

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

### **Instituto Superior Técnico – MEIC-A**

#### *Sprint 4 Report*

In this Sprint I had to create the Payment Service to debit the prepaid amount from the user account and to trigger the dunning process whenever any client exhausts the plafond to meet the integration between rating and dunning process requirement. In addition, I created the operator provision workflows to meet the providers provision requirements. The improvement that I made was to create a Load Account workflow that loads the user account and stops the dunning process when the user balance becomes bigger than 0 again.

#### **1. Definition of the mobility operators and respective messages**

Mobility as a Service is an analogy of the usual Software as a Service model popularized by the Cloud for the software industry. The idea is the same, people will use the transport network they see fitting better their needs, be it the Public Transport Operators like Metro or Buses, Taxis or the new players such as Uber or Cabify and all other innovative alternatives for personal transportation like rental bikes, scooters, motorcycles, etc.

The innovation is the seamless use of all of them without the usual difficulties of different ticketing and payments systems, and the negative incentive for such use due to incompatibilities between cards, apps, tariffs, monthly subscriptions, etc.

The main idea of Maas is that one can take any transportation system and in the background his usage is being registered and one will pay for mobility according to the schema that best suits his needs.

As part of the proposal, I aim to integrate all the different transportation operators that provide services in Lisbon. For that purpose, I have defined the architecture of the Maas Operator based on an analysis I made of the types of services that are provided by each operator. I can generalize these services by gathering all of them in three types:

- **Type 0:** Check in and check out method
  - This method is used by Metropolitano de Lisboa and CP, for example
  - The user must have a ticket or a pass that is validated when he enters and when he leaves the transport and the price is fixed

Types of messages:

```
{
  "event": {
    "eventType": "t0-check-in",
    "operator": "Metro",
    "info": {
      "Id": "1",
      "Station": "Odivelas",
      "Timestamp": "2020-02-29 18:23:41.278"
    }
  }
}
```

```
}  
}  
  
{  
  "event": {  
    "eventType": "t0-check-out",  
    "operator": "Metro",  
    "info": {  
      "Id": "1",  
      "Station": "Alameda",  
      "Timestamp": "2020-02-29 18:23:47.718"  
    }  
  }  
}
```

- **Type 1:** Distance and time dependent method
  - This method is used by Uber and Cabify, for example
  - The price of the trip is mainly calculated using the distance and time of the trip

Types of messages:

```
{  
  "event": {  
    "eventType": "t1",  
    "operator": "Uber",  
    "info": {  
      "Id": "1",  
      "Price": "20.63",  
      "Timestamp": "2020-02-29 19:45:58.638"  
    }  
  }  
}
```

- **Type 2:** Time dependent method
  - This method is used by GIRA and Lime, for example
  - The price of the trip depends on how much time do you spend using the transport

Types of messages:

```
{  
  "event": {  
    "eventType": "t2",  
    "operator": "Gira",  
    "info": {  
      "Id": "1",  
      "Time": "3600",  
      "Price": "12.60",  
      "Timestamp": "2020-02-29 20: 57: 10.294"  
    }  
  }  
}
```

To have a representative example of how the system would work in the “real world”, I chose one operator of each type to make sure that the system would cope with every type of public transportation operating in Lisbon. I chose **Metro, Uber and GIRA**.

## 2. Definition of the event queueing integration: Topics, Partitions

In my system I have defined that every operator has a topic because of two main reasons:

- One topic for company enforces the event load to be more distributed, one topic for all operators would have too much load
- It helps identifying the operator that sent the message

In addition, I have identified the type of service in the name of the topic: **T0\_Metro, T1\_Uber** and **T2\_GIRA**. This helps the consumer services to know the type of processing that is needed for the event simply by reading the name of the topic.

The **TripCosts** topic will be used for the TripCost events, the Customer Management Service will produce this type of events to this topic for them to be consumed by the Operator Management Service.

The **Debit** topic will be used for the Debit events, the Operator Management Service will produce this type of events to this topic for them to be consumed by the Payment Service.

To provide some parallelism to my system I have created all the topics with 3 partitions and a replication factor of 3.

## 3. Event flow diagram

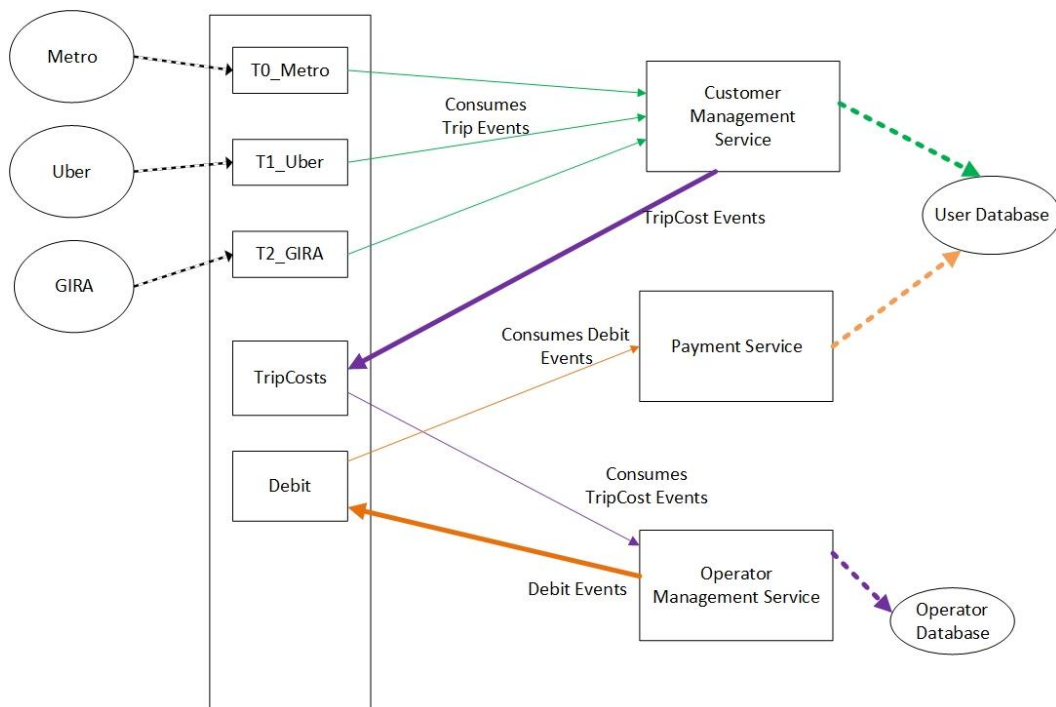


Fig. 1 – Taxation Event flows diagram

#### 4. Microservices input and output

##### Customer Management Service:

##### Input

This service takes as input Trip events that are consumed from the operator topics.

- **Trip Events**

There are 3 types of trip events:

- **Type 0 operators** (Check in and check out method)

These events contain as main fields:

- **eventType** to say if it is a check-in or a check-out
- **operator** that contains the name of the operator
- **info** that contains information about a trip. Inside this field there are fields with the id of the customer, station, and a timestamp.

Examples:

Check-in in metro

##### JSON

```
{
  "event": {
    "eventType": "t0-check-in",
    "operator": "Metro",
    "info": {
      "Id": "1",
      "Station": "Odivelas",
      "Timestamp": "2020-02-29 18:23:41.278"
    }
  }
}
```

##### XML

```
<event>
  <eventType>t0-check-in</eventType>
  <info>
    <Station>Odivelas</Station>
    <Timestamp>2020-02-29 18:23:41.278</Timestamp>
    <Id>1</Id>
  </info>
  <operator>Metro</operator>
</event>
```

Check-out in metro

#### JSON

```
{
  "event": {
    "eventType": "t0-check-out",
    "operator": "Metro",
    "info": {
      "Id": "1",
      "Station": "Alameda",
      "Timestamp": "2020-02-29 18:23:47.718"
    }
  }
}
```

#### XML

```
<event>
  <eventType>t0-check-out</eventType>
  <info>
    <Station>Alameda</Station>
    <Timestamp>2020-02-29 18:23:47.718</Timestamp>
    <Id>1</Id>
  </info>
  <operator>Metro</operator>
</event>
```

- **Type 1 operators** (Distance and time dependent method)

These events contain as main fields:

- **eventType** that says "t1"
- **operator** that contains the name of the operator
- **info** that contains information about a trip. Inside this there are fields with the id of the customer, price, and a timestamp.

Example:

#### JSON

```
{
  "event": {
    "eventType": "t1",
    "operator": "Uber",
    "info": {
      "Id": "1",
      "Price": "20.63",
      "Timestamp": "2020-02-29 19:45:58.638"
    }
  }
}
```

## XML

```
<event>
  <eventType>t1</eventType>
  <info>
    <Price>20.63</Price>
    <Timestamp>2020-02-29 19:45:58.638</Timestamp>
    <Id>1</ Id >
  </info>
  <operator>Uber</operator>
</event>
```

- **Type 2 operators** (Time dependent method)

These events contain as main fields:

- **eventType** that says "t2"
- **operator** that contains the name of the operator
- **info** that contains information about a trip. Inside this field there are fields with the id of the customer, time spent with the vehicle, price of the ride and a timestamp.

Example:

## JSON

```
{
  "event": {
    "eventType": "t2",
    "operator": "Gira",
    "info": {
      "Id": "1",
      "Time": "3600",
      "Price": "12.60",
      "Timestamp": "2020-02-29 20: 57: 10.294"
    }
  }
}
```

## XML

```
<event>
  <eventType>t2</eventType>
  <info>
    <Price>12.60</Price>
    <Time>3600</Time>
    <Timestamp>2020-02-29 20: 57: 10.294</Timestamp>
    <Id>1</Id>
  </info>
  <operator>Gira</operator>
</event>
```

## Output

This service has as output a TripCost Event. This event has the goal provide information to the Operator Management Service to compute how much money should be debited from the user account.

- **TripCost Events**

These events contain as main fields:

- **eventType** that says “trip-cost”
- **info** that contains information about a trip and the user that made that trip. The info field contains: cost of the trip (null if operator type is t0), id of the user, plan type, operator name and timestamp.

Example:

### JSON

```
{
  "event": {
    "eventType": "trip-cost",
    "info": {
      "cost": "23",
      "id": "69c594cfdeeaedd220",
      "planType": "pre-paid",
      "operatorName": "Uber",
      "timeStamp": "2020-02-29 20:57:10.294"
    }
  }
}
```

### XML

```
<event>
  <eventType>trip-cost</eventType>
  <info>
    <cost>23</cost>
    <operatorName>Uber</operatorName>
    <planType>pre-paid</planType>
    <timeStamp>2020-02-29 20:57:10.294</timeStamp>
    <id>1</id>
  </info>
</event>
```

### Operator Management Service:

This service has as input TripCost Events. This event has the goal provide information to the Operator Management Service to compute how much money should be debited from the user account.

#### **Input**

- **TripCost Events**

(Described before)

#### **Output**

This service has as output a Debit Event. This event has the goal to provide information to the Payment Service to debit money from the user account.

- **Debit Events**

These events contain as main fields:

- **eventType** that says “debit”
- **info** that contains information about the debit. The info field contains information about the amount to debit, plan type and the id of the user.

Example:

#### **JSON**

```
{
  "event": {
    "eventType": "debit",
    "info": {
      "id": "1",
      "planType": "pre-paid",
      "amount": "20"
    }
  }
}
```

#### **XML**

```
<event>
  <eventType>debit</eventType>
  <info>
    <amount>20</amount>
    <planType>pre-paid</planType>
    <id>1</id>
  </info>
</event>
```



#### Payment Service:

This service has as input Debit Events. This event has the goal to debit the amount in the event from the user account considering its plan type.

#### **Input**

- **Debit Events**

(Described before)

The microservices described before are just used for the taxation event flow handling. I had to create other microservices as Lambda functions for the executable business processes:

- Create Discount Service
- Create Operator Service
- Blacklist User Service
- Load Account Service
- Check User Service (checks if user exists in the system)
- Get Email Service
- Remove User Service
- Unique Id Service (checks if the user already exists in the system)
- User Registration Service

(The inputs and outputs of these microservices are present in the Swagger files folder and the code is in the Microservices folder)

### **5. Definition of the fault tolerance requirements for Kafka**

In order to provide **high availability** in my system and as every topic has 3 partitions (number of partitions has to be less or equal to the number of brokers), I've created **3 brokers** in the Maas operator, added a **replication factor** of 3 to every topic and configured a **minimum of 2 ISR**. This way I can have a **leader of each topic in every broker** and **I guarantee that there will be another ISR even if the leader fails**, providing a normal service to the users even if two of the brokers fail.

