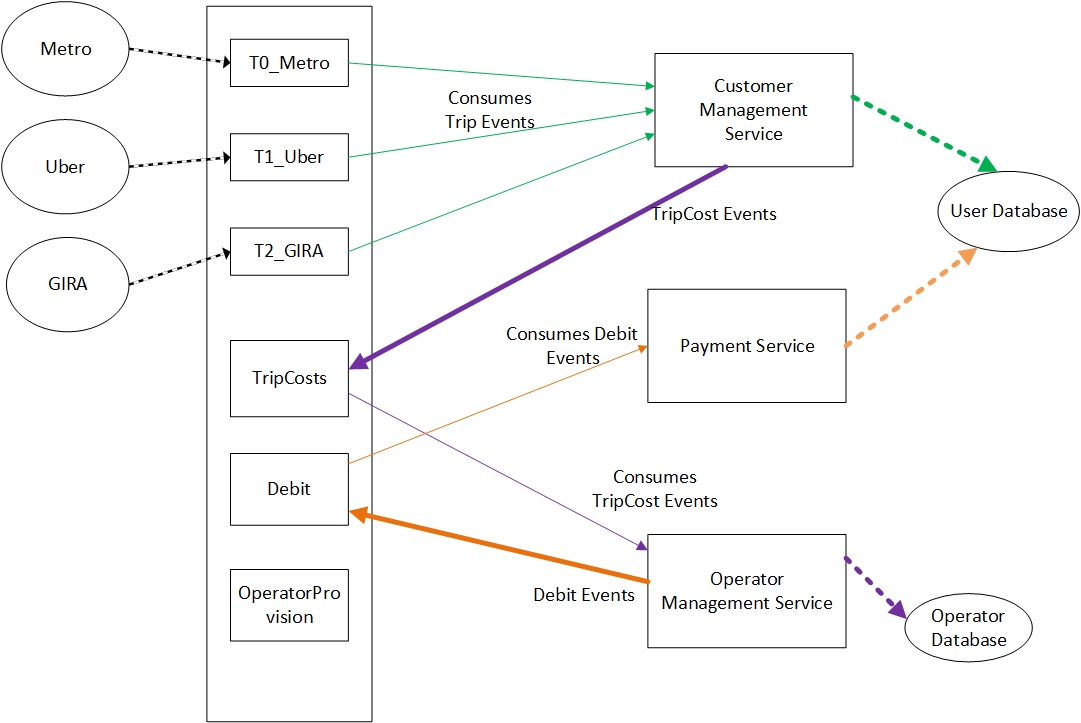
**Enterprise Integration (MEIC-A, 2019-20, 2º semestre)**

