

Dokumen Analisis dan Desain

Flight Booking

IF4050 Pembangunan Perangkat Lunak Berorientasi Service



Dibuat oleh:

| | |
|------------------------------|----------|
| Afif Bambang Prasetya | 13515058 |
| Winarto | 13515061 |
| Gianfranco Fertino Hwandiano | 13515118 |

**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2018**

1. Problem Description

Bagi masyarakat yang sering berpergian dari suatu kota ke kota lain atau bahkan dari suatu negeri ke negeri lain, biaya bepergian menjadi hal yang penting. Selain biaya, hal yang tidak kalah penting adalah kemudahan dalam mendapatkan tiket. Oleh karena itu, sekarang terdapat banyak perusahaan yang menempatkan diri sebagai pihak ketiga dalam proses pembelian tiket pesawat. Perusahaan ini memiliki tujuan untuk memudahkan pembeli untuk membeli tiket pesawat dari sekian banyak maskapai penerbangan tanpa harus membelinya langsung di bandara atau *website* maskapai tersebut. Oleh sebab itu, untuk meningkatkan angka penjualan tiket, maskapai penerbangan harus menyediakan *service* yang dapat digunakan dengan mudah sehingga perusahaan pihak ketiga tertarik untuk memasukan maskapai penerbangan sebagai opsi pembelian tiket.

2. Business Requirements

Business requirements merupakan aktivitas penting yang harus dilakukan oleh sebuah organisasi atau perusahaan untuk mencapai objektif atau tujuan yang telah ditetapkan. Jenis *requirement* ini memiliki pandangan yang lebih luas dari sisi bisnis dan lebih fokus pada “*why*”. *Business requirements* kemudian dapat diturunkan menjadi *functional requirements* yang lebih fokus pada “*what*” dan umumnya lebih bersifat teknis, serta jenis *requirements* lainnya.

Scope dari servis ini yaitu sebagai perantara. *Service* menyediakan layanan perantara antar pengguna yang ingin membeli tiket pesawat secara instan melalui elektronik dengan maskapai penerbangan yang menjual tiket pesawat. Berikut ini adalah *business requirements* dari *service flight booking* ini:

1. Meningkatkan penjualan tiket sebanyak 30% dalam 6 bulan sejak terdaftar.
2. Mempercepat proses pembelian tiket pesawat dari metode sebelumnya sebanyak 50% lebih cepat. Metode sebelumnya yaitu membeli langsung di bandara atau *online* lewat *website* maskapai.
3. Mengurangi biaya operasi yang perlu dikeluarkan maskapai sebanyak 20%. Hal tersebut dikarenakan maskapai dapat mengurangi usahanya pada *platform* pembelian tiket.

Berdasarkan *business requirements* tersebut, didapatkan *functional requirement* sebagai berikut:

1. Sistem dapat menerima pesanan pembelian tiket pesawat.
2. Sistem dapat mengembalikan daftar penerbangan yang ada beserta informasi lengkapnya.

3. Sistem dapat menerima permintaan pembatalan tiket pesawat.
4. Sistem dapat menerima pembayaran dengan bantuan sistem lain.
5. Sistem dapat menerima permintaan penurunan tiket harga berdasarkan kode promo.

3. Servis Bisnis dan Model Bisnis Proses

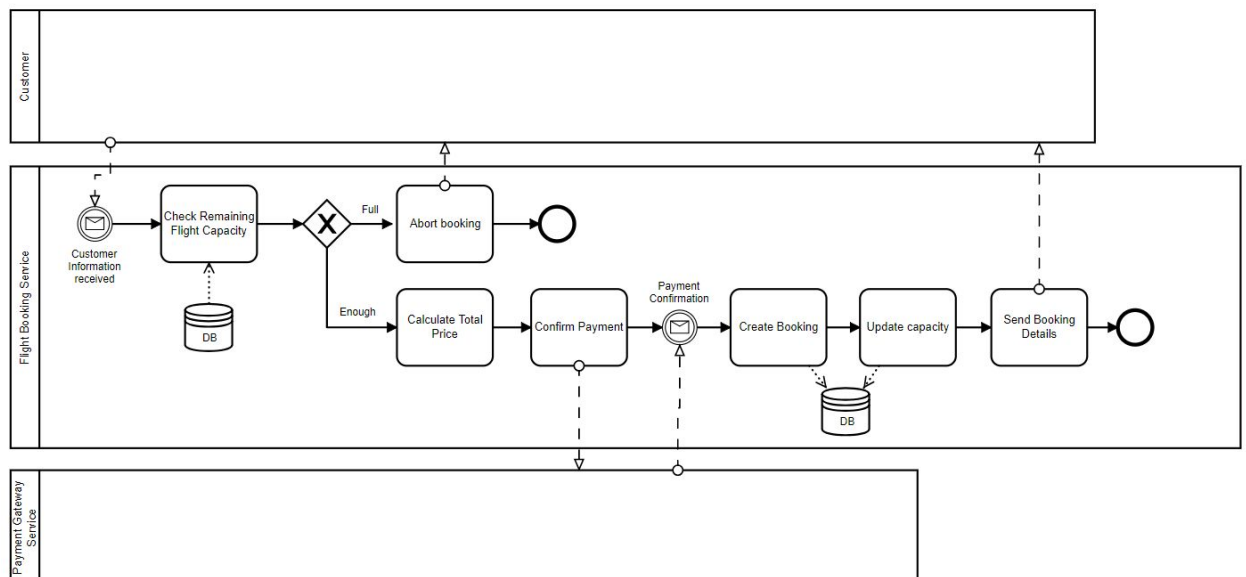
3.1 Task Centric

3.1. 1 Booking Ticket

Input: Flight information, Customer

Output: Flight ticket digital

Deskripsi: Servis menerima *input* berupa tanggal penerbangan, jam penerbangan, kelas, lokasi awal, dan lokasi tujuan. Kemudian servis akan mengecek apakah penerbangan tersebut tersedia atau tidak. Jika iya maka total harga yang perlu dibayar akan dihitung untuk kemudian diteruskan ke *payment gateway service*. Setelah menerima konfirmasi pembayaran, maka servis akan membuat *booking* baru dan memperbaharui kapasitas penerbangan tersebut. *Booking details* akhirnya dikirim ke pengguna.