The Kafka cluster durably persists all published records using a configurable retention period. I have set the **retention period** for 48 hours, so for the two days after the record is published, it is available for consumption, after which it will be discarded to free up space. I think that 48 hours more than enough for the records to be consumed.

### **6. 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
```

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

(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

(I'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.

(I'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.**

## 7. 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

## 8. 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
```

## 9. 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

## 10. 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 user
Info(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) REFERENCE
S 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) REFERENCE
S 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) REFERENCE
S history(tripID, time_stamp) on DELETE CASCADE
);
```

## 11. Operator database creation script

```
DROP DATABASE IF EXISTS operatordb;
CREATE DATABASE IF NOT EXISTS operatordb;
```

```
USE operatordb;
```

```
CREATE TABLE operator(
    operatorName VARCHAR(100) NOT NULL,
    operatorType VARCHAR(2) NOT NULL,
    price DECIMAL (4, 2),
    CONSTRAINT pk_operator PRIMARY KEY (operatorName)
```

```
);

CREATE TABLE discount(
    discountId VARCHAR(100) NOT NULL UNIQUE,
    discountName VARCHAR(100) NOT NULL UNIQUE,
    value INT NOT NULL,
    beginAt DATETIME NOT NULL,
    endAt DATETIME NOT NULL,
    CONSTRAINT pk_discount PRIMARY KEY (discountId)
);

CREATE TABLE planType(
    plan VARCHAR(20) NOT NULL,
    CONSTRAINT pk_planType PRIMARY KEY (plan)
);

CREATE TABLE discount_planType
(
    discountId VARCHAR(100) NOT NULL,
    plan VARCHAR(20) NOT NULL,
    CONSTRAINT pk_discount_planType PRIMARY KEY (discountId,plan),
    CONSTRAINT fk_planType FOREIGN KEY (plan) REFERENCES planType(plan)
) on DELETE CASCADE
);

CREATE TABLE operator_discount(
    operatorName VARCHAR(100) NOT NULL,
    discountId VARCHAR(100) NOT NULL,
    CONSTRAINT pk_operator_discount PRIMARY KEY (operatorName,discount
Id),
    CONSTRAINT fk_operator FOREIGN KEY (operatorName) REFERENCES opera
tor(operatorName) on DELETE CASCADE,
    CONSTRAINT fk_discount FOREIGN KEY (discountId) REFERENCES discoun
t(discountId) on DELETE CASCADE
);
```

## 12. 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'
```

### 13. Executable processes

#### Costumer Provision process:

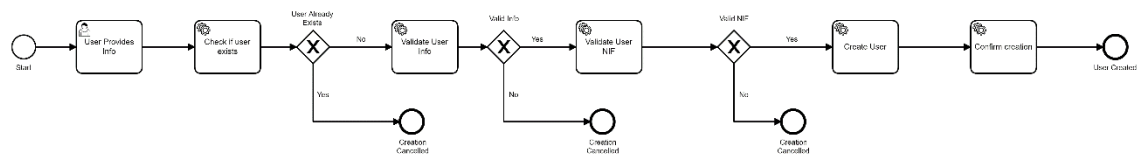


Fig. 2 – User Registration Process



### Dunning Process:

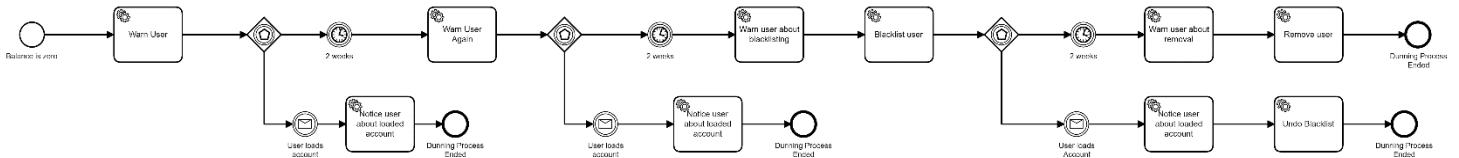


Fig. 3 – Dunning Process

### Account Loading Process:

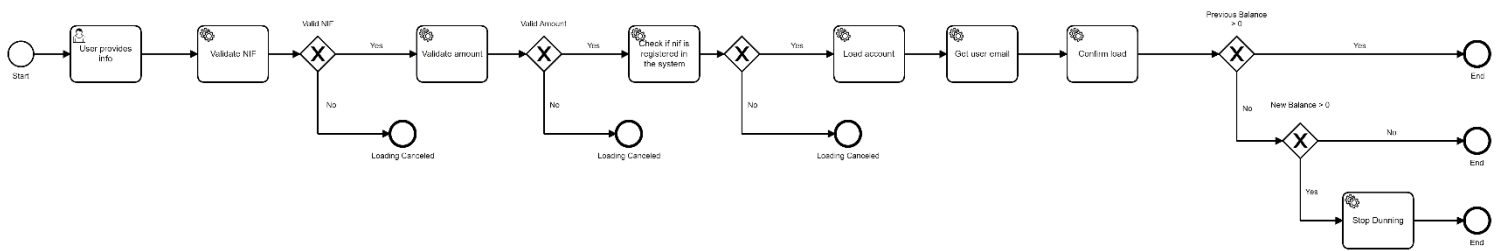


Fig. 4 – Account Loading Process

### Operator Creation Process:

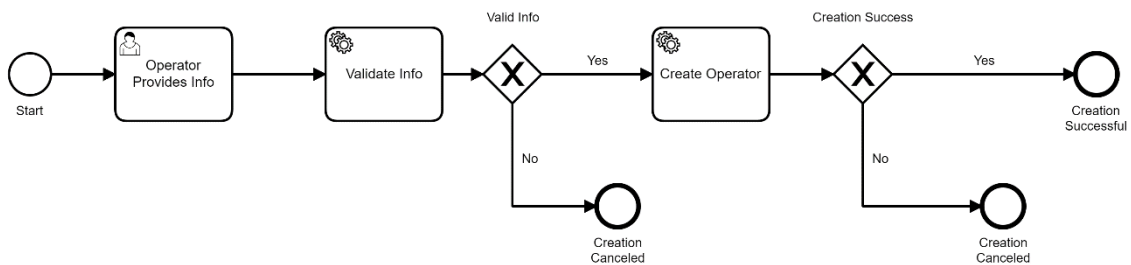


Fig. 5 – Operator Creation Process

### Discount Creation Process:

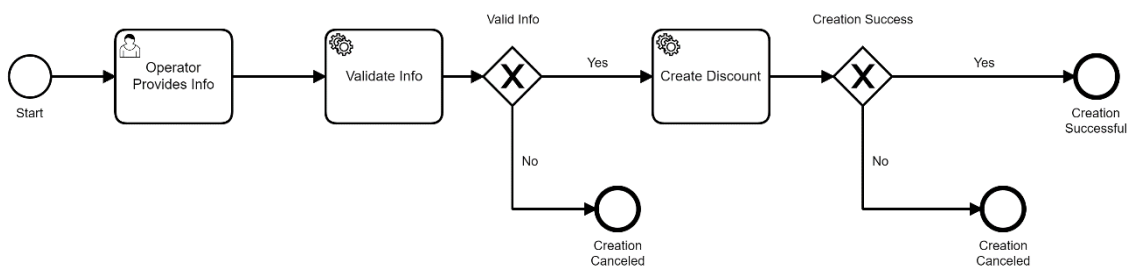


Fig. 6 – Discount Creation Process

## 14. Functional Testing

### Trip Event Handling

Database Initial State:

	id	balance	blackListed
▶	1	116.56	0
	2	500.00	0
	3	500.00	0
	4	200.00	0
★	NULL	NULL	NULL

	id	nif	email	firstName	lastName	planType	address
▶	1	215963548	eil@gmail.com	Jose	Mario	pre-paid	Rua 1
	2	215923548	idsl@gmail.com	Manuel	Esteves	post-paid	Rua 2
	3	259433548	kds@gmail.com	Joana	Simões	generalPass	Rua 3
	4	507712439	jhmffreitas@gmail.com	Rui	Pinheiro	pre-paid	Rua 4
★	NULL	NULL	NULL	NULL	NULL	NULL	NULL

1. I started by creating an Uber Producer and produced a trip event for user with id 4

```
PS C:\Users\jhmff\desktop\IST\MEIC\lano\2semestre\IE\Projeto\Sprint2\Producers\Uber\ProducerProviderAbstractWithoutTokenControl\target> java -jar ProducerProvider2-0.0.1-SNAPSHOT.jar --provider-name Uber --broker-list ec2-52-202-49-153.compute-1.amazonaws.com:9092,ec2-52-202-49-153.compute-1.amazonaws.com:9093,ec2-52-202-49-153.compute-1.amazonaws.com:9094 --topic T1_Uber --id-list 4 --throughput 1 --typeMessage JSON
The following arguments are accepted:
--provider-name=Uber
--broker-list=ec2-52-202-49-153.compute-1.amazonaws.com:9092,ec2-52-202-49-153.compute-1.amazonaws.com:9093,ec2-52-202-49-153.compute-1.amazonaws.com:9094
--topic=T1_Uber
--id-list=[4]
--throughput=1
--typeMessage=JSON
----- Processing starting -----
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
This is the message to send = {"event":{"eventType":"t1","operator":"Uber","info":{"Id": "4", "Price": "53.203564", "Timestamp": "2020-05-27 23:06:38.482"} }}
Sending new message to Kafka... with key=1590617198495
Sent...
Waiting...2020-05-27 23:06:41.712
```

2. Then I started the UserManagementService to consume this event and produce a TripCost Event

```
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Subscribing to topic ->T2_GIRA
Subscribing to topic ->T0_Metro
Subscribing to topic ->T1_Uber
handleRequest: Begin!!
handleRequest: topic = T1_Uber, partition = 2, offset = 77, customer = 1590617198495, message = {"event":{"eventType":"t1",
"operator":"Uber", "info":{"Id": "4", "Price": "53.203564", "Timestamp": "2020-05-27 23:06:38.482"} }}
parseEvent(operator):Uber
parseEvent(eventType):t1
parseEvent(info):{"Price":"53.203564","Id":"4","Timestamp":"2020-05-27 23:06:38.482"}
produceTripCostEvent : send event !
produceTripCostEvent : Sent -> {"event":{"eventType":"trip-cost", "info":{"cost": "53.203564", "id": "4", "planType": "pre-paid",
"operatorName": "Uber", "timeStamp": "2020-05-27 23:06:38.482"} }}
END RequestId: 4857321a-f456-4fe8-a9e8-15a0617cc218
REPORT RequestId: 4857321a-f456-4fe8-a9e8-15a0617cc218 Duration: 20004.20 ms Billed Duration: 20000 ms Memory Size: 3008 MB
Max Memory Used: 133 MB Init Duration: 31.53 ms
```

It successfully gathered all the correct information and sent the TripCost event to the OperatorManagementService

### 3. Then I started the OperatorManagementService to consume this event and produce a Debit Event

```
[ec2-user@ip-172-31-38-219 OperatorWebservice]$ mvn exec:java -D exec.mainClass="WebService.OperatorManagementServicePublisher"
[INFO] Scanning for projects...
[INFO] -----< projeto-ie:OperatorManagementService >-----
[INFO] Building OperatorManagementService 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO] --- exec-maven-plugin:1.6.0:java (default-cli) @ OperatorManagementService ---
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
startService: Begin!!

startService: topic = TripCosts, partition = 1, offset = 182, customer = TripCostsKey, message = {"event":{"eventType":"trip-cost", "info":{"cost": "53.203564", "id": "4", "planType": "pre-paid", "operatorName": "Uber", "timeStamp": "2020-05-27 23:06:38.482" }}}

processEvent(eventType):trip-cost

processEvent(info):{"timeStamp":"2020-05-27 23:06:38.482","cost":"53.203564","planType":"pre-paid","id":"4","operatorName":"Uber"}

getDiscount : Start searching for discount

getDiscount : No discount found

BaseCost:53.203564

produceDebitEvent : Sent -> {"event":{"eventType":"debit", "info":{"id": "4", "planType": "pre-paid", "amount": "53.203564" }}}

```

It gathered the correct information and sent the Debit event to the PaymentService

