

5BLOC Documentation technique ET Guide d'utilisation

Projet de :
282810 – ISAAC Wesley
280804 – NDIONGUE El hadj
288189 – Kpognon Timothé
230089 – MPOYI Audry
286968 – REYAL Kevin

I. Architecture et technologie du projet

1. La blockchain

La blockchain utilisée est une blockchain Ethereum privée créée avec Geth. 3 nœuds y sont connectés. La blockchain tourne sur le port 8543 en localhost (127.0.0.1).

```
[INFO] [04-05 08:33:00: 99,159] Looking for peers peercount=1 tried=69 static=0
[INFO] [04-05 08:33:16, 718] Looking for peers peercount=0 tried=70 static=0
[INFO] [04-05 08:33:20, 968] Looking for peers peercount=0 tried=59 static=0
[INFO] [04-05 08:33:30, 971] Looking for peers peercount=0 tried=53 static=0
[INFO] [04-05 08:33:41, 014] Looking for peers peercount=0 tried=60 static=0
[INFO] [04-05 08:33:51, 056] Looking for peers peercount=0 tried=51 static=0
[INFO] [04-05 08:34:03, 169] Looking for peers peercount=0 tried=53 static=0
[INFO] [04-05 08:34:13, 332] Looking for peers peercount=0 tried=47 static=0
[INFO] [04-05 08:34:23, 345] Looking for peers peercount=0 tried=44 static=0
[INFO] [04-05 08:34:33, 418] Looking for peers peercount=1 tried=73 static=0
[INFO] [04-05 08:34:43, 464] Looking for peers peercount=0 tried=59 static=0
[INFO] [04-05 08:34:53, 564] Looking for peers peercount=0 tried=54 static=0
[INFO] [04-05 08:35:03, 654] Looking for peers peercount=2 tried=56 static=0
[INFO] [04-05 08:35:13, 692] Looking for peers peercount=1 tried=44 static=0
[INFO] [04-05 08:35:23, 917] Looking for peers peercount=0 tried=36 static=0
[INFO] [04-05 08:35:34, 199] Looking for peers peercount=0 tried=65 static=0
[INFO] [04-05 08:35:44, 247] Looking for peers peercount=0 tried=59 static=0
[INFO] [04-05 08:35:55, 258] Looking for peers peercount=0 tried=68 static=0
[INFO] [04-05 08:36:06, 035] Looking for peers peercount=0 tried=47 static=0
[INFO] [04-05 08:36:16, 193] Looking for peers peercount=0 tried=52 static=0
[INFO] [04-05 08:36:26, 459] Looking for peers peercount=0 tried=38 static=0
[INFO] [04-05 08:36:36, 661] Looking for peers peercount=0 tried=51 static=0
[INFO] [04-05 08:36:46, 726] Looking for peers peercount=0 tried=51 static=0
[INFO] [04-05 08:36:56, 852] Looking for peers peercount=0 tried=46 static=0
[INFO] [04-05 08:37:07, 388] Looking for peers peercount=1 tried=43 static=0
[INFO] [04-05 08:37:17, 660] Looking for peers peercount=0 tried=68 static=0
[INFO] [04-05 08:37:27, 791] Looking for peers peercount=0 tried=54 static=0
[INFO] [04-05 08:37:37, 835] Looking for peers peercount=0 tried=35 static=0
[INFO] [04-05 08:37:47, 835] Looking for peers peercount=1 tried=33 static=0
[INFO] [04-05 08:37:57, 973] Looking for peers peercount=0 tried=71 static=0
[INFO] [04-05 08:38:09, 067] Looking for peers peercount=0 tried=38 static=0
[INFO] [04-05 08:38:11, 879] Updated mining threads threads=8
[INFO] [04-05 08:38:11, 879] Updated mining threads threads=8
[INFO] [04-05 08:38:11, 879] Updated mining threads threads=8
[INFO] [04-05 08:38:11, 879] Commit new mining work price=10000000000
[INFO] [04-05 08:38:11, 879] Commit new mining work price=10000000000
[INFO] [04-05 08:38:11, 974] Successfully sealed new block number=2282 sealhash=898231..af9b40 uncles=0 txs=0 gas=0 fees=0 elapsed=171.356μs
