

# Voting System

Blockchain project

## ► Cuprins

## Introducere

Acest proiect cuprinde un sistem de votare implementat folosind tehnologia Blockchain și este creat de: [Ana-Maria Comorașu](#), [Alexandru Țifui](#), [Laura Tender](#).

Proiectul a fost dezvoltat în cadrul cursului Blockchain din anul al III-lea la Facultatea de Matematică și Informatică, Universitatea din București.

Sistemul de votare permite unui proprietar să își creeze o alegere cu scop sau un titlu și să adauge candidați. Proprietarul este cel care deschide și închide alegerile și cel care trebuie să autorizeze persoanele care participă la vot. Sistemul nostru de votare cuprinde și o interfață pentru a arată progresul alegerilor.

## Tehnologii

- [Solidity](#)
- [Javascript](#)
- [Truffle](#)
- [Ganache](#)
- [MetaMask](#)
- [Web3.js](#)
- [Nodejs](#)

## Usage

Trebuie să instalăm următoarele: truffle, ganache, metamask.

Comenzile de care avem nevoie pentru a porni aplicația:

```
1 | conda activate blockchain
2 | ganache -p 7545 -i 5777
3 | truffle migrate --reset all
4 | node client/src/script.js
```

Apoi intrăm pe localhost:3000 și ne conectăm cu MetaMask.

Pentru a porni alegerile, adminul folosește truffle console astfel:

```
1 | truffle console
2 | ballot = await Ballot.deployed()
3 | ballot.grantVoter(adresaUser)
4 | ballot.startBallot()
```

Putem verifica că alegerile au fost pornite astfel:

```
1 | ballot.electionState()
2 | >in_progress
```

Pentru a închide alegerile:

```
1 | ballot.endBallot()
```

## Implementarea sistemului de votare

Implementarea sistemului de votare poate fi găsită în fișierul `Ballot.sol`.

- **Crearea, deschiderea și închiderea alegerilor**

Această parte este realizată de *Ana Comorașu* și cuprinde crearea clasei `Ballot`, constructorul, deschiderea, închiderea și starea alegerilor.

În constructor, alegerea noastră cuprinde un titlu, un proprietar, o listă de candidați și starea votării este marcată ca `registered`.

```
1 | constructor(string memory _title, bytes32[] memory _candidateList){
2 |     title = _title;
3 |     chairperson = msg.sender;
4 |     for(uint i = 0; i < _candidateList.length; i++){
5 |         candidatesList.push(Candidate({
6 |             name: _candidateList[i],
7 |             numVotes: 0
8 |         }));
9 |     }
10 |     state = VotingState.Registered;
11 | }
12 |
```

Votarea poate avea trei stări: nu a început (registered), în desfășurare, încheiată. Verificarea stărilor are loc prin modificarii: `inProgress`, `notStarted`, `finished`. Funcția `startBallot` deschide alegerile, iar funcția `endBallot` le încheie.

```
1  function startBallot() public notStarted isChairperson {
2      state = VotingState.Progress;
3  }
4
5  function endBallot() public inProgress {
6      state = VotingState.Completed;
7  }
```

- **Verificarea dreptului de proprietar și autorizarea votanților**

Această parte este realizată de *Laura Tender*.

Verificarea dreptului de proprietar este implementată cu ajutorul modifierului `isChairperson`, prin care ne asigurăm că senderul este persoană care a creat alegerile. Acest modifier este folosit pentru deschiderea și închiderea alegerilor, cât și pentru autorizarea votanților.

```
1  modifier isChairperson(){
2      require(msg.sender == chairperson, "Only the chairperson is
3      allowed.");
4      _;
```

Un votant se află în una dintre stările: neautorizat (`NotGranted`), autorizat (`Granted`) și care a votat deja (`Voted`). Votanții sunt reținuți într-un dicționar în care cheia este adresa lor, iar valoarea este starea sa. Autorizarea unui votant are loc în funcția `grantVoter` prin care votantul este adăugat în dicționar cu starea `granted`.

```
1  function grantVoter(address _voter) public notStarted isChairperson
2  {
3      alreadyVoted[_voter] = VoterState.Granted;
```

## • **Votarea și rezultatele alegerilor**

Această parte este realizată de *Alexandru Țifui*.

Votarea are loc prin funcția `vote`, dacă alegerile sunt încă în desfășurare și dacă persoană are drept de vot. Odată ce este condiții sunt îndeplinite, verificăm că opțiunea de vot este una validă, creștem cu un vot numărul de voturi pentru acest candidat și schimbăm starea votantului din `Granted` în `Voted`.

```
1  function vote(uint _choice) public inProgress canVote {
2      // * check if the choice is in the candidacy list
3      require(_choice < candidatesList.length, "There is no choice for
this");
4      // * add vote
5      candidatesList[_choice].numVotes++;
6      alreadyVoted[msg.sender] = VoterState.Voted;
7  }
```

Funcțiile `nthCandidate` și `nthNoVotes` returnează numele, respectiv numărul de voturi ale candidatului cu indexul `n`. Acestea verifică dacă candidatul cu indexul `n` există.

```
1  function nthCandidate(uint _id) public view finished returns (string
memory) {
2      require(_id < candidatesList.length, "Candidate index out of
range");
3      return candidatesList[_id].name;
4  }
```

## **Client**

### • **Migrații**

Proiectul cuprinde 2 migrații: `1_initial_migration.js` și `2_deploy_ballot.js`.

Prima migrație face deploy la contractul `Migrations` (care poate fi găsit în `contracts/Migrations.sol`).

A doua migrație face deploy la contractul `Ballot` (are poate fi găsit în `contracts/Ballot.sol`). Astfel sunt create alegeri cu titlul "USA Election 1904" și lista de candidați "Candidate 1", "Candidate 2", ..., "Candidate 10".

- **Legarea front end-ului cu smart contractul**

În fișierul `utils.js` am creat funcțiile:

- `getWeb3`: pentru a instanția un obiect de tip `web3`. Am folosit `window.ethereum` ca provider și `window.ethereum.request` pentru a cere permisiunea pentru a accesa conturile.
- `getContract`: pentru a crea o instanță a contractului.

- **Interacționarea cu smart contractul**

Fișierul `api-calls.js` conține câteva funcții ajutătoare pentru a apela metodele din smart contract.

Funcțiile `getElectionState`, `callVote`, `getCandidatesNo`, `getNthCandidate` și `getNthResult` apelează metodele corespunzătoare și tratează excepțiile.

Fișierul `index.js` conține funcțiile `constructErrorCard`, `connectToContract`, `vote`, `getVotes`, `getResults` și `createCandidateResult` și face legătura dintre `html` și funcționalități.