

E-Notary

Membrii echipei

- Anghelache Vlad-Alexandru (332)
- Cațaron Andrei-Vlad (311)

Repartizare taskuri:

- **Anghelache Vlad-Alexandru:**
 - frontend;
 - crearea legăturii dintre backend și frontend;
 - backend: ajustarea codului și rezolvarea unor bug-uri ;
- **Cațaron Andrei-Vlad:**
 - backend;
 - research tool-uri și tehnologii;

Link GitHub: <https://github.com/vladanghelache/NotaryDapp>

Descrierea proiectului

Aplicație web descentralizată creată cu ajutorul tehnologiei blockchain care, prin semnarea unui document (încărcarea documentului în blockchain), atestă existența unui document la o anumită data și integritatea acestuia de la momentul semnării.

Tehnologii folosite

- Angular
- Ganache
- Truffle

- Solidity

Backend

Fișierul *Authenticity.sol* conține structura de date *Signature*, în care sunt encapsulate următoarele date:

- author: adresa deținătorului documentului;
- hash: hashul documentului;
- timestamp: data semnării documentului;
- size: mărimea documentului în bytes;
- docType: tipul documentului.

De asemenea, fișierul conține funcțiile care determină funcționalitățile principale ale aplicației:

- *signDocument*:
 - primește ca parametri hash-ul, mărimea și tipul documentului;
 - verifică cu ajutorul lui *require()* dacă documentul a fost deja semnat;
 - dacă documentul nu a mai fost semnat în trecut, se adaugă semnătura acestuia în *signaturesMap*, map folosit pentru stocarea semnăturilor;
 - se emite evenimentul *DocumentSigned*;
 - cod:

```
function signDocument(uint size, string memory hash, string memory
docType) public payable {
    require((signaturesMap[hash].size == 0 &&
        keccak256(abi.encodePacked(signaturesMap[hash].hash))
== keccak256(abi.encodePacked("")) &&

keccak256(abi.encodePacked(signaturesMap[hash].docType)) ==
keccak256(abi.encodePacked(""))),
        "document already signed");
```

```
    Signature memory newSignature = Signature(msg.sender, hash,
block.timestamp, size, docType);
    signaturesMap[hash] = newSignature;
    signaturesArray.push(newSignature);
    emit DocumentSigned(msg.sender, hash, block.timestamp, size,
docType);
```

```
}
```

- *verifyDocument*:

- primește ca parametru hash-ul unui document;
- returnează semnătura documentului corespunzător hash-ului dat;
- în cazul în care hash-ul nu corespunde unui document semnat, funcția va returna un obiect *Signature* cu campurile goale;
- cod:

```
function verifyDocument(string memory hash) public view returns
(Signature memory) {

    Signature memory newSignature = signaturesMap[hash];

    return newSignature;
}
```

- *getSignatures*:

- returnează un array de *Signature* care conține semnăturile corespunzătoare contului conectat;
- cod:

```
function getSignatures() public view returns (Signature[] memory) {
    uint count = 0;
    for (uint i = 0; i < signaturesArray.length; i++) {
        Signature storage signature = signaturesArray[i];
        if(keccak256(abi.encodePacked(signature.author)) ==
keccak256(abi.encodePacked(msg.sender))) {
            count++;
        }
    }
}
```

```
Signature[] memory newArray = new Signature[] (count);
uint j = 0;
for (uint i = 0; i < signaturesArray.length; i++) {
    Signature storage signature = signaturesArray[i];
```

```

    if (keccak256(abi.encodePacked(signature.author)) ==
keccak256(abi.encodePacked(msg.sender))) {
        newArray[j] = signature;
        j = j + 1;
    }
}
return newArray;
}

```

Frontend

Fișierul *authenticity.service.ts* este intermediar între frontend și backend, asigurând conectarea la MetaMask și apelarea funcțiilor smart-contractului *Authenticity*.

Aplicația conține 3 componente principale:

- *sign-document*:
 - utilizatorul poate încărca un document, pentru care se calculează hash-ul (folosind funcția SHA256), care se transmite către backend împreună cu tipul și mărimea documentului;
- *verify-document*:
 - utilizatorul poate încărca un document, pentru care se calculează hash-ul (folosind funcția SHA256), care se transmite către backend, pentru a verifica dacă documentul a fost semnat;
 - dacă documentul a fost semnat se vor afișa datele primite de la backend;
- *history*:
 - sunt afișate semnăturile corespunzătoare contului conectat.