[INFO] [04-05 08:38:11, 974] Block reached canonical chain number=2282 sealhash=898231..af9b40 hash=f092a3..6523da elapsed=103.739ms
[INFO] [04-05 08:38:11, 974] Mined potential block number=2282 sealhash=f092a3..6523da
[INFO] [04-05 08:38:11, 974] Commit new mining work number=2283 sealhash=d309e3..8104d6 uncles=0 txs=0 gas=0 fees=0 elapsed=165.096μs
[INFO] [04-05 08:38:11, 974] Successfully sealed new block number=2283 sealhash=d309e3..8104d6 hash=6c56a9..093e57 elapsed=413.578ms
[INFO] [04-05 08:38:12, 388] Block reached canonical chain number=2276 hash=162852..7ecc8
[INFO] [04-05 08:38:12, 388] Mined potential block number=2283 hash=c656a9..093e57
[INFO] [04-05 08:38:12, 388] Commit new mining work number=2284 sealhash=d6d809..445382 uncles=0 txs=0 gas=0 fees=0 elapsed=177.38μs
[INFO] [04-05 08:38:12, 424] Successfully sealed new block number=2284 sealhash=d6d809..445382 hash=8c9fc3..43e467 elapsed=36.429ms
[INFO] [04-05 08:38:12, 424] Mining too far in the future wait=2s
[INFO] [04-05 08:38:12, 424] Block reached canonical chain number=2277 hash=529c2d..1a9b99
[INFO] [04-05 08:38:12, 424] Mined potential block number=2284 hash=8c9fc3..43e467
[INFO] [04-05 08:38:14, 461] Commit new mining work number=2285 sealhash=315f5fc..e881f0 uncles=0 txs=0 gas=0 fees=0 elapsed=2.001s
[INFO] [04-05 08:38:14, 461] Block reached canonical chain number=2278 hash=935e3..ed7cf7
[INFO] [04-05 08:38:14, 461] Mined potential block number=2285 hash=76040e..e17cf1 elapsed=35.456ms
[INFO] [04-05 08:38:14, 461] Commit new mining work number=2286 sealhash=3d092e..a67162 uncles=0 txs=0 gas=0 fees=0 elapsed=248.027μs
[INFO] [04-05 08:38:14, 828] Successfully sealed new block number=2286 sealhash=3d092e..a67162 hash=c784e0..5ba5a elapsed=366.459ms
[INFO] [04-05 08:38:14, 828] Mining too far in the future wait=2s
[INFO] [04-05 08:38:14, 828] Block reached canonical chain number=2279 hash=a29a2b..d46d1a
[INFO] [04-05 08:38:14, 828] Mined potential block number=2286 hash=c784e0..5ba5a
[INFO] [04-05 08:38:16, 832] Commit new mining work number=2287 sealhash=901b92..048dbd uncles=0 txs=0 gas=0 fees=0 elapsed=2.004s
[INFO] [04-05 08:38:16, 832] Successfully sealed new block number=2287 sealhash=901b92..048dbd hash=9e0ed9..66caf6 elapsed=1.307s
[INFO] [04-05 08:38:18, 140] Block reached canonical chain number=2288 hash=5fe44..720b01
[INFO] [04-05 08:38:18, 140] Mined potential block number=2287 hash=9e0ed9..66caf6
[INFO] [04-05 08:38:18, 140] Commit new mining work number=2288 sealhash=f635ce..a29d65 uncles=0 txs=0 gas=0 fees=0 elapsed=171.605μs
[INFO] [04-05 08:38:18, 149] Successfully sealed new block number=2288 sealhash=f635ce..a29d65 hash=5457c8..1fa6fa elapsed=356.360ms
[INFO] [04-05 08:38:18, 149] Mined potential block number=2281 hash=a3c7f..de0e01
[INFO] [04-05 08:38:18, 149] Commit new mining work number=2288 sealhash=5457c8..1fa6fa
[INFO] [04-05 08:38:18, 149] Successfully sealed new block number=2289 sealhash=160a17..bf0cb0 uncles=0 txs=0 gas=0 fees=0 elapsed=166.976μs
[INFO] [04-05 08:38:18, 149] Block reached canonical chain number=2289 sealhash=160a17..bf0cb0 hash=ae7777..e05e68 elapsed=651.667μs
[INFO] [04-05 08:38:18, 149] Mined potential block number=2282 hash=f092a3..6523da
[INFO] [04-05 08:38:18, 149] Commit new mining work number=2289 sealhash=ae7777..e05e68
```

