Hyperledger Voting Platform

PlasticTwist – An open platform for plastic lifecycle awareness, monetization and sustainable innovation

Thomas Sermpinis – Oswinds AUTh Team

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Tel** +30 6978089151 | serbinistom007@gmail.com | https://cr0wsplace.com |
|  |  |  |

Contents

[Introduction 1](#_Toc534286473)

[Overall Description 2](#_Toc534286474)

[Platform Characteristics 4](#_Toc534286475)

[Installation and Execution Instructions 6](#_Toc534286476)

[Ethereum Alternative Implementation 7](#_Toc534286477)

[Comparison 8](#_Toc534286478)

[Contact Information 9](#_Toc534286479)

# Introduction

## Purpose

The purpose of this document is to describe the exact specifications of the Hyperledger voting platform developed under the Horizon 2020 programme, PTwist, in order to support and expand the capabilities of the system.

## Project Scope

This project includes decentralized voting capabilities to support the decentralized addition of new organizations in the final Plastic Twist network. It uses the blockchain technology to exploit its capabilities and the Hyperledger Fabric framework, to be easily applicable to the initial system created by Almerys. The voting system was developed with several aspects in mind, including:

* Interoperability: The system must work with other systems in the Plastic Twist platform
* Ease of use: Non-technical oriented users (like the pilots of Plastic Twist network) must be able to use the voting platform with ease with all the technical processes happening in the background.
* Decentralization: The voting system must be completely decentralized, with no central authorities managing the votes or the new organization addition process.
* Immutability: Data like votes and new organization addition transactions have to remain in the blockchain with no way of modification, for users to be able to query the ledger and be sure for the authenticity and integrity of the voting process.

## References

* Platform GitHub Repo: https://github.com/Cr0wTom/New-Org-Voting-Platform

# Overall Description

A close up of a map

Description generated with very high confidence  
Figure 1: The flow of the processes taking place in the voting platform by different organizations of the consortium.

The Plastic Twist voting platform follows a specific process to achieve the decentralized voting characteristics, which is based on the flow chart of Figure 1. The process goes as follows:

1. Organizations of the consortium navigate to the web interface of the platform to vote for new organization addition.
2. Organizations vote yes or no in the available votings of the network.
   1. If the vote is no, there is no further process.
   2. If the vote is yes, the new policy gets signed and transferred to the next organization, by the *New Org add script* as seen in Figure 1.
3. After the completion of the voting by the majority of the consortium members, the node setup of the organization that has been chosen by the consortium (Organization 3 in Figure 1), automatically executes a script that signs the New Policy document and makes a transaction for the new policy to get executed in the network and be distributed to the members of the consortium.

Also, there is a process for new votings creation, where users navigate to the Votings page of the web interface to create a new voting or query the available votings that are active. The process is the following:

1. The Organization that needs to start a new voting for a new organization addition to the consortium, navigates to the active\_votings.html page of the platform (more information in the Platform Characteristics section) and fills the information of the new voting he wants to create.
2. A new transaction is generated automatically using the NodeJS API which states the characteristics of the new voting.
3. Now, users that want to vote for this voting, navigate to the vote page of the platform and select the new option from the drop-down menu of the available votings.

# Platform Characteristics

The platform described in this report has the following characteristics:

* Organization representatives (consortium members of the Plastic Twist program) can use their already owned private key obtained for the platform to:
  + Create new votings for new organization addition in the consortium.
  + Vote for new organization addition in the consortium, by selecting one of the active votings.
  + Query the blockchain for past voting results.
  + Query the blockchain for active voting statistics.
* The platform manages new votings and votes with a smart contract that applies business logic to the decisions that are made, with no need for a central authority.
* The platform adds the new organization automatically when the voting finishes.

The platform consists of two basic components:

* Web Interface: The web interface is the part that users interact with. It comes with easy to understand ui elements for voting and querying and communicates with the backend system via NodeJS code implemented in the background. The web interface comes with the pages described in Table 1 bellow.

*Table 1:* The pages that make up the web interface of the Plastic Twist voting platform.

|  |  |
| --- | --- |
| *index.html* | Selection of the voting addition of the vote page of the platform. |
| *active\_votings.html* | New voting creation by supplying the ID and the name of the new voting, and querying of the available votings that exist in the blockchain. |
| *voting.html* | Vote for new organization addition by the user by supplying the voter name and the voting that he wants to vote in. Also, query of the votes in the blockchain and chart creation with statistics of the voting selected. Votings get selected from a drop-down menu which gets filled by automatically querying the blockchain for active votings at the time of page load. |

* Backend System: The backend system uses Hyperledger fabric to communicate with the blockchain, make new transactions and query. It communicates using the NodeJS API, with a smart contract written in Go for business logic implementation. The parts of the backend system can be seen in Table 2 bellow.