### 4. Finally, I started the PaymentService to consume this event and debit 53.20€ from the user account

```
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
handleRequest: Begin!!
handleRequest: topic = Debit, partition = 0, offset = 181, customer = DebitKey, message = {"event":{"eventType":"debit", "info":{"id": "4", "planType": "pre-paid", "amount": "53.203564" }}}
parseEvent(eventType):debit
parseEvent(info):{"amount":"53.203564","planType":"pre-paid","id":"4"}
Processing debit event
User Balance -> 200.00
Trip Cost -> 53.203564
Debited Account of user: 4 New Account Balance: 146.79643
END RequestId: 6410f139-1bf1-429f-aae4-021f4ef9bcda
REPORT RequestId: 6410f139-1bf1-429f-aae4-021f4ef9bcda Duration: 45030.88 ms Billed Duration: 45000 ms Memory Size: 3008 MB
Max Memory Used: 131 MB Init Duration: 45.10 ms
2020-05-27T22:11:39.432Z 6410f139-1bf1-429f-aae4-021f4ef9bcda Task timed out after 45.03 seconds

```

Database final state:

The amount was debited from the user account:

	id	balance	blackListed
▶	1	116.56	0
	2	500.00	0
	3	500.00	0
	4	146.80	0
•	NULL	NULL	NULL

The user history was also updated:

tripID	time_stamp	price
4-20052701520151	2020-05-27 13:52:02	63.86
4-20052701530151	2020-05-27 13:53:02	25.51
4-20052701540151	2020-05-27 13:54:02	10.42
4-20052701550151	2020-05-27 13:55:02	87.23
4-20052701560151	2020-05-27 13:56:02	54.25
4-20052701570151	2020-05-27 13:57:02	78.76
4-200527030812512	2020-05-27 15:08:12	67.13
4-200527110638538	2020-05-27 23:06:38	53.20

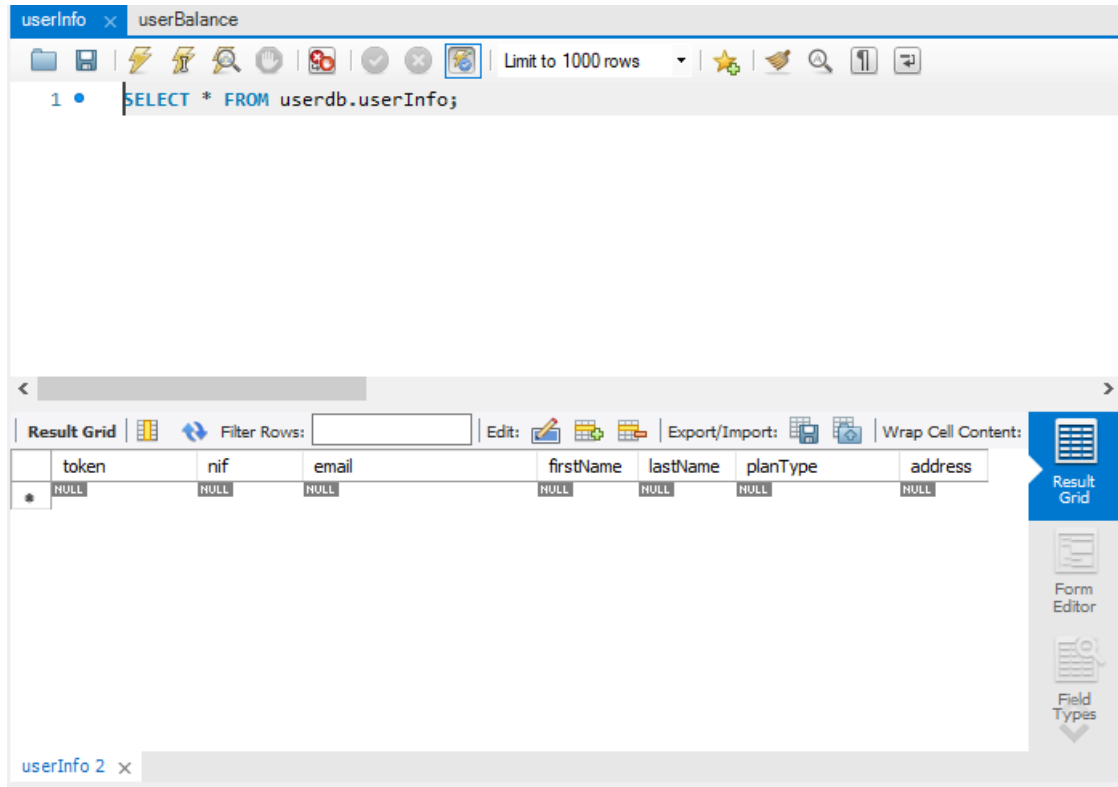
tripID	id	operatorName	time_stamp
4-20052701520151	4	Uber	2020-05-27 13:52:02
4-20052701530151	4	Uber	2020-05-27 13:53:02
4-20052701540151	4	Uber	2020-05-27 13:54:02
4-20052701550151	4	Uber	2020-05-27 13:55:02
4-20052701560151	4	Uber	2020-05-27 13:56:02
4-20052701570151	4	Uber	2020-05-27 13:57:02
4-200527030812512	4	Uber	2020-05-27 15:08:12
4-200527110638538	4	Uber	2020-05-27 23:06:38

## Customer Provision - User Registration Process

- User is created successfully:
1. When I start the process, the following form shows up on Camunda Tasklist, this corresponds to the User Provides Info Task. In this case I will provide correct inputs to all fields to show how this process works when everything goes well

The screenshot shows the Camunda Tasklist interface. On the left, there is a sidebar with 'My Tasks (1)' and a list of task groups: 'My Group Tasks', 'Accounting', 'John's Tasks', 'Mary's Tasks', 'Peter's Tasks', and 'All Tasks'. The main area displays a task titled 'User Provides Info' with a sub-header 'User Registration'. Below this, there are tabs for 'Form', 'History', 'Diagram', and 'Description', with 'Form' being the active tab. The form contains several input fields: 'Token' (id456123753), 'Email' (jhenffreitas@gmail.com), 'First\_Name' (Joao), 'Last\_Name' (Freitas), 'Plan\_Type' (generalPass), 'Balance' (250), 'NIF' (506475069), and 'Address' (Rua Teste 2). At the bottom right of the form, there are 'Save' and 'Complete' buttons. The footer of the interface includes the text 'Data and Time displayed in local timezone: Europe/Lisbon' and 'Powered by camunda BPM / v7.15.0-OSNAPSHOT'.

At the beginning of the process the database tables were empty:



2. The first task in the workflow is to check if the user exists in the database:

This image shows the console output when I ran the validation client

```
mai 15, 2020 3:33:37 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$0
INFO: Unique ID Validation Started
mai 15, 2020 3:33:41 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$0
INFO: Finished with HTTP error code : 200
HTTP/1.1 200 OK [Content-Type: application/json, Content-Length: 18, Connection: keep-alive, Date: Fri, 15 May 2020 02:33:41 GMT,
mai 15, 2020 3:33:41 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$0
INFO: Unique ID:true
```

This image shows part of the request made to the validation service retrieved from the CloudWatch Logs, which confirms the call made to the service simply by analyzing the timestamp and the request body.

```
requestContext": {
  "resourceId": "xox5zr",
  "resourcePath": "/UniqueIDValidation",
  "httpMethod": "POST",
  "extendedRequestId": "MjTcOGZlIAMF-Rw=",
  "requestTime": "15/May/2020:02:33:37 +0000",
  "path": "/default/UniqueIDValidation",
  "accountId": "618315746481",
  "protocol": "HTTP/1.1",
  "stage": "default",
  "domainPrefix": "8gyz42fgd6",
  "requestTimeEpoch": 1589510017397,
  "requestId": "a62be13c-e9b8-4563-9d22-ed7799fd91cc",
  "identity": {
    "cognitoIdentityPoolId": null,
    "accountId": null,
    "cognitoIdentityId": null,
    "caller": null,
    "sourceIp": "54.236.120.160",
    "principalOrgId": null,
    "accessKey": null,
    "cognitoAuthenticationType": null,
    "cognitoAuthenticationProvider": null,
    "userArn": null,
    "userAgent": "Apache-HttpClient/4.5.11 (Java/1.8.0_251)",
    "user": null
  },
  "domainName": "8gyz42fgd6.execute-api.us-east-1.amazonaws.com",
  "apiId": "8gyz42fgd6"
},
"resource": "/UniqueIDValidation",
"httpMethod": "POST",
"queryStringParameters": null,
"stageVariables": null,
"body": "{\"id\":\"id456123753\"}"
```

3. The next task is to validate the userInfo by checking if there are any empty or null fields:

I can see in the console that the validation is made and succeeds

```
mai 15, 2020 3:33:41 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$1
INFO: Validation Started
mai 15, 2020 3:33:41 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$1
INFO: Data is valid
```

4. Next, I have the validation of the NIF task:

In the console I can verify that indeed this validation was successful, thus can be confirmed by checking the last log line. The “success” printed in extracted directly from the webservice response result.

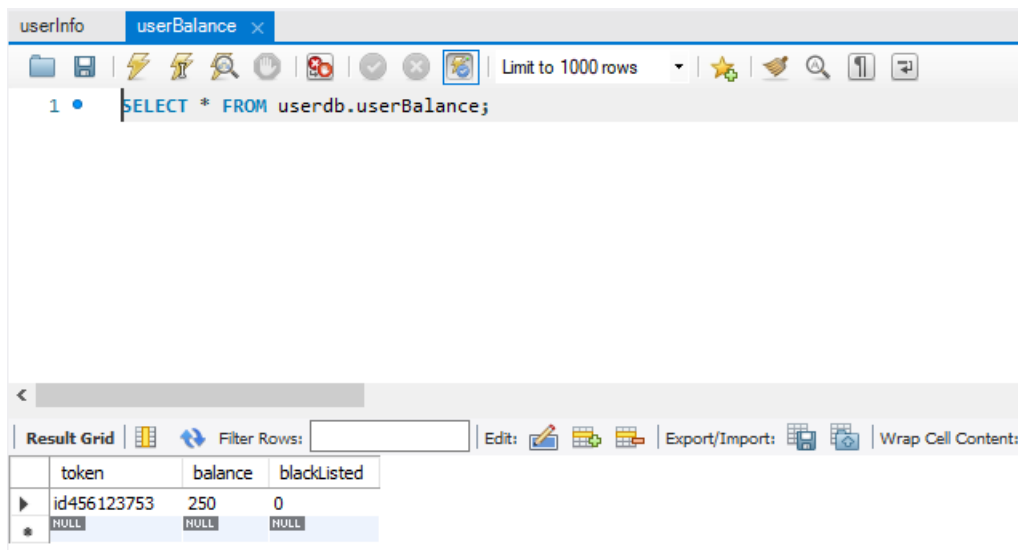
```
INFO: NIF Validation Started
mai 15, 2020 3:33:42 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$2
INFO: Finished with HTTP error code : 200
HTTP/1.1 200 OK [Server: nginx, Date: Fri, 15 May 2020 02:33:42 GMT, Content-Type: application/json;
mai 15, 2020 3:33:42 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$2
INFO: Valid NIF:succes
```

5. Finally, I have the creation of the user task:

In the console I can see that the operation succeeded.

```
mai 15, 2020 3:33:42 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$3
INFO: Creation Started
mai 15, 2020 3:33:46 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$3
INFO: Finished with HTTP error code : 200
HTTP/1.1 200 OK [Content-Type: application/json, Content-Length: 103, Connection: keep-alive, Date: Fri, 15 May 2020 02:33:46 GMT,
mai 15, 2020 3:33:46 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$3
INFO: response body = {"headers":{"x-custom-header":"User Registration"},"body":{"message":"Success"},"statusCode":200}
```





Now I am going to present the failure cases:

- User already exists

I will start the process again and try to insert a user with the same data as the previous one.