Ci-dessus, la fenêtre de terminal avec la blockchain geth qui tourne et des opérations de minage de blocks

Ci-dessus, un noeud connecté à la blockchain se présente sous cette forme. les différentes commandes affichés sur la fenêtre permettent de lancer le minage de blocks, l'arrêter et déverrouiller le compte principal sur la blockchain.

2) Le Smart-contrat

Il est écrit en Solidity et déployé sur la blockchain Geth au moyen de Truffle en utilisant le compilateur solidity 0.4.26, nous laissant un peu plus de liberté que les nouvelles versions, notamment au niveau de l'usage string dans le constructeur du contrat, dans la version 0.6 nous aurions été limité à du byte32 que nous aurions du convertir, par exemple, avec web3js.

Il regroupe les fonctions qui sont utilisables au sein de la blockchain. Les candidats aux élections doivent y être définis dans le constructeur **avant** son déploiement.

La particularité de notre architecture fait que le contrat dois être compilé deux fois dans notre projet afin de pouvoir être utilisé : 1 fois dans le front-end, afin d'obtenir le json compilé et utilisable dans le dossier build pour que l'application l'utilise, et une fois dans le dossier truffle, afin de le déployer sur la blockchain.

Ci-dessous le code du contrat.

```
supVote-frontend > contracts > voteContract.sol
1 pragma solidity 0.4.25;
2
3 contract VoteContract {
4     struct Candidate {
5         uint id;
6         string name;
7         uint voteNumber;
8     }
9
10
11     // Store accounts that have voted
12     mapping(address => bool) public voters;
13
14     mapping(uint => Candidate) public candidates;
15
16     uint public incrementVoteForCandidate;
17     event candidateHasVoted (
18         uint indexed _candidateId
19     );
20
21     constructor () public {
22         candidateCreator("Macron");
23         candidateCreator("Fillon");
24         candidateCreator("Le Pen");
25     }
26
27     function candidateCreator (string _name) private {
28         incrementVoteForCandidate++;
29         candidates[incrementVoteForCandidate] = Candidate(incrementVoteForCandidate, _name, 0);
30     }
31
32     function voteForCandidate (uint _candidateId) public {
33         require(!voters[msg.sender]);
34         require(_candidateId > 0 && _candidateId <= incrementVoteForCandidate);
35         voters[msg.sender] = true;
36         candidates[_candidateId].voteNumber++;
37
38         emit candidateHasVoted(_candidateId);
39     }
40
41 }
```

3) L'application

C'est une application faite en Node.JS avec plusieurs librairies dont Web3JS, celle-ci tourne sur le port 3000. Elle permet aux utilisateurs de voter mais aussi de consulter le nombre de votes par candidats aux élections.

Ci-dessous l'interface utilisateur de SupVote. on peut y retrouver le compte avec lequel nous sommes connectés. Dans l'exemple sur cette capture le bouton « voter » n'y est pas disponible car l'Application détecte qu'un vote a déjà été effectuer au sein de la blockchain avec le compte connecté. Chaque utilisateur ne peut voter qu'une seule fois.

SupVote - élection via blockchain

Résultat des élections

Nom du candidat	Nombre de Votes
Macron	1
Fillon	0
Le Pen	0

Vous êtes connecté avec le compte : 0x39b74d0de8339200857fc2200ce6252edc1df1dc

Vous avez déjà voté avec ce compte ! Chaque compte ne peut voter qu'une fois !

Ci-dessous, avec un autre compte sur la blockchain, l'application offre la possibilité de voter car ce compte n'a pas encore voter.

SupVote - élection via blockchain

Résultat des élections

Nom du candidat	Nombre de Votes
Macron	1
Fillon	0
Le Pen	0

Veuillez selectionner un candidat

Vote

Vous êtes connecté avec le compte : 0xbcd3886f61e56be1e9783cc5ed781edb7d33939

II) fonctionnement et déploiement

1) la blockchain

Pour lancer la blockchain, il faut se placer dans le dossier blockchain-2 du projet, et lancer la commande suivante :

```
geth --port 4321 --networkid 1234 --datadir=./blockchain --rpc --  
rpcport 8543 --rpcaddr 127.0.0.1 --rpccorsdomain "*" --rpcapi  
"eth,net,web3,personal,miner" --allow-insecure-unlock
```

Pour lancer la connexion d'un noeud à la blockchain il faut lancer la commande suivante dans un autre terminal : geth attach <http://127.0.0.1:8543>

C'est depuis ce nouveau terminal que nous pouvons effectuer des actions comme lancer le minage de block ou déverrouiller des comptes sur la blockchain.

