Enterprise Software Infrastructure Project Report

Scholarship Apply Process with Blockchain

Prof. Andrea Morichetta

XU ZHUOMING 106816

zhuoming.xu@studenti.unicam.it

NGUYEN THANH NGOC 106817

thanhngoc.nguyen@studenti.unciam.it

June,2020

Contents

Introduction

In this project, a scholarship application process is modeling with BPMN. The process is further deployed to Camunda REST Engine, moreover, a blockchain technology is integrated inside the process as record the student application information.

Use Case

A student login the unicam scholarship system, after filling the pre-required information, he/she send the request to UNICAM_BDS department. The automation checking system is installed in two different sub-divisions, the first one is BDS_student_office, where the first round check is achieved, GPA is the fundamental requirement for the application, if the student's GPA is less than 3.0, the system will refuse the student application immediately and send the notification to students, if it is greater than 3.0, it will goes to BDS_segeteria_office system for the second round check, in this part, another hard-core requirement is Comprehensive Score, if a student's comprehensive score is less than 5.0, the system will deny the application as well even though GPA is greater than 3.0, if the students meet both fundamental requirements , the system will send the congrats notification to student, and the application

information will be archived by the staff in BDS_segeteria_office.

The BPMN collaboration model is shown in figure 1.

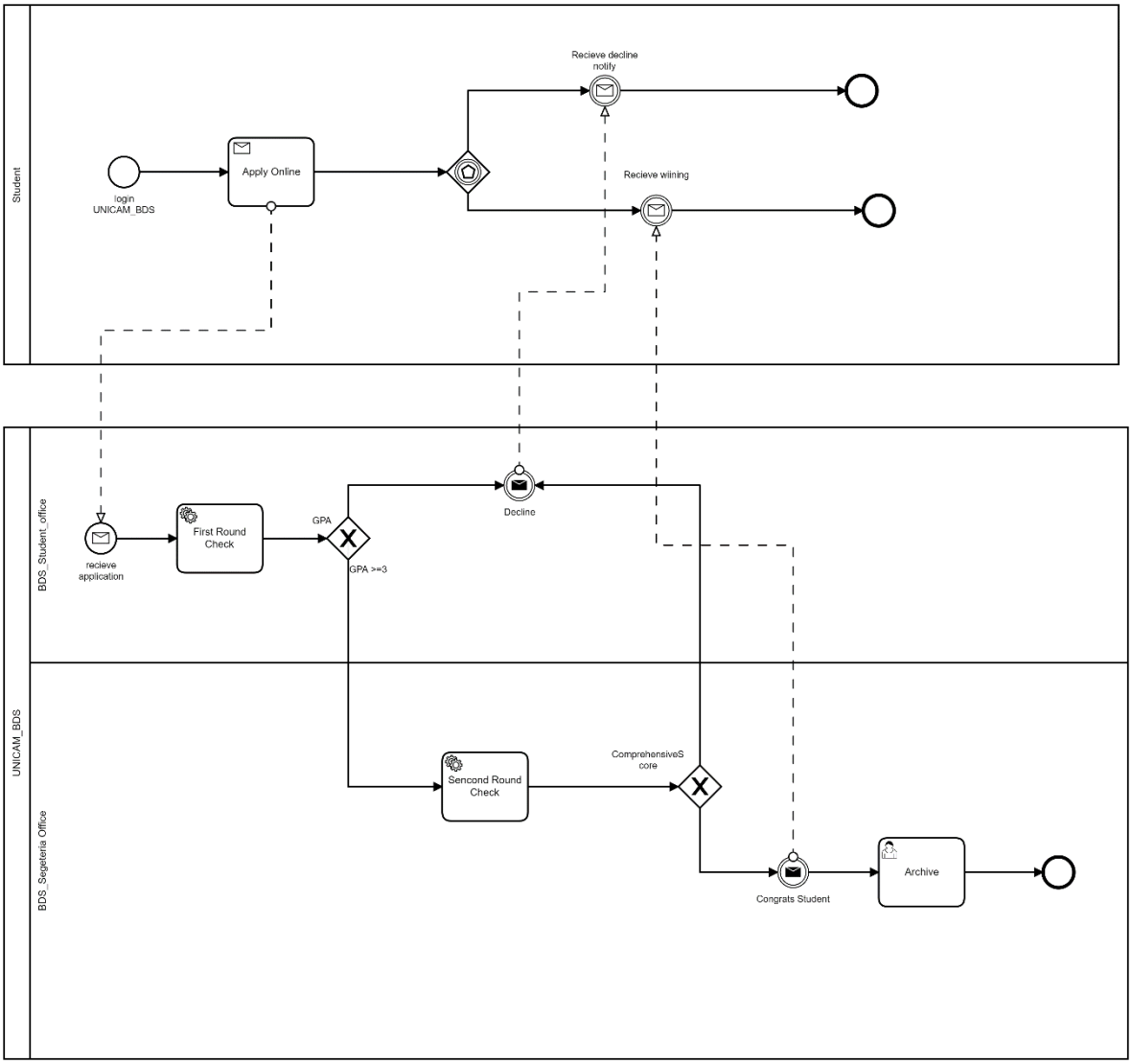


Figure1.1 ApplyScholarship Process

Model Deployment and Interaction

