

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

Instituto Superior Técnico – MEIC-A

Sprint 3 Report

1. Open API specification files

(Files inside SwaggerFiles folder)

2. Configuration of the Kong hq endpoints to serve each microservice

(File inside KongConfigurations folder)

--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'
```

-----**(Will be needed in the next sprint)**

--CREATES OPERATOR MANAGEMENT SERVICE (deals with operator provision and cost processing)

```
curl -i -X POST --url http://localhost:8001/services/ --data 'name=OperatorManagementService' --data 'url=http://ec2-3-80-233-61.compute-1.amazonaws.com:9997/operatorManagementService'
```

--ADDS ROUTE FOR OPERATOR MANAGEMENT SERVICE OPERATIONS

```
curl -i -X POST --url http://localhost:8001/services/OperatorManagementService/routes --data 'hosts[]=operatorManagementService.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'
```

-----**(Will be needed in the next sprint)**

--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'
```

3. Modelling of all the MaaS business processes using Camunda modeler

Costumer Provision process:

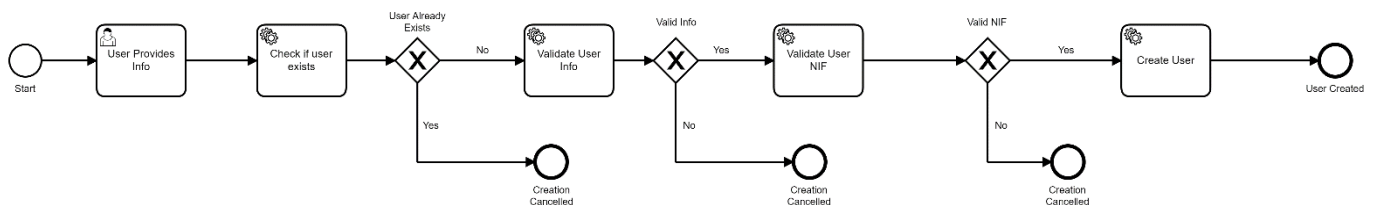


Fig. 1 – User Registration Process

Dunning Process:

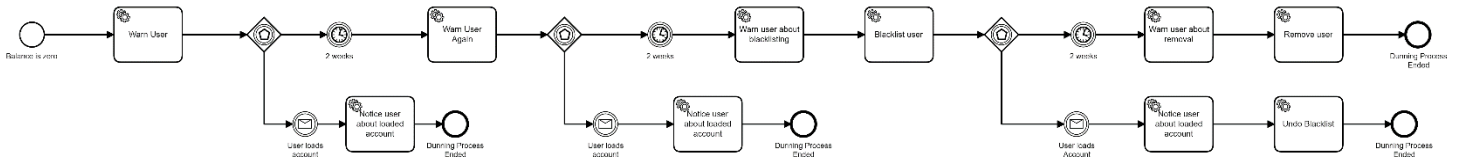


Fig. 2 – Dunning Process

4. Development of one executable business processes regarding the process provisioning of customers or providers, using Camunda modeller and deployed in Camunda engine

(Code inside ProvisioningProcess folder)

This workflow was integrated with an external webservice that validates nifs (<https://www.nif.pt/>)

5. Development of one executable business processes regarding the dunning process of customers, using Camunda modeller and deployed in Camunda engine

(Code inside DunningProcess folder)

Besides developing code for blacklisting and removing the user in the Java Delegate class, I integrated the Camunda model with connectors that use Mailgun to send emails to the user.

In the next sprint I intend to execute this process automatically in the PaymentService when the balance of an account reaches 0€. As I do not have that service implemented yet, I invoke this process remotely with curl.

6. Development of the required databases, microservices, kafka topics, etc.

(Files inside RequiredDevelopment folder)

Changes since previous sprint:

- **Databases:**
 - I added two columns to the userInfo table which were the nif and address of the user.
 - I added one column to the userBalance table to tell if a user is blacklisted or not. The decision to create the column in this table is due to the separation that I made in the database (UserManagementService only accesses userInfo table and PaymentService only accesses userbalance

table). As the blacklisting of a client is related to payments, I created it in this table.

- **Microservices:**

- I had to refactor UserManagementService to separate the handling of trip events and the creation of users, as a result I created a Lambda function to create users
- RequiredDevelopment/Microservices/UserManagementService only deals with trip event processing now
- RequiredDevelopment/Microservices/OperatorManagementService now provides endpoints for creation of topics and discounts
- The rest of the code contained in the RequiredDevelopment/Microservices are Lambda functions that I had to create for the different operations needed in the developed workflows

- **Kafka Topics, Zookeeper configurations and Kafka configurations:**

- Remained the same since the previous sprint

7. Functional Testing

In this section I will demonstrate the correct execution of the two executable process:

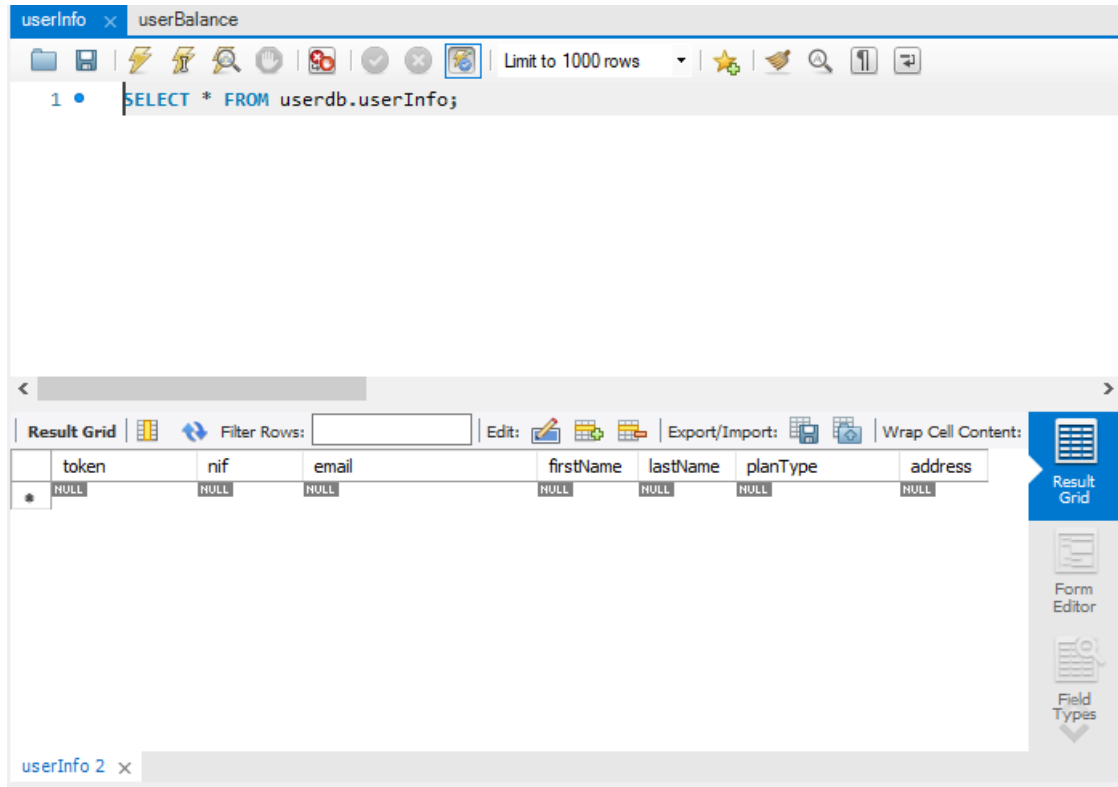
User Registration Process

- User is created successfully:

1. When we 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 displays the Camunda Tasklist web application. On the left, a sidebar shows a list of tasks under 'My Tasks (1)', including 'User Provides Info' which is currently selected. The main area shows the 'User Provides Info' task form, which is part of the 'User Registration' process. The form includes fields for 'Token' (id456123753), 'Email' (jhmfrtias@gmail.com), 'First_Name' (Joao), 'Last_Name' (Frtias), '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 interface also shows a 'Created' timestamp of 'Created 5 minutes ago' and a 'Demo Demo' label.

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: "MjTcOGZlIiAMF-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,
    cognitoAuthProvider: 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:

We 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, we have the validation of the NIF task:

In the console we 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.

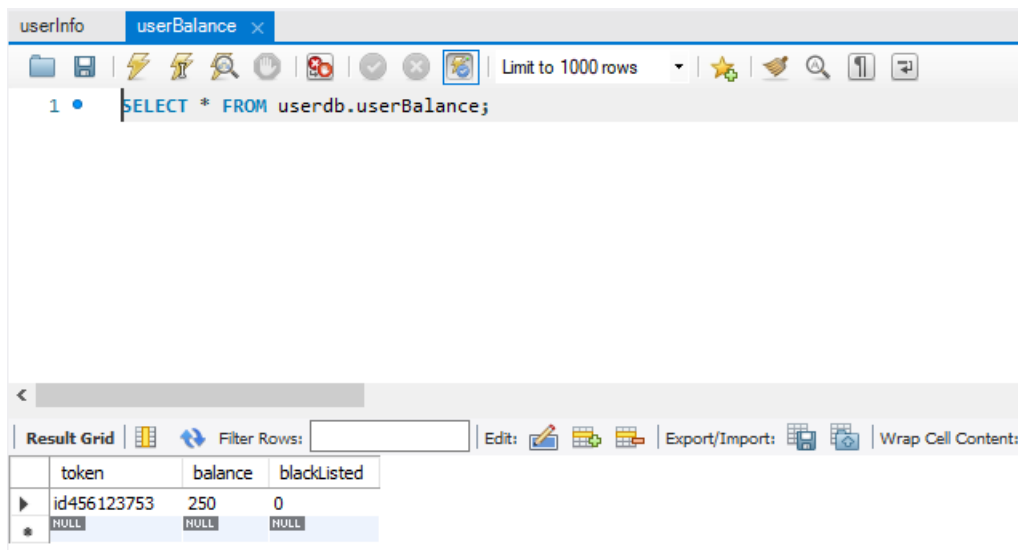
```
INFO: NIF Validation Started
mai 15, 2020 3:33:42 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker lambda$2
INFO: Finished with HTTP error code : 200
HTTP/1.1 200 OK [Server: nginx, Date: Fri, 15 May 2020 02:33:42 GMT, Content-Type: application/json;
mai 15, 2020 3:33:42 AM org.camunda.bpm.pools.userRegister.UserRegisterWorker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

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

In the console we 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 }
```

[illegible]



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 we 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

