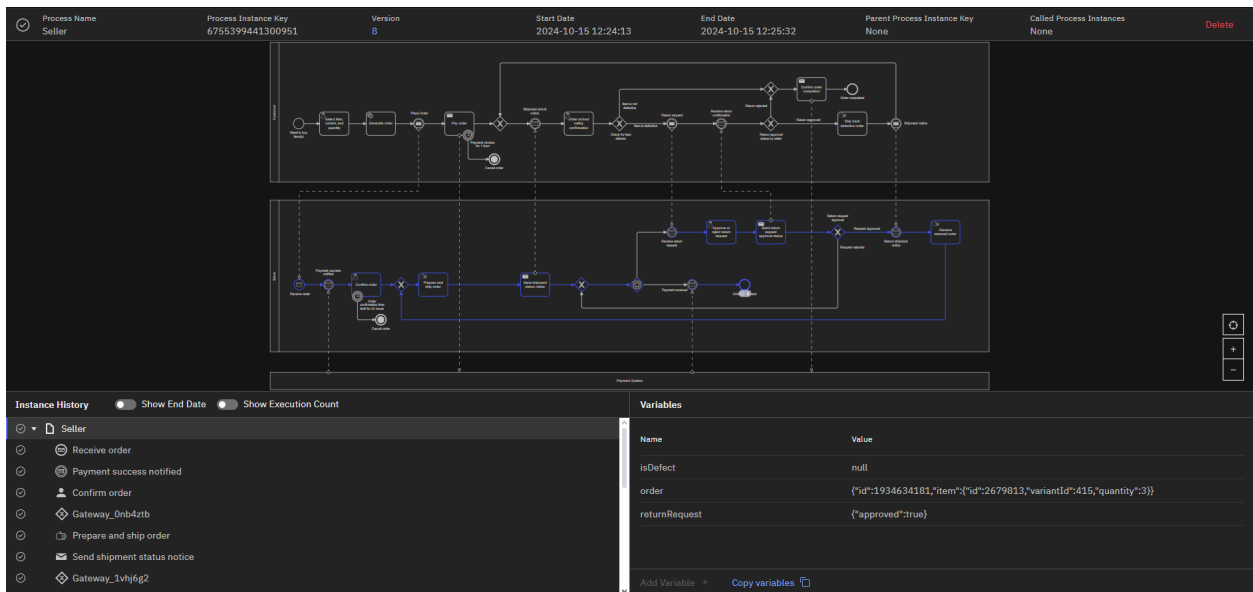
```
private static void RegisterWorkers() {  
    ...  
    workers.Add(MessageThrowJob<SellerProcVariables>(  
        "ReturnApproval", "msgSellerReturnApproval",  
        correlationKey: vars => vars.Order.Id!.ToString()!,  
        sentVariables: JsonConvert.SerializeObject,  
        preCall: vars => logger.LogInformation("Return request  
        {Approval} for order ID: {OrderId}",  
        (vars.ReturnRequest?.Approved ?? false) ? "approved" :  
        "rejected", vars.Order.Id  
    )  
    ));  
    ...  
}
```

### 3. Hasil *Deployment*

Berikut merupakan hasil *deployment* BPMN:



Gambar 3.1 Hasil Eksekusi Proses *Customer*



Gambar 3.2 Hasil Eksekusi Proses *Seller*

2024-10-15 12:24:12.7006	INFO	ECommerceClient.Program	Generated order ID 1934634181 for order with item ID 2679813, Variant ID 415, Qty 3
2024-10-15 12:24:13.0952	INFO	ECommerceClient.Program	Place order for item ID: 2679813, Variant ID: 415, Qty: 3
2024-10-15 12:24:13.7633	INFO	ECommerceClient.Program	Payment for order ID: 1934634181
2024-10-15 12:24:39.7157	INFO	ECommerceClient.Program	Sent shipment status for order ID: 1934634181
2024-10-15 12:24:59.2474	INFO	ECommerceClient.Program	Return requested for order ID: 1934634181
2024-10-15 12:25:16.2976	INFO	ECommerceClient.Program	Return request approved for order ID: 1934634181
2024-10-15 12:25:16.6339	INFO	ECommerceClient.Program	Sent back returned item for order ID: 1934634181
2024-10-15 12:25:16.9044	INFO	ECommerceClient.Program	Sent shipment status for order ID: 1934634181
2024-10-15 12:25:32.5049	INFO	ECommerceClient.Program	Order completion confirmed for order ID: 1934634181

Gambar 3.3 Log Eksekusi Program *Client*



#### 4. Lampiran

Tautan *repository* GitHub: <https://github.com/JerichoFletcher/IF4052-Tugas1-ECommerce>

Tautan *file* model BPMN dan formulir-formulir yang digunakan:

<https://drive.google.com/file/d/1w55eCqVrCYHxGp7HfaxACxIBIRjbzCYo/view?usp=sharing>