1.1 Si le contrat n'est pas encore déployé sur la blockchain

Si le contrat n'est pas encore déployé, ou que l'on souhaite redéployé le contrat, il faut se placer en premier lieu dans le dossier blockchain-2 et **se mettre dans la console truffle** (commande « truffle console ») puis dans lancer la compilation du contrat avec la commande « Truffle compile », puis migrer le contrat avec « truffle migrate —reset » le tout lorsque l'on est en train de miner des blocks

Ensuite il faudra se rendre dans l'application dans supVote-frontend compiler le contrat au moyen de truffle compile, ce qui créera un fichier json que l'application utilisera de son coté, car elle n'est pas dans le même dossier que la blockchain.

2) l'application

L'application a été conçue en Node.js avec web3js et d'autres dépendances. le backend de l'application se situe dans le fichier app.js , le frontend dans index.html.

Pour lancer l'application il suffit de se placer dans le dossier supVote-frontend et de lancer la commande : npm run dev , cela ouvrira normalement le navigateur sur localhost sur le port 3000 une fois le chargement des fichiers fini par le serveur de fichiers.

```

App.contracts.VoteContract.deployed().then(function(instance) {
    supStance = instance;
    return supStance.incrementVoteForCandidate();
}).then(function(incrementVoteForCandidate) {
    var resultsPerCandidate = $("#resultsPerCandidate");
    resultsPerCandidate.empty();
    var choosenCandidate = $('#choosenCandidate');
    choosenCandidate.empty();

    for (var i = 1; i <= incrementVoteForCandidate; i++) {
        supStance.candidates(i).then(function(candidate) {
            var id = candidate[0];
            var name = candidate[1];
            var voteNumber = candidate[2];
            var htmlForCandidate = `<tr><td class='${name}'> ${name} </td><td> ${voteNumber} </td></tr>`;

            resultsPerCandidate.append(htmlForCandidate);

            var candidateOption = "<option value='" + id + "' >" + name + "</ option>"
            choosenCandidate.append(candidateOption);
        });
    }
    return supStance.voters(App.account);
}).then(function(alreadyVoted) {
    if(alreadyVoted) {
        $('form').hide();
        $("#votedOrNot").html("Vous avez déjà voté avec ce compte ! Chaque compte ne peut voter qu'une fois !  ");
    }
    loadingBar.hide();
    htmlContent.show();
}).catch(function(error) {
    console.warn(error);
});

```

Ci-dessus une partie du code du back-end, gérant l'affichage des candidats, si le compte connecté a déjà voté ou non, ou encore la fonction de vote.

une partie du code du Front-end ci-dessous, du html avec bootstrap.

```
<div class="row">
  <div class="col-lg-12">
    <h1 class="text-center">SupVote – élection via blockchain</h1>
    <h1 class="text-center">Résultat des élections</h1>
    <hr/>
    <br/>
    <div id="loadingBar">
      <p class="text-center">Chargement en cours ...</p>
    </div>
    <div id="htmlContent" style="display: none;">
      <table class="table">
        <thead>
          <tr>
            <th scope="col">Nom du candidat</th>
            <th scope="col">Nombre de Votes</th>
          </tr>
        </thead>
        <tbody id="resultsPerCandidate">
        </tbody>
      </table>
    <hr/>
    <form onSubmit="App.loadVote(); return false;">
      <div class="form-group">
        <label for="chooseCandidate">Veuillez selectionner un candidat</label>
        <select class="form-control" id="chooseCandidate">
        </select>
      </div>
      <button type="submit" class="btn btn-primary">Vote</button>
      <hr />
    </form>
```

!!! ATTENTION !!! :

Si les nodes modules ne sont pas inclus dans le projet, il faut lancer la commande npm install pour installer les dépendances présente dans le package.json.

la blockchain dois impérativement être lancée en premier et avoir un compte déverrouillé et en train de miner des blocks (miner.start() dans un noeud de la blockchain) afin que l'application puisse fonctionner correctement!

Afin de voter il est nécessaire de se connecter avec un compte sur la blockchain qui n'a pas encore voter, les deux comptes présent ayant déjà voter, il est possible de créer un autre compte avec la commande : personal.newAccount('passphrase') ou 'passphrase' dois être remplacé par le mot de passe de notre choix.

Lorsque l'utilisateur clique sur voter, la transaction est envoyée à la blockchain

Les deux comptes présents portent les addresses suivantes :

« 0x39b74d0de8339200857fc2200ce6252edc1df1dc" ayant pour mot de passe « lamalama »
« 0xbed3886f61e56be1e9783ccd5ed781edb7d33939 » ayant pour mot de passe « kingofcod »

Pour changer le compte connecté à la blockchain il suffit d'utiliser la commande
miner.setEtherbase(« 0x39b74d0de8339200857fc2200ce6252edc1df1dc") en remplaçant avec
l'ID du compte que l'on souhaite afficher.