Maha Hamza Glsi-A

Compte-rendu tp1 Ethereum

Use Case: Electronic vote

Step One: Installing Dependencies :

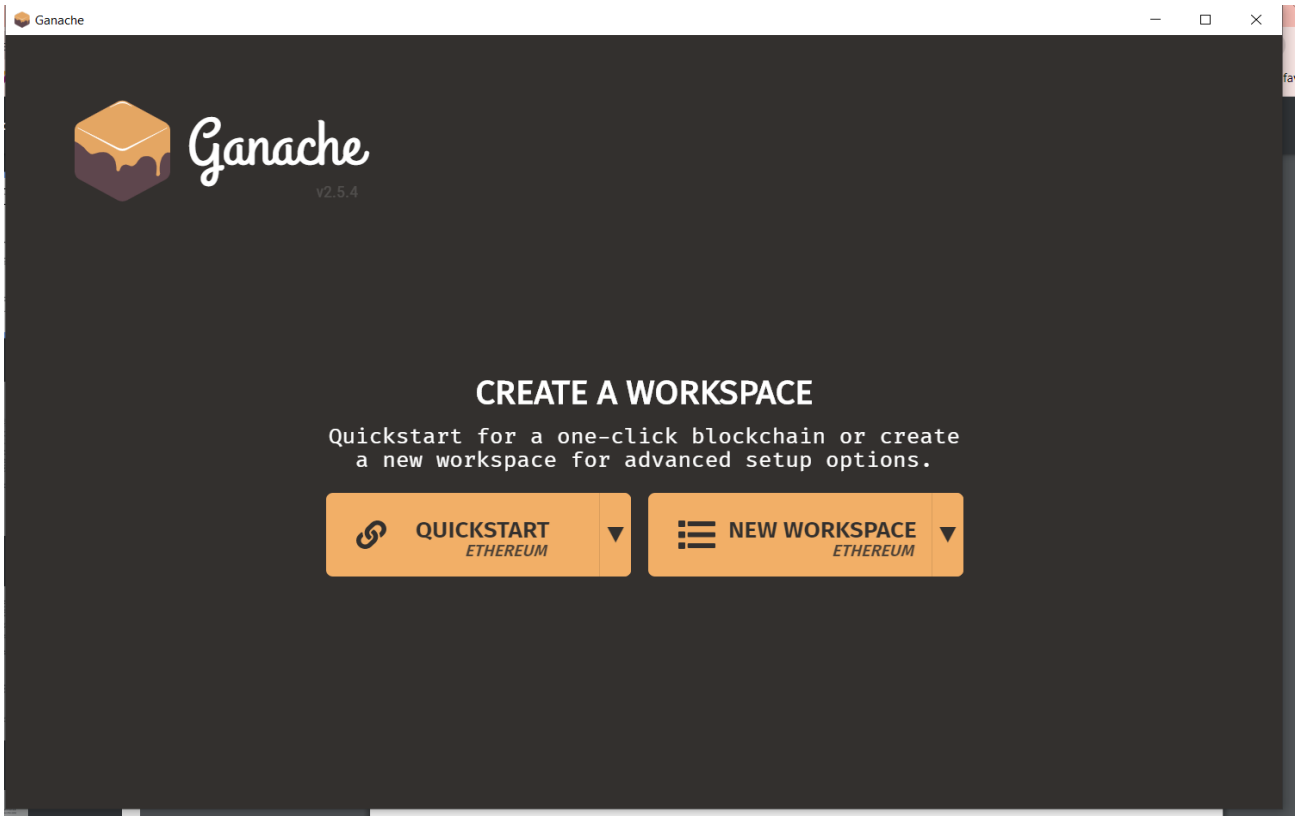
Truffle :

```
npm install -g truffle
```

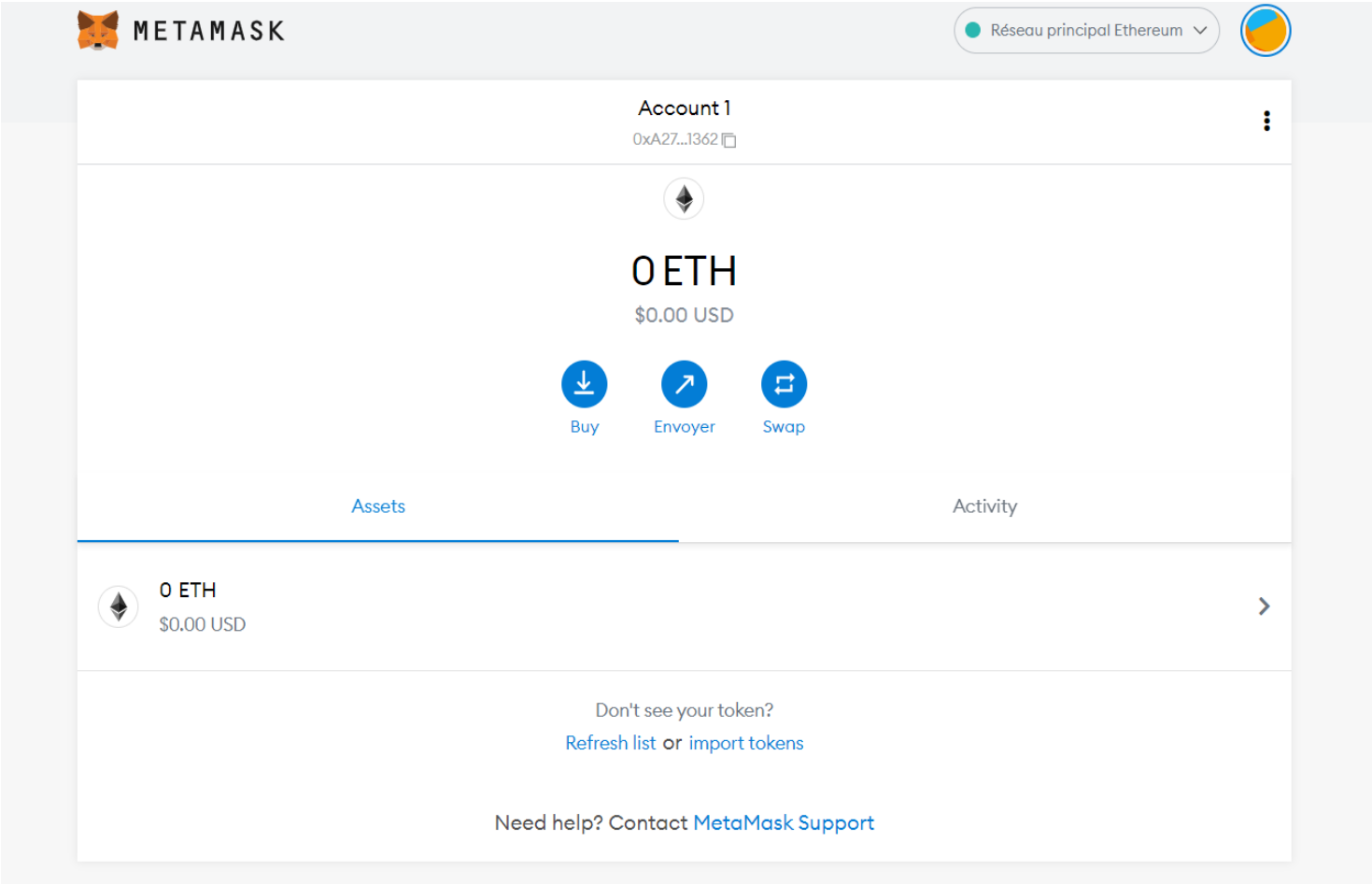
C:\Windows\System32\cmd.exe

```
C:\Users\lenovo\tpBlockchainEthereum>npm i -g truffle
npm WARN deprecated graphql-tools@6.2.6: This package has been deprecated and now it only exports makeExecutableSchema.\
nAnd it will no longer receive updates.\nWe recommend you to migrate to scoped packages such as @graphql-tools/schema, @
graphql-tools/utils and etc.\nCheck out https://www.graphql-tools.com to learn what package you should use instead
npm WARN deprecated uuid@2.0.1: Please upgrade  to version 7 or higher.  Older versions may use Math.random() in certain
 circumstances, which is known to be problematic.  See https://v8.dev/blog/math-random for details.
npm WARN deprecated mkdirp-promise@5.0.1: This package is broken and no longer maintained. 'mkdirp' itself supports prom
ises now, please switch to that.
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated uuid@3.3.2: Please upgrade  to version 7 or higher.  Older versions may use Math.random() in certain
 circumstances, which is known to be problematic.  See https://v8.dev/blog/math-random for details.
npm WARN deprecated multicodec@0.5.7: This module has been superseded by the multiformats module
npm WARN deprecated cids@0.7.5: This module has been superseded by the multiformats module
npm WARN deprecated multicodec@1.0.4: This module has been superseded by the multiformats module
npm WARN deprecated multibase@0.6.1: This module has been superseded by the multiformats module
npm WARN deprecated multibase@0.7.0: This module has been superseded by the multiformats module
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
npm WARN deprecated @ensdomains/ens@0.4.3: Please use @ensdomains/ens-contracts
npm WARN deprecated @ensdomains/resolver@0.2.4: Please use @ensdomains/ens-contracts
npm WARN deprecated testrpc@0.0.1: testrpc has been renamed to ganache-cli, please use this package from now on.
npm WARN deprecated ethereumjs-testrpc@6.0.3: ethereumjs-testrpc has been renamed to ganache-cli, please use this packag
e from now on.
npm WARN deprecated querystring@0.2.0: The querystring API is considered Legacy. new code should use the URLSearchParams
 API instead.
npm WARN deprecated chokidar@2.1.8: Chokidar 2 will break on node v14+. Upgrade to chokidar 3 with 15x less dependencies
```