The process is further deployed to Camunda Process Engine with endpoint http://locahost:8080/engin -rest/. The tasks activities in the process are defined as External Task type and implemented with JavaScript as task subscriber and Java Application as message notification since the notification task may be also achieved through other applications.

In the project, the application process is simulated with Postman, a student starts application through Camunda Rest API web service, using http://locahost:8080/engin -rest/message message-start event to post the data to the system to start the application process. Figure 1.2 shows the that the token is pending and wait to be subscribed.

```
{
  "messageName": "Start",
  "businessKey": "2",
  "processVariables": {
    "gpa": {"value": "6", "type": "Integer"},
    "comprehensive":{"value":"8","type":"Integer"},
    "studentID":{"value":"10010", "type":"String"}
  }
}
```

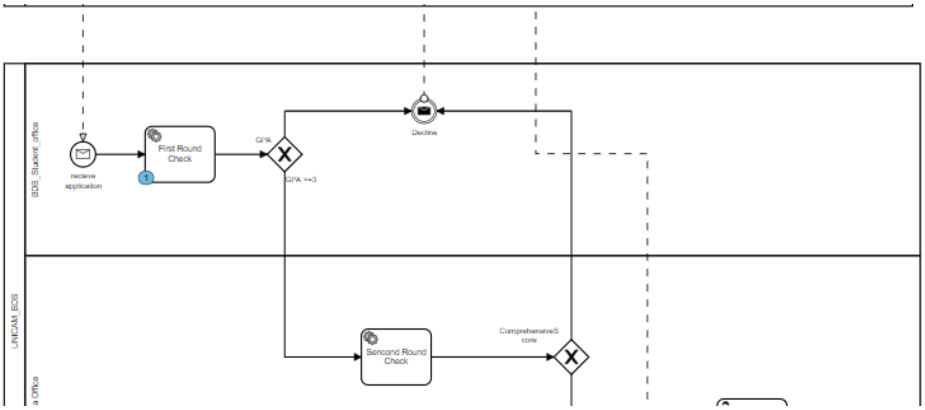


Figure1.2 Process Start

Then the external worker subscribe the topic and check the application data, the subscribe succeed info is shown in VS console as **✓ subscribed to topic FirstRoundCheck,** after the first round the check, the first task is completed as **✓ completed task f1d39e01-b329-11ea-bb3b-005056c00008**, then the token will based on the data gateway to decide the flow, figure 1.3 shows the external works has subscribed the second round check task **✓ subscribed to topic SecondRoundCheck,** after second round check , the task will be completed **✓ completed task 58948c89-b32a-11ea-bb3b-005056c00008.** Sending notification will be two intermediate message throw event executed as external task running as Java Applications. The token will complete after the running the Java applications, figure1.3 and 1.4 shows the pending token in two message events.

