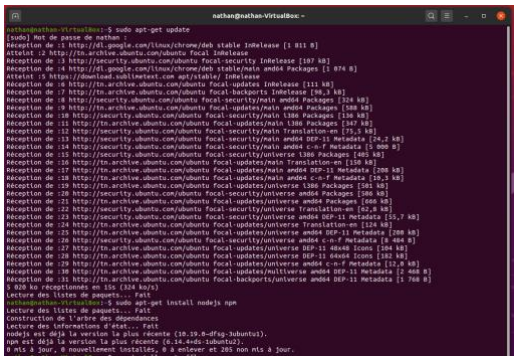
COMPTE RENDU – E-VOTING

Avant toute chose, je tenais à annoncer que j'ai réalisé le TP avant de prendre les captures d'écran. De ce fait, les image de migrations ou d'installation etc indiquent être déjà à jour. Ensuite, je n'explique pas dans le détail le fonctionnement des fonctions car c'est expliqué ici : https://www.dappuniversity.com/videos/3681ZYbDSSk

MISE EN PLACE DE L'ENVIRONNEMENT

Installation npm et nodeJS



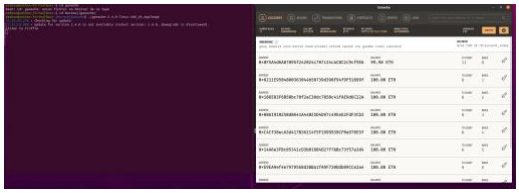
sudo apt-get update
sudo apt-get install nodejs npm

Installation truffle

sudo npm install -g truffle

```
nathan@nathan-VirtualBox:~$ sudo npm install -g truffle
npm WARN deprecated flat@4.1.0: Fixed a prototype pollution security issue in 4.1.0, please upgrade to ^4.1.1 or ^5.0.1.
/usr/local/bin/truffle -> /usr/local/lib/node_modules/truffle/build/cli.bundled.js

> truffle@5.1.47 postinstall /usr/local/lib/node_modules/truffle
> node ./scripts/postinstall.js
```

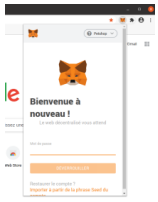
Installation ganache



Télécharger ganache via le lien : https://www.trufflesuite.com/ganache , on obtient une image du type ganache-1.3.0-x86_64.AppImage.

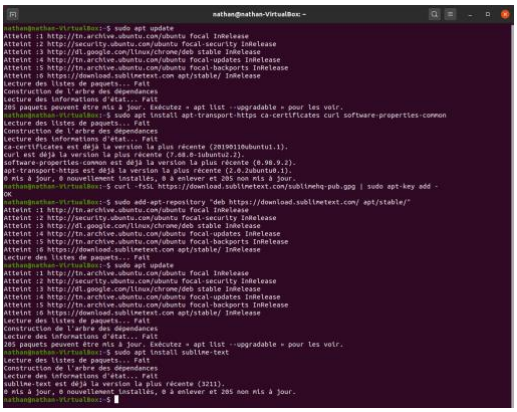
Ensuite, saisir : chmod a+x ganache-1.3.0-x86_64.AppImage et ./ganache-1.3.0-x86_64.AppImage

Installation metamask



Ajouter le plugin metamask depuis chrome extension

Installation sublime text



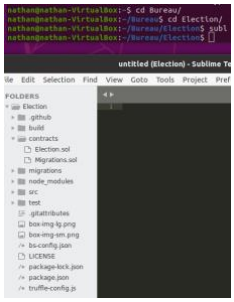
sudo apt update
sudo apt install apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://download.sublimetext.com/sublimehq-pub.gpg | sudo apt-key add -
sudo add-apt-repository "deb https://download.sublimetext.com/ apt/stable/"
sudo apt updates
udo apt install sublime-text

Vérification des versions

truffle version

```
nathan@nathan-VirtualBox:~$ truffle version
Truffle v5.1.47 (core: 5.1.47)
Solidity v0.5.16 (solc-js)
Node v10.19.0
Web3.js v1.2.1
nathan@nathan-VirtualBox:~$
```

MISE EN PLACE DU PROJET



Voici les premières étapes à suivre :