Ganache:



Metamask:



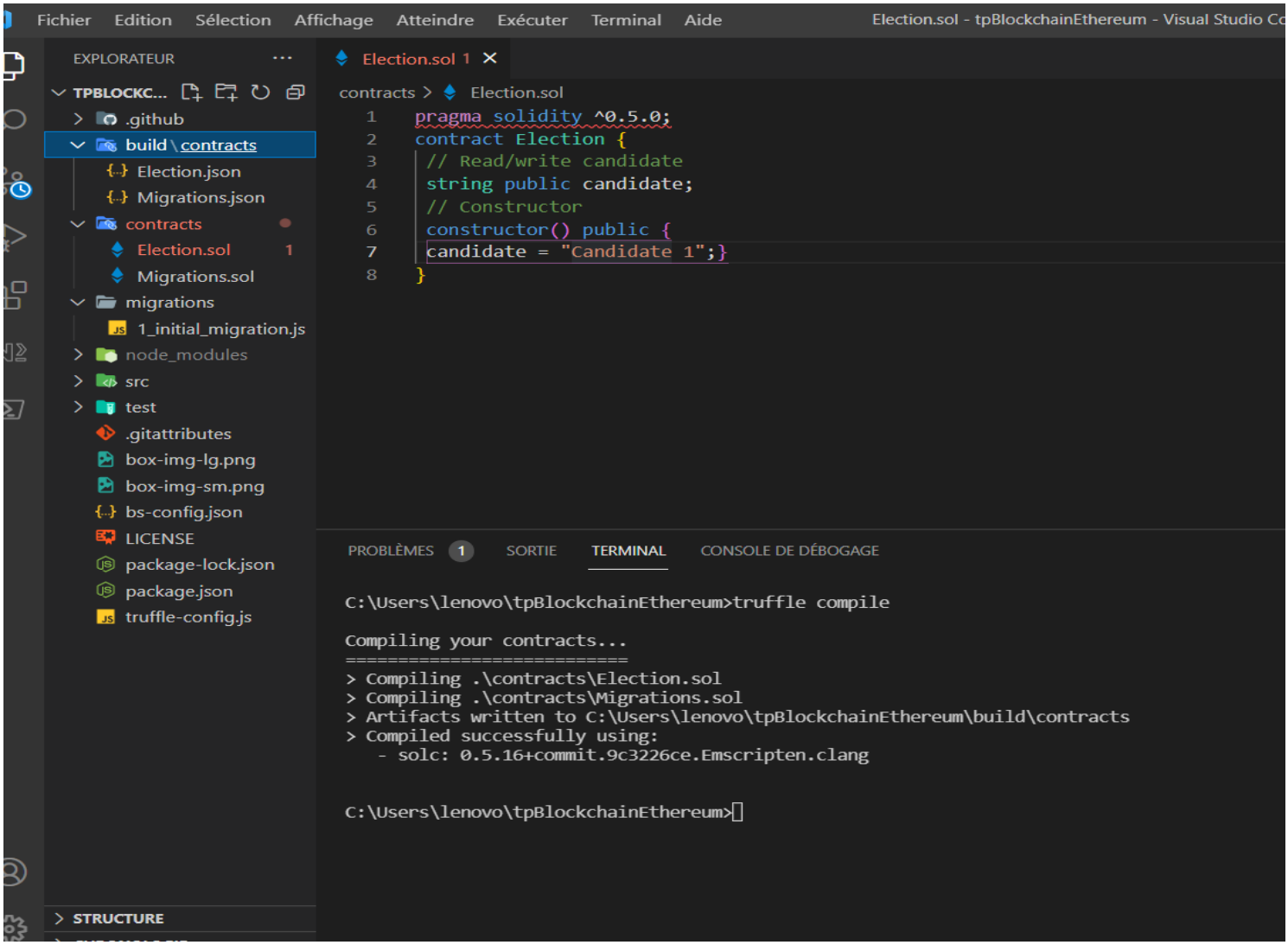
Step two: Creating the project

truffle unbox pet-shop

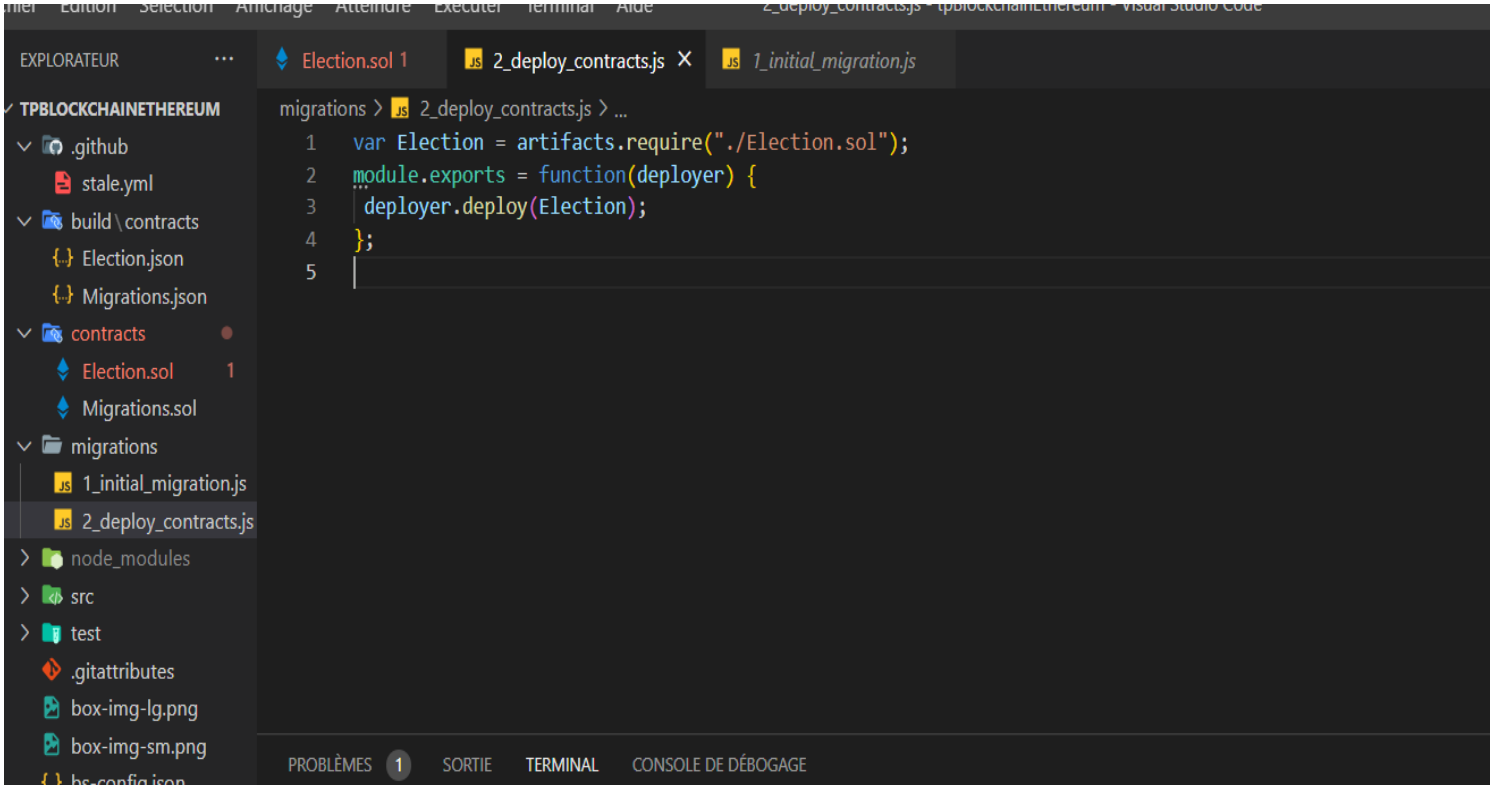
C:\Windows\System32\cmd.exe

```
C:\Users\lenovo\tpBlockchainEthereum>truffle unbox pet-shop

Starting unbox...
=================

√ Preparing to download box
√ Downloading
npm WARN pet-shop@1.0.0 No description
npm WARN pet-shop@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.4 