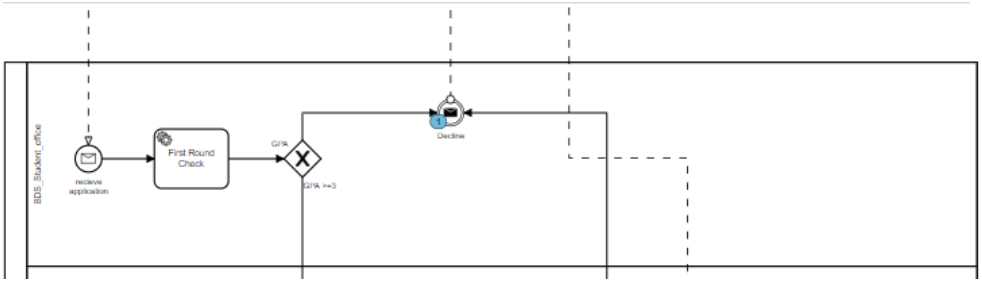


Figure1.3 Decline Toke

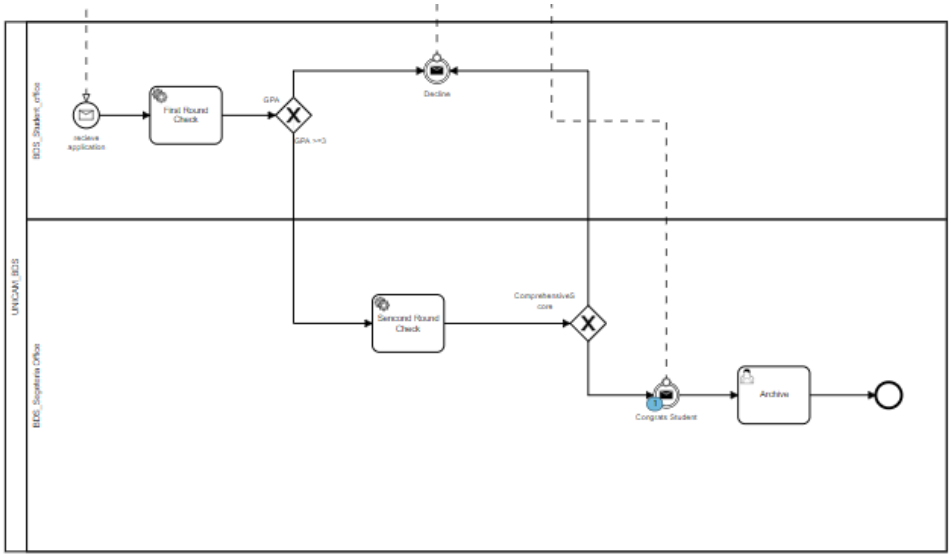


Figure1.4 Congrats Token

Students will get the notification through intermediate catch event , if a student does not meet the requirements , or pass the scholarship qualification checking, a message will be sent. Figure 1.5 and 1.6 show the message after running the notification application.

```
5114 [main] INFO org.camunda.bpm.client - TASK/CLIENT-01026 Discovered data format provider: org.camunda.bpm.
5117 [main] INFO org.camunda.bpm.client - TASK/CLIENT-01025 Discovered data format: org.camunda.bpm.client.va
5117 [main] INFO org.camunda.bpm.client - TASK/CLIENT-01026 Discovered data format provider: org.camunda.bpm.
5118 [main] INFO org.camunda.bpm.client - TASK/CLIENT-01025 Discovered data format: org.camunda.bpm.client.va
5118 [main] INFO org.camunda.bpm.client - TASK/CLIENT-01026 Discovered data format provider: org.camunda.bpm.
5792 [main] INFO org.camunda.bpm.client - TASK/CLIENT-01025 Discovered data format: org.camunda.bpm.client.va
Jun 15, 2020 10:58:18 AM com.camunda.project.SecondRoundCheck lambda$0
INFO: We sorry to inform you that your requirements is not qualified
```

Figure 1.5 Decline Message

```
4911 [main] INFO org.camunda.bpm.client - TASK/CLIENT-01025 Discovered data format: org.camunda.bpm.client.va
4911 [main] INFO org.camunda.bpm.client - TASK/CLIENT-01026 Discovered data format provider: org.camunda.bpm.
5031 [main] INFO org.camunda.bpm.client - TASK/CLIENT-01025 Discovered data format: org.camunda.bpm.client.va
Jun 15, 2020 10:26:59 AM com.camunda.project.CongratsStudent lambda$0
INFO: Congratulations!
Jun 15, 2020 10:29:20 AM com.camunda.project.CongratsStudent lambda$0
INFO: Congratulations!
Jun 15, 2020 11:07:29 AM com.camunda.project.CongratsStudent lambda$0
INFO: Congratulations!
```