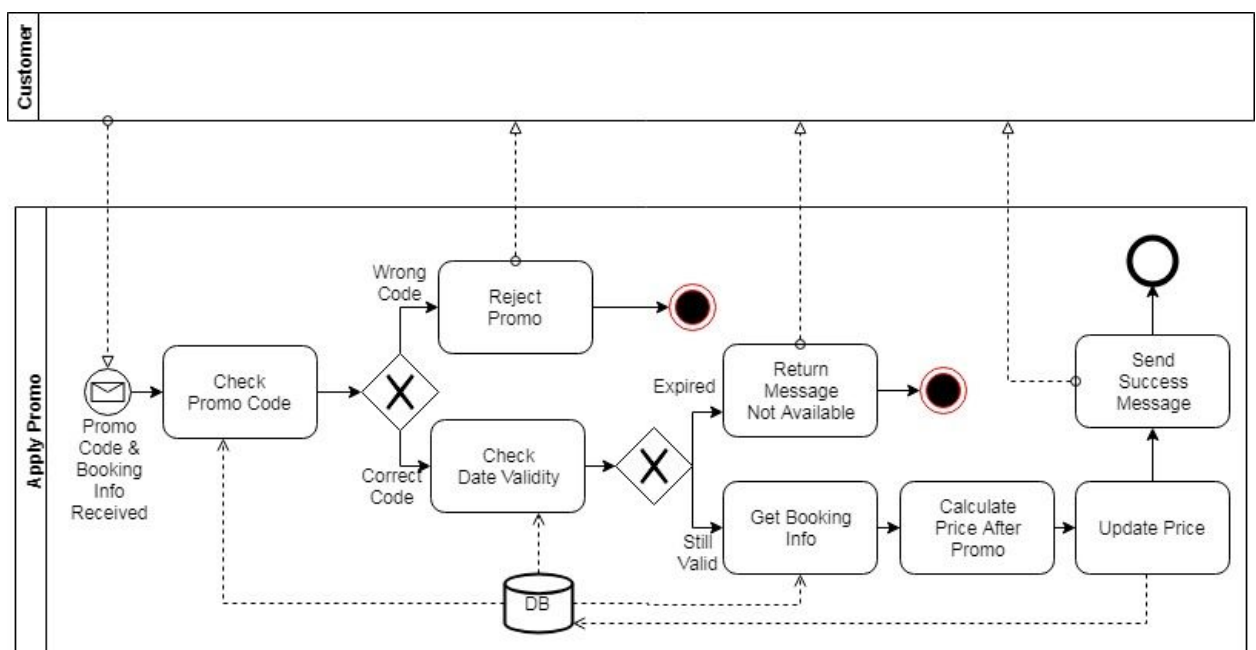


3.1. 2 Apply Promo

Input: Promo Code dan Booking ID

Output: Success / Fail Message

Deskripsi: Servis menerima input berupa *promo code* dan *booking id*. Kemudian servis akan mengecek apakah *promo code* yang dimasukan benar dan masih berlaku. Setelah itu servis akan mengambil *booking info* dari basis data, mengkalkulasi harga setelah promo, dan memperbaharui harga *booking* dari pengguna. Pesan berhasil akan kemudian dikirim ke pengguna.



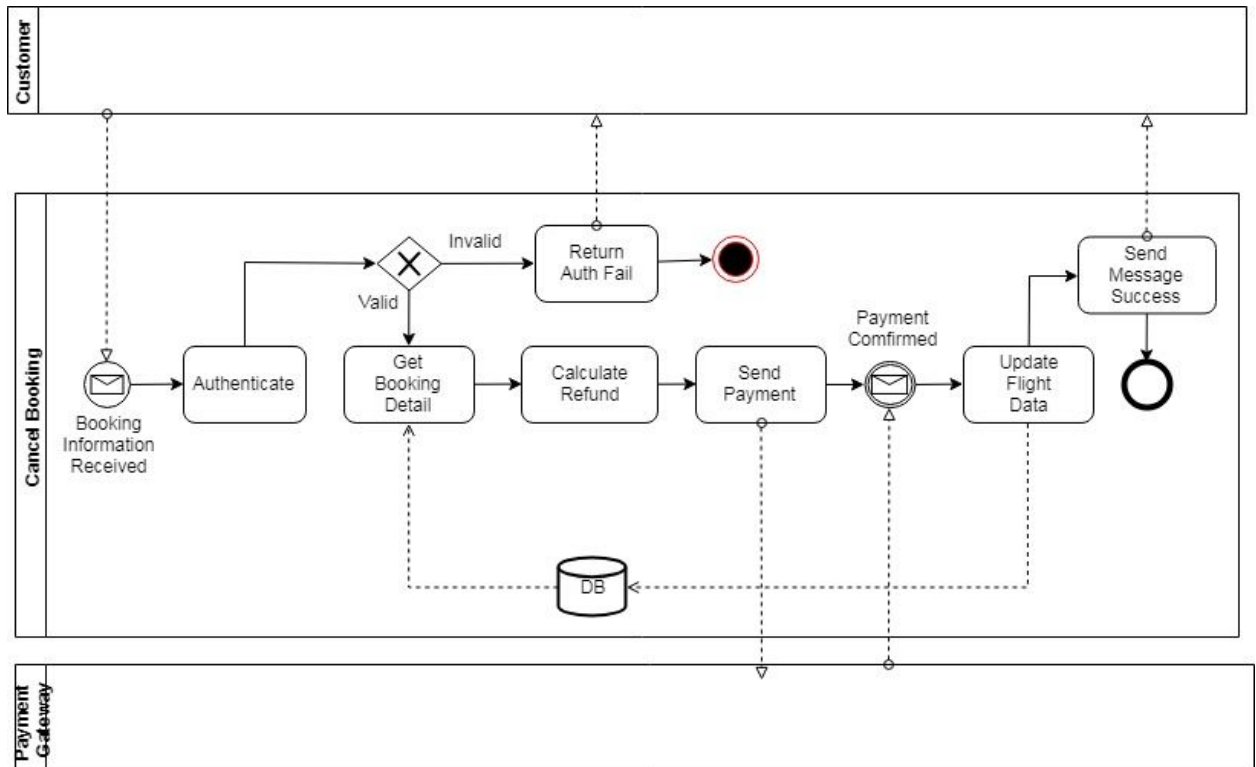
3.1. 3 Cancel Booking

Input: Booking ID, ticket ID, booking passcode

Output: Message (success / fail)

Deskripsi: Servis menerima *input* booking ID untuk mengenali booking, kemudian ticket ID untuk mengenali tiket yang dibatalkan, dan booking passcode untuk keperluan autentikasi. Servis akan mengautentikasi *booking passcode* yang ada agar tidak terjadi pembajakan. Kemudian booking detail akan diambil dari

database berdasarkan *booking ID*. Setelah itu servis akan mengkalkulasi *refund* yang akan diperoleh *customer* berdasarkan rumus yang telah ditetapkan. *Payment* kemudian diteruskan ke *service payment gateway*. Setelah pembayaran selesai, maka data penerbangan diperbaharui dan pesan sukses dikirim ke *customer*.