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin","arch":"any
"} (current: {"os":"win32","arch":"x64"})

√ Cleaning up temporary files
√ Setting up box

Unbox successful, sweet!

Commands:

  Compile:        truffle compile
  Migrate:        truffle migrate
  Test contracts: truffle test
```

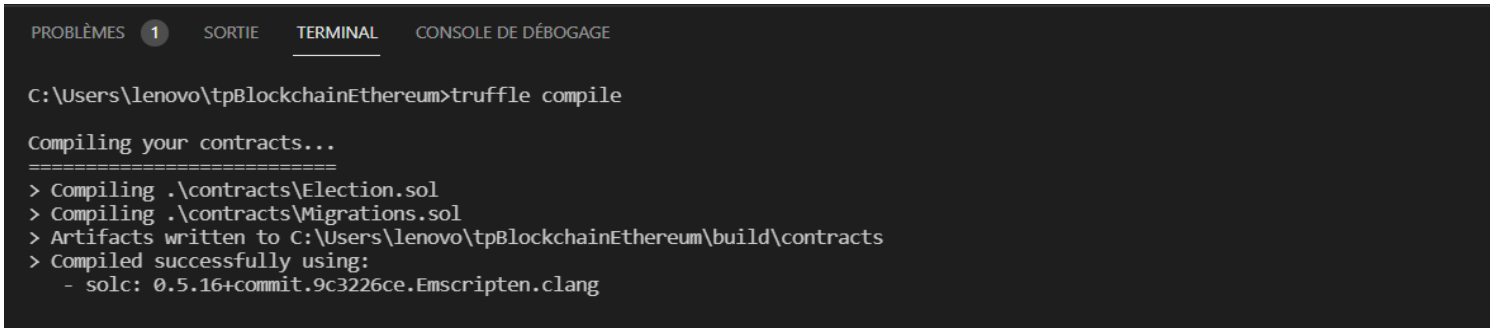
- Election.sol file :

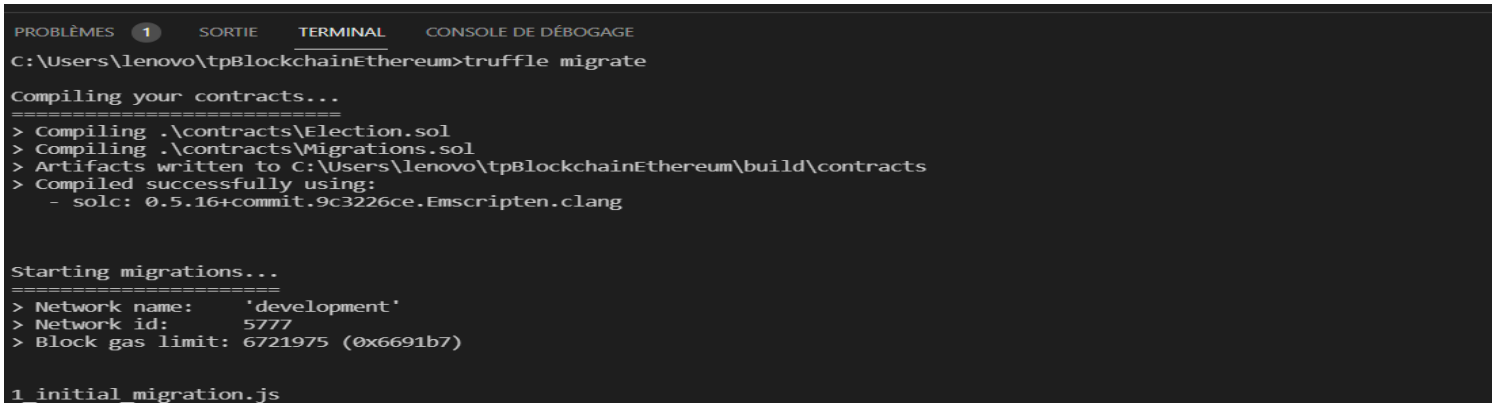


- migrations/2_deploy_contracts.js:



- **truffle compile & truffle migrate:**





- **truffle console :**

```
PROBLÈMES 1   SORTIE   TERMINAL   CONSOLE DE DÉBOGAGE

C:\Users\lenovo\tpBlockchainEthereum>truffle console
truffle(development)> Election.deployed().then(function(instance) { app = instance })
undefined
truffle(development)> app.candidate()
'Candidate 1'
truffle(development)>
```

=> first smart contract

Step 3: Preparing Candidates List

Model "Candidate"

```
EXPLORATEUR
TPBLOCKC...
.github
stale.yml
build\contracts
Election.json
Migrations.json
contracts
Election.sol 1
Migrations.sol
migrations
1_initial_migration.js
2_deploy_contracts.js
node_modules
src
test
.gitattributes
box-img-lg.png
box-img-sm.png
bs-config.json
LICENSE
package-lock.json
package.json
truffle-config.js

Election.sol 1 ×   2_deploy_contracts.js   1_initial_migration.js
contracts > Election.sol

1  pragma solidity ^0.5.0;
2
3  contract Election {
4
5  /*  // Read/write candidate
6   string public candidate; */
7
8  // Read/write Candidates
9   mapping(uint => Candidate) public candidates;
10
11  // Constructor
12  constructor() public {
13
14  candidate = "Candidate 1";
15  }
16  // Model a Candidate
17  struct Candidate {
18  uint id;
19  string name;
20  uint voteCount;
21  }
22  }
23

PROBLÈMES 1   SORTIE   TERMINAL   CONSOLE DE DÉBOGAGE   node

C:\Users\lenovo\tpBlockchainEthereum>truffle console
truffle(development)> Election.deployed().then(function(instance) { app = instance })
undefined
truffle(development)> app.candidate()
'Candidate 1'
```

Read/write Candidates(mapping) + add candidates to the mapping we've created :(Add two candidates to our election by calling the "addCandidate" function twice inside the constructor function) :

```
Election.sol 1 ×
contracts > Election.sol
// Read/write Candidates
 mapping(uint => Candidate) public candidates;
// Store accounts that have voted
mapping(address => bool) public voters;
 // Constructor
 constructor() public {

 //candidate = "Candidate 1";
 addCandidate("Maha");
 addCandidate("Camélia");
 }
 // Model a Candidate
 struct Candidate {
 uint id;
 string name;
 uint voteCount;
}

// Store Candidates Count
 uint public candidatesCount;

function addCandidate (string memory _name) private {
 candidatesCount ++;
 candidates[candidatesCount] = Candidate(candidatesCount, _name, 0);
 }


}
```

migration will execute when we deploy the contract to the blockchain, and populate our election with two candidates :

```
PROBLÈMES 1   SORTIE   TERMINAL   CONSOLE DE DÉBOGAGE

C:\Users\lenovo\tpBlockchainEthereum>truffle migrate

Compiling your contracts...
===========================
> Compiling .\contracts\Election.sol
> Compiling .\contracts\Migrations.sol
> Artifacts written to C:\Users\lenovo\tpBlockchainEthereum\build\contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang


Starting migrations...
======================
> Network name:    'development'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)


1_initial_migration.js
======================

   Deploying 'Migrations'
   ----------------------
```

Step4: Client-Side Application

"index.html" file :

Election.sol 1 | index.html | app.js

src > index.html > html

```html
<!DOCTYPE html><html lang="en">
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Election Results</title>
<!-- Bootstrap -->
<link href="css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
<div class="container" style="width: 650px;"> <div class="row"> <div class="col-lg-12">
<h1 class="text-center">Election Results</h1>
<hr/><br/>
<div id="loader"><p class="text-center">Loading...</p></div>
<div id="content" style="display: none;">
<table class="table">
<thead><tr><th scope="col">#</th><th scope="col">Name</th>
<th scope="col">Votes</th></tr></thead>
<tbody id="candidatesResults"></tbody>
</table>
<hr/>
<p id="accountAddress" class="text-center"></p></div> </div></div></div>
<!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<!-- Include all compiled plugins (below), or include individual files as needed -->
<script src="js/bootstrap.min.js"></script>
<script src="js/web3.min.js"></script>
<script src="js/truffle-contract.js"></script>
<script src="js/app.js"></script>
</body></html>
```

"app.js" file :

Election.sol 1 | index.html | app.js

src > js > app.js > render > then() callback > then() callback > candidateTemplate

```javascript
App = {
  web3Provider: null,
  contracts: {},
  account: '0x0',
  init: function() {
  return App.initWeb3();
  },
  initWeb3: function() {
  if (typeof web3 !== 'undefined') {
  // If a web3 instance is already provided by Meta Mask.
  App.web3Provider = web3.currentProvider;
  web3 = new Web3(web3.currentProvider);
  } else {
  // Specify default instance if no web3 instance provided
  App.web3Provider = new Web3.providers.HttpProvider('http://localhost:7545');
  web3 = new Web3(App.web3Provider);
  }
  return App.initContract();
  },
  initContract: function() {
  $.getJSON("Election.json", function(election) {
  // Instantiate a new truffle contract from the artifact
  App.contracts.Election = TruffleContract(election);
  // Connect provider to interact with contract
  App.contracts.Election.setProvider(App.web3Provider);
  return App.render();
  });
  },
  render: function() {
  var electionInstance;
  var loader = $("#loader");
  var content = $("#content");
  loader.show();
```

truffle migrate –reset :

```
PROBLÈMES 1   SORTIE   TERMINAL   CONSOLE DE DÉBOGAGE
C:\Users\lenovo\tpBlockchainEthereum>truffle migrate --reset

Compiling your contracts...
===========================
> Compiling .\contracts\Election.sol
> Compiling .\contracts\Migrations.sol
> Artifacts written to C:\Users\lenovo\tpBlockchainEthereum\build\contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang



Starting migrations...
======================
> Network name:    'development'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)


1_initial_migration.js
======================

   Replacing 'Migrations'
   ----------------------
   > transaction hash:    0xb61d3603bc455a49ca6b79165a48f7ffbc44fe958aa74a3cd9e09beb8e69ed88
   > Blocks: 0            Seconds: 0
   > contract address:    0xf2C1B3D701C45bc8f5Cf7a1b2255F51e21704AFa
```

start development server from the command : npm run dev :

```
PROBLÈMES 1   SORTIE   TERMINAL   CONSOLE DE DÉBOGAGE

C:\Users\lenovo\tpBlockchainEthereum>npm run dev

> pet-shop@1.0.0 dev C:\Users\lenovo\tpBlockchainEthereum
> lite-server

** browser-sync config **
{
  injectChanges: false,
  files: [ './**/*.{html,htm,css,js}' ],
  watchOptions: { ignored: 'node_modules' },
  server: {
    baseDir: [ './src', './build/contracts' ],
    middleware: [ [Function], [Function] ]
  }
}
[Browsersync] Access URLs:
 --------------------------------
  Local: http://localhost:3000
 --------------------------------
     UI: http://localhost:3001
 --------------------------------
[Browsersync] Serving files from: ./src
[Browsersync] Serving files from: ./build/contracts
[Browsersync] Watching files...
21.11.30 02:15:33 200 GET /index.html
21.11.30 02:15:33 200 GET /js/bootstrap.min.js
```

