Sûrchain :

Store valuable objects with blockchain technology

Contexte :

Valuable objects

Bank

Safety deposit box

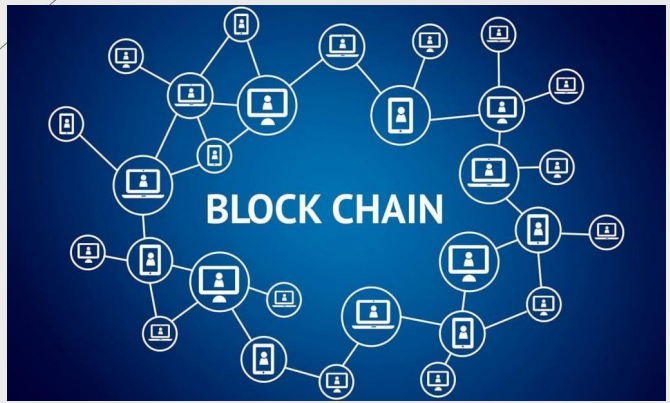
Issues :

- Status of bank
- Lack of privacy
- Account opening request
- Security deposit request
- Write a contract on a paper sheet

Problem : How to secure our personal property by limiting the risk of theft, expropriation, and facilitating its implementation

Summary :

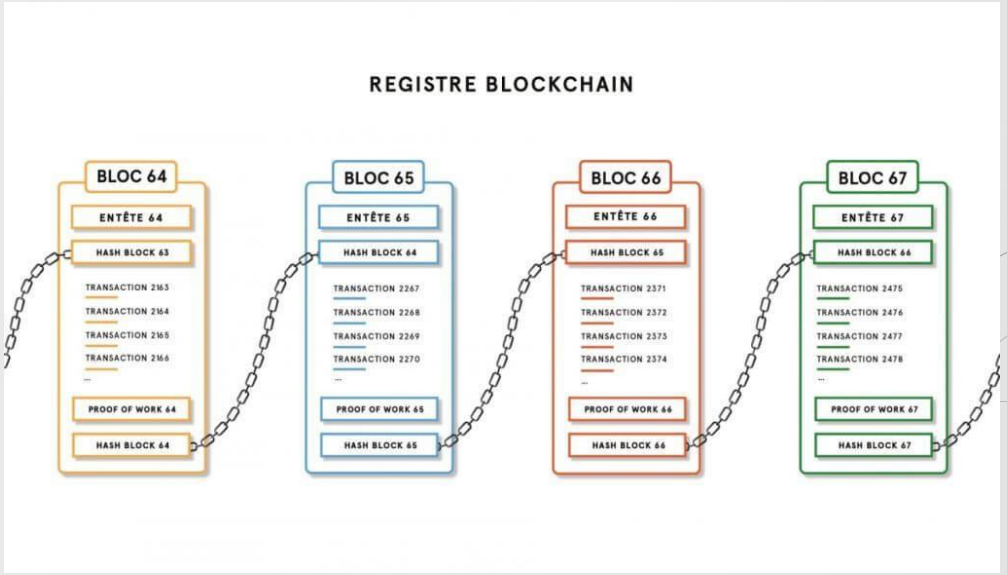
Solution provided :



Permet de résoudre les contraintes engendrées par les banques