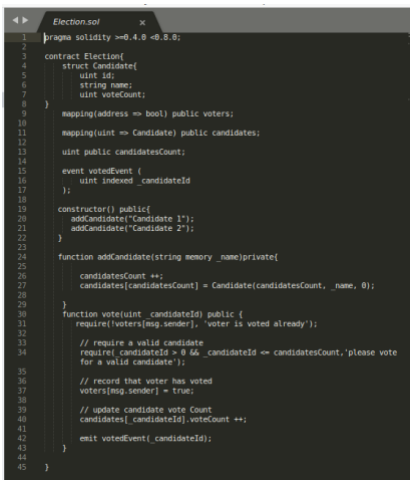
- Création du dossier : mkdir Election et aller dedans : cd Bireau/Election

- Lancement de Sublime Text : subl

- Création du smart contrat : touch contracts/Election.sol

Ce smart contrat permettra de lire les données à partir de la Blockchain et d'en écrire. Ici, il permettra de lister les candidats qui participent à l'élection, et gérer les votants et leur vote.

MISE EN PLACE DU CODE



```
pragma solidity >=0.4.0 <0.8.0;

contract Election{
    struct Candidate{
        uint id;
        string name;
        uint voteCount;
    }
    mapping(address => bool) public voters;

    mapping(uint => Candidate) public candidates;

    uint public candidatesCount;

    event votedEvent (
        uint indexed _candidateId
    );

    constructor() public{
      addCandidate("Candidate 1");
      addCandidate("Candidate 2");
    }

    function addCandidate(string memory _name)private{

        candidatesCount ++;
        candidates[candidatesCount] = Candidate(candidatesCount, _name, 0);

    }
    function vote(uint _candidateId) public {
      require(!voters[msg.sender], 'voter is voted already');

        // require a valid candidate
        require(_candidateId > 0 && _candidateId <= candidatesCount,'please vote
        for a valid candidate');

        // record that voter has voted
        voters[msg.sender] = true;

        // update candidate vote Count
        candidates[_candidateId].voteCount ++;

        emit votedEvent(_candidateId);
    }
}
```

Taper le code suivant dans Election.sol :

Ce code permet globalement de créer les candidats, ajouter des candidats, voter et comptabiliser les votes et évidement de mettre à jour chacune de ces actions.

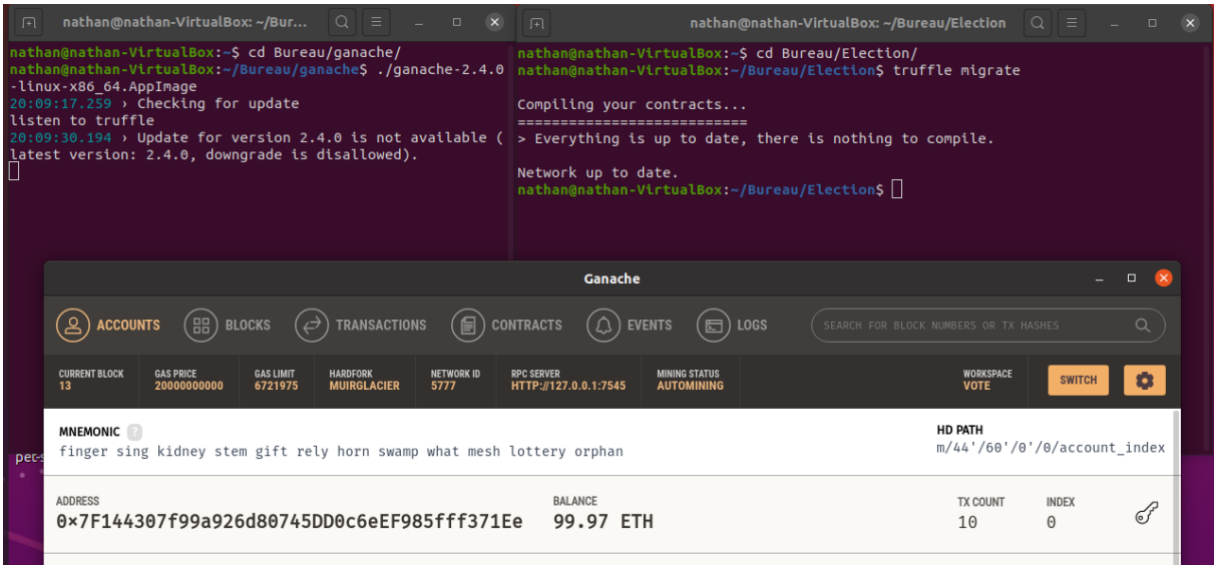
Création du fichier 2 deploy contracts.js sous le répertoire migration
Ce code permet de vérifier le déploiement du fichier sol sur la blockchain

```
var Election = artifacts.require("./Election.sol");

module.exports = function(deployer) {
  deployer.deploy(Election);
};
```