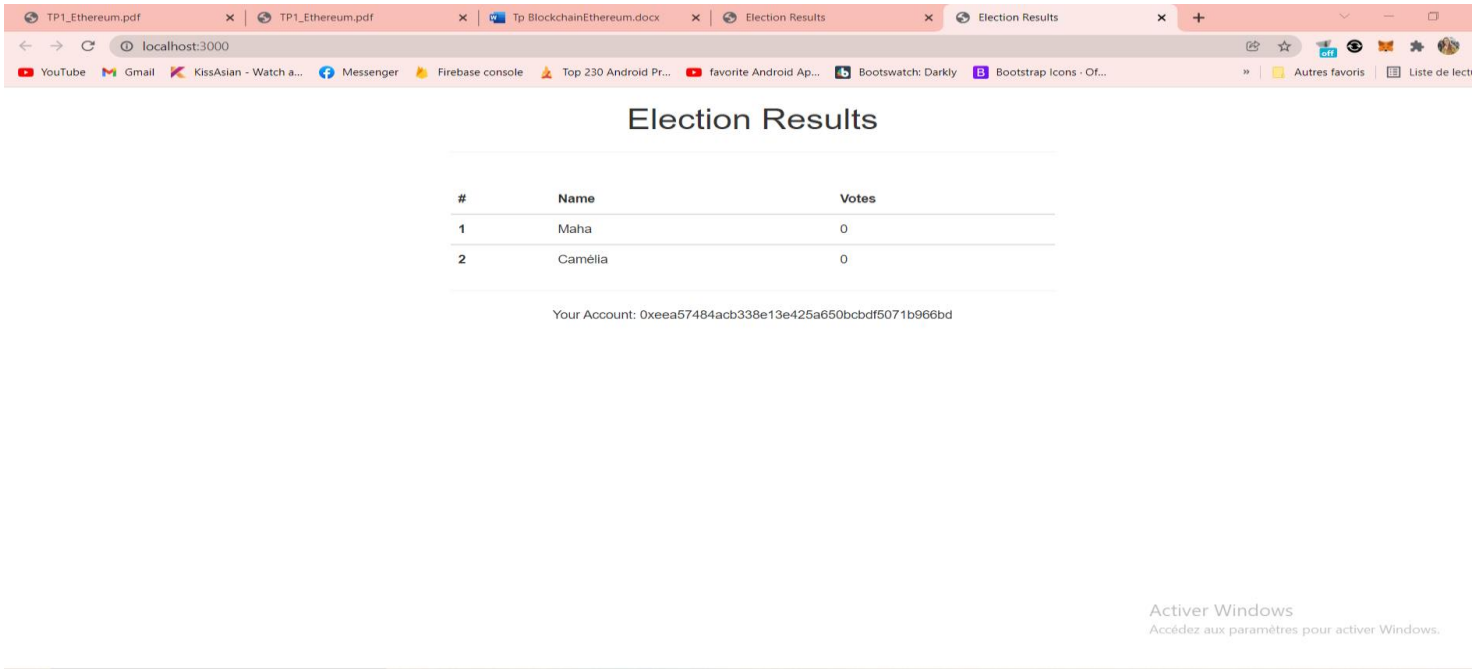
_Ethereum.pdf | MetaMask | TP1_Ethereum.pdf | tekupde-my.sharepoint.com | Election Results

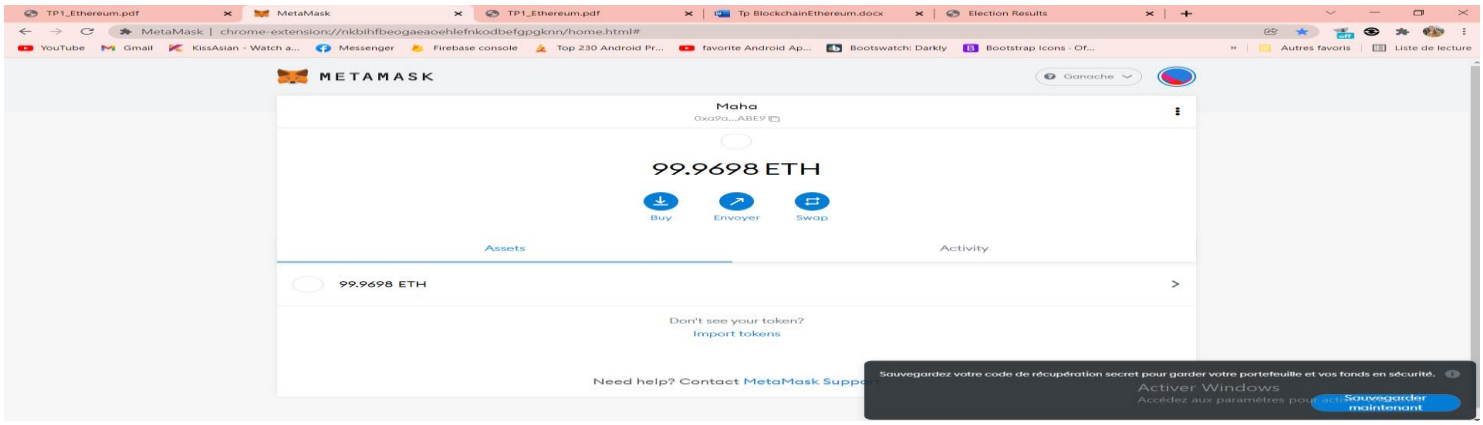
localhost:3000

Gmail | KissAsian - Watch a... | Messenger | Firebase console | Top 230 Android Pr... | favorite Android Ap... | Bootswatch: Darkly | Bootstrap Icons · Of... | Autres favoris | Liste de lect

Election Results

Loading...