3.2 Entity Centric

3.2.1 Passenger

| No. | Method | Path | Input [type] | Output | Deskripsi |
|-----|--------|------------|--------------------|---|--------------------------------------|
| 1. | GET | /passenger | Id_passenger [int] | Id_passenger [int] Fullname [string] Phone [string] Email [string] Birthdate [date] Gender [male/female] | Servis mengembalikan data penumpang. |
| 2. | POST | /passenger | Id_passenger [int] | Message [string] | Servis menerima data |

| | | | | | |
|--|--|--|---|--|-----------------|
| | | | Fullname [string] Phone [string] Email [string] Birthdate [date] Gender [male/female] | | penumpang baru. |
|--|--|--|---|--|-----------------|

3.2.2 Flight

| No. | Method | Path | Input [type] | Output | Deskripsi |
|-----|--------|----------|--|---|--|
| 1. | GET | /flights | Date [date] Class [economy/business/first class] Origin [string] Destination [string] | List of [Flight id [int] Date [date] Class [economy/business/first class] Origin [string] Destination [string] Time [time] Price [float]] | Servis mengembalikan daftar penerbangan dan informasinya. |
| 2. | GET | /flights | Flight id [int] | Flight id [int] Date [date] Class [economy/business/first class] Origin [string] Destination [string] Time [time] Price [float] | Servis mengembalikan informasi detail dari suatu penerbangan |

3.2.3 Promotion

| No. | Method | Path | Input [type] | Output | Deskripsi |
|-----|--------|--------|---------------------|--|--|
| 1. | GET | /promo | - | List of [Promo Code [string] AvailableFrom [date] AvailableUntil [date] Discount [float]] | Servis mengembalikan daftar promo yang masih berlaku. |
| 2. | GET | /promo | Promo Code [string] | Promo Code [string] AvailableFrom [date] AvailableUntil [date] Discount [float] | Servis mengembalikan informasi detail dari suatu promo |
| 3. | POST | /promo | Promo Code [string] | Message [string] | Servis membuat |

| | | | | | |
|--|--|--|---|--|-------------|
| | | | AvailableFrom [date] AvailableUntil [date] Discount [float] | | promo baru. |
|--|--|--|---|--|-------------|

3.2.4 Booking

| No. | Method | Path | Input [type] | Output | Deskripsi |
|-----|--------|----------|---|---|---|
| 1. | GET | /booking | Booking ID [int] | Booking ID [int] Passenger ID [int] Number of passenger [int] Total price [float] Flight ID [int] Date [date] Class [economy/business/first class] Origin [string] Destination [string] Time [time] Price [float] | Servis mengembalikan informasi suatu booking. |
| 2. | POST | /booking | Passenger ID [int] Number of passenger [int] Total price [float] Flight ID [int] | Message [string] | Servis membuat booking baru dari data yang dimasukan. |
| 3. | DELETE | /booking | Booking ID [int] | Message [string] | Servis menghapus suatu booking tertentu. |

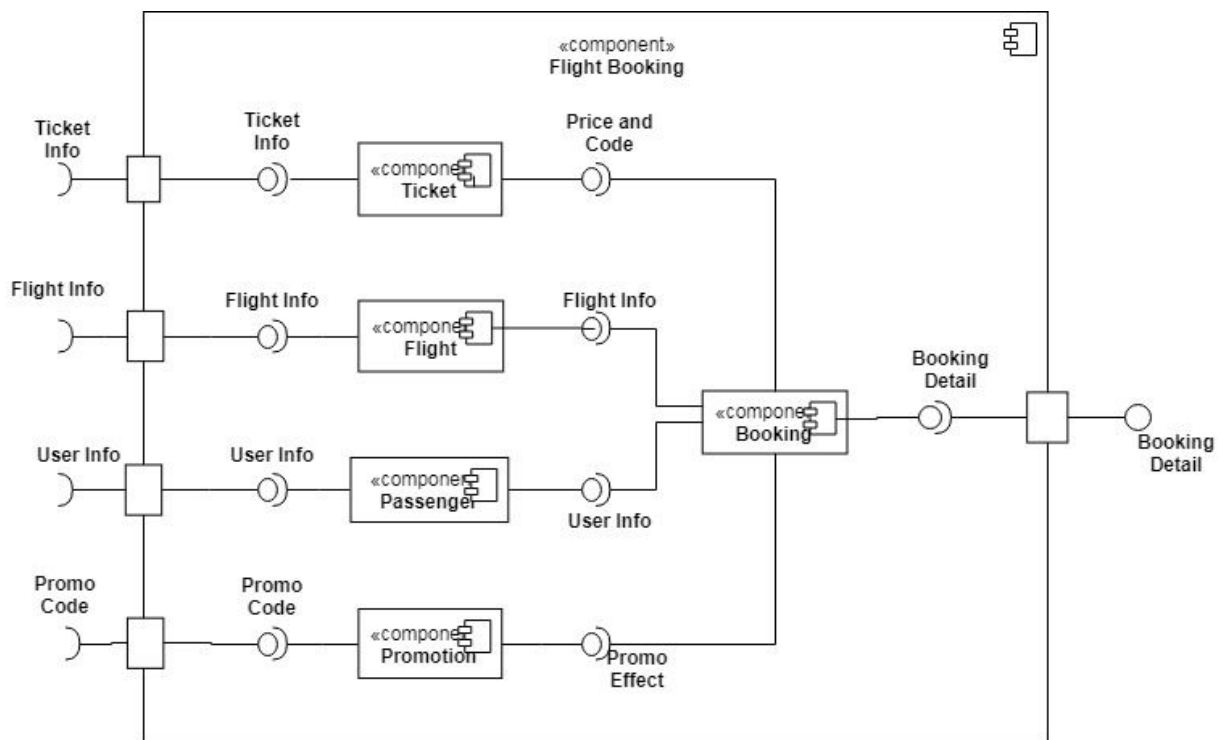
3.2.5 Ticket

| No. | Method | Path | Input [type] | Output | Deskripsi |
|-----|--------|---------|--|--|---|
| 1. | POST | /ticket | Passenger ID [int] Booking ID [int] | List of [Ticket Code [string]] | Servis membuat tiket baru berdasarkan booking pengguna yang jumlahnya sesuai penumpang. |
| 2. | GET | /ticket | Ticket Code [string] | Passenger ID [int] Booking ID [int] Number of passenger [int] Total price [float] Flight ID [int] Date [date] | Servis mengembalikan informasi penerbangan dari suatu tiket. |

| | | | | | |
|--|--|--|--|--|--|
| | | | | Class [economy/business/first class] Origin [string] Destination [string] Time [time] Price [float] | |
|--|--|--|--|--|--|