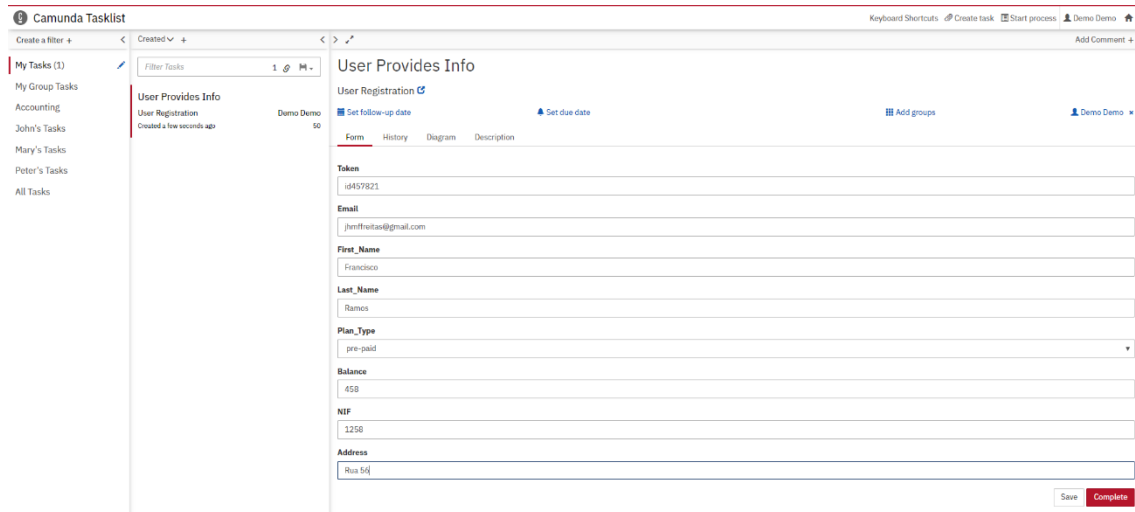
I will start the process again and try to insert a user with different token but with empty fields in the first name and last name for example

As we 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



As we 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
```

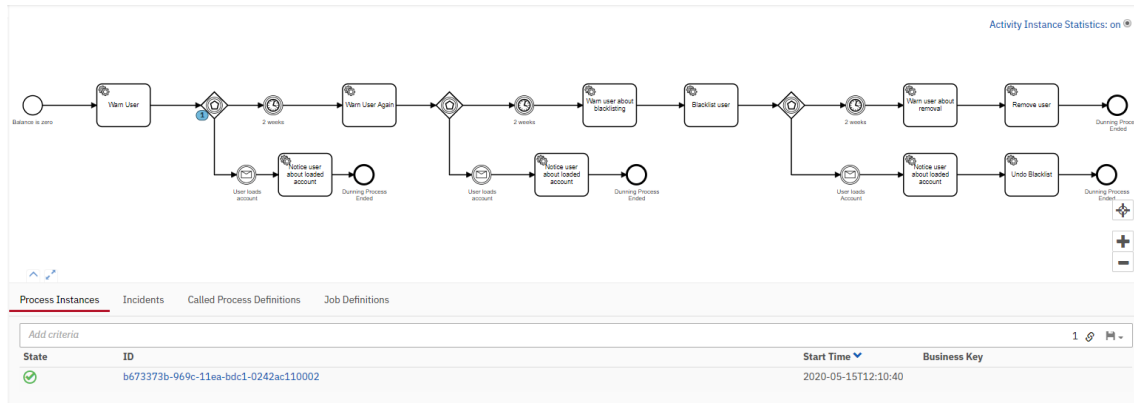
Dunning Process

- 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:

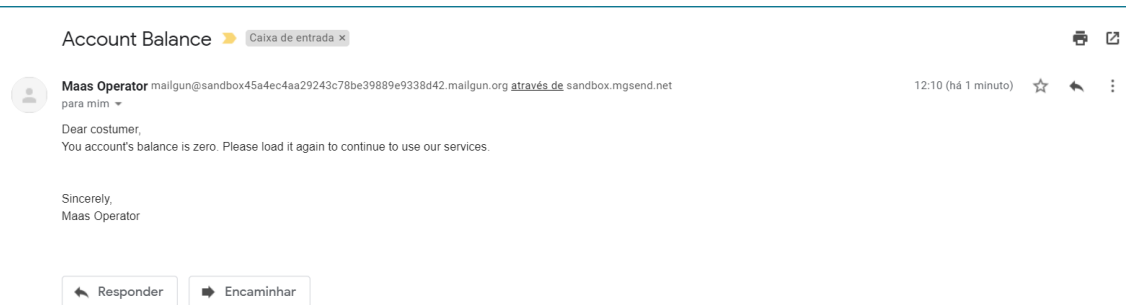
```
curl -H "Content-Type: application/json" -X POST -d '{"variables": {"fromEmail": {"value": "Maas Operator"}}
```

