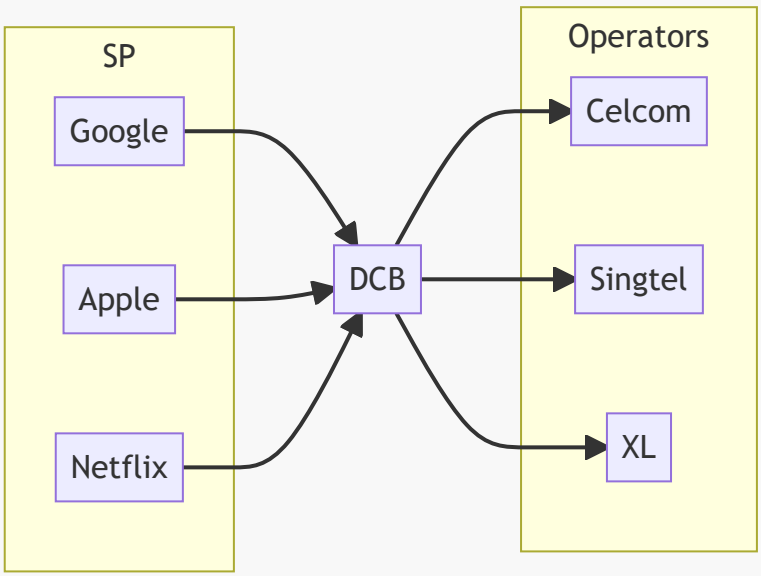
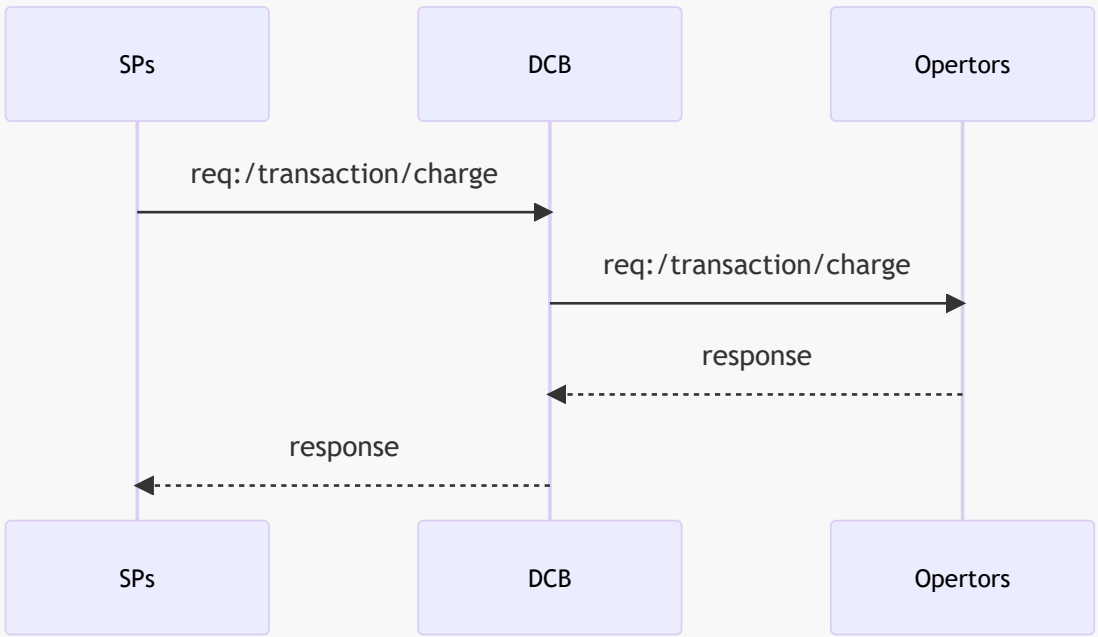
DCB Java Developer - Technical Test

- **Usage Case**: Boost Connect `Direct Carrier Billing platform` (DCB) stands in between `Service Providers` (ex: Google Play, Apple Store, Netflix...) and `Telecom Opertators` (ex: Celcom, XL, Singtel...).



The platform consists of multiple services working together to process online transaction by charging payment to mobile phone carrier bill.

Operator Adapter



To perform a transaction, each SP will send a `transaction/charge` request to DCB. Hence, DCB will need to forward the same to the operators's backend.

DCB defines a `standard request structure` for SP to be used. However, the operator backend may not understand our request. To solve this issue, we need to build an `Operator Adapter Service` to transform the request before sending it to corresponding operator backend.

Problem 1: Charge Request Handling

Please help to build a service using `Spring Boot` which receives the request at `/api/transaction/charge`, do the transformation and send the request to the operator backend. Sample request & response are listed in Appendix-A. The transaction should be tracked in database and should be updated upon receiving the response.

Problem 2 (`Optional`): Authentication

Operator's backend only accepts authenticated request. Thus, the adapter needs to get an `access_token` from the operator and put it in the `Authorization` header before calling operator. Refer to Appendix-B for the `generateToken` API.

Please help to enhance your adapter to handle `access_token` for ensuring all the requests being sent with a valid token.

NOTE:

- We will focus on the main flow, thus validation rule is flexible.
- You can choose either `MySQL` or `Postgres` as database
- Other info if not mention, you can make your own assumption.

Appendix-A

- `POST /<your-service>/api/transaction/charge`

```
Sample Request:

{
    "chargingRequest": {
        "customerInfo": {
            "mobileNo": "6281990449203"
        },
        "transactionInfo": {
            "transactionId": "ff0ced5b-eb31-4eea-8105-d4ebe9557c88",
            "item": "SPOTIFY_G",
            "itemDescription": "SPOTIFY_G",
            "balanceType": "AirBalance",
            "amount": "1000",
            "currency": "IDR"
        }
}
```

```
Sample Response:

{
    "chargingResponse": {
        "customerInfo": {
            "mobileNo": "6281990449203"
        },
        "transactionInfo": {
            "transactionId": "XL20210523100000000002441568162819904492O3",
            "responseStatus": "Success", //or Failed
            "responseDesc": "Request has been execute successfully"
        }
    }
}
```

- `POST /<operator-backend-service>/carrier/payment`

```
Sample Request

#header
Authorization: <access_token>

#body
{
    "transactionInfo": {
        "msisdn": "6281990449203", //mobileNo
        "refId": "ff0ced5b-eb31-4eea-8105-d4ebe9557c88", //transactionId
        "item": "SPOTIFY_G",
        "itemDescription": "SPOTIFY_G",
        "balanceType": "AirBalance",
        "chargeAmount": "1000",
        "currency": "IDR"
    }
}

Sample Response
{
    "transactionInfo": {
        "msisdn": "6281990449203",
        "refId": "ff0ced5b-eb31-4eea-8105-d4ebe9557c88",
        "responseCode": "00",
        "responseDesc": "Request has been execute successfully"
    }
}
```

- `responseCode`:
  - 00: successful
  - 11: failed

Appendix-B

- `GET /<operator-backend-service>/generateToken?service_key=<provided_service_key>&service_name=<provided_service_name>`

```
Sample Response:

{
    "access_token": "8710103f-40a0-39ad-871a-f1ea84691f28",
    "scope": "am_application_scope default",
    "expires_in": 3600, //seconds
    "token_type": "Bearer",
}
```

- `provided_service_key` and `provided_service_name` are fixed values, shared by the operator.