4. Service Architecture

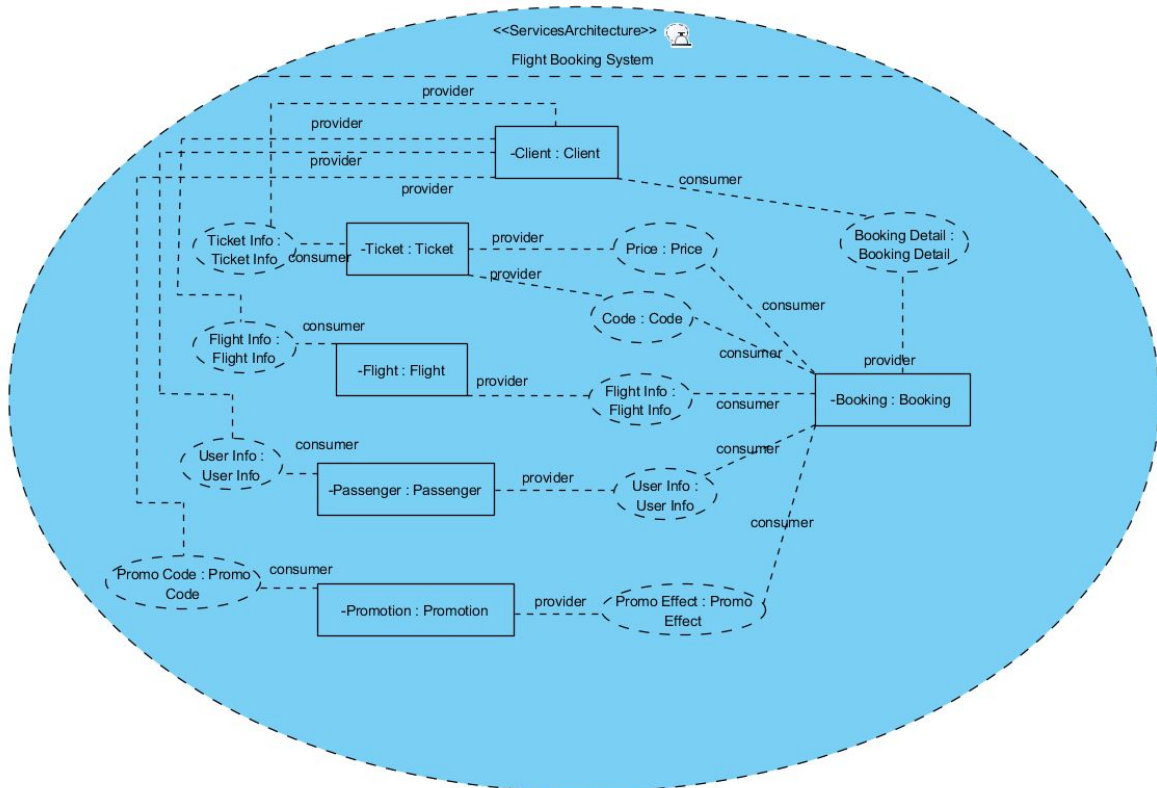
4.1 Component Diagram



Gambar diatas menggambarkan *component diagram* dari sistem pemesanan tiket pesawat ini. Secara garis besar, sistem terdiri atas 5 komponen utama. Komponen Ticket membutuhkan informasi *ticket* yang ingin dipesan untuk kemudian menghasilkan total harga serta kode tiket (digital). Komponen Flight membutuhkan informasi penerbangan untuk kemudian diteruskan ke komponen Booking. Komponen Passenger membutuhkan informasi pengguna yang memesan untuk kemudian diteruskan juga ke komponen Booking. Komponen Promo membutuhkan informasi kode promo yang digunakan untuk

kemudian dicek kebenarannya dan diteruskan ke komponen Booking. Komponen Booking kemudian mengaggregasi semua informasi yang diterima dari komponen lain untuk menghasilkan suatu data *booking* yang utuh dari suatu pengguna.

4.2 SoaML



Gambar di atas menggambarkan diagram SoaML dari servis pemesanan tiket pesawat ini. Sama seperti sebelumnya, terdiri dari 5 komponen / entitas utama. Hubungan antar tiap komponen dapat dilihat langsung pada diagram. Terlihat jelas bahwa siapa yang merupakan *consumer* dan *provider* antar suatu hubungan dari dua komponen. Selain itu terdapat juga data yang menjadi pusat hubungan kedua komponen / entitas.

5. Service Description

5.1 Task Service Description

Berikut adalah *task service description* yang disepakati dengan 2 kelompok lainnya dalam domain *flight booking*:

- Booking Ticket
- Cancel Booking

Selain dua *task-centric service* tersebut, terdapat sebuah *task-centric service description* tambahan (pribadi) yaitu Service Apply Promotion

5.1.1 Booking Ticket

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="com.flightbooking.ws"
xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
xmlns:wsp="http://www.w3.org/ns/ws-policy"
xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
targetNamespace="com.flightbooking.ws" name="BookingServiceImplService">
  <types>
    <xsd:schema>
      <xsd:import namespace="com.flightbooking.ws"
schemaLocation="http://localhost:8000/ws/booking?xsd=1" />
    </xsd:schema>
  </types>
  <message name="bookingFlight">
    <part name="parameters" element="tns:bookingFlight" />
  </message>
  <message name="bookingFlightResponse">
    <part name="parameters" element="tns:bookingFlightResponse" />
  </message>
  <message name="validatePayment">
    <part name="parameters" element="tns:validatePayment" />
  </message>
  <message name="validatePaymentResponse">
    <part name="parameters" element="tns:validatePaymentResponse" />
  </message>
```

```

</message>
<message name="abortBooking">
  <part name="parameters" element="tns:abortBooking" />
</message>
<message name="abortBookingResponse">
  <part name="parameters" element="tns:abortBookingResponse" />
</message>
<portType name="BookingServiceImpl">
  <operation name="bookingFlight">
    <input
wsam:Action="com.flightbooking.ws/BookingServiceImpl/bookingFlightRequest"
message="tns:bookingFlight" />
    <output
wsam:Action="com.flightbooking.ws/BookingServiceImpl/bookingFlightResponse"
message="tns:bookingFlightResponse" />
    </operation>
    <operation name="validatePayment">
      <input
wsam:Action="com.flightbooking.ws/BookingServiceImpl/validatePaymentRequest"
message="tns:validatePayment" />
      <output
wsam:Action="com.flightbooking.ws/BookingServiceImpl/validatePaymentResponse"
message="tns:validatePaymentResponse" />
      </operation>
    <operation name="abortBooking">
      <input
wsam:Action="com.flightbooking.ws/BookingServiceImpl/abortBookingRequest"
message="tns:abortBooking" />
      <output
wsam:Action="com.flightbooking.ws/BookingServiceImpl/abortBookingResponse"
message="tns:abortBookingResponse" />
      </operation>
    </portType>
    <plnk:partnerLinkType name="FlightBookingPLT">
      <plnk:role name="BookingService" portType="BookingServiceImpl"/>
    </plnk:partnerLinkType>
    <binding name="BookingServiceImplPortBinding" type="tns:BookingServiceImpl">
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"
/>