```
<mailgun@sandbox45a4ec4aa29243c78be39889e9338d42.mailgun.org>"},"email": {"value": "jhmfreytas@gmail.com"}, "token": {"value": "id456123753"} } }' http://192.168.99.100:8080/engine-rest/process-definition/key/dunning-process/start
```

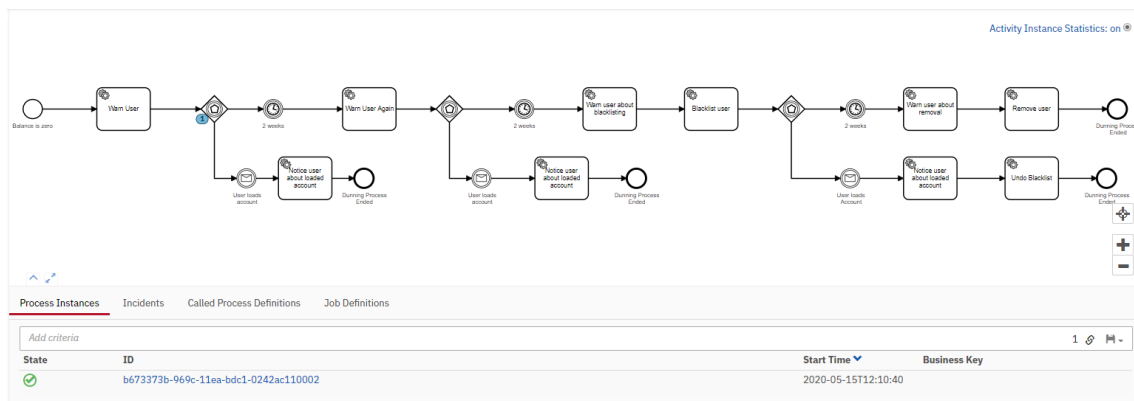
3. The process started at 12:10:



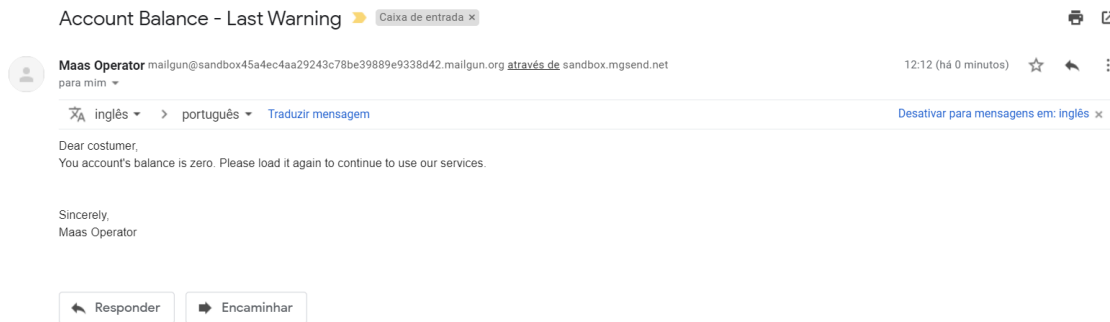
When the process started, I received this email:



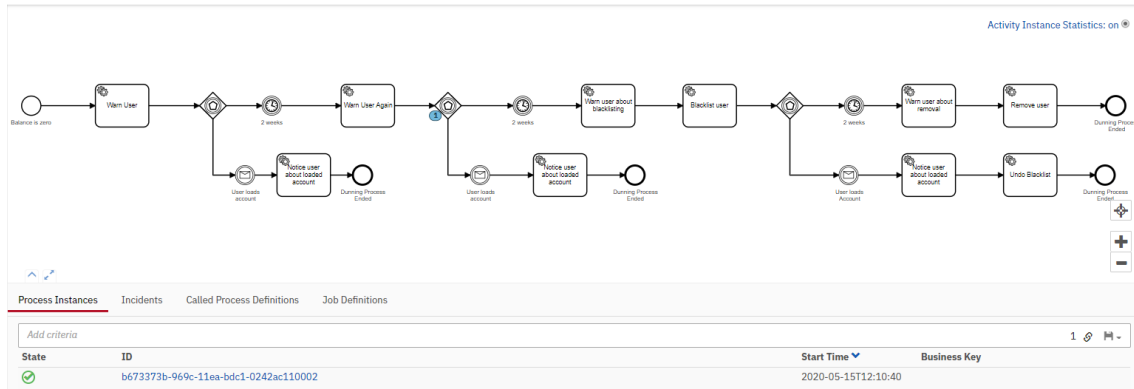
Then it remained waiting for 2 weeks to pass (60 seconds):



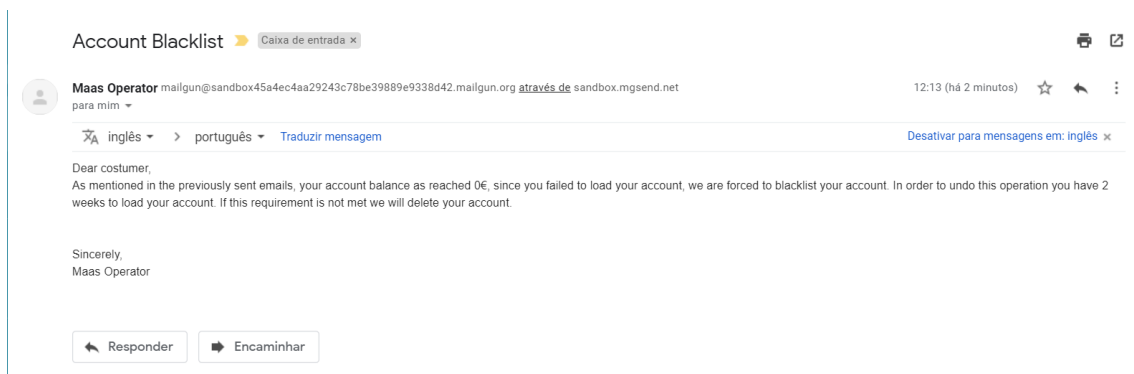
4. As I did not load the account it carried on warning the user again:



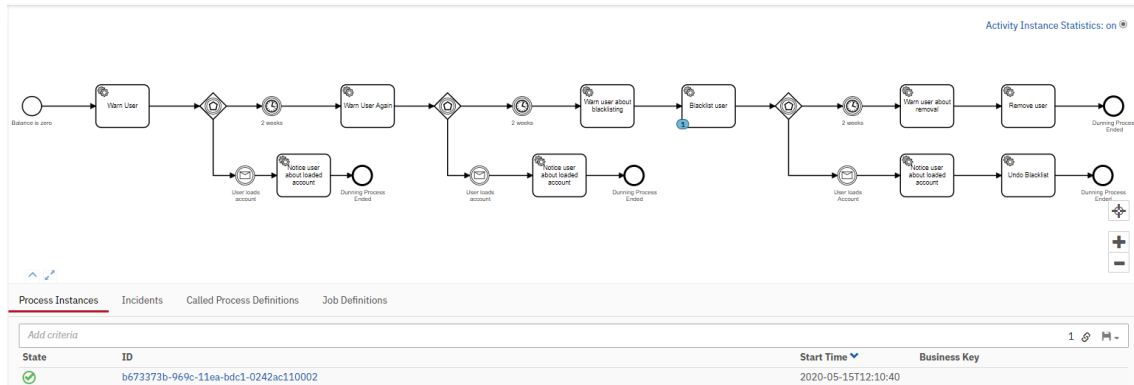