Figure 1.6 Congratulation Message

At the end, the system will archive the winners' information as a user task, waiting for the staff in BDS_Segeteria_office to final confirm and archive the file as shown in figure1.7 and 1.8

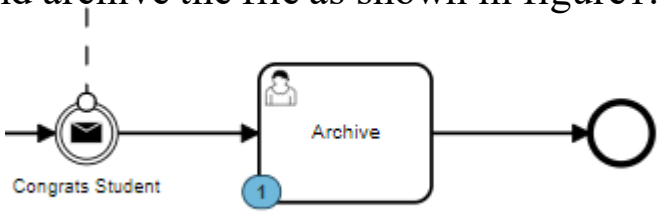


Figure 1.7 Archive User Task

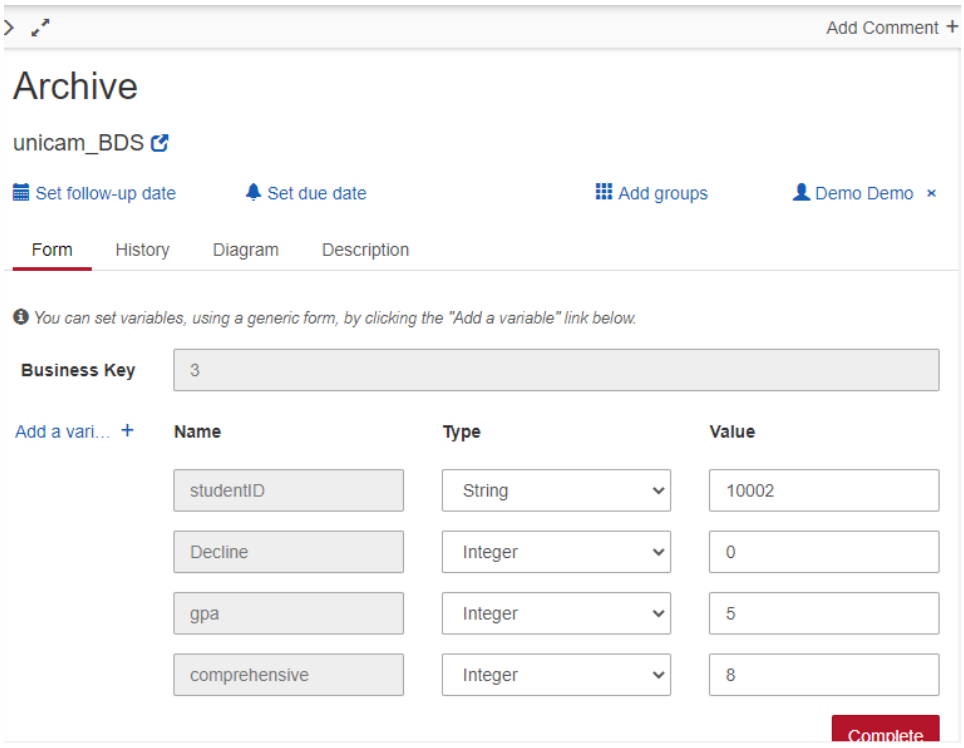


Figure 1.8 Staff Archive and complete the task

Integrate Blockchain in Ganache

A blockchain is a distributed ledger that is structured into a linked list of blocks, each blocks contains an ordered set of transactions, typical solutions use cryptographic hashes to secure the link from a block to its processor, in this project, the blockchain technology is implemented in the business process, the students application information will record in the blocks and store in the blockchain network. To achieve this, we use Web3 JS, a JavaScript library provided by Ethereum, it encapsulates the Remote Procedure Call(RPC) communication API and provides methods to interact with the blockchain, and makes it simple to interact with Ethereum using JavaScript.