Instituto Superior Técnico – MEIC-A

*Sprint 4 Report*

1. **Event flows diagrams**

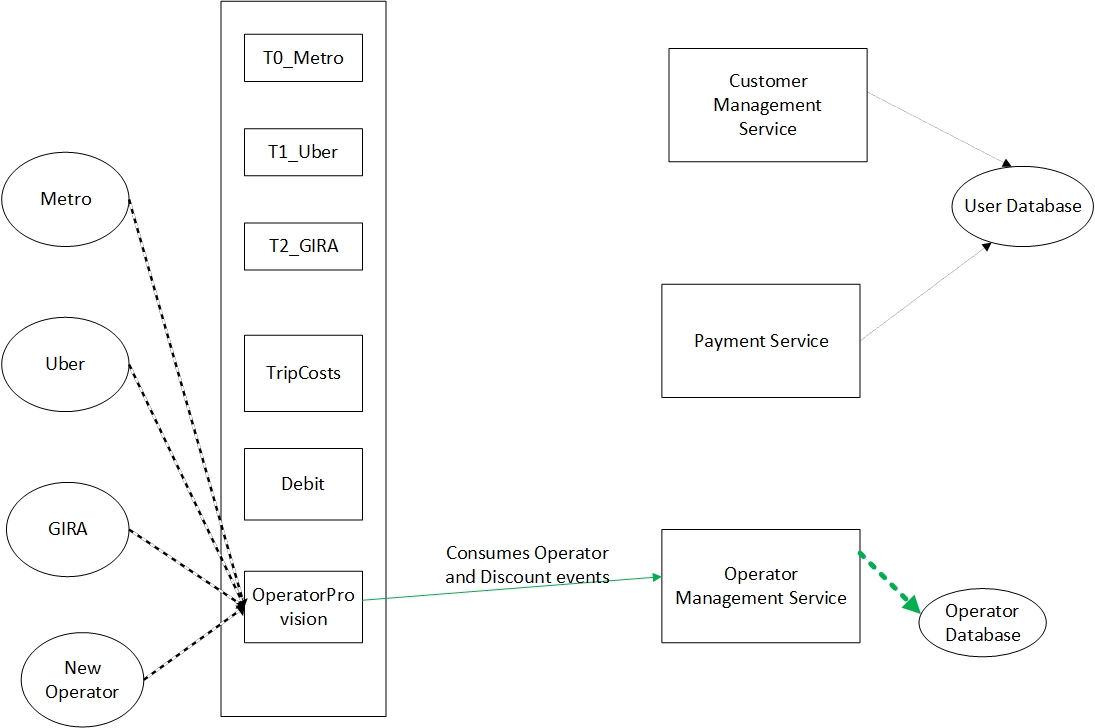
Fig. 1 – Taxation Event flows diagram

Fig. 2 – Operator and Discount Event flows diagram

1. **Kafka configuration**

* **3 brokers were created**

cp /usr/local/kafka/config/server.properties /usr/local/kafka/config/server-1.properties

cp /usr/local/kafka/config/server.properties /usr/local/kafka/config/server-2.properties

cp /usr/local/kafka/config/server.properties /usr/local/kafka/config/server-3.properties

sudo nano /usr/local/kafka/config/server-1.properties

sudo nano /usr/local/kafka/config/server-2.properties

sudo nano /usr/local/kafka/config/server-3.properties

|  |  |  |
| --- | --- | --- |
| Broker-1:  config/server-1.properties:  **broker.id=0**  **listeners=PLAINTEXT://*<Public DNS>*:9092**  **offsets.topic.replication.factor=3**  **transaction.state.log.replication.factor=3**  **transaction.state.log.min.isr=2**  **log.dir=/tmp/kafka-logs-0**  **log.retention.hours = 48**  **zookeeper.connect=localhost:2181** | Broker-2:  config/server-2.properties:  **broker.id=1**  **listeners=PLAINTEXT://*<Public DNS>*:9093**  **offsets.topic.replication.factor=3**  **transaction.state.log.replication.factor=3**  **transaction.state.log.min.isr=2**  **log.dir=/tmp/kafka-logs-1**  **log.retention.hours = 48**  **zookeeper.connect=localhost:2181** | Broker-3:  config/server-3.properties:  **broker.id=2**  **listeners=PLAINTEXT://*<Public DNS>*:9094**  **offsets.topic.replication.factor=3**  **transaction.state.log.replication.factor=3**  **transaction.state.log.min.isr=2**  **log.dir=/tmp/kafka-logs-2**  **log.retention.hours = 48**  **zookeeper.connect=localhost:2181** |

**(The parameters not mentioned here remained with the default values)**

**broker.id –** broker id

**listeners –** the address the broker socket listens on

**offsets.topic.replication.factor -** specify the replication factor for the \_\_consumer\_offsets topic. This topic stores information about committed offsets for each topic:partition per group of consumers

(We’ve set this value to 3 to take advantage of having 3 brokers, providing more redundancy for this information)

**transaction.state.log.replication.factor -** the replication factor for the transaction topic. Internal topic creation will fail until the cluster size meets this replication factor requirement.