Then it remained waiting for 2 weeks to pass (60 seconds):



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



Then started the blacklisting task:



We 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:

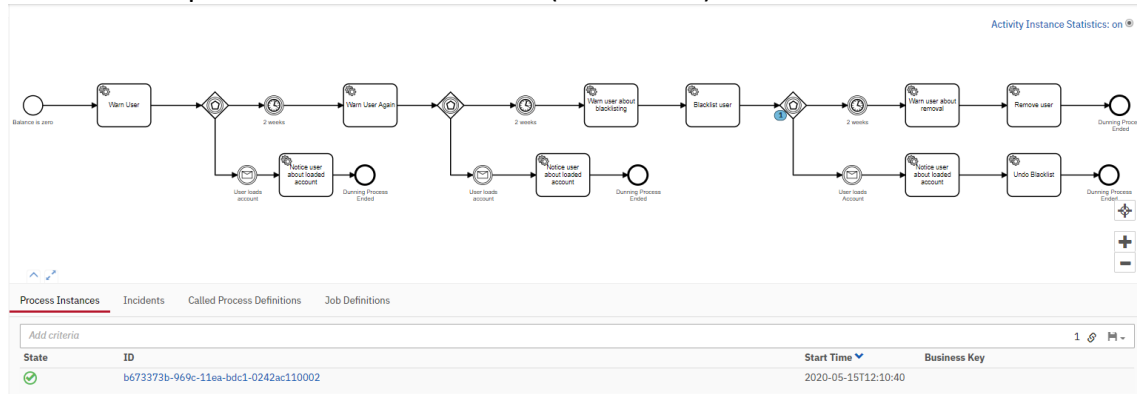
```
2020-05-15 11:15:23
{
  "requestContext": {
    "resourceId": "1yxzqd",
    "resourcePath": "/BlackListUser",
    "httpMethod": "POST",
    "extendedRequestId": "Mkf3MGjyIAMFfLQ=",
    "requestTime": "15/May/2020:11:15:19 +0000",
    "path": "/default/BlackListUser",
    "accountId": "618315746481",
    "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 we can check the database, the blacklist flag is set to true:

The screenshot shows a database query tool interface. At the top, there is a toolbar with various icons and a dropdown menu set to "Limit to 1000 rows". Below the toolbar, a SQL query is entered: `1 • SELECT * FROM userdb.userBalance;`. The results are displayed in a table with the following data:

token	balance	blackListed
d456123753	250	1
NULL	NULL	NULL

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



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

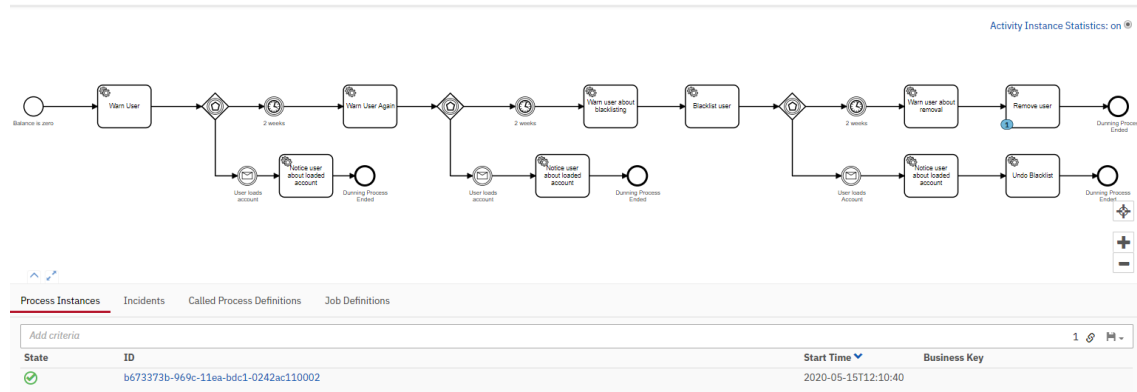
The screenshot shows an email message titled "Account Removal". The sender is "Maas Operator" with the email address "mailgun@sandbox45a4ec4aa29243c78be39889e9338d42.mailgun.org". The message is addressed to "mim". The content of the email is:

Dear customer,
Your account will be removed. Thank you for using our services.

Sincerely,
Maas Operator

At the bottom of the email, there are two buttons: "Responder" and "Encaminhar".