As I can see, as the user already existed, the process ended after the first task by following the Yes branch in the “User Already Exists” gateway

```
mai 15, 2020 4:06:21 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$0
INFO: Unique ID Validation Started
mai 15, 2020 4:06:25 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$0
INFO: Finished with HTTP error code : 200
HTTP/1.1 200 OK [Content-Type: application/json, Content-Length: 19, Connection: keep-alive]
mai 15, 2020 4:06:25 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$0
INFO: Unique ID:false
```

- Invalid data is provided

I will start the process again and try to insert a user with different token but with empty fields in the first name and last name for example

As I can see, because of the empty fields, the process ended after the second task by following the No branch in the “Valid Info” gateway

```
mai 15, 2020 4:13:29 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$0
INFO: Unique ID Validation Started
mai 15, 2020 4:13:30 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$0
INFO: Finished with HTTP error code : 200
HTTP/1.1 200 OK [Content-Type: application/json, Content-Length: 18, Connection: keep-alive]
mai 15, 2020 4:13:30 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$0
INFO: Unique ID:true
mai 15, 2020 4:13:30 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$1
INFO: Validation Started
mai 15, 2020 4:13:30 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$1
INFO: Data is invalid
```



- Invalid NIF is provided

I will start the process again and try to create a user with an invalid NIF but the other parameters are valid

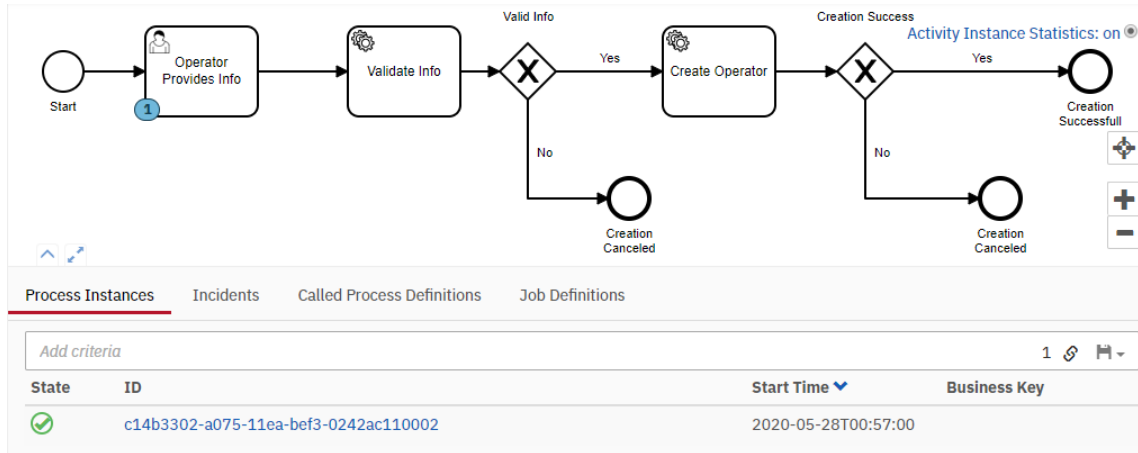
The screenshot shows the Camunda Tasklist web application. On the left, there is a sidebar with a 'My Tasks (1)' section and a list of task groups: 'My Group Tasks', 'Accounting', 'John's Tasks', 'Mary's Tasks', 'Peter's Tasks', and 'All Tasks'. The main area displays a task titled 'User Provides Info' with a sub-header 'User Registration'. Below this, there are tabs for 'Form', 'History', 'Diagram', and 'Description', with 'Form' being the active tab. The form contains several input fields: 'Token' (value: id4579821), 'Email' (value: jhmfritas@gmail.com), 'First\_Name' (value: Francisco), 'Last\_Name' (value: Ramos), 'Plan\_Type' (value: pre-paid), 'Balance' (value: 450), 'NIF' (value: 1250), and 'Address' (value: Rua 54). At the bottom right of the form, there are 'Save' and 'Complete' buttons.

As I can see, as the NIF is invalid, the process ended after the third task by following the 'No branch' in the 'Valid NIF' gateway

```
mai 15, 2020 4:44:08 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$0
INFO: Unique ID Validation Started
mai 15, 2020 4:44:09 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$0
INFO: Finished with HTTP error code : 200
HTTP/1.1 200 OK [Content-Type: application/json, Content-Length: 18, Connection: keep-alive, Date: Fri, 15 May 2020 03:44:09 GMT]
mai 15, 2020 4:44:09 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$0
INFO: Unique ID:true
mai 15, 2020 4:44:09 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$1
INFO: Validation Started
mai 15, 2020 4:44:09 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$1
INFO: Data is valid
mai 15, 2020 4:44:09 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$2
INFO: NIF Validation Started
mai 15, 2020 4:44:10 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$2
INFO: Finished with HTTP error code : 200
HTTP/1.1 200 OK [Server: nginx, Date: Fri, 15 May 2020 03:44:10 GMT, Content-Type: application/json; charset=utf-8]
mai 15, 2020 4:44:10 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$2
INFO: Valid NIF:false
```

## Operator Provision – Operator Creation Process

1. I started the process:



2. Then I provided the information needed:

The screenshot shows the 'Operator Provides Info' form within the 'Operator Creation' process. The form has tabs for 'Form', 'History', 'Diagram', and 'Description'. The 'Form' tab is active, showing three input fields: 'Operator Name' (with the value 'Cabify'), 'Operator Type' (a dropdown menu with 'T1' selected), and 'Price (If T1 or T2 price = null)' (with the value 'null'). At the bottom right, there are 'Save' and 'Complete' buttons.

3. Then I checked the logs of the worker, it seems that the creation succeeded

```
mai 28, 2020 12:58:10 AM org.camunda.bpm.pools.operatorCreation.OperatorCreationWorker lambda$0
INFO: Info Validation Started
mai 28, 2020 12:58:10 AM org.camunda.bpm.pools.operatorCreation.OperatorCreationWorker lambda$0
INFO: Data is valid
mai 28, 2020 12:58:10 AM org.camunda.bpm.pools.operatorCreation.OperatorCreationWorker lambda$1
INFO: Create Operator Started!
mai 28, 2020 12:58:14 AM org.camunda.bpm.pools.operatorCreation.OperatorCreationWorker lambda$1
INFO: Finished with HTTP error code : 200
HTTP/1.1 200 OK [Content-Type: application/json, Content-Length: 21, Connection: keep-alive, Date: Wed, 27 May 2020 23:58:14 GMT, x-:
mai 28, 2020 12:58:14 AM org.camunda.bpm.pools.operatorCreation.OperatorCreationWorker lambda$1
INFO: Creation Succeeded:Success
```

It created the topic in kafka:

```
tec2-user@ip-172-31-38-219 ~]$ sudo /usr/local/kafka/bin/kafka-topics.sh --list --zookeeper localhost:2181
OpenJDK 64-Bit Server VM warning: If the number of processors is expected to increase from one, then you should configure the number of parallel GC threads appropriately using -XX:ParallelGCThreads=N
Debit
T0_Metro
T1_Cabify
T1_Uber
T2_Gira
TripCosts
__consumer_offsets
tec2-user@ip-172-31-38-219 ~]$
```

It also created the operator in the database:

operatorName	operatorType	price
Cabify	T1	NULL
Gira	T2	NULL
Metro	T0	2.25
Uber	T1	NULL
NULL	NULL	NULL

- Invalid data is provided

If I try to create an Operator of type T1 with a non null value for the price for example, it detects invalid data and cancels the process

### Operator Provides Info

Operator Creation [🔗](#)

[📅 Set follow-up date](#) [🔔 Set due date](#) [👤 Add groups](#) [👤 Demo Demo](#) ×

**Form** History Diagram Description

**Operator Name**

**Operator Type**

T1 ▼

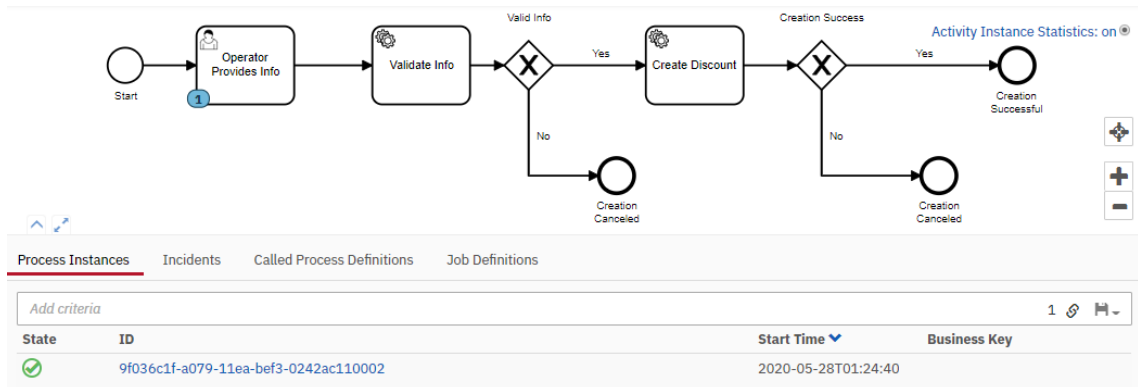
**Price (If T1 or T2 price = null)**

[Save](#) [Complete](#)

```
mai 28, 2020 1:09:54 AM org.camunda.bpm.pools.operatorCreation.OperatorCreationWorker lambda$0
INFO: Info Validation Started
mai 28, 2020 1:09:54 AM org.camunda.bpm.pools.operatorCreation.OperatorCreationWorker lambda$0
INFO: Data is invalid
```

## Operator Provision – Discount Creation Process

1. I started the process:



2. Then I provided the information needed:

**Operator Provides Info**

Discount Creation

Set follow-up date Set due date Add groups Demo Demo

Form History Diagram Description

**Discount ID**  
desconto-Maas-1

**Discount Name**  
Mês a Metade

**Value**  
50

**Start Date (format: 2020-06-05 00:00:0.000)**  
2020-06-01 00:00:00

**End Date (format: 2020-06-05 23:59:59.999)**  
2020-07-01 00:00:00

**Plan Type**  
Pre Paid

Save Complete

3. Then I checked the log:

```
mai 28, 2020 1:29:54 AM org.camunda.bpm.pools.discountCreation.DiscountCreationWorker lambda$0
INFO: Info Validation Started
mai 28, 2020 1:29:54 AM org.camunda.bpm.pools.discountCreation.DiscountCreationWorker lambda$0
INFO: Data is valid
mai 28, 2020 1:29:54 AM org.camunda.bpm.pools.discountCreation.DiscountCreationWorker lambda$1
INFO: Create Discount Started!
mai 28, 2020 1:29:59 AM org.camunda.bpm.pools.discountCreation.DiscountCreationWorker lambda$1
INFO: Finished with HTTP error code : 200
HTTP/1.1 200 OK [Content-Type: application/json, Content-Length: 21, Connection: keep-alive, Date: Thu, 28 May 2020 00:29:59 GMT,
mai 28, 2020 1:29:59 AM org.camunda.bpm.pools.discountCreation.DiscountCreationWorker lambda$1
INFO: Discount Creation:Success
```

The creation succeeded, then I checked the database:

	discountId	discountName	value	beginAt	endAt
▶	desconto-Maas-1	Mês a Metade	50	2020-06-01 00:00:00	2020-07-01 00:00:00
	dia-ambiente	Dia do Ambiente	13	2020-07-05 00:00:00	2020-07-06 00:00:00
	dia-sem-poluicao	Dia Sem Poluicao	45	2020-07-25 00:00:00	2020-07-30 00:00:00
*	NULL	NULL	NULL	NULL	NULL

- Invalid data is provided

1. I tried to create a discount with a value equal to 0:

Operator Provides Info

Discount Creation [🔗](#)

[Set follow-up date](#) [Set due date](#) [Add groups](#) [Demo Demo](#) ✕

[Form](#) [History](#) [Diagram](#) [Description](#)

**Discount ID**

id2

**Discount Name**

DescontoErro

**Value**

0

**Start Date (format: 2020-06-05 00:00:0.000)**

2020-07-05 00:00:0.000

**End Date (format: 2020-06-05 23:59:59.999)**

2020-07-05 23:59:59.999

**Plan Type**

Pre Paid

[Save](#) [Complete](#)

2. The creation was cancelled:

```
mai 28, 2020 1:37:14 AM org.camunda.bpm.pools.discountCreation.DiscountCreationWorker lambda$0
INFO: Info Validation Started
mai 28, 2020 1:37:14 AM org.camunda.bpm.pools.discountCreation.DiscountCreationWorker lambda$0
INFO: Data is invalid
```