(We’ve set this value to 3 to take advantage of having 3 brokers, providing more redundancy for this information)

**transaction.state.log.min.isr -** minimum ISR for this topic

(All the topics will have at least the leader and one replica in sync to continue to provide service)

**log.dir** **–** the directory in which the log data is kept

**log.retention.hours** **-** the number of hours to keep a log file before deleting it

(As mentioned before we’ve set it to 48, the messages will be kept for 48 hours before they are deleted)

**zookeeper.connect -** ZooKeeper connection string

(Contains the addresses for the zookeeper nodes)

* **Opened the in-bound ports 9092, 9093 and 9094 in the AWS EC2 console for the Kafka brokers.**

1. **Zookeeper configuration**

* **I created new directory with the command**

sudo mkdir -p /var/lib/zookeeper

* **I created the baseline zookeeper server configuration with the command**

cat > /usr/local/zookeeper/conf/zoo.cfg << EOF

* **And then wrote the following content to the file directly in the command line**

tickTime=2000

dataDir=/var/lib/zookeeper

clientPort=2181

EOF

* **I opened the in-bound ports 2181 in the AWS EC2 console for the zookeeper nodes**

1. **Topics configuration**

sudo /usr/local/kafka/bin/kafka-topics.sh --create --zookeeper localhost:2181 -replication-factor 3 --partitions 3 --topic T0\_METRO

sudo /usr/local/kafka/bin/kafka-topics.sh --create --zookeeper localhost:2181 -replication-factor 3 --partitions 3 -–topic T1\_UBER

sudo /usr/local/kafka/bin/kafka-topics.sh --create --zookeeper localhost:2181 -replication-factor 3 --partitions 3 –-topic T2\_GIRA

sudo /usr/local/kafka/bin/kafka-topics.sh --create --zookeeper localhost:2181 -replication-factor 3 --partitions 3 --topic TripCosts

sudo /usr/local/kafka/bin/kafka-topics.sh --create --zookeeper localhost:2181 -replication-factor 3 --partitions 3 --topic Debit

sudo /usr/local/kafka/bin/kafka-topics.sh --create --zookeeper localhost:2181 -replication-factor 3 --partitions 3 --topic OperatorProvision

1. **Database configuration**

UserDB Database configurations:

* + Engine Type: MySQL
  + Version: 5.7.22
  + Template: Free Tier
  + DB instance identifier: userdb
  + DB instance size: db.t2.micro
  + Storage type: General Purpose (SSD)
  + Allocated storage: 20 GiB
  + Enable storage autoscaling: true
  + Maximum storage threshold: 1000 GiB
  + Virtual Private Cloud (VPC): Default VPC
  + Subnet group: default-vpc-8af6c4f0
  + Publicly accessible: Yes
  + VPC Security Groups: default and launch-kafka
  + Availability zone: No preference
  + Database port: 3306
  + Database authentication options: Password Authentication

OperatorDB Database configurations:

* + Engine Type: MySQL
  + Version: 5.7.22
  + Template: Free Tier
  + DB instance identifier: operatorDB
  + DB instance size: db.t2.micro
  + Storage type: General Purpose (SSD)
  + Allocated storage: 20 GiB
  + Enable storage autoscaling: true
  + Maximum storage threshold: 1000 GiB
  + Virtual Private Cloud (VPC): Default VPC
  + Subnet group: default-vpc-8af6c4f0
  + Publicly accessible: Yes
  + VPC Security Groups: default and launch-kafka
  + Availability zone: No preference
  + Database port: 3306
  + Database authentication options: Password Authentication

I also added an inbound rule in the security group on port 3306 for the database connections

1. **User database creation script**

DROP DATABASE IF EXISTS userdb;

CREATE DATABASE IF NOT EXISTS userdb;

USE userdb;

DROP TABLE IF EXISTS userInfo;

CREATE TABLE userInfo

(

    token VARCHAR(100) NOT NULL,

    nif VARCHAR(9) UNIQUE NOT NULL,

    email VARCHAR(50) NOT NULL,

    firstName VARCHAR(20) NOT NULL,

    lastName VARCHAR(20) NOT NULL,

    planType VARCHAR(20) NOT NULL,

    address VARCHAR(100) NOT NULL,

    CONSTRAINT pk\_userInfo PRIMARY KEY (token)

);

DROP TABLE IF EXISTS userBalance;

CREATE TABLE userBalance

(

    token VARCHAR(100) NOT NULL,

    balance INT NOT NULL,

    blackListed BOOLEAN NOT NULL,

    CONSTRAINT pk\_userBalance PRIMARY KEY (token),

    CONSTRAINT fk\_userInfo\_userBalance FOREIGN KEY (token) REFERENCES userInfo(token) on DELETE CASCADE

);

DROP TABLE IF EXISTS history;