Step5: Cast Votes

Store accounts that have voted:

```
// Store accounts that have voted
mapping(address => bool) public voters;
```

add a "vote" function:

```
Election.sol 1 ×   index.html   app.js
contracts > Election.sol
    candidatesCount ++;
    candidates[candidatesCount] = Candidate(candidatesCount, _name, 0);
    }

    function vote (uint _candidateId) public {
    // require that they haven't voted before
    require(!voters[msg.sender]);
    // require a valid candidate
    require(_candidateId > 0 && _candidateId <= candidatesCount);
    // record that voter has voted
    voters[msg.sender] = true;
    // update candidate vote Count
    candidates[_candidateId].voteCount ++;
    }
    }
    }
```

Step5: Client-Side Voting

"index.html" file (ajouter un formulaire):

```
Election.sol 1   index.html ×   app.js
src > index.html > html > body > form
<!DOCTYPE html><html lang="en">
  <head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Election Results</title>
  <!-- Bootstrap -->
  <link href="css/bootstrap.min.css" rel="stylesheet">
  </head>
  <body>
      <form onSubmit="App.castVote(); return false;">
          <div class="form-group">
          <label for="candidatesSelect">Select Candidate</label>
          <select class="form-control" id="candidatesSelect">
          </select>
          </div>
          <button type="submit" class="btn btn-primary">Vote</button>
          <hr />
```

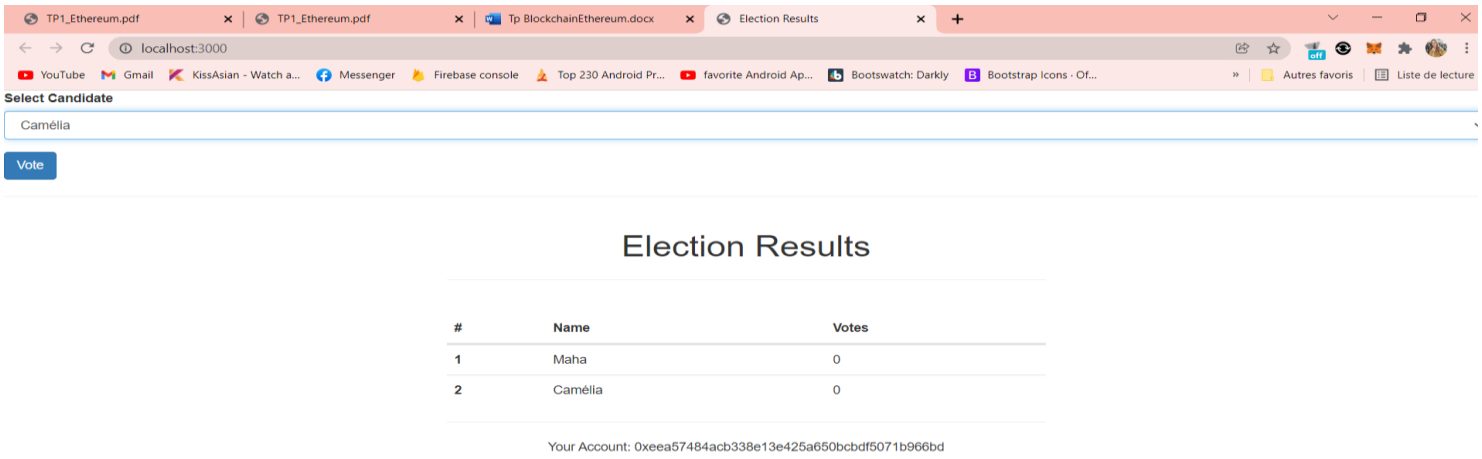
. We create the form with an empty select element. We will populate the select options with the candidates provided by our smart contract in our "app.js" file.

The form has an "onSubmit" handler that will call the "castVote" function. We will define this in our "app.js" file

Election.sol 1 | index.html | app.js

src > js > app.js > ...

```javascript
render: function() {
  var electionInstance;
  var loader = $("#loader");
  var content = $("#content");
  loader.show();
  content.hide();
  // Load account data
  web3.eth.getCoinbase(function(err, account) {
  if (err === null) {
  App.account = account;
  $("#accountAddress").html("Your Account: " + account);
  }
  });
  // Load contract data
  App.contracts.Election.deployed().then(function(instance) {
  electionInstance = instance;
  return electionInstance.candidatesCount();
  }).then(function(candidatesCount) {
  var candidatesResults = $("#candidatesResults");
  candidatesResults.empty();
  var candidatesSelect = $('#candidatesSelect');
  candidatesSelect.empty();
  for (var i = 1; i <= candidatesCount; i++) {
  electionInstance.candidates(i).then(function(candidate) {
  var id = candidate[0];
  var name = candidate[1];
  var voteCount = candidate[2];
  // Render candidate Result
  var candidateTemplate = "<tr><th>" + id + "</th><td>" + name + "</td><td>" + voteCount + "</td></tr>"
  candidatesResults.append(candidateTemplate);
```



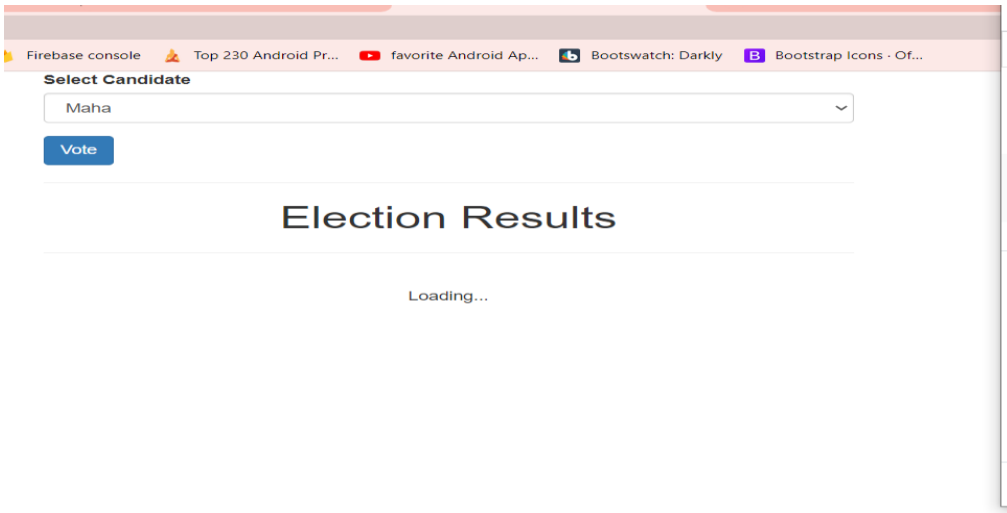
Next, we want to write a function that's called whenever the form is submitted:

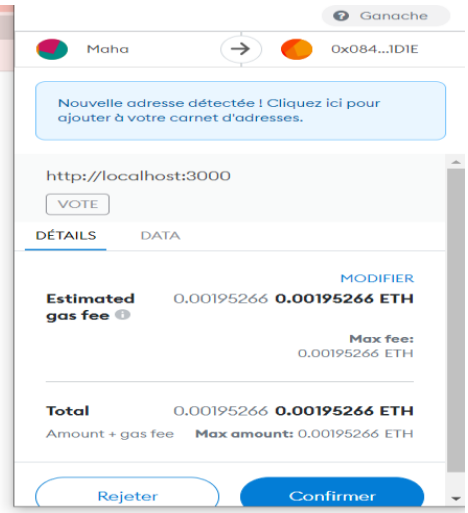
```
Election.sol 1    index.html    app.js
src > js > app.js > render
29  > render: function() {···
75      },
76      castVote: function() {
77        var candidateId = $('#candidatesSelect').val();
78        App.contracts.Election.deployed().then(function(instance)
79        {
80        return instance.vote(candidateId, { from: App.account })
81        ;
82        }).then(function(result) {
83        // Wait for votes to update
84        $("#content").hide();
85        $("#loader").show();
86        }).catch(function(err) {
87        console.error(err);
88        });
89        }
90
91    };
92    $(function() {
93    $(window).load(function() {
94    App.init();
95    });
96    });
```