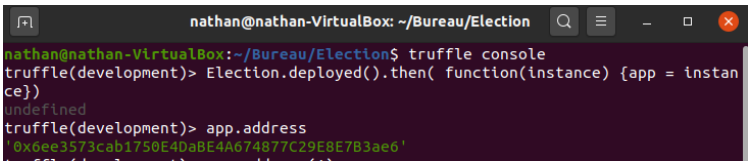
VERIFICATION FONCTIONNEMENT

Lancement de ganache : ./ganache-2-4-0-linux-x86_64.AppImage

Lancement de la migration vers la blockchain : truffle migrate



Lancer la console truffle : truffle console afin de tester le fonctionnement du fichier Election.sol avec la création d'une instance : Election.deployed().then(function(instance) {app = instance}) et vérifier que la variable a été assigné et qu'elle a une adresse.



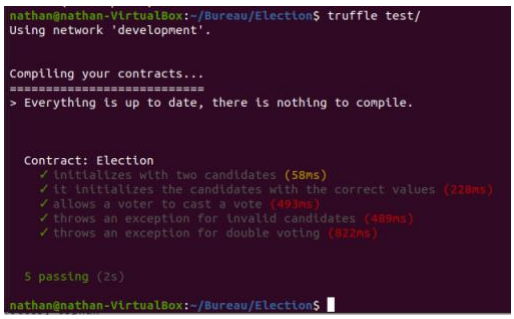
ANIMATION INTERFACE CLIENT

Créer un fichier election.js dans Test

Si des modifications ont été effectuées sur le smart contract, taper :truffle migrate -reset

Ce fichier javascript met en place toute les animations et les vérifications à effectuer sur l'interface client.

Lancer l'exécution su scripte : truffle test



VISUEL INTERFACE CLIENT

Modifier le fichier index.html

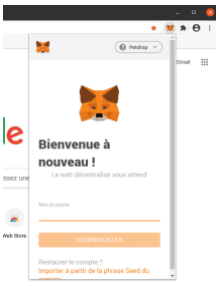
Ce fichier permet de mettre en place l'aspect esthétique de l'interface client ainsi que l'appel des différents fichiers qui intégreront la page web.

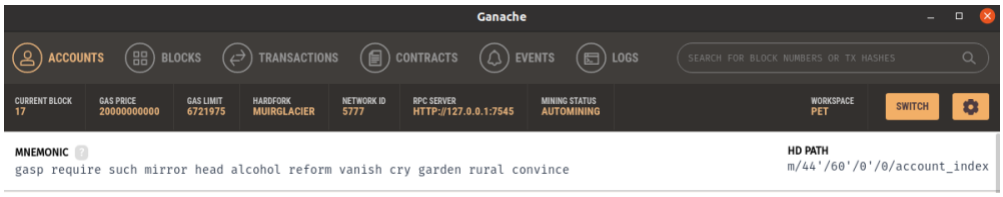
Remplacer le contenu du fichier app.js

C'est un fichier de configuration de la page web

CONFIGURATION METAMSK

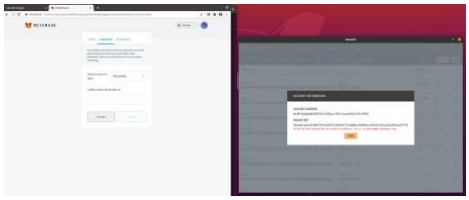
Il faut installer chrome et l'extension metamask. Il faut choisir l'option qui indique que l'on a déjà un compte et entrer la « seed phrase » que l'on trouve sur ganache.

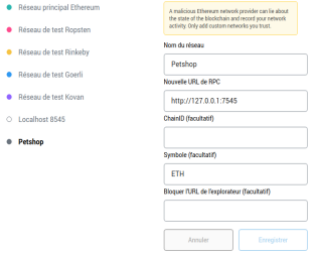




Ensuite, il faut créer un nouveau réseau et indiquer l'adresse RPC server que l'on trouve sur ganache **http://127.0.0.1:7545**

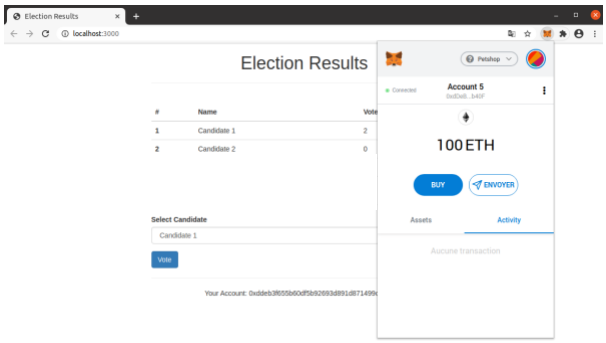
Ensuite, on peut importer les comptes de ganache en cliquant sur importer un compte en entrant la clé privée de l'un des 10 comptes possédants chacun 100 ETH





PHASE FINALE

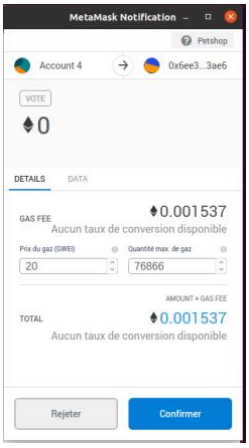
Lancer le serveur : npm run dev

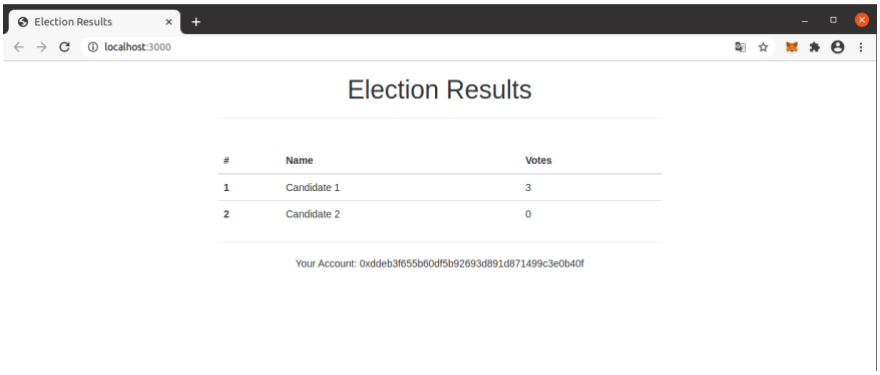



```
nathan@nathan-VirtualBox:~$ cd Bureau/Election/
nathan@nathan-VirtualBox:~/Bureau/Election$ npm run dev

> pet-shop@1.0.0 dev /home/nathan/Bureau/Election
> 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
    External: http://10.0.2.15:3000
 --------------------------------
          UI: http://localhost:3001
 UI External: http://localhost:3001
 --------------------------------
[Browsersync] Serving files from: ./src
[Browsersync] Serving files from: ./build/contracts
[Browsersync] Watching files...
20.10.08 22:05:55 304 GET /index.html
20.10.08 22:05:56 304 GET /css/bootstrap.min.css
20.10.08 22:05:56 304 GET /js/bootstrap.min.js
20.10.08 22:05:56 304 GET /js/web3.min.js
20.10.08 22:05:56 304 GET /js/truffle-contract.js
20.10.08 22:05:56 304 GET /js/app.js
20.10.08 22:05:57 304 GET /Election.json
20.10.08 22:06:16 304 GET /index.html
20.10.08 22:06:16 304 GET /css/bootstrap.min.css
20.10.08 22:06:16 304 GET /js/bootstrap.min.js
20.10.08 22:06:16 304 GET /js/web3.min.js
20.10.08 22:06:16 304 GET /js/truffle-contract.js
20.10.08 22:06:16 304 GET /js/app.js
20.10.08 22:06:16 304 GET /Election.json
```

On remarque que le serveur ouvre automatiquement une page chrome en locale.

Lorsque l'on clique sur voter, une fenêtre metamask s'ouvre et nous demande de confirmer la transaction pour que le vote soit pris en compte.





Chaque compte est identifié de manière unique. Ce numéro est inscrit en bas de la page. Chacune des transaction (vote) est ainsi enregistré dans la blockchain avec l'identifiant du compte et le vote.