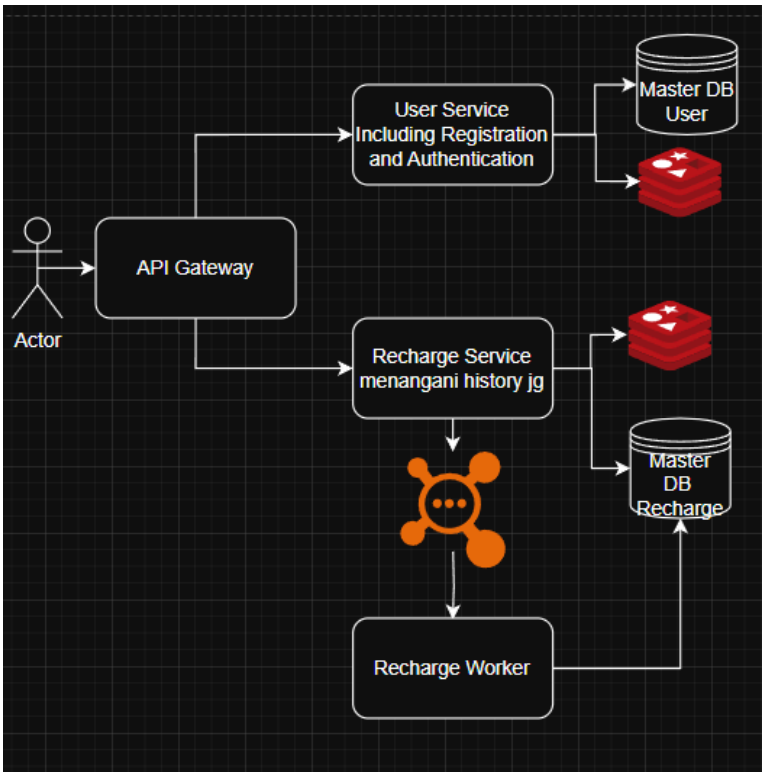
Implementation scope

- Authorization will use copy and paste code to shorten development time instead of creating a library for dependency.
- Login will use username & password instead
- There are no password salting
- 3rd party service is never called. Mock is only have the class that should contains the http request to the 3rd party
- There are no load balancers for POC thus there are only one instance per service too.
- SMS service is omitted
- There are no OTP to shorten development time
- DB will not have master slave structure
- KMS is not used, private key is copy pasted
- DLQ will not be used.
- Not doing sanitizing
- Webhook need manual trigger by user.
- Entity will be passed as result return to shorten delivery time.
- Will use this document as documentation. Swagger will not be used.

POC Final design



API Gateway

Api gateway can:

- route the request to user service or recharge service based on the path.
- First JWT validation

User Service

User service can:

- Register user
- Authenticate
- Get new access token from refresh token

Recharge Service

Recharge service can:

- Send recharge request
- Get recharge status
- Get recharge history
- Updating recharge status from webhook

Recharge Worker

Recharge service can:

- Process recharge request

API Endpoint

URL :/users/register
Method :POST
Header :-
Body Example:

```
{
   "username": "user2",
   "password": "1",
   "name": "user2"
}
```

Note : There are no way to update user right now.

URL :/users/authenticate
Method :POST
Header :-
Body Example:

```
{
   "username": "user2",
   "password": "1"
```

}
Note : There are refresh token in httponly cookies for refresh later

URL :/users/refresh-token
Method :POST
Header :-
Body Example :-
Note : make sure httponly cookies is included.

URL :/recharge/request
Method :POST
Header :Authorization bearer token
Body Example :

```
{
  "operatorType": "TELKOMSEL",
  "amount": 10000
}
```

Note : Supported operator type is TELKOMSEL, TRI, XL

URL :/recharge/request
Method :POST
Header :Authorization bearer token
Body Example :

```
{
  "operatorType": "TELKOMSEL",
  "amount": 10000
}
```

Note : make sure httponly cookies is included.

URL :recharge/recharge-id/{recharge-id}
Method :GET
Header :Authorization bearer token
Body Example :-
Note :-

URL :/recharge/histories/me?page={page-number}&size={item-in-each-page}
Method :GET
Header :Authorization bearer token
Body Example :-
Note :-

URL :/recharge-webhook/update
Method :POST
Header :x-third-party-key = tLrEudbKq+GFl5TNgGRg03GEF6jX04vXTqsvAcZ5/5B=
Body Example :

```
{
  "id": "TELKOMSEL-1",
```

```
    "status": "SUCCESS"
}
```


Note :

- Supported status is SUCCESS and FAILED
- Id can be seen using recharge/recharge-id/{recharge-id} endpoint after the queue successfully updating the third party id.

How to set up

There are docker compose and image that we can use. If there are docker then simply do
> docker compose up --build -d
Docker compose not exposing any container port except for the api gateway port. Adjust this as needed.

If there are no docker then make sure there are

- JAVA 17 JRE
- Rabbit MQ 3.12.1
- Redis 7.2
- Build and run each project. For needed environment variable can be checked inside the docker compose file.