Install Web3 package in NodeJs:

```
npm install web3

npm install ethereum-js
```

A smart contract is a program stored and executable in the blockchain written in Solidity Programming language, the smart contract used in this project is :

```
1. // SPDX-License-Identifier: MIT
2. pragma solidity ^0.6.10;
3.
4. contract ApplyScholarship {
5.     struct Task {
```

```
6.          string businessKey;
7.          string name;
8.          string executor;
9.          string additionalInfo;
10.     }
11.
12.     mapping(bytes32 => Task[]) private instances;
13.
14.     function createCollaboration(string memory businessKey)
    public view returns (bytes32 instanceID){
15.         instanceID = keccak256(abi.encode(businessKey,
    block.timestamp));
16.         return instanceID;
17.     }
18.
19.     function registerActivity(bytes32 instanceID, string
    memory businessKey, string memory taskName, string memory
    executor, string memory additionalInfo) public{
20.         instances[instanceID].push(Task(businessKey ,taskName,
    executor, additionalInfo));
21.     }
22. }
```

To get the smart contract data, it is compiled in Remix Ethereum, and generates the bytecode as the further transaction data. The http provider we use is Ganache CLI **http://127.0.0.1:8545.** **Using** Ganache, an account address and private key is generated:

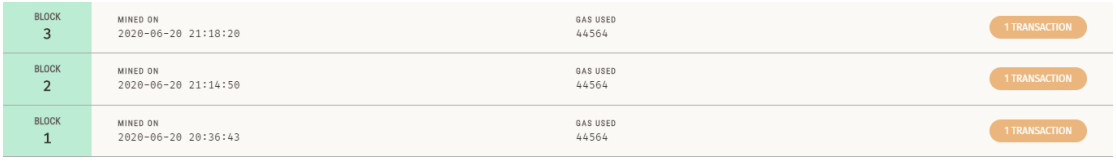
Account:0xa20be716f842e587Fe449eE75EE97765A1314c94

The application process is considered as a transaction , once a student applies online, and the application info is record in the system, a transaction presented as **txHash** will run automatically and record in the block

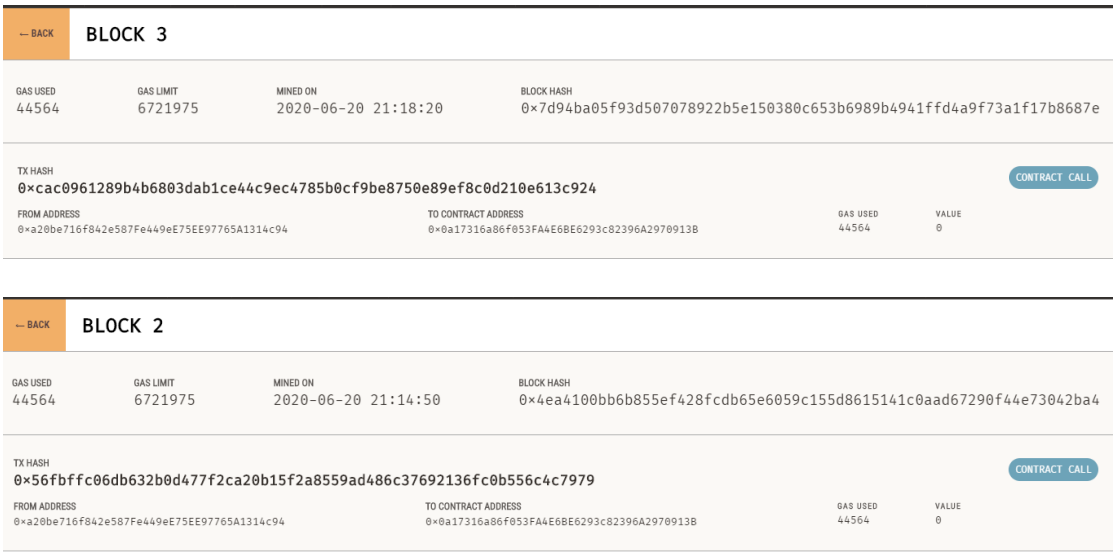
```
PS G:\ESI PROJECT\external task> node .\firstcheck.js
✓ subscribed to topic FirstRoundCheck
err null txHash: 0x56fbffc06db632b0d477f2ca20b15f2a8559ad486c37692136fc0b556c4c7979
✓ completed task f1d39e01-b329-11ea-bb3b-005056c00008
```

```
PS G:\ESI PROJECT\external task> node .\secondchek.js
✓ subscribed to topic SecondRoundCheck
✓ completed task 58948c89-b32a-11ea-bb3b-005056c00008
err null txHash: 0xcac0961289b4b6803dab1ce44c9ec4785b0cf9be8750e89ef8c0
d210e613c924
```