*Table 2:* The technical parts of the Plastic Twist platform.

|  |  |
| --- | --- |
| *create\_voting.js* | new voting creation by the user, by calling the createVoting() function of the smart contract |
| *query\_votings.js* | query of available votings by the user, by calling the queryAllVotings() function of the smart contract. This function is also called by the voting page of the web interface automatically, for the correct votings to be selected by the user. |
| *query.js* | query of the votes happened in the blockchain of a certain active voting selected by the user. |
| *vote.js* | vote (yes or no) for new organization addition in the consortium network. |
| *neworgadd.sh* | after a user of the consortium votes yes in an active voting the neworgadd.sh script gets executed automatically, in order to sign the new transaction of the addition of the new organization with the private key of the user that voted yes. |

# Installation and Execution Instructions

To execute the network for testing purposes there is a process that must be followed. With **bold** letters, the exact commands to be executed are presented in the following instructions:

1. If this is the first time of execution the prerequisites must be installed by executing **./prereqs-ubuntu.sh** in an ubuntu machine.
2. The folder of the repo has to be placed in the *fabric-samples* folder and the **voting.go** file has to be place in the folder *fabric-samples/chaincode/voting.*
3. To install all the required files of NodeJs we execute **npm install**
4. To start the network entities required for the platform, we execute **./startFabric.sh**
5. To create an admin identity for the network, we execute **node enrollAdmin.js**
6. To create a second entity that will create transactions in the network, we execute **node registerUser.js**
7. To start the NodeJS server that the Web platform needs in order to communicate with the fabric network, we execute **node app.js** and we leave it running in the background.
8. We now open the index.html with a browser and navigate through the different available options of the platform. The .js files described in the Platform characteristics section can be also used using the node command in order to perform transactions and platform specific tasks with a terminal window.

# Ethereum Alternative Implementation

An alternative implementation has been developed using smart contracts in the Ethereum platform, written in Solidity (<https://github.com/Cr0wTom/Ethereum-Voting>). The purpose of this implementation is to find the weak spots of the original Hyperledger platform and compare the two implementations to find the possible use cases for each of them.

**Setting up an Ethereum Private Network**

To be able to compare an Ethereum implementation of our voting system with our Hyperledger implementation we have to implement it in a private network. This is possible in the Ethereum blockchain but there are some steps that must be followed before executing our smart contract.

The tools needed for this project are the following:

* **Homebrew**: <https://brew.sh/>
* **Node and npm:** <https://nodejs.org/en/>
* **Geth (go-etheruem):**
  + brew tap ethereum/Ethereum
  + brew install ethereum
* **Truffle:**
  + sudo npm install -g truffle
* **Solidity compiler:** 
  + sudo npm install -g solc

To start we must generate the genesis block and initiate the chain data in the blockchain. To do so we start by executing the following:

* *mkdir voting*
* *cd voting*
* *touch genesis.json*

We now open the genesis.json file and paste the following info:

{

"config": {

"chainId": 143,

"homesteadBlock": 0,

"eip155Block": 0,

"eip158Block": 0

},

"alloc": {},

"difficulty" : "0x20000",

"gasLimit" : "0x8880000"

}

We continue by creating the blockchain folder by executing:

* *mkdir blockchain*
* *geth --datadir blockchain init genesis.json*

Now that we have the configuration ready we execute the following steps (commands that need execution in *Italics*):

1. To start the private Ethereum blockchain we execute the following command inside the voting folder:
   * *geth --port 3000 --networkid 58343 --nodiscover --datadir=./blockchain --maxpeers=0 --rpc --rpcport 8543 --rpcaddr 127.0.0.1 --rpccorsdomain "\*" --rpcapi "eth,net,web3,personal,miner"*
2. To connect to our private Ethereum network using the Geth Javascript console we execute the following command in the voting folder:
   * *geth attach* [*http://127.0.0.1:8543*](http://127.0.0.1:8543)
3. No in the console we have to create and account and mine for Ether in our blockchain:
   * *personal.newAccount('seed')*
   * *personal.unlockAccount(web3.eth.coinbase, "seed", 15000)*
   * *miner.start()*
4. Initialize Truffle to be able to deploy our smart contract in our network. In the voting folder we execute:
   * *mkdir truffle*
   * *cd truffle*
   * *truffle init*
5. We update the truffle.js file with our account and network details that we used earlier. (We can run *personal.listAccounts[0]* to list these information again)
6. We place the contract in the contracts folder of the truffle directory.
7. We place the 2\_deploy\_contracts.js file in the “migrations” folder of the “truffle” directory.
8. We compile and deploy the contract with the following commands:
   * *truffle compile*
   * *truffle migrate*

# Comparison

To compare the two systems, we set some categories and we evaluated them for each solution.

* Consensus
* Speed
* Customizability
* Ease of integration
* Support

# Contact Information

**Thomas Sermpinis – Oswinds AUTh Team**

**Tel** +30 6978089151

https://cr0wsplace.com

serbinistom007@gmail.com