    <operation name="bookingFlight">
      <soap:operation soapAction="" />
      <input>
        <soap:body use="literal" />
      </input>

```

```

        <output>
            <soap:body use="literal" />
        </output>
    </operation>
    <operation name="validatePayment">
        <soap:operation soapAction="" />
        <input>
            <soap:body use="literal" />
        </input>
        <output>
            <soap:body use="literal" />
        </output>
    </operation>
    <operation name="abortBooking">
        <soap:operation soapAction="" />
        <input>
            <soap:body use="literal" />
        </input>
        <output>
            <soap:body use="literal" />
        </output>
    </operation>
</binding>
<service name="BookingServiceImplService">
    <port name="BookingServiceImplPort"
binding="tns:BookingServiceImplPortBinding">
        <soap:address location="http://localhost:8000/ws/task" />
    </port>
</service>
</definitions>

```

5.1.2 Apply Promo

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://api.flightbooking.com/FlightBookingService/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="FlightBookingService"
targetNamespace="http://api.flightbooking.com/FlightBookingService/">
    <wsdl:types>
        <xsd:schema targetNamespace="http://api.flightbooking.com/FlightBookingService/">
            <xsd:element name="Validate">

```

```

        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="in" type="xsd:string"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="ValidateResponse">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="out" type="xsd:boolean"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
</wsdl:types>
<wsdl:message name="ValidateRequest">
    <wsdl:part element="tns:Validate" name="promoCode"/>

    <wsdl:part name="bookingInfo" type="xsd:string"></wsdl:part>
</wsdl:message>
<wsdl:message name="ValidateResponse">
    <wsdl:part element="tns:ValidateResponse" name="parameters"/>
</wsdl:message>
<wsdl:portType name="CheckPromoCodeService">
    <wsdl:operation name="Validate">
        <wsdl:input message="tns:ValidateRequest"/>
        <wsdl:output message="tns:ValidateResponse"/>
    </wsdl:operation>

</wsdl:portType>
<wsdl:binding name="FlightBookingServiceSOAP" type="tns:CheckPromoCodeService">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="Validate">
        <soap:operation
soapAction="http://api.flightbooking.com/FlightBookingService/Validate"/>
        <wsdl:input>
            <soap:body use="literal"/>
        </wsdl:input>
        <wsdl:output>
            <soap:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
<wsdl:service name="FlightBookingService">

```

```

    <wsdl:port binding="tns:FlightBookingServiceSOAP"
name="FlightBookingServiceSOAP">
        <soap:address location="https://api.flightbooking.com"/>
    </wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

5.1.3 Cancel Booking

```

<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="com.flightbooking.ws"
xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
xmlns:wsp="http://www.w3.org/ns/ws-policy"
xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
targetNamespace="com.flightbooking.ws" name="CancelServiceImplService">
    <types>
        <xsd:schema>
            <xsd:import namespace="com.flightbooking.ws"
schemaLocation="http://localhost:8000/ws/cancel?xsd=1" />
        </xsd:schema>
    </types>
    <message name="validateBookingNumber">
        <part name="parameters" element="tns:validateBookingNumber" />
    </message>
    <message name="validateBookingNumberResponse">
        <part name="parameters" element="tns:validateBookingNumberResponse" />
    </message>
    <message name="rejectCancelation">
        <part name="parameters" element="tns:rejectCancelation" />
    </message>
    <message name="rejectCancelationResponse">
        <part name="parameters" element="tns:rejectCancelationResponse" />
    </message>
    <message name="calculateRefund">
        <part name="parameters" element="tns:calculateRefund" />
    </message>
    <message name="calculateRefundResponse">
        <part name="parameters" element="tns:calculateRefundResponse" />
    </message>

```

```

</message>
<portType name="CancelServiceImpl">
  <operation name="validateBookingNumber">
    <input
wsam:Action="com.flightbooking.ws/CancelServiceImpl/validateBookingNumberRequest"
message="tns:validateBookingNumber" />
    <output
wsam:Action="com.flightbooking.ws/CancelServiceImpl/validateBookingNumberResponse"
message="tns:validateBookingNumberResponse" />
    </operation>
    <operation name="rejectCancelation">
      <input
wsam:Action="com.flightbooking.ws/CancelServiceImpl/rejectCancelationRequest"
message="tns:rejectCancelation" />
      <output
wsam:Action="com.flightbooking.ws/CancelServiceImpl/rejectCancelationResponse"
message="tns:rejectCancelationResponse" />
      </operation>
      <operation name="calculateRefund">
        <input
wsam:Action="com.flightbooking.ws/CancelServiceImpl/calculateRefundRequest"
message="tns:calculateRefund" />
        <output
wsam:Action="com.flightbooking.ws/CancelServiceImpl/calculateRefundResponse"
message="tns:calculateRefundResponse" />
        </operation>
      </portType>
      <plnk:partnerLinkType name="CancelBookingPLT">
        <plnk:role name="CancelService" portType="CancelServiceImpl"/>
      </plnk:partnerLinkType>
      <binding name="CancelServiceImplPortBinding" type="tns:CancelServiceImpl">
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"
/>

        <operation name="validateBookingNumber">
          <soap:operation soapAction="" />
          <input>
            <soap:body use="literal" />
          </input>
          <output>
            <soap:body use="literal" />
          </output>
        </operation>
        <operation name="rejectCancelation">
          <soap:operation soapAction="" />

```

```

        <input>
            <soap:body use="literal" />
        </input>
        <output>
            <soap:body use="literal" />
        </output>
    </operation>
    <operation name="calculateRefund">
        <soap:operation soapAction="" />
        <input>
            <soap:body use="literal" />
        </input>
        <output>
            <soap:body use="literal" />
        </output>
    </operation>
</binding>
<service name="CancelServiceImplService">
    <port name="CancelServiceImplPort" binding="tns:CancelServiceImplPortBinding">
        <soap:address location="http://localhost:8000/ws/cancel" />
    </port>
</service>
</definitions>

```

5.2 Entity Service Description

5.2.1 Passenger