CREATE TABLE history

(

    tripID VARCHAR(100) NOT NULL,

    token VARCHAR(100) NOT NULL,

    operatorName VARCHAR(30) NOT NULL,

    time\_stamp DATETIME NOT NULL,

    CONSTRAINT pk\_history PRIMARY KEY (tripID, time\_stamp),

    CONSTRAINT fk\_userInfo\_history FOREIGN KEY (token) REFERENCES userInfo(token) on DELETE CASCADE

);

DROP TABLE IF EXISTS T0\_History;

CREATE TABLE T0\_History

(

    tripID VARCHAR(100) NOT NULL,

    time\_stamp DATETIME NOT NULL,

    station VARCHAR(15) NOT NULL,

    isCheckIn BOOLEAN NOT NULL,

    CONSTRAINT pk\_historyt0 PRIMARY KEY (tripID, time\_stamp),

    CONSTRAINT fk\_historyt0 FOREIGN KEY (tripID, time\_stamp) REFERENCES history(tripID, time\_stamp) on DELETE CASCADE

);

DROP TABLE IF EXISTS T1\_History;

CREATE TABLE T1\_History

(

    tripID VARCHAR(100) NOT NULL,

    time\_stamp DATETIME NOT NULL,

    price DECIMAL (4, 2) NOT NULL,

    CONSTRAINT pk\_historyt1 PRIMARY KEY (tripID, time\_stamp),

    CONSTRAINT fk\_historyt1 FOREIGN KEY (tripID, time\_stamp) REFERENCES history(tripID, time\_stamp) on DELETE CASCADE

);

DROP TABLE IF EXISTS T2\_History;

CREATE TABLE T2\_History

(

    tripID VARCHAR(100) NOT NULL,

    time\_stamp DATETIME NOT NULL,

    time BIGINT NOT NULL,

    price DECIMAL(4, 2) NOT NULL,

    CONSTRAINT pk\_historyt2 PRIMARY KEY (tripID, time\_stamp),

    CONSTRAINT fk\_historyt2 FOREIGN KEY (tripID, time\_stamp) REFERENCES history(tripID, time\_stamp) on DELETE CASCADE

);

1. **Operator database creation script**
2. **Kong configurations**

**--CREATES USER REGISTRATION SERVICE**

curl -i -X POST --url http://localhost:8001/services/ --data 'name=UserRegistration' --data 'url=https://7zp5uskhi8.execute-api.us-east-1.amazonaws.com/default/UserRegistration'

**--ADDS ROUTE FOR NEW USER CREATION**

curl -i -X POST --url http://localhost:8001/services/UserRegistration/routes --data 'hosts[]=new-user.com'

**--CREATES USER UNIQUE ID VALIDATION SERVICE**

curl -i -X POST --url http://localhost:8001/services/ --data 'name=UniqueIDValidation' --data 'url=https://8gyz42fgd6.execute-api.us-east-1.amazonaws.com/default/UniqueIDValidation'

**--ADDS ROUTE FOR UNIQUE ID VALIDATION**

curl -i -X POST --url http://localhost:8001/services/UniqueIDValidation/routes --data 'hosts[]=unique-id.com'

**--CREATES LOAD ACCOUNT SERVICE**

curl -i -X POST --url http://localhost:8001/services/ --data 'name=LoadAccountService' --data 'url=https://zhx0o69m0i.execute-api.us-east-1.amazonaws.com/default/LoadAccountService'

**--ADDS ROUTE FOR ACCOUNT LOADING**

curl -i -X POST --url http://localhost:8001/services/LoadAccountService/routes --data 'hosts[]=load-account.com'

**--CREATES BLACKLIST USER SERVICE**

curl -i -X POST --url http://localhost:8001/services/ --data 'name=BlacklistUserService' --data 'url=https://z62m3l4rh2.execute-api.us-east-1.amazonaws.com/default/BlackListUser'

**--ADDS ROUTE FOR USER BLACKLISTING**

curl -i -X POST --url http://localhost:8001/services/BlacklistUserService/routes --data 'hosts[]=blacklist-user.com'

**--CREATES USER REMOVAL SERVICE**

curl -i -X POST --url http://localhost:8001/services/ --data 'name=UserRemovalService' --data 'url=https://hbjv9al4p4.execute-api.us-east 1.amazonaws.com/default/UserRemoval'

**--ADDS ROUTE FOR USER REMOVAL**

curl -i -X POST --url http://localhost:8001/services/UserRemovalService/routes --data 'hosts[]=remove-user.com'

**--CREATES CHECK USER SERVICE**

curl -i -X POST --url http://localhost:8001/services/ --data 'name=CheckUserService' --data 'url=https://qokp7xifaf.execute-api.us-east-1.amazonaws.com/default/CheckUser'

**--ADDS ROUTE FOR USER CHECKING**

curl -i -X POST --url http://localhost:8001/services/CheckUserService/routes --data 'hosts[]=check-user.com'

**--CREATES USER EMAIL SERVICE**

curl -i -X POST --url http://localhost:8001/services/ --data 'name=GetUserEmailService' --data 'url=https://xgtr8btwm8.execute-api.us-east-1.amazonaws.com/default/GetUserEmail'

**--ADDS ROUTE FOR USER EMAIL RETRIEVAL**

curl -i -X POST --url http://localhost:8001/services/GetUserEmailService/routes --data 'hosts[]=get-email.com'

**--CREATES OPERATOR CREATION SERVICE**

curl -i -X POST --url http://localhost:8001/services/ --data 'name=OperatorCreationService' --data 'url=https://ysuo5wgpg1.execute-api.us-eaault/OperatorCreation'

**--ADDS ROUTE FOR OPERATOR CREATION**

curl -i -X POST --url http://localhost:8001/services/OperatorCreationService/routes --data 'hosts[]=create-operator.com'

**--CREATES DISCOUNT CREATION SERVICE**

curl -i -X POST --url http://localhost:8001/services/ --data 'name=DiscountCreationService' --data 'url=https://5jgqj5i8z3.execute-api.us-eaault/DiscountCreation'

**--ADDS ROUTE FOR DISCOUNT CREATION**

curl -i -X POST --url http://localhost:8001/services/DiscountCreationService/routes --data 'hosts[]=create-discount.com'

1. **Executable processes**

**Costumer Provision process**:

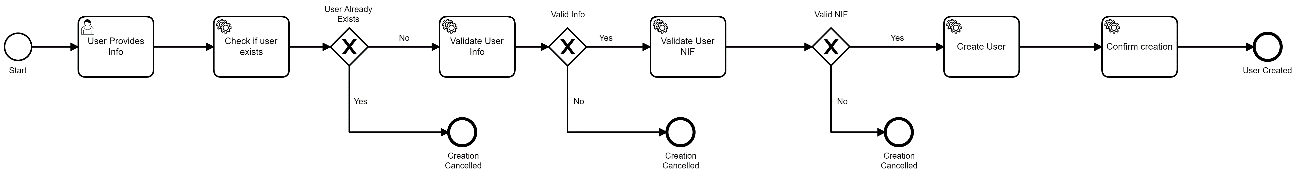


Fig. 3 – User Registration Process

**Dunning Process:**

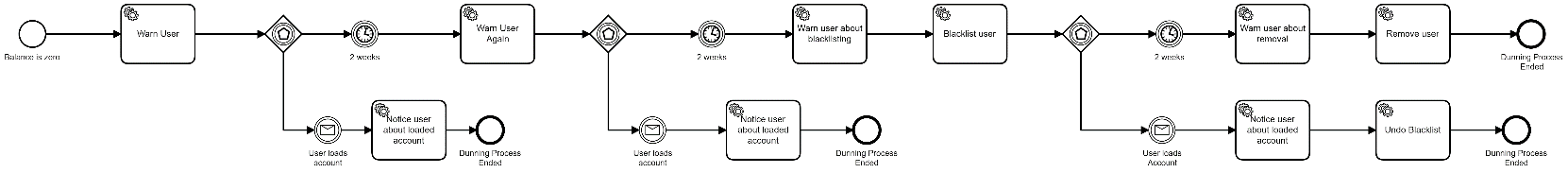
****

Fig. 4 – Dunning Process

**Account Loading Process:**

Fig. 5 – Account Loading Process