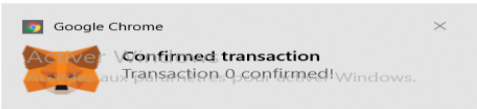
Then we migrate :

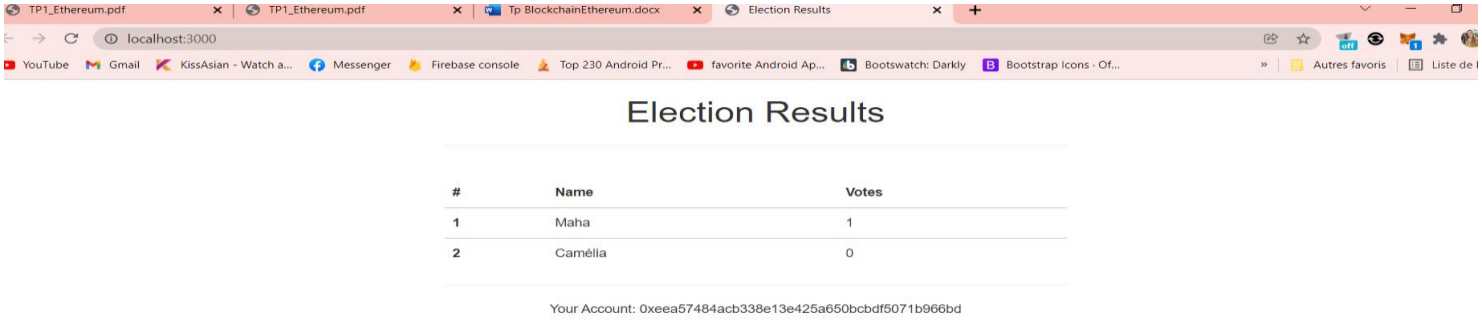
```
PROBLÈMES 1    SORTIE    TERMINAL    CONSOLE DE DÉBOGAGE

C:\Users\lenovo\tpBlockchainEthereum>truffle migrate --reset
```

Then we lance the server & vote :

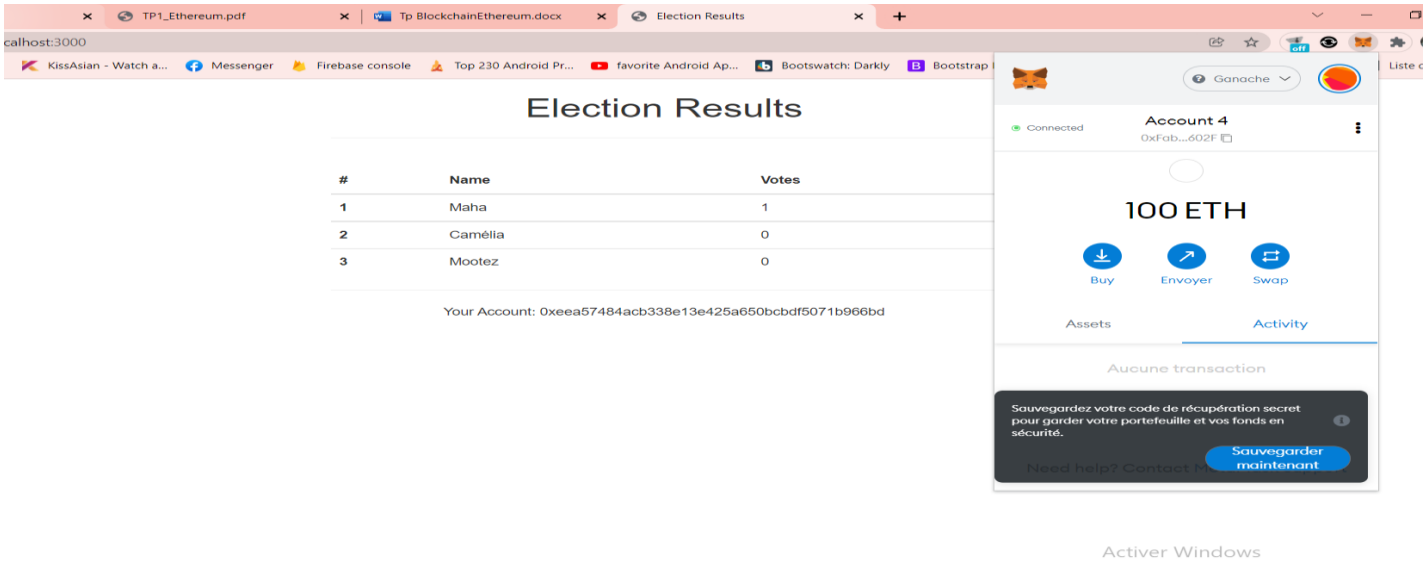


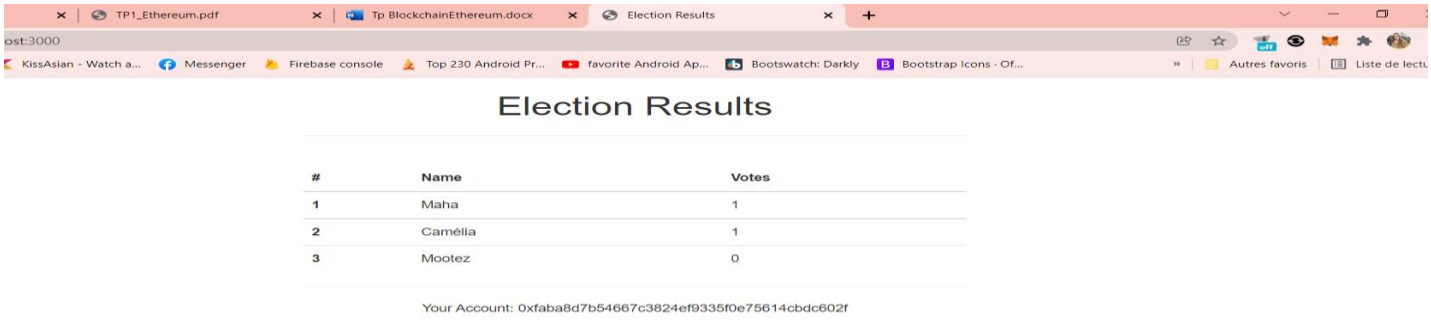






Try with an other account :





Step 6 : Watch Events

We define an event variable:

```
Election.sol 1 ×   index.html   app.js
contracts > Election.sol
    string name;
    uint voteCount;
  }
  event votedEvent (
    uint indexed _candidateId
    );