```

<resource path="passengers" id="/passengers">
    <doc xml:lang="en" title="/passengers"/>
    <method name="GET" id="GET">
        <doc xml:lang="en" title="GET"/>
        <request/>
        <response status="">
            <representation mediaType="application/json"/>
        </response>
    </method>
    <method name="POST" id="POST">
        <doc xml:lang="en" title="POST"/>
        <request>
            <representation mediaType="application/x-www-form-urlencoded"/>
        </request>
    </method>

```



```
</request>
<response status="">
  <representation mediaType="application/x-www-form-urlencoded"/>
  <representation mediaType="application/json"/>
</response>
</method>
</resource>
```

5.2.2 Flight

```
<resource path="flights" id="/flights">
  <doc xml:lang="en" title="/flights"/>
  <resource path="{flight-number}" id="/{flight-number}">
    <doc xml:lang="en" title="/{flight-number}"/>
    <param name="seller-id" type="xs:int" required="false" default=""
style="template" xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
    <method name="GET" id="GET">
      <doc xml:lang="en" title="GET"/>
      <request/>
      <response status="">
        <representation mediaType="application/json"/>
      </response>
    </method>
  </resource>
<method name="GET" id="GET">
  <doc xml:lang="en" title="GET"/>
  <request/>
  <response status="">
    <representation mediaType="application/json"/>
  </response>
</method>
</resource>
```

5.2.3 Promotion

```
<resource path="promotions" id="/promotions">
  <doc xml:lang="en" title="/promotions"/>
  <resource path="{promo-code}" id="/{promo-code}">
    <doc xml:lang="en" title="/{promo-code}"/>
```

```

    <param name="buyer-id" type="xs:int" required="false" default="" style="template"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
    <method name="GET" id="GET">
        <doc xml:lang="en" title="GET"/>
        <request/>
        <response status="">
            <representation mediaType="application/json"/>
        </response>
    </method>
</resource>
<method name="GET" id="GET">
    <doc xml:lang="en" title="GET"/>
    <request/>
    <response status="">
        <representation mediaType="application/json"/>
    </response>
</method>
<method name="POST" id="POST">
    <doc xml:lang="en" title="POST"/>
    <request>
        <representation mediaType="application/x-www-form-urlencoded"/>
    </request>
    <response status="">
        <representation mediaType="application/x-www-form-urlencoded"/>
        <representation mediaType="application/json"/>
    </response>
</method>
</resource>

```

5.2.4 Booking

```

<resource path="bookings" id="/bookings">
    <doc xml:lang="en" title="/bookings"/>
    <resource path="{booking-id}" id="/{booking-id}">
        <doc xml:lang="en" title="/{booking-id}"/>
        <param name="order-id" type="xs:int" required="false" default="" style="template"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
        <method name="DELETE" id="DELETE">
            <doc xml:lang="en" title="DELETE"/>
            <request/>
            <response status="">
                <representation mediaType="application/json"/>
            </response>
        </method>
    </resource>
</resource>

```

```

    </response>
  </method>
  <method name="GET" id="GET">
    <doc xml:lang="en" title="GET"/>
    <request/>
    <response status="">
      <representation mediaType="application/json"/>
    </response>
  </method>
</resource>
<method name="POST" id="POST">
  <doc xml:lang="en" title="POST"/>
  <request>
    <representation mediaType="application/x-www-form-urlencoded"/>
  </request>
  <response status="">
    <representation mediaType="application/x-www-form-urlencoded"/>
    <representation mediaType="application/json"/>
  </response>
</method>
</resource>

```

5.2.5 Ticket

```

<resource path="tickets" id="/tickets">
  <doc xml:lang="en" title="/tickets"/>
  <resource path="{ticket-id}" id="/{ticket-id}">
    <doc xml:lang="en" title="/{ticket-id}"/>
    <param name="buyer-id" type="xs:int" required="false" default="" style="template"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
    <method name="GET" id="GET">
      <doc xml:lang="en" title="GET"/>
      <request/>
      <response status="">
        <representation mediaType="application/json"/>
      </response>
    </method>
  </resource>
  <method name="POST" id="POST">
    <doc xml:lang="en" title="POST"/>
    <request>
      <representation mediaType="application/x-www-form-urlencoded"/>
    </request>
  </method>
</resource>

```

```
</request>
<response status="">
  <representation mediaType="application/x-www-form-urlencoded"/>
  <representation mediaType="application/json"/>
</response>
</method>
</resource>
```

6. Implementasi

6.1. Task Service

Implementasi task service dengan menggunakan Business Process Engine Camunda. Kami memodelkan ketiga service yang akan kami kembangkan yaitu Apply Promo, Booking Ticket, dan Cancel Ticket dengan menggunakan Camunda Modeler. Dan untuk setiap task ditentukan mana yang menjadi service task dan yang didefinisikan sebagai message task.

Implementasi setiap service task menggunakan bahasa Java. Untuk service task yang membutuhkan koneksi ke database menggunakan jdbc mysql connector. Proses akan dieksekusi jika menerima input dari pengguna. Setelah menerima input, maka service akan menjalankan setiap task. *Passing* output dari suatu task ke task lain dengan menggunakan *setVariable* yang disediakan oleh Camunda.

Untuk menerima input dari pengguna, kami membuat sebuah soap wrapper yang *wsdl*nya diekspose ke pengguna. WSDL tersebut dapat dikonsumsi oleh pengguna, kemudian yang dilakukan oleh server adalah menerima input tersebut kemudian diteruskan ke engine-rest Camunda. Detail mengenai cara pemakaian dan format input serta kode program dapat dilihat di <http://gitlab.informatika.org/IF4050-FlightBooking/ppls-flightbookingservice>.

Service task ini dideploy di *Virtual Private Server* DigitalOcean pada webserver tomcat dan alamat <http://178.128.80.25:8080>. Untuk menjalankan task service secara manual dapat dengan melakukan request POST dengan Content-Type: application/json ke alamat <http://178.128.80.25:8080/engine-rest/message> dengan request body sebagai berikut

Apply Promo Input

```
{
  "messageName" : "Apply Promo",
  "businessKey" : "12345",
  "processVariables" : {
    "promoCode" : {
```

```
        "value" : "BAKULBDG",
        "type": "String"
    },
    "bookingId" : {
        "value" : 2,
        "type": "Integer"
    }
},
"resultEnabled" : true
}
```

Output