Then it started the removal task:



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

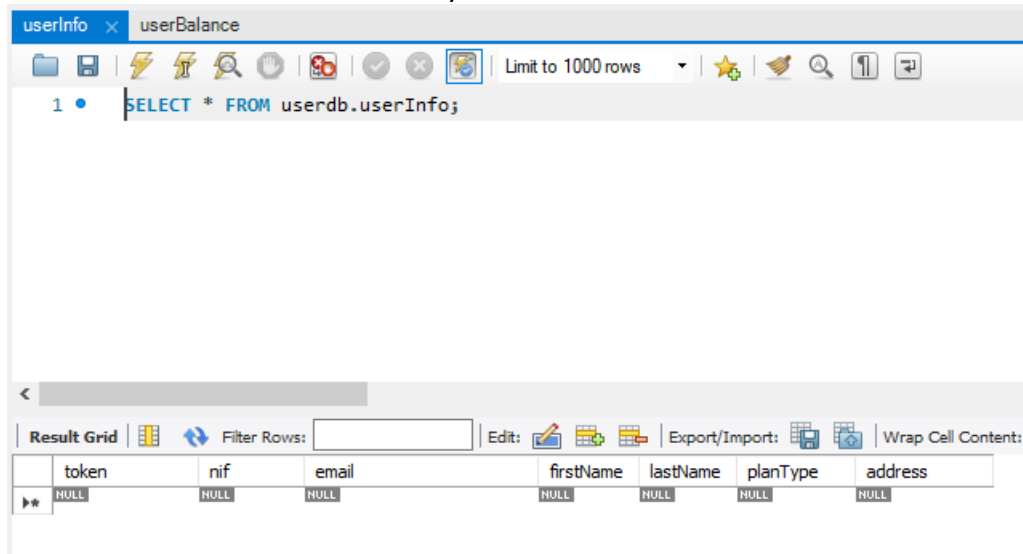
```
mai 15, 2020 12:17:31 PM org.camunda.bpm.pools.dunningProcess.DunningProcessWorker lambda$2
INFO: Remove User Started!
mai 15, 2020 12:17:36 PM org.camunda.bpm.pools.dunningProcess.DunningProcessWorker lambda$2
INFO: Finished with HTTP error code : 200
HTTP/1.1 200 OK [Content-Type: application/json,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

Here is the CloudWatch Log of the Removal Service:

2020-05-15 11:17:37

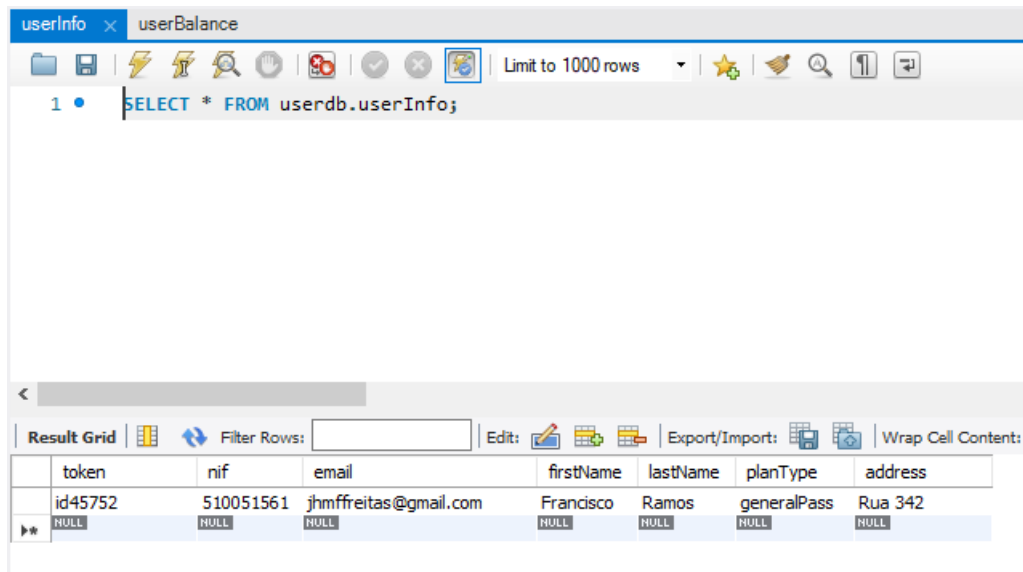
```
{
  "requestContext": {
    "resourceId": "uwauco",
    "resourcePath": "/UserRemoval",
    "httpMethod": "POST",
    "extendedRequestId": "MkgMEHr7oAMFp7g=",
    "requestTime": "15/May/2020:11:17:33 +0000",
    "path": "/default/UserRemoval",
    "accountId": "618315746481",
    "protocol": "HTTP/1.1",
    "stage": "default",
    "domainPrefix": "hbjev9a14p4",
    "requestTimeEpoch": 1589541453153,
    "requestId": "7d85c701-c6a6-4bbd-a2f1-4e466c498e18",
    "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": "hbjev9a14p4.execute-api.us-east-1.amazonaws.com",
  "apiId": "hbjev9a14p4"
},
{
  "resource": "/UserRemoval",
  "httpMethod": "POST",
  "queryStringParameters": null,
  "stageVariables": null,
  "body": "{\"id\":\"id456123753\"}"
}
```

The user is not in the database anymore:



- User Loads account after first warning

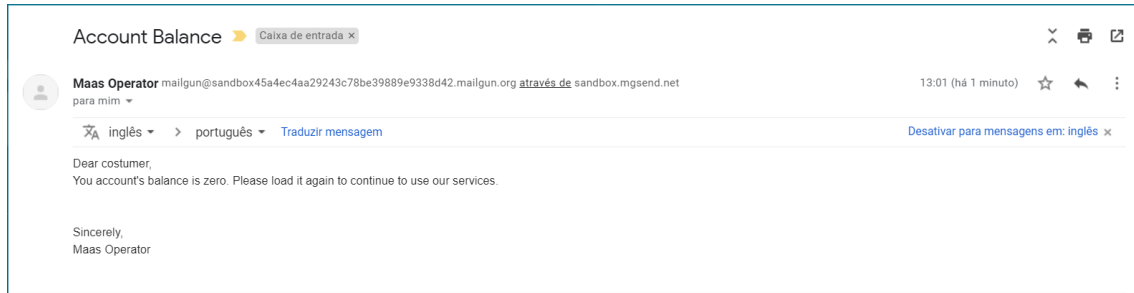
1. I created a new user in the database:



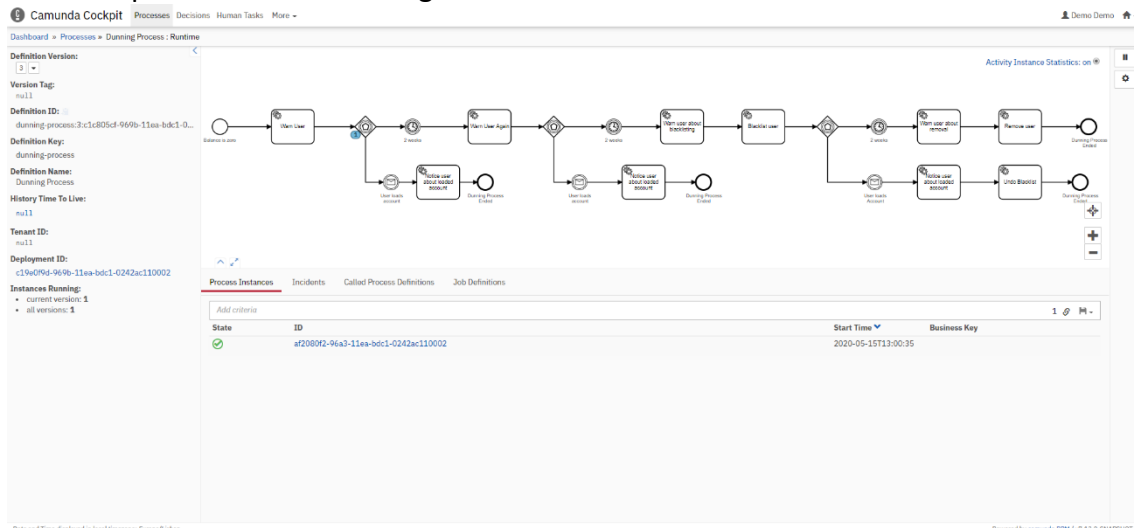
I started the process again with the command:

```
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:



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"  
}
```

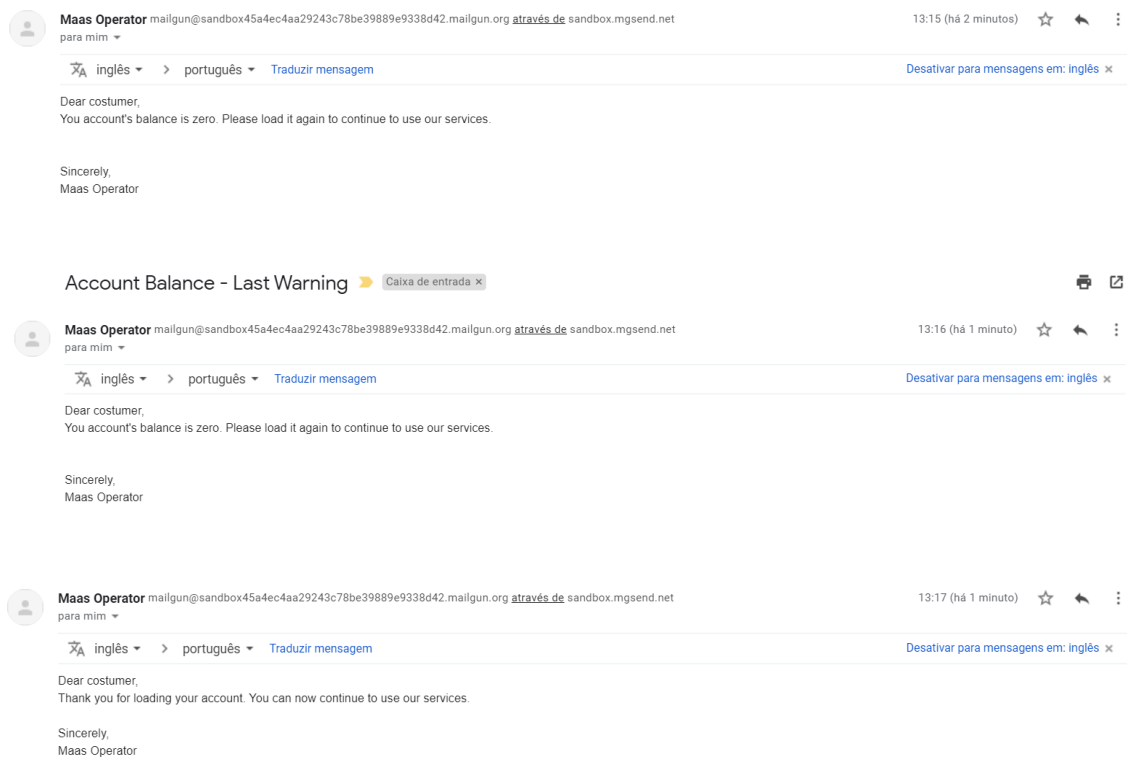
After this message was sent, I received an email saying this:



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.

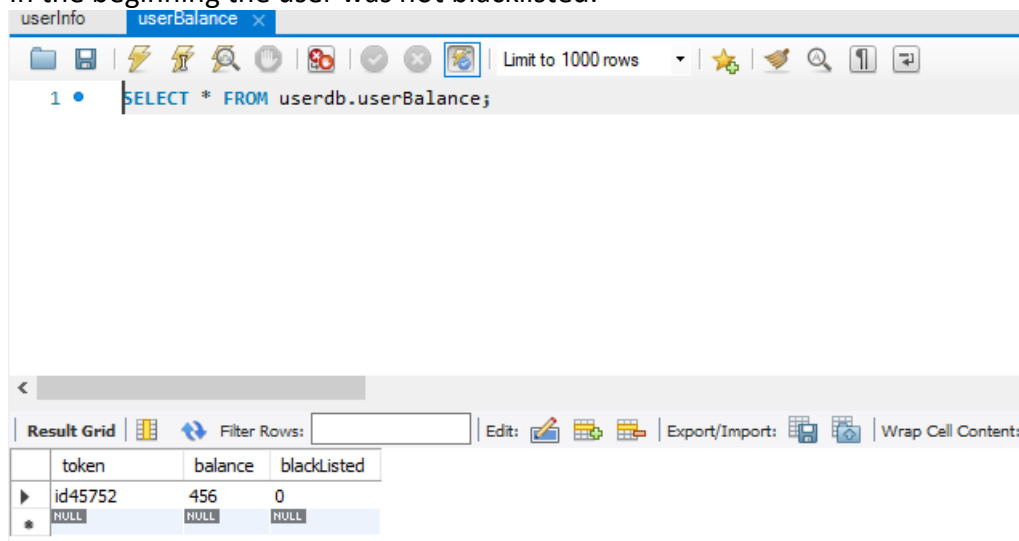
- User Loads account after second warning

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:



- User loads account after blacklist

In the beginning the user was not blacklisted:



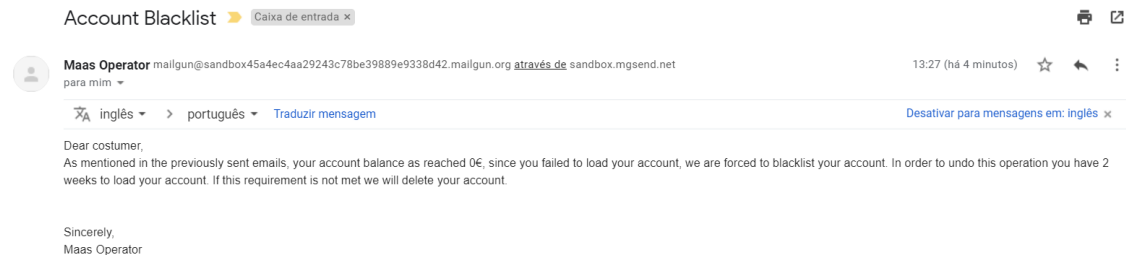
1. I started the process with the previous command and received the first email:



2. After 60 seconds I received the second one:



3. Then I received a warning about the blacklist operation



I checked the console:

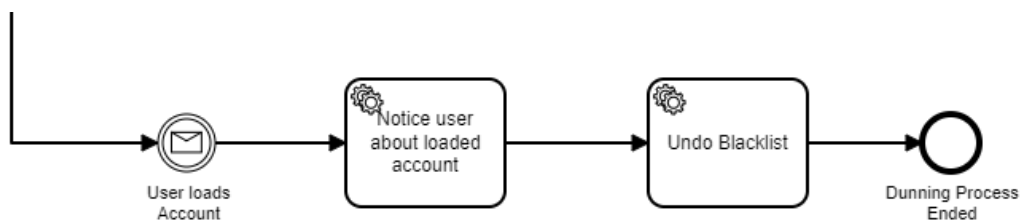
```
mai 15, 2020 1:27:54 PM org.camunda.bpm.pools.dunningProcess.DunningProcessWorker lambda$0
INFO: Blacklist User Started!
mai 15, 2020 1:27:59 PM org.camunda.bpm.pools.dunningProcess.DunningProcessWorker lambda$0
INFO: Finished with HTTP error code : 200
HTTP/1.1 200 OK [Content-Type: application/json, 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 single query: `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 displays a table with three columns: 'token', 'balance', and 'blackListed'. The first row shows 'id45752', '456', and '1'. The second row shows 'NULL', 'NULL', and 'NULL'.

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:

The screenshot shows an email from 'Maas Operator' with the email address 'mailgun@sandbox45a4ec4aa29243c78be39889e9338d42.mailgun.org'. The email is addressed to 'mim'. The email body contains the following text: 'Dear costumer, Thank you for loading your account. You can now continue to use our services. Sincerely, Maas Operator'. The email is in Portuguese, and there are links to 'Traduzir mensagem' and 'Desativar para mensagens em: inglês'.

Then I checked the console:

```
INFO: Undo Blacklist Started!
mai 15, 2020 1:28:35 PM org.camunda.bpm.pools.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.pools.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": "MkqmJHy5oAMFWtg=",
    "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