in Ganache UI, we are able to observe the details of transaction



Each transaction is recorded in a new block



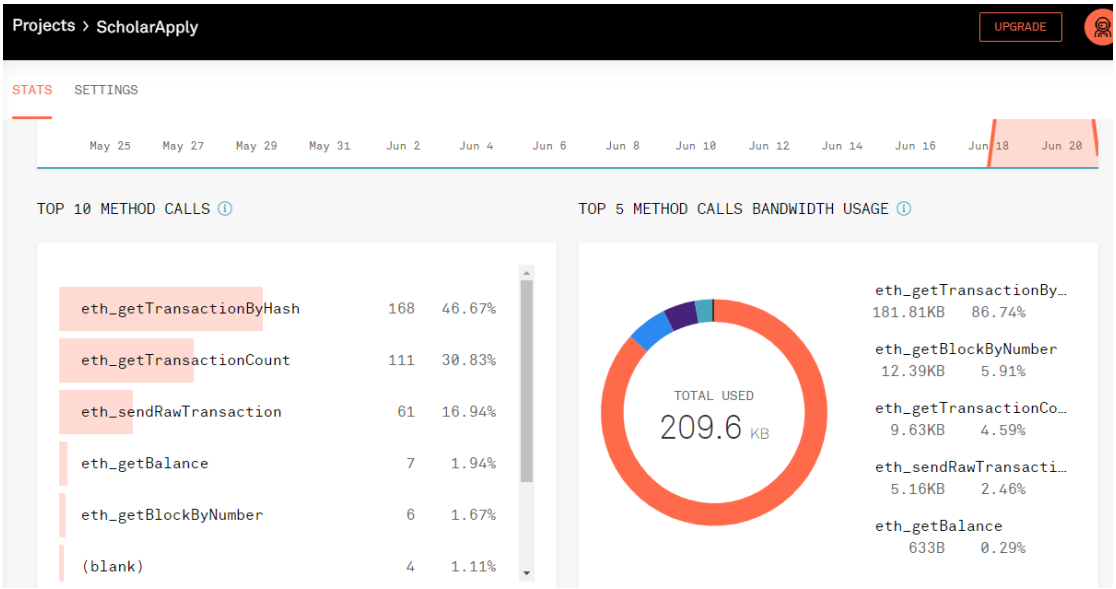
1. err null txHash: 0x56fbffc06db632b0d477f2ca20b15f2a8559ad486c37692136fc0b556c4c7979
2. err null txHash: 0xcac0961289b4b6803dab1ce44c9ec4785b0cf9be8750e89ef8c0d210e613c924

The data used as transaction is recorded in the block. Behind these transactions, the Web3 Ethereum is considered as the topic and business logic integrated in the Camunda BPMN External Worker, once the topic is subscribed, all the processes are automatically generated.

Integrate Blockchain in Infura Rinkeby

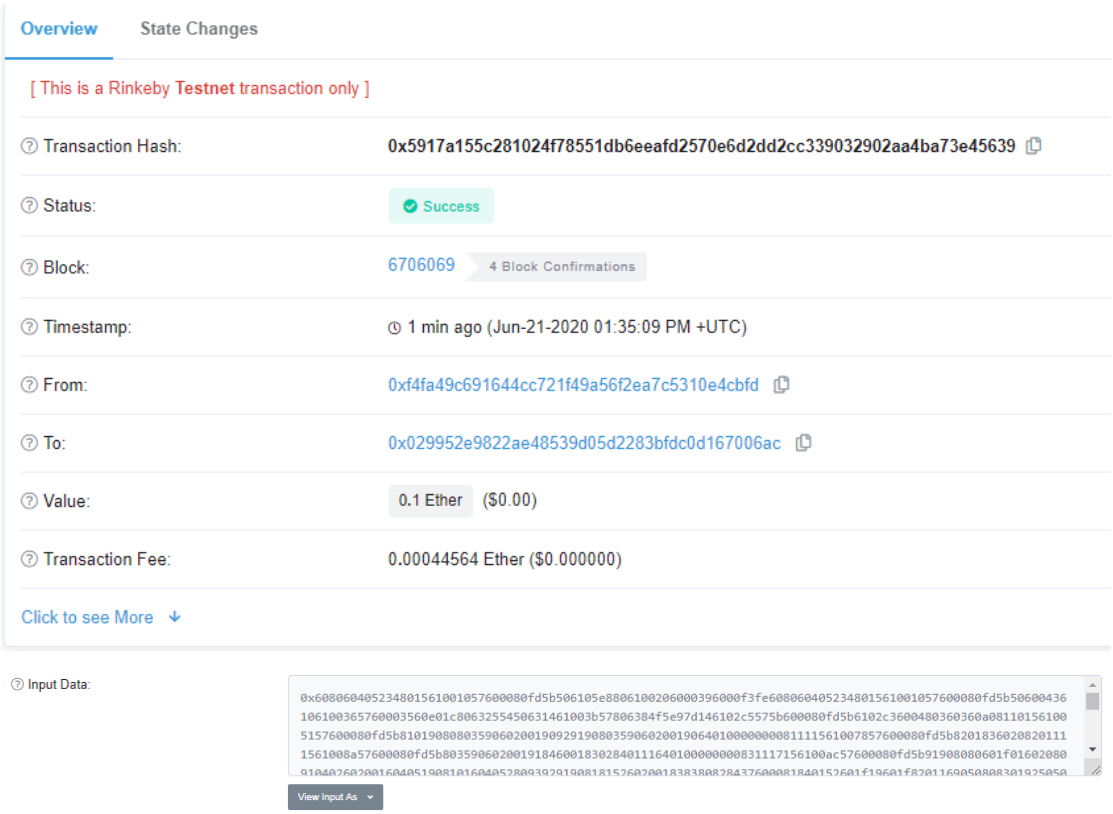
In the aforementioned chapter, the transaction is achieved in localhost, therefore in the chapter, the student application transaction is broadcast to Etherscan Rinkeby Testnet Network. First of all, we use infura.io as HttpProvider in Web3 JS.

https://rinkeby.infura.io/v3/912db4e761714175adda50403cac5bfc



MetaMask as a sender sends the transaction to address: **0x029952e9822Ae48539d05d2283bfDc0d167006aC** containing the data of smart contract. Once student apply through Rest API, the task topic will be subscribed and based on the data and gateway, task will be completed and send the transaction to the address as a block in Etherscan network automatically

```
PS G:\ESI PROJECT\external task> node .\firstcheck.js
✓ subscribed to topic FirstRoundCheck
✓ completed task b25571b0-b3c3-11ea-bd65-005056c00008
✓ completed task 43348a9c-b3c4-11ea-bd65-005056c00008
err null txHash: 0x73f1c577cfe6615caed69aad3285588b586c16d5c57d27d0f1067f36262a812e
```



Conclusion

In the project, we implemented integration between Camunda BPM and Blockchain technology with a Scholarship Application scenario. This integration is achieve in Camunda External worker with Camunda Rest Web Service, blockchain as part of business logic, run together with the process topic subscription with the use of Web3 JS connecting Ethereum test network port 8545.

Programming language in the project: JavaScript, Java, Solidity

Camunda Engine Endpoint: **localhost:8080/engine-rest**

Ethereum testrpc: **http://127.0.0.1:8545**

Transaction Sender:

0xa20be716f842e587Fe449eE75EE97765A1314c94

Transaction Receiver:

0x0a17316a86f053FA4E6BE6293c82396A2970913B

In Rinkeby Testnet network:

Http host:

https://rinkeby.infura.io/v3/912db4e761714175adda50403cac5bfc

Sender Account:

0xF4Fa49C691644Cc721f49A56F2ea7c5310E4Cbfd

Receiver Address:

0x029952e9822Ae48539d05d2283bfDc0d167006aC