```
[
  {
    "resultType": "ProcessDefinition",
    "execution": null,
    "processInstance": {
      "links": [],
      "id": "c770988d-f2fe-11e8-856c-02421d4d19d0",
      "definitionId":
"ApplyPromo:3:70188742-f26e-11e8-856c-02421d4d19d0",
      "businessKey": "12345",
      "caseInstanceId": null,
      "ended": true,
      "suspended": false,
      "tenantId": null
    }
  }
]
```

Booking Ticket Input

```
{
  "messageName" : "Booking Ticket",
  "businessKey" : "12345",
  "processVariables" : {
    "numOfPassenger" : {
      "value" : 1,
      "type": "Integer"
    },
    "passengerId" : {
      "value" : 1,
      "type": "Integer"
    },
    "flightId" : {
      "value" : 2,
      "type": "Integer"
    }
  }
}
```

```
    }
  },
  "resultEnabled" : true
}
```

Output

```
[
  {
    "resultType": "ProcessDefinition",
    "execution": null,
    "processInstance": {
      "links": [],
      "id": "70384656-f2f7-11e8-856c-02421d4d19d0",
      "definitionId":
"BookingTicket:3:701f6516-f26e-11e8-856c-02421d4d19d0",
      "businessKey": "12345",
      "caseInstanceId": null,
      "ended": false,
      "suspended": false,
      "tenantId": null
    }
  }
]
```

Cancel Booking Input

```
{
  "messageName" : "Cancel Booking",
  "businessKey" : "12345",
  "processVariables" : {
    "bookingId" : {
      "value" : 2,
      "type": "Integer"
    }
  },
  "resultEnabled" : true
}
```

Output

```
[
  {
    "resultType": "ProcessDefinition",
    "execution": null,
    "processInstance": {
```

```

        "links": [],
        "id": "4db75970-f31e-11e8-856c-02421d4d19d0",
        "definitionId":
"CancelBooking:3:701bbb94-f26e-11e8-856c-02421d4d19d0",
        "businessKey": "12345",
        "caseInstanceId": null,
        "ended": false,
        "suspended": false,
        "tenantId": null
    }
}
]

```

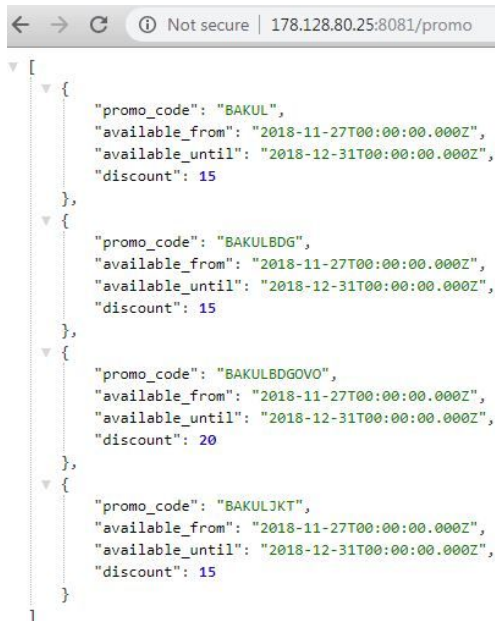
6.2. Entity Service

Pertama-tama, kami mendefinisikan *route* atau alamat dari servis ini. Karena berbasis entitas, maka kami membagi *route* menjadi 5 partisi utama sesuai jumlah entitas yang ada. Kemudian untuk tiap entitasnya, kami menentukan operasi mana saja yang sekiranya diperlukan dan tidak diperlukan. Operasi tidak harus penuh CRUD (Create, Read, Update, Delete). Misalnya, perusahaan biasanya enggan menghapus data pengguna oleh karena itu operasi *Delete* tidak diterapkan pada entitas pengguna. Definisi lengkap dari *route entity service* dapat dilihat pada bagian 3.1.

Kemudian kami memutuskan *technology stack* yang akan digunakan untuk membangun servis ini. Untuk *server side language* kami memilih NodeJS karena kami lebih familiar dan lebih memakan waktu lebih singkat dalam pengembangannya ketimbang bahasa lain. Untuk DBMS kami memilih MySQL karena dirasa cocok dengan jenis data yang kami simpan yang bersifat relasional.

Design pattern yang digunakan berbasiskan *model driven development*. Artinya aplikasi dirancang sehingga terbagi menjadi sejumlah komponen-komponen sesuai entitas / model yang ada. Kode lengkapnya dapat dilihat pada http://gitlab.informatika.org/IF4050-FlightBooking/ppls-flightbookingservice/tree/master/entity_service.

Entity Service telah di-deploy pada *Virtual Private Server* yaitu Digital Ocean dengan alamat <http://178.128.80.25:8081>. Berikut ini kami berikan salah satu contoh *entity promo* yaitu GET semua promo yang ada.



```
[
  {
    "promo_code": "BAKUL",
    "available_from": "2018-11-27T00:00:00.000Z",
    "available_until": "2018-12-31T00:00:00.000Z",
    "discount": 15
  },
  {
    "promo_code": "BAKULBDG",
    "available_from": "2018-11-27T00:00:00.000Z",
    "available_until": "2018-12-31T00:00:00.000Z",
    "discount": 15
  },
  {
    "promo_code": "BAKULBDGGOV0",
    "available_from": "2018-11-27T00:00:00.000Z",
    "available_until": "2018-12-31T00:00:00.000Z",
    "discount": 20
  },
  {
    "promo_code": "BAKUL3KT",
    "available_from": "2018-11-27T00:00:00.000Z",
    "available_until": "2018-12-31T00:00:00.000Z",
    "discount": 15
  }
]
```

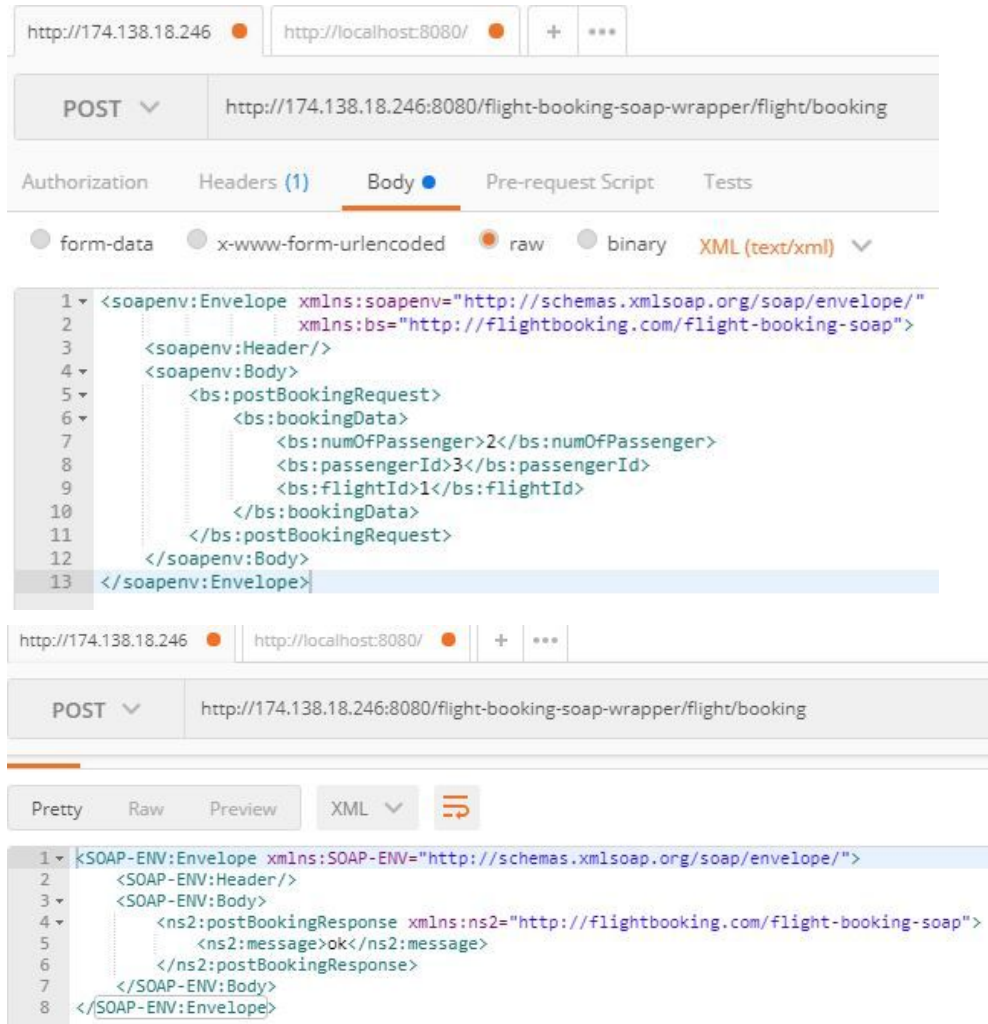
6.3. SOAP Wrapper Service

Dikarenakan Task Service yang dibangun oleh Camunda bersifat RESTful, maka kami membangun sebuah SOAP *service* yang bersifat sebagai *wrapper* dari Task Service tersebut. Service SOAP ini dapat menerima *request* dengan format data XML atau *request* secara langsung menggunakan WSDL yang disediakan. Service SOAP ini hanya bertugas menerima *request* XML, mengubahnya menjadi JSON dan kemudian meneruskan *request* ke Task-based Service Camunda.

Service dibangun menggunakan Java Spring Boot dengan alasan *compatibility* dengan transaksi data bertipe XML. Kodenya dapat dilihat langsung di <http://gitlab.informatika.org/IF4050-FlightBooking/ppls-flightbookingservice/tree/master/flght-booking-soap>. Service telah di-*deploy* di Virtual Private Server yang sama dengan Service lainnya tapi dengan *port* berbeda, yaitu:

1. <http://174.138.18.246:8080/flight-booking-soap-wrapper/flight/booking> untuk *booking*.
2. <http://174.138.18.246:8080/flight-booking-soap-wrapper/flight/cancel-booking> untuk *cancel booking*.
3. <http://174.138.18.246:8080/flight-booking-soap-wrapper/flight/apply-promo> untuk *apply promo*.

Request body untuk masing-masing *task service* dapat dilihat pada direktori `flight-booking-soap/src/main/resources` pada *file* `booking-request.xml`, `cancel-booking-request.xml`, dan `apply-promo-request.xml`. Berikut adalah hasil contoh *request* pada *task service booking*.



Untuk *request* juga dapat dilakukan melalui WSDL yang ada.

7. Integrasi

Dari ketiga *task centric service* yang kami kembangkan, hanya 2 *service* yang memerlukan integrasi dengan *service* lain yaitu *service* Booking Ticket dan Cancel Booking. *Service* yang kami integrasikan adalah *service* Payment Gateway EasyPay yang dokumentasinya dapat diakses pada alamat <https://github.com/nieltg/easypay>.

Karena *service* payment gateway easypay tidak menyediakan url callback yang dapat digunakan untuk melanjutkan eksekusi bp engine camunda dengan *task intermediate message catch event*. Maka, kami membuat sebuah wrapper baru yang menerima request rest dari *send message task* camunda. Ketika wrapper ini menerima request dari camunda, maka yang dilakukan adalah wrapper akan melakukan request ke easypay untuk melakukan pembayaran. Setelah pembayaran sukses, wrapper ini akan mengirimkan pesan ke camunda untuk melanjutkan proses karena pembayaran telah berhasil dilakukan.

Implementasi wrapper ini menggunakan node js yang dapat potongan kodenya sebagai berikut

```
router.get('/', function(req, res, next) {
  let paymentMethod = req.query.paymentMethod;
  let amount = req.query.amount;
  let service = req.query.service;
  let businessKey = req.query.businessKey;
  let xml = getXMLRequest(paymentMethod, amount);
  res.send(xml);
  request.post(
    {url: url,
     body : xml,
     headers: {'Content-Type': 'text/xml'}}
  ),
  function (error, response, body) {
    if (!error) {
      bodyMessage["businessKey"] = businessKey;
      if (service == "payment"){
        bodyMessage["messageName"] = "Confirm Payment";
      } else if(service == "refund"){
        bodyMessage["messageName"] = "Confirm Refund";
      }
      request.post(
        {url: "http://178.128.80.25:8080/engine-rest/message",
         body : JSON.stringify(bodyMessage),
         headers: {'Content-Type': 'application/json'}}
      ),
      function (error, response, body) {
        if (!error) {
          console.log("ok");
        }
      }
    }
  }
});
});
```

Implementasi service task pada Java untuk melakukan request get terhadap wrapper:

```
public void execute(DelegateExecution execution) throws Exception {
  Booking booking = (Booking)
  execution.getVariable("BookingDetail");
  String urlToRead =
  "http://localhost:8082/?paymentMethod=bank&amount=" + booking.getTotalPrice()
  + "&businessKey=" + execution.getBusinessKey() + "&service=payment";
  execution.setVariable("payment", true);
  StringBuilder result = new StringBuilder();
  URL url = new URL(urlToRead);
  HttpURLConnection conn = (HttpURLConnection) url.openConnection();
```

```
        conn.setRequestMethod("GET");
        BufferedReader rd = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
        String line;
        while ((line = rd.readLine()) != null) {
            result.append(line);
        }
        rd.close();
    }
```