```

Now we can trigger this "voted" event inside our "vote" function :

```
Election.sol 1 ×   index.html   app.js
contracts > Election.sol
  }
  function vote (uint _candidateId) public {
    // require that they haven't voted before
    require(!voters[msg.sender]);
    // require a valid candidate
    require(_candidateId > 0 && _candidateId <= candidatesCount);
    // record that voter has voted
    voters[msg.sender] = true;
    // update candidate vote Count
    candidates[_candidateId].voteCount ++;

  // trigger voted event
     emit votedEvent(_candidateId);
  }

  }
```

update the client-side application to listen for the voted event and fire a page refresh any time that it is triggered. We can do that with a "listenForEvents" function ,

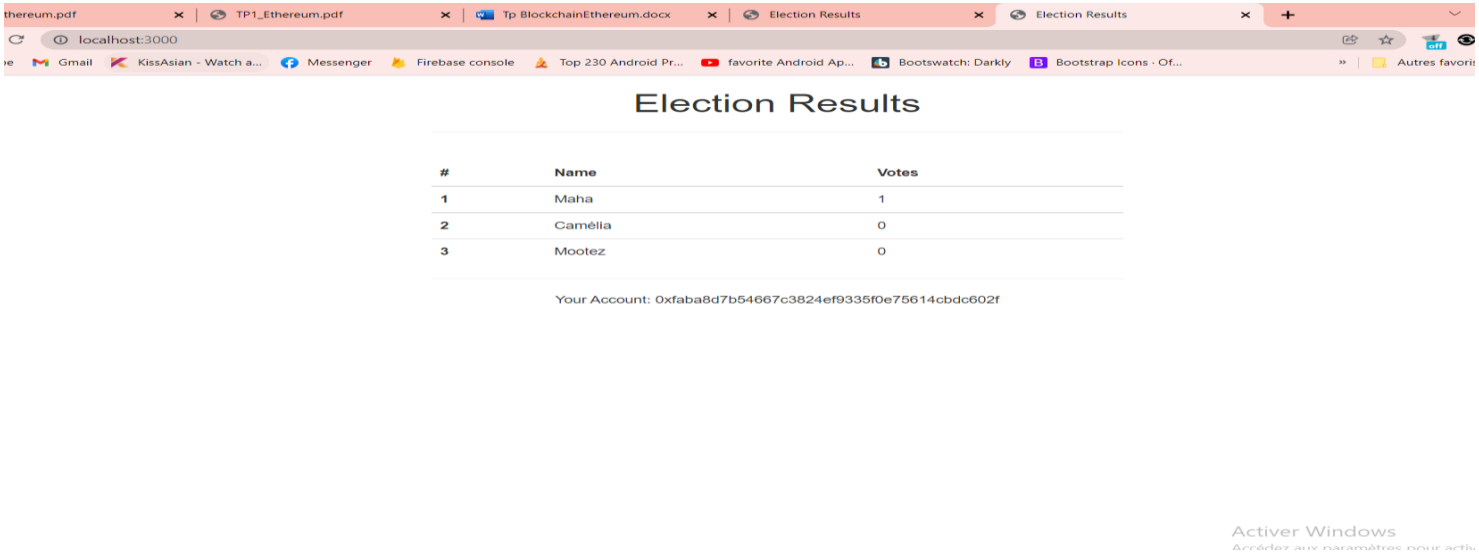
And finally, we can call this function whenever we initialize the contract:

```
Election.sol 1        app.js
src > js > app.js > initContract > $.getJSON("Election.json") callback
76   castVote: function() {···
89       },
90   listenForEvents: function() {
91           App.contracts.Election.deployed().then(function(instance) {
92           instance.votedEvent({}, {
93           fromBlock: 0,
94           toBlock: 'latest'
95           }).watch(function(error, event) {
96           console.log("event triggered", event)
97           // Reload when a new vote is recorded
98           App.render();
99           });
100          });
101        },
102    initContract: function() {
103          $.getJSON("Election.json", function(election) {
104          // Instantiate a new truffle contract from the artifact
105          App.contracts.Election = TruffleContract(election);
106          // Connect provider to interact with contract
107          App.contracts.Election.setProvider(App.web3Provider);
108          App.listenForEvents();
109
110          });
111        }
```

Now that we've updated our contract, we must run the <u>migrations</u> command :

```
111        }
112
PROBLÈMES 1    SORTIE    TERMINAL    CONSOLE DE DÉBOGAGE

Terminer le programme de commandes (O/N) ? o

C:\Users\lenovo\tpBlockchainEthereum>truffle migrate --reset

Compiling your contracts...
===========================
> Compiling .\contracts\Election.sol
> Compiling .\contracts\Election.sol
> Compiling .\contracts\Migrations.sol
> Artifacts written to C:\Users\lenovo\tpBlockchainEthereum\build\contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang
```

Now, you can vote on your client-side application, and watch the votes recorded in real time :



Partie Notée :

- Ajouter une interface pour ajout des nouveaux candidats à partir d'un formulaire

Add a form :

Election.sol | index.html | app.js

src > index.html

```html
            <hr />
          </form>
    <!-- //partie notée : ************************************************************************ -->
    <form onSubmit="App.addCandidate(); return false;">
      <div class="form-group">
        <label for="cname">Enter the Candidate name : </label>
        <input
          type="text"
          id="cname"
          name="name"
          placeholder="Enter candidate name"
          required
          size="10"
        />
        <br />
        <br />
      </div>
      <button type="submit" class="btn btn-primary">Add</button>
      <hr />
    </form>
  <!--  ************************************************************************ -->
          <h1 class="text-center">Election Results</h1>
          <hr />
          <br />
          <div id="loader"><p class="text-center">Loading...</p></div>
```

- Permetter seulement aux deux premières addresses du ganache d'effecteuer cet ajout sinon les autres addresses ne peuvent que voter.

Function addCandidate :

Election.sol | index.html | app.js

src > js > app.js

```
App.contracts.Election.setProvider(App.web3Provider);
App.listenForEvents();

});
},
//partie notée :*****************************************************************************
addCandidate: function(){
account1 = web3.eth.accounts[0];
account2 = web3.eth.accounts[1];
var candidateName = $("#cname").val();
App.contracts.Election.deployed()
.then(function (instance) {
  if(App.account==account1 || App.account==account2){
  return instance.addCandidate(candidateName, { from: App.account})
}
}
)
  // Reload when a new vote is recorded
App.render();
},
};
/* ************************************************************************** */
$(function() {
$(window).load(function() {
App.init();
```

PROBLÈMES SORTIE TERMINAL CONSOLE DE DÉBOGAGE Powe

localhost:3000

YouTube Gmail KissAsian - Watch a... Messenger Firebase console favorite Android Ap... Bootswatch: Darkly Bootstrap Icons · Of... Plus Circle Icon | Fo... Freelance.com » Autres favoris Liste de lecture

Select Candidate

Maha

Vote

Enter the Candidate name : nourhene

Add

Election Results

#	Name	Votes
1	Maha	0
2	Camélia	1
3	Mootez	0

Your Account: 0xaa512039fd859a828f8ce086c0b4c2a685861bc5