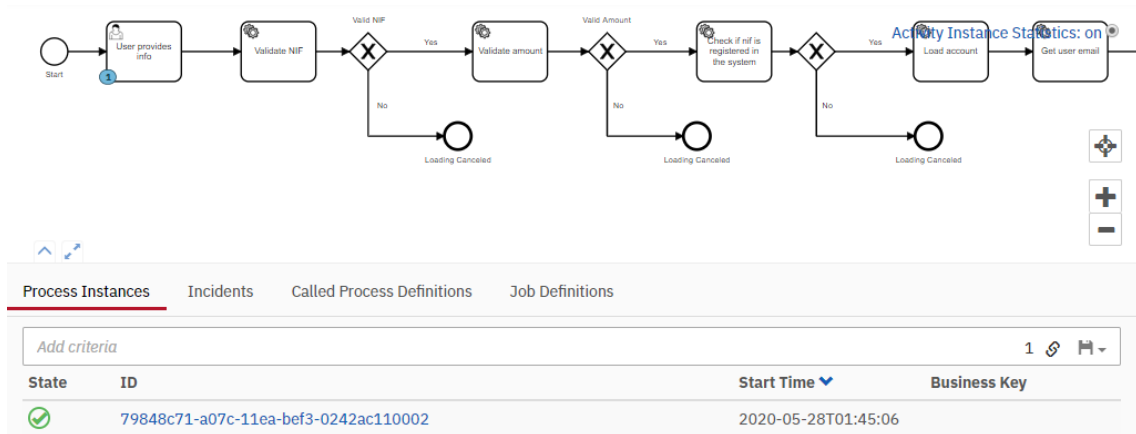
## Load Account Process

I used the information from user with id 4 for this test:

id	nif	email	firstName	lastName	planType	address
1	215963548	eil@gmail.com	Jose	Mario	pre-paid	Rua 1
2	215923548	idsl@gmail.com	Manuel	Esteves	post-paid	Rua 2
3	259433548	kds@gmail.com	Joana	Simões	generalPass	Rua 3
4	507712439	jhmffreitas@gmail.com	Rui	Pinheiro	pre-paid	Rua 4
NULL	NULL	NULL	NULL	NULL	NULL	NULL

id	balance	blackListed
1	116.56	0
2	500.00	0
3	500.00	0
4	146.80	0
NULL	NULL	NULL

1. I started the process:



2. Provided the needed info:

User provides info

Load Account

Set follow-up date Set due date Add groups Demo Demo x

Form History Diagram Description

NIF

507712439

Balance

100

Save Complete

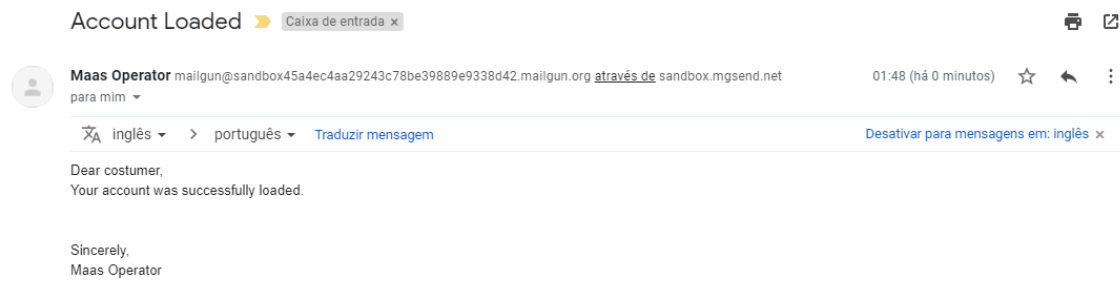
3. Then I checked the log:

```
mai 28, 2020 1:48:42 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$0
INFO: NIF Validation Started
mai 28, 2020 1:48:42 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$0
INFO: Finished with HTTP error code : 200
HTTP/1.1 200 OK [Server: nginx, Date: Thu, 28 May 2020 00:48:42 GMT, Content-Type: application/json; charset=utf-8, Transfer-Encoding
mai 28, 2020 1:48:42 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$0
INFO: Valid NIF:true
mai 28, 2020 1:48:42 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$1
INFO: Amount Validation Started
mai 28, 2020 1:48:42 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$1
INFO: Amount is valid
mai 28, 2020 1:48:42 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$2
INFO: Check NIF Started!
mai 28, 2020 1:48:47 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$2
INFO: Finished with HTTP error code : 200
HTTP/1.1 200 OK [Content-Type: application/json, Content-Length: 30, Connection: keep-alive, Date: Thu, 28 May 2020 00:48:47 GMT, x-a
mai 28, 2020 1:48:47 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$2
INFO: NIF is registered:Success
mai 28, 2020 1:48:47 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$4
INFO: Load Account Started!
mai 28, 2020 1:48:52 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$4
INFO: Finished with HTTP error code : 200
HTTP/1.1 200 OK [Content-Type: application/json, Content-Length: 99, Connection: keep-alive, Date: Thu, 28 May 2020 00:48:52 GMT, x-a
mai 28, 2020 1:48:52 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$4
INFO: Previous Balance: 146.8€
mai 28, 2020 1:48:52 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$4
INFO: Amount to deposit: 100€
mai 28, 2020 1:48:52 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$4
INFO: Loaded Account:Success! Account loaded: 246.8€
mai 28, 2020 1:48:52 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$3
INFO: Get email Started!
mai 28, 2020 1:48:57 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$3
INFO: Finished with HTTP error code : 200
HTTP/1.1 200 OK [Content-Type: application/json, Content-Length: 35, Connection: keep-alive, Date: Thu, 28 May 2020 00:48:57 GMT, x-a
mai 28, 2020 1:48:57 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$3
INFO: Got email:jhmffreitas@gmail.com
```

The creation succeeded, then I checked the database to verify the user balance:

id	balance	blackListed
1	116.56	0
2	500.00	0
3	500.00	0
4	246.80	0
NULL	NULL	NULL

I also received the confirmation email:

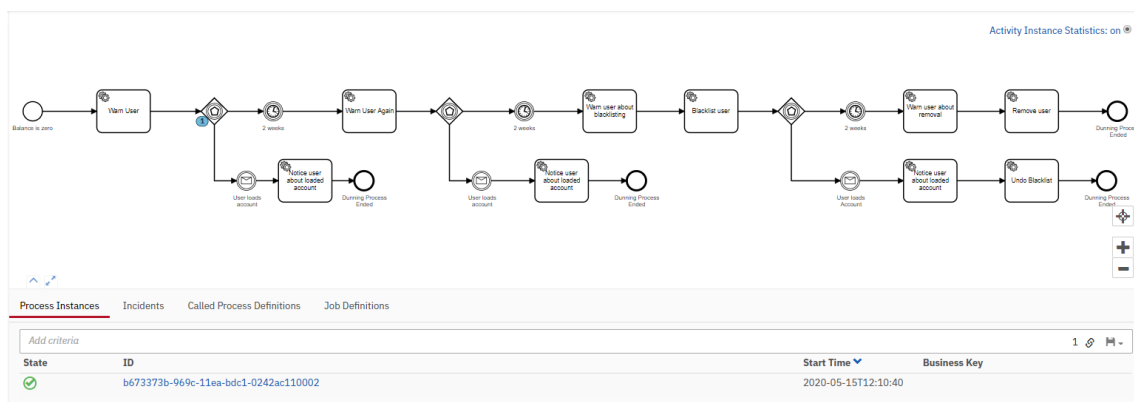


## Dunning Process

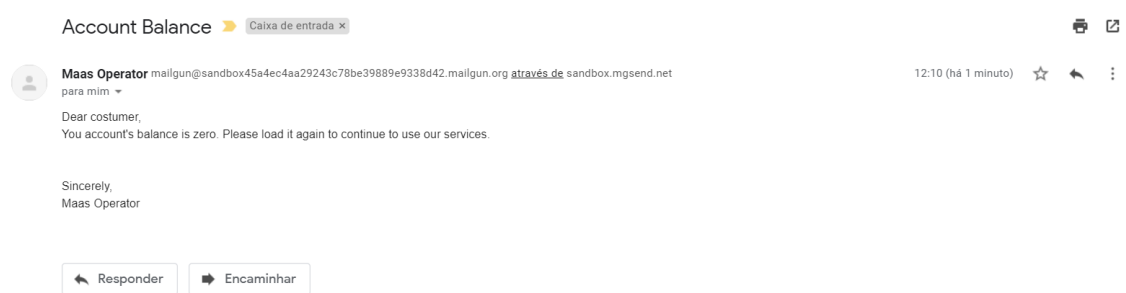
- User never loads the account
1. I will use the previously created user to execute the dunning process. In addition, I will use my personal email to check if the emails are sent.
  2. I started the process with this command:

```
curl -H "Content-Type: application/json" -X POST -d '{"variables": {"fromEmail": {"value": "Maas Operator <mailgun@sandbox45a4ec4aa29243c78be39889e9338d42.mailgun.org>"}, "email": {"value": "jhmffreitas@gmail.com"}, "token": {"value": "id456123753"} }}' http://192.168.99.100:8080/engine-rest/process-definition/key/dunning-process/start
```

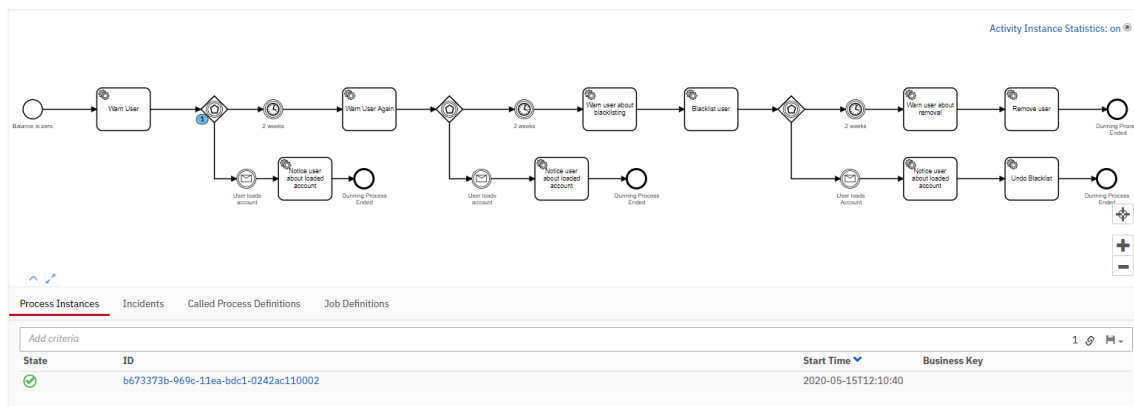
3. The process started at 12:10:



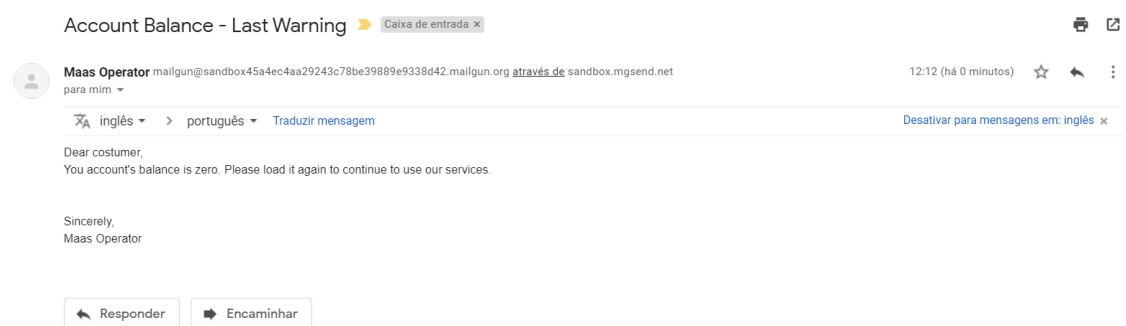
When the process started, I received this email:



Then it remained waiting for 2 weeks to pass (60 seconds):



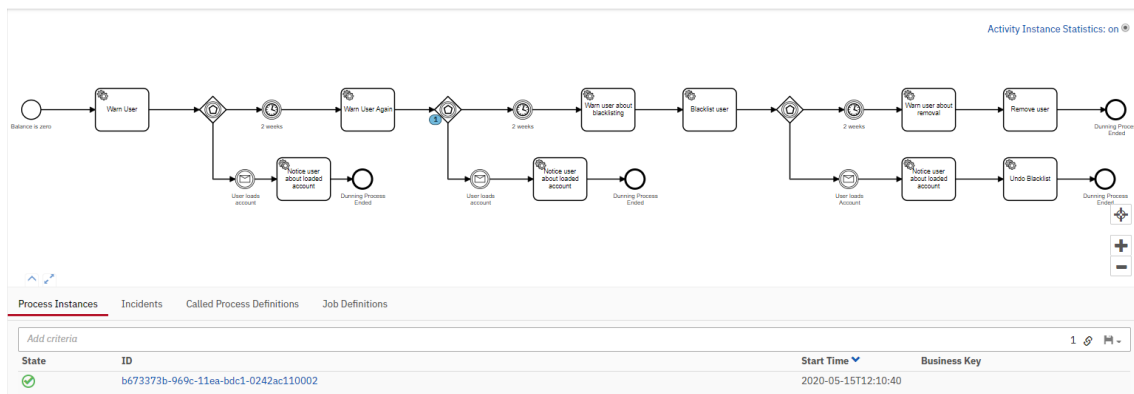
4. As I did not load the account it carried on warning the user again:



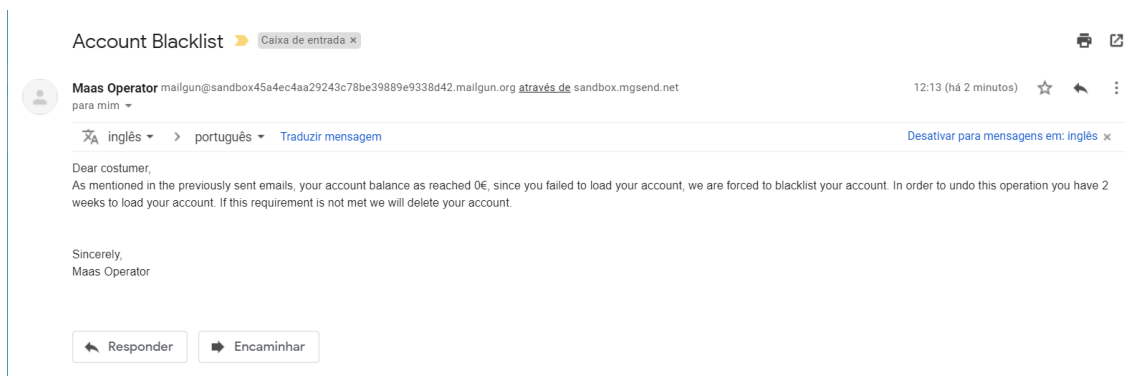
Then it remained waiting for 2 weeks to pass (60 seconds):



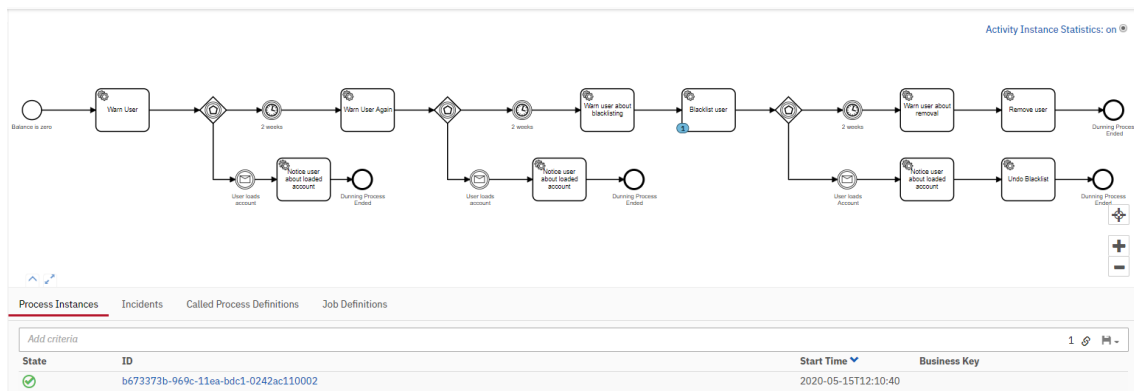
IE – MEIC-A – 2019/20, 2ºsem  
Sprint 4 – Grupo 3 – 87671  
Instituto Superior Técnico



5. As I did not load the account again it sent the user a notice about blacklisting the account:



Then started the blacklisting task:



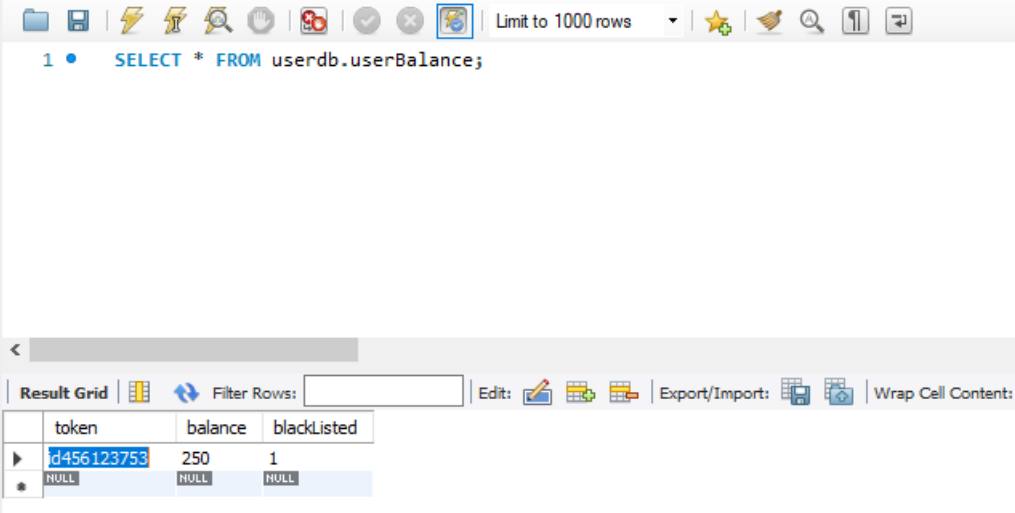
I can check in the console:

```
mai 15, 2020 12:15:18 PM org.camunda.bpm.pools.dunningProcess.DunningProcessWorker lambda$0
INFO: Blacklist User Started!
mai 15, 2020 12:15:23 PM org.camunda.bpm.pools.dunningProcess.DunningProcessWorker lambda$0
INFO: Finished with HTTP error code : 200
HTTP/1.1 200 OK [Content-Type: application/json, Content-Length: 21, Connection: keep-alive, Da
mai 15, 2020 12:15:23 PM org.camunda.bpm.pools.dunningProcess.DunningProcessWorker lambda$0
INFO: Blacklisted ID:Success
```

Here is the CloudWatch Log of the Blacklisting Service:

```
2020-05-15 11:15:23
{
  "requestContext": {
    "resourceId": "1yxzqd",
    "resourcePath": "/BlackListUser",
    "httpMethod": "POST",
    "extendedRequestId": "Mkf3MGjyIAMFfLQ=",
    "requestTime": "15/May/2020:11:15:19 +0000",
    "path": "/default/BlackListUser",
    "accountId": "618315746481",
    "protocol": "HTTP/1.1",
    "stage": "default",
    "domainPrefix": "z62m3l4rh2",
    "requestTimeEpoch": 1589541319516,
    "requestId": "d37ce392-2db8-414f-b402-a997fdc9698a",
    "identity": {
      "cognitoIdentityPoolId": null,
      "accountId": null,
      "cognitoIdentityId": null,
      "caller": null,
      "sourceIp": "54.236.120.160",
      "principalOrgId": null,
      "accessKey": null,
      "cognitoAuthenticationType": null,
      "cognitoAuthenticationProvider": null,
      "userArn": null,
      "userAgent": "Apache-HttpClient/4.5.11 (Java/1.8.0_251)",
      "user": null
    }
  },
  "domainName": "z62m3l4rh2.execute-api.us-east-1.amazonaws.com",
  "apiId": "z62m3l4rh2"
},
{
  "resource": "/BlackListUser",
  "httpMethod": "POST",
  "queryStringParameters": null,
  "stageVariables": null,
  "body": "{\"id\":\"id456123753\",\"operation\":\"blacklist\"}"
}
```

After this I can check the database, the blacklist flag is set to true:

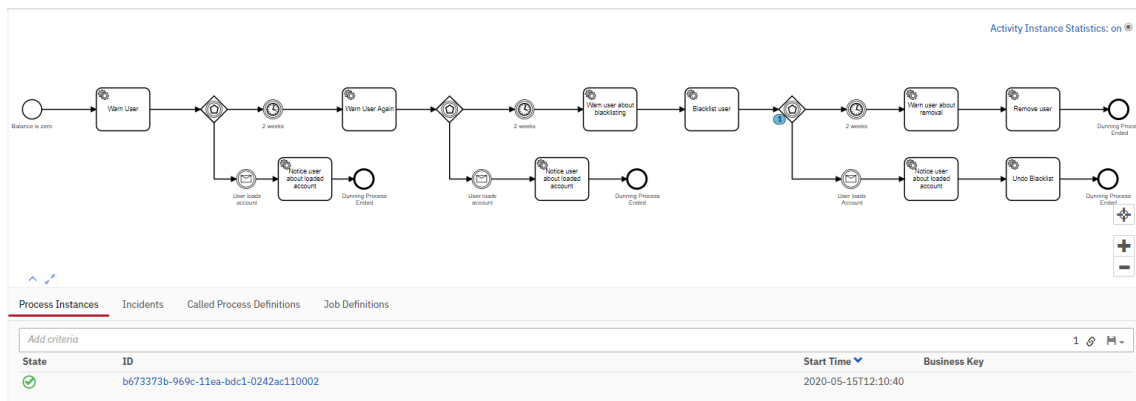


The screenshot shows a database query interface. The SQL query entered is `SELECT * FROM userdb.userBalance;`. The results are displayed in a table with three columns: `token`, `balance`, and `blackListed`. The first row shows a token of `d456123753`, a balance of `250`, and a `blackListed` value of `1`. The second row shows `NULL` for all three fields.

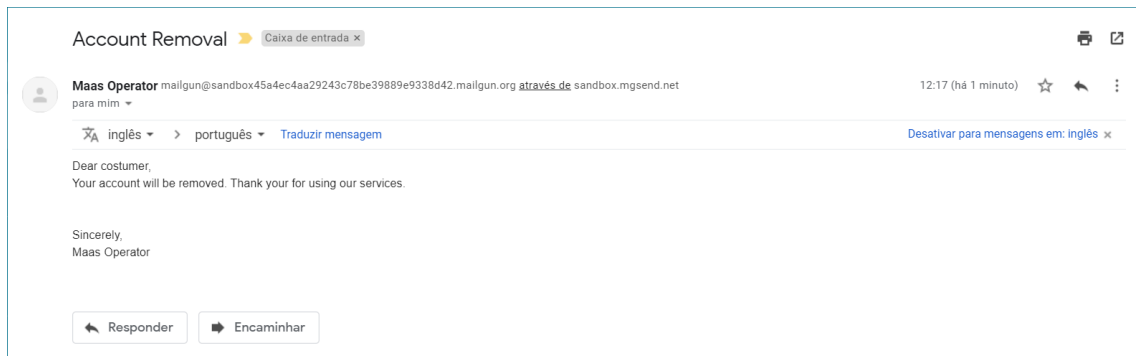
token	balance	blackListed
d456123753	250	1
NULL	NULL	NULL

After this the process waited for 2 weeks (60 seconds):

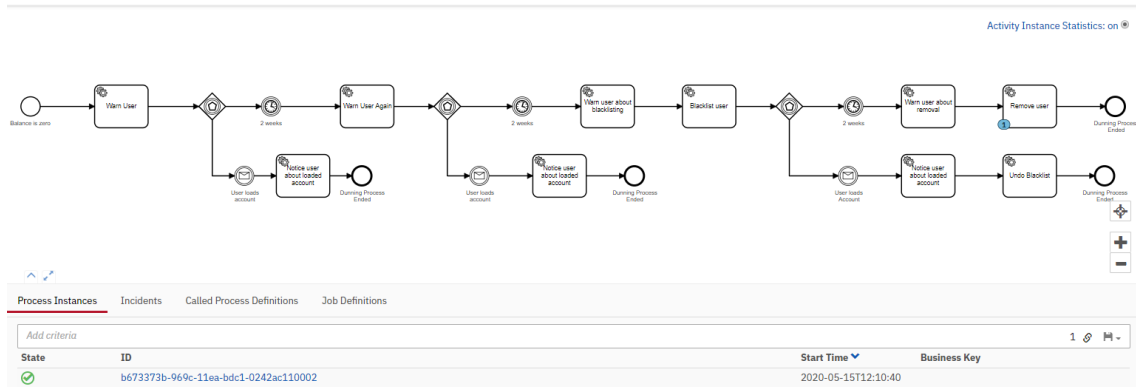
IE – MEIC-A – 2019/20, 2ºsem  
Sprint 4 – Grupo 3 – 87671  
Instituto Superior Técnico



6. As I did not load the account again during this period, it warned the user about the removal:



Then it started the removal task:



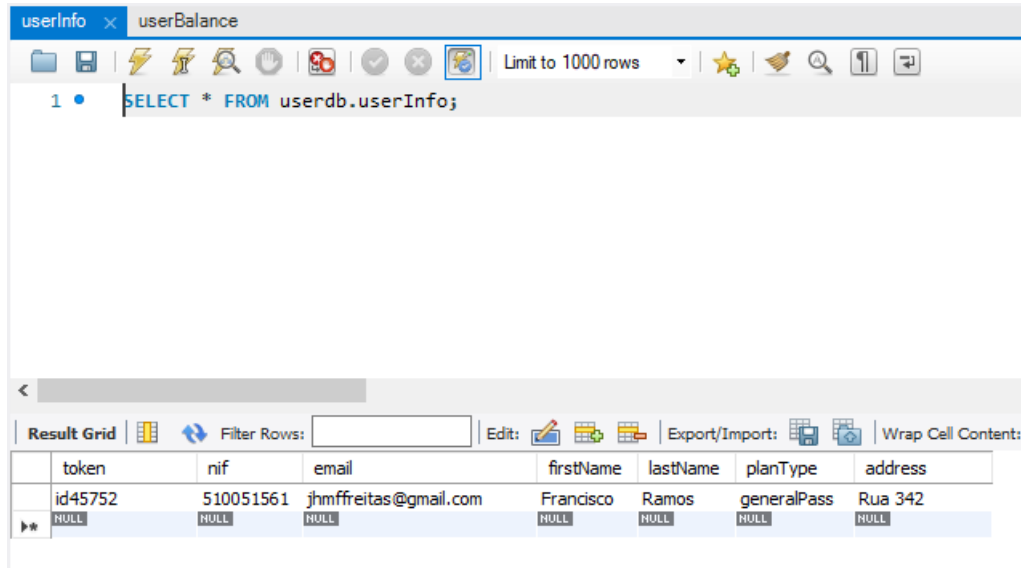
I can check in the console that the operation was successful, and the process ended:

```
mai 15, 2020 12:17:31 PM org.camunda.bpm.pools.dunningProcess.DunningProcessWorker lambda$2
INFO: Remove User Started!
mai 15, 2020 12:17:36 PM org.camunda.bpm.pools.dunningProcess.DunningProcessWorker lambda$2
INFO: Finished with HTTP error code : 200
HTTP/1.1 200 OK [Content-Type: application/json, Content-Length: 21, Connection: keep-alive, D:
mai 15, 2020 12:17:36 PM org.camunda.bpm.pools.dunningProcess.DunningProcessWorker lambda$2
INFO: Remove User:Success
```

[illegible]

- User Loads account after first warning

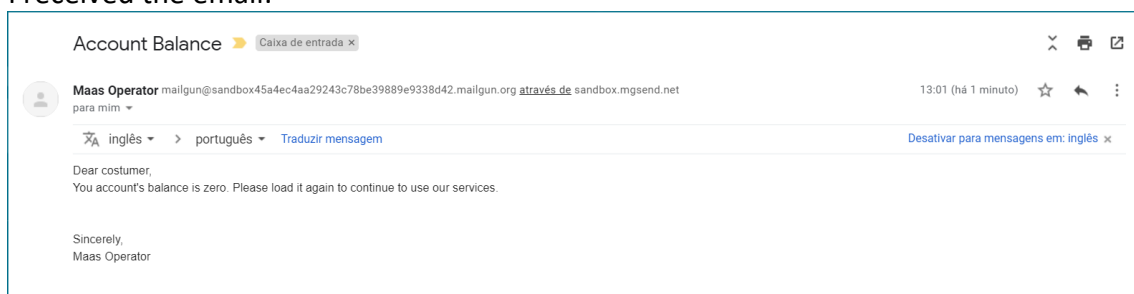
1. I created a new user in the database:



I started the process again with the command:

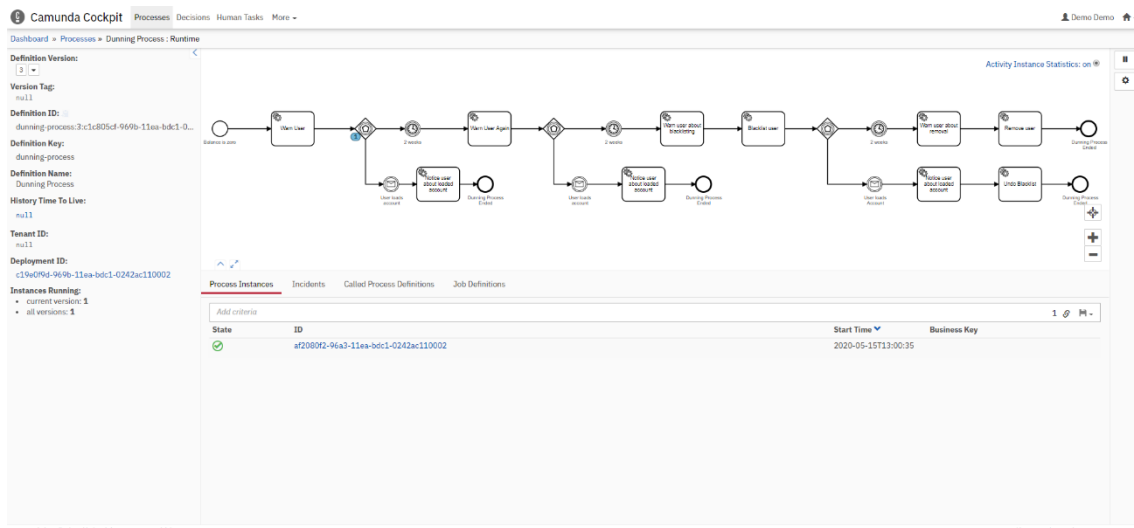
```
curl -H "Content-Type: application/json" -X POST -d '{"variables":  
{"fromEmail": {"value": "Maas Operator  
<mailgun@sandbox45a4ec4aa29243c78be39889e9338d42.mailgun.org>"}, "email  
": {"value": "jhmffreitas@gmail.com"}, "token": {"value": "id45752"} } }'  
http://192.168.99.100:8080/engine-rest/process-definition/key/dunning-  
process/start
```

I received the email:



Then the process started waiting for the two weeks:

IE – MEIC-A – 2019/20, 2ºsem  
Sprint 4 – Grupo 3 – 87671  
Instituto Superior Técnico



2. When the process was waiting for the two weeks to pass, I ran this command:

```
curl -H "Content-Type: application/json" -X POST --data  
@userLoadsAccount.json http://192.168.99.100:8080/engine-rest/message
```

userLoadsAccount.json content:

```
{  
  "messageName": "UserLoadsAccountMessage"  
}
```

After this message was sent, I received an email saying this:



This message will be used by the Payment Service when the user loads an account to end the dunning process. After this the process ended.

- User Loads account after second warning

This case is the same as before, with the difference that the user receives two warnings instead of one. For this case I used the user created in the previous case as it was not deleted from the database. Just to illustrate I will show the emails received with the corresponding timestamps:

IE – MEIC-A – 2019/20, 2ºsem  
Sprint 4 – Grupo 3 – 87671  
Instituto Superior Técnico

**Maas Operator** mailgun@sandbox45a4ec4aa29243c78be39889e9338d42.mailgun.org [através de](#) sandbox.mgsend.net  
para mim ▾ 13:15 (há 2 minutos) ☆ ↶ ⋮

🌐 inglês ▾ > português ▾ [Traduzir mensagem](#) [Desativar para mensagens em: inglês](#) ✕

Dear costumer,  
Your account's balance is zero. Please load it again to continue to use our services.

Sincerely,  
Maas Operator

**Account Balance - Last Warning** 📧 [Caixa de entrada](#) ✕

**Maas Operator** mailgun@sandbox45a4ec4aa29243c78be39889e9338d42.mailgun.org [através de](#) sandbox.mgsend.net  
para mim ▾ 13:16 (há 1 minuto) ☆ ↶ ⋮

🌐 inglês ▾ > português ▾ [Traduzir mensagem](#) [Desativar para mensagens em: inglês](#) ✕

Dear costumer,  
Your account's balance is zero. Please load it again to continue to use our services.

Sincerely,  
Maas Operator

**Maas Operator** mailgun@sandbox45a4ec4aa29243c78be39889e9338d42.mailgun.org [através de](#) sandbox.mgsend.net  
para mim ▾ 13:17 (há 1 minuto) ☆ ↶ ⋮

🌐 inglês ▾ > português ▾ [Traduzir mensagem](#) [Desativar para mensagens em: inglês](#) ✕

Dear costumer,  
Thank you for loading your account. You can now continue to use our services.

Sincerely,  
Maas Operator

- User loads account after blacklist

In the beginning the user was not blacklisted:

userInfo userBalance ✕

Limit to 1000 rows

1 • `SELECT * FROM userdb.userBalance;`

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

	token	balance	blackListed
▶	id45752	456	0
*	NULL	NULL	NULL

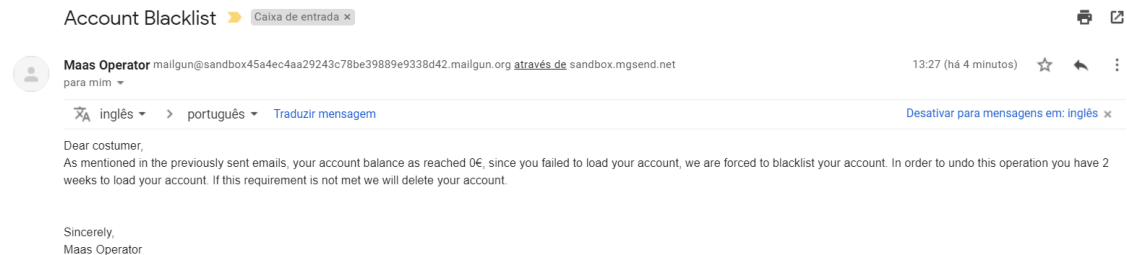
1. I started the process with the previous command and received the first email:



2. After 60 seconds I received the second one:



3. Then I received a warning about the blacklist operation

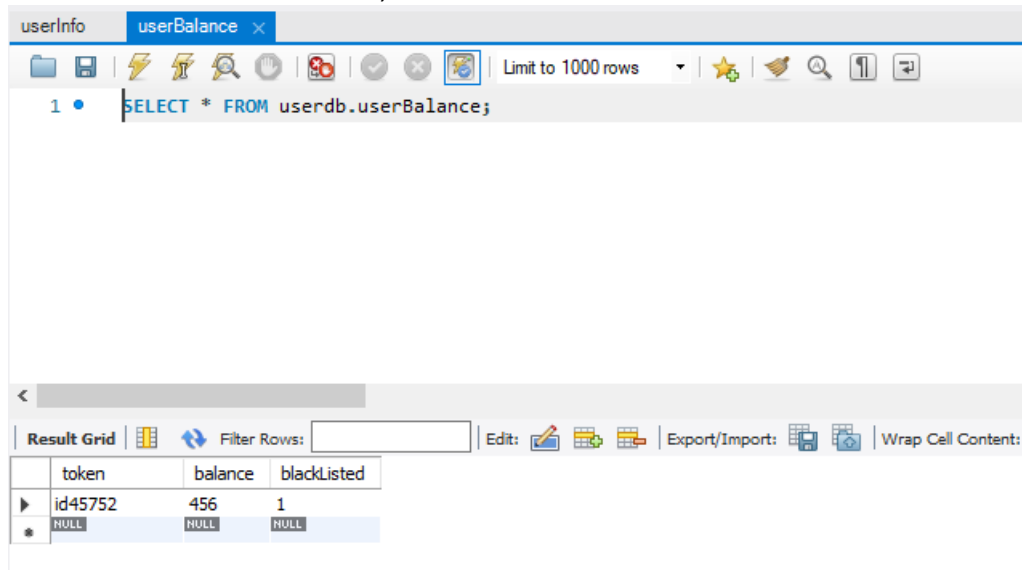


I checked the console:

```
mai 15, 2020 1:27:54 PM org.camunda.bpm.pools.dunningProcess.DunningProcessWorker lambda$0
INFO: Blacklist User Started!
mai 15, 2020 1:27:59 PM org.camunda.bpm.pools.dunningProcess.DunningProcessWorker lambda$0
INFO: Finished with HTTP error code : 200
HTTP/1.1 200 OK [Content-Type: application/json, Content-Length: 21, Connection: keep-alive, D
mai 15, 2020 1:27:59 PM org.camunda.bpm.pools.dunningProcess.DunningProcessWorker lambda$0
INFO: Blacklisted ID:Success
```



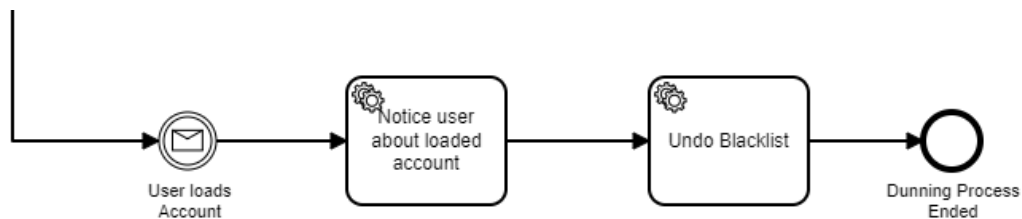
Then I checked the database, the user is blacklisted:



The screenshot shows a database query tool interface. At the top, there are tabs for 'userInfo' and 'userBalance'. Below the tabs is a toolbar with various icons and a 'Limit to 1000 rows' dropdown. The query editor shows a SQL statement: `SELECT * FROM userdb.userBalance;`. Below the query editor is a 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Edit' button, an 'Export/Import' button, and a 'Wrap Cell Content' button. The result grid contains a table with the following data:

token	balance	blackListed
id45752	456	1
NULL	NULL	NULL

4. After this the process was waiting for the next 60 seconds but I sent a `loadAccountMessage`, so it went to this branch:



I received the following email:



Then I checked the console:

```
INFO: Undo Blacklist Started!
mai 15, 2020 1:28:35 PM org.camunda.bpm.pool.dunningProcess.DunningProcessWorker lambda$1
INFO: Finished with HTTP error code : 200
HTTP/1.1 200 OK [Content-Type: application/json, Content-Length: 21, Connection: keep-alive, I
mai 15, 2020 1:28:35 PM org.camunda.bpm.pool.dunningProcess.DunningProcessWorker lambda$1
INFO: Undo Blacklist:Success
```

Here is the CloudWatch Log of the undo-blacklist operation:

```
2020-05-15 12:28:36
{
  "requestContext": {
    "resourceId": "1yxzqd",
    "resourcePath": "/BlackListUser",
    "httpMethod": "POST",
    "extendedRequestId": "MkqmJHySoAMFWtg=",
    "requestTime": "15/May/2020:12:28:36 +0000",
    "path": "/default/BlackListUser",
    "accountId": "618315746481",
    "protocol": "HTTP/1.1",
    "stage": "default",
    "domainPrefix": "z62m3l4rh2",
    "requestTimeEpoch": 1589545716062,
    "requestId": "3e31b3ea-76f4-411f-816b-6f2026daa03e",
    "identity": {
      "cognitoIdentityPoolId": null,
      "accountId": null,
      "cognitoIdentityId": null,
      "caller": null,
      "sourceIp": "54.236.120.160",
      "principalOrgId": null,
      "accessKey": null,
      "cognitoAuthenticationType": null,
      "cognitoAuthenticationProvider": null,
      "userArn": null,
      "userAgent": "Apache-HttpClient/4.5.11 (Java/1.8.0_251)",
      "user": null
    },
    "domainName": "z62m3l4rh2.execute-api.us-east-1.amazonaws.com",
    "apiId": "z62m3l4rh2"
  },
  "resource": "/BlackListUser",
  "httpMethod": "POST",
  "queryStringParameters": null,
  "stageVariables": null,
  "body": "{\"id\":\"id45752\",\"operation\":\"undo-blacklist\"}"
}
```

Finally, I checked the database again, the user was not blacklisted anymore:

userInfo userBalance x

Limit to 1000 rows

1 • SELECT \* FROM userdb.userBalance;

Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:

	token	balance	blackListed
▶	id45752	456	0
*	NULL	NULL	NULL

## Automatic start of Dunning Process and stop with Load Account Process

1. I started by having a user in the database with a low balance and produced a trip event for that user

id	balance	blackListed
1	116.56	0
2	500.00	0
3	500.00	0
4	1.80	0
NULL	NULL	NULL

```
PS C:\Users\jhmff\Desktop\IST\MEIC\1ano\2semestre\IE\Projeto\Sprint2\Producers\Uber\ProducerProviderAbstractWithoutTokenControl\target> java -jar ProducerProvider2-0.0.1-SNAPSHOT.jar --provider-name Uber --broker-list ec2-35-173-139-11.compute-1.amazonaws.com:9092,ec2-35-173-139-11.compute-1.amazonaws.com:9093,ec2-35-173-139-11.compute-1.amazonaws.com:9094 --topic T1_Uber --id-list 4 --throughput 1
--typeMessage JSON
The following arguments are accepted:
--provider-name=Uber
--broker-list=ec2-35-173-139-11.compute-1.amazonaws.com:9092,ec2-35-173-139-11.compute-1.amazonaws.com:9093,ec2-35-173-139-11.compute-1.amazonaws.com:9094
--topic=T1_Uber
--id-list=[4]
--throughput=1
--typeMessage=JSON
----- Processing starting -----
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
This is the message to send = {"event":{"eventType":"t1", "operator":"Uber", "info":{"Id": "4", "Price": "97.91515", "Timestamp": "2020-05-28 02:21:32.567" }}}
Sending new message to Kafka... with key=1590628892576
Sent...
Waiting...2020-05-28 02:21:33.194
```

2. The trip event was successfully handled as I can see by the logs :

### UserManagementService:

```
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Subscribing to topic ->T2_GIRA
Subscribing to topic ->T0_Metro
Subscribing to topic ->T1_Uber
handleRequest: Begin!!
handleRequest: topic = T1_Uber, partition = 1, offset = 97,customer = 1590628892576,message = {"event":{"eventType":"t1",
"operator":"Uber", "info":{" Id": "4", "Price": "97.91515", "Timestamp": "2020-05-28 02:21:32.567" }}}
parseEvent(operator):Uber
parseEvent(eventType):t1
parseEvent(info):{"Price":"97.91515","Id":"4","Timestamp":"2020-05-28 02:21:32.567"}
produceTripCostEvent : send event !
produceTripCostEvent : Sent -> {"event":{"eventType":"trip-cost", "info":{" cost": "97.91515", "id": "4", "planType": "pre-paid",
"operatorName": "Uber", "timeStamp": "2020-05-28 02:21:32.567" }}}
END RequestId: ce603721-989b-4865-8bf6-59281de9eb00
REPORT RequestId: ce603721-989b-4865-8bf6-59281de9eb00 Duration: 20011.22 ms Billed Duration: 20000 ms Memory Size: 3008 MB
Max Memory Used: 130 MB Init Duration: 29.61 ms
2020-05-28T01:24:07.234Z ce603721-989b-4865-8bf6-59281de9eb00 Task timed out after 20.01 seconds
```

### OperatorManagementService:

```
startService: Begin!!
startService: topic = TripCosts, partition = 1, offset = 183,customer = TripCostsKey,message = {"event":{"eventType":"trip-cost", "info":{" cost": "97.91515", "id": "4", "planType": "pre-pa
id", "operatorName": "Uber", "timeStamp": "2020-05-28 02:21:32.567" }}}
processEvent(eventType):trip-cost
processEvent(info):{"timeStamp":"2020-05-28 02:21:32.567","cost":"97.91515","planType":"pre-paid","id":"4","operatorName":"Uber"}
getDiscount : Start searching for discount
getDiscount : No discount found
BaseCost:97.91515
produceDebitEvent : Sent -> {"event":{"eventType":"debit", "info":{" id": "4", "planType": "pre-paid", "amount": "97.91515" }}}

```

Database:

3. I started a Load Account Process with this info:

## User provides info

### Load Account

Set follow-up date

Set due date

Add groups

Demo Demo

Form

History

Diagram

Description

NIF

507712439

Balance

200

Save

Complete

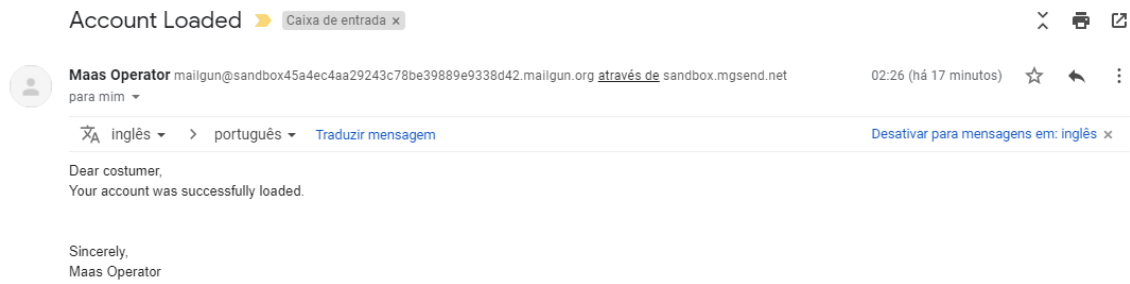
4. I checked the log and the loading succeeded:

```
mai 28, 2020 2:25:55 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$0
INFO: NIF Validation Started
mai 28, 2020 2:25:56 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$0
INFO: Finished with HTTP error code : 200
HTTP/1.1 200 OK [Server: nginx, Date: Thu, 28 May 2020 01:25:56 GMT, Content-Type: application/json; charset=utf-8, Transfer-Encoding: gzip]
mai 28, 2020 2:25:56 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$0
INFO: Valid NIF:true
mai 28, 2020 2:25:56 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$1
INFO: Amount Validation Started
mai 28, 2020 2:25:56 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$1
INFO: Amount is valid
mai 28, 2020 2:25:56 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$2
INFO: Check NIF Started!
mai 28, 2020 2:26:01 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$2
INFO: Finished with HTTP error code : 200
HTTP/1.1 200 OK [Content-Type: application/json, Content-Length: 30, Connection: keep-alive, Date: Thu, 28 May 2020 01:26:01 GMT,
mai 28, 2020 2:26:01 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$2
INFO: NIF is registered:Success
mai 28, 2020 2:26:01 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$4
INFO: Load Account Started!
mai 28, 2020 2:26:06 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$4
INFO: Finished with HTTP error code : 200
HTTP/1.1 200 OK [Content-Type: application/json, Content-Length: 102, Connection: keep-alive, Date: Thu, 28 May 2020 01:26:06 GMT,
mai 28, 2020 2:26:06 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$4
INFO: Previous Balance: -96.12€
mai 28, 2020 2:26:06 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$4
INFO: Amount to deposit: 200€
mai 28, 2020 2:26:06 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$4
INFO: Loaded Account:Success! Account loaded: 103.88€
mai 28, 2020 2:26:06 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$3
INFO: Get email Started!
mai 28, 2020 2:26:11 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$3
INFO: Finished with HTTP error code : 200
HTTP/1.1 200 OK [Content-Type: application/json, Content-Length: 35, Connection: keep-alive, Date: Thu, 28 May 2020 01:26:11 GMT,
mai 28, 2020 2:26:11 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$3
INFO: Got email:jhmffreitas@gmail.com
```

The user now has positive balance:

id	balance	blackListed
1	116.56	0
2	500.00	0
3	500.00	0
4	103.88	0
NULL	NULL	NULL

I also received this email:



5. It also detected that the dunning process was active, so it sent a message to stop it:

```
mai 28, 2020 2:26:11 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$5
INFO: Stop Dunning Started!
mai 28, 2020 2:26:11 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$5
INFO: Post request: {"messageName":"UserLoadsAccountMessage","businessKey":"4"}
mai 28, 2020 2:26:11 AM org.camunda.bpm.pools.loadAccount.LoadAccountWorker lambda$5
INFO: Finished with HTTP error code : 204
HTTP/1.1 204 [Date: Thu, 28 May 2020 01:26:11 GMT, Keep-Alive: timeout=20, Connection: keep-alive]
```

After this the dunning process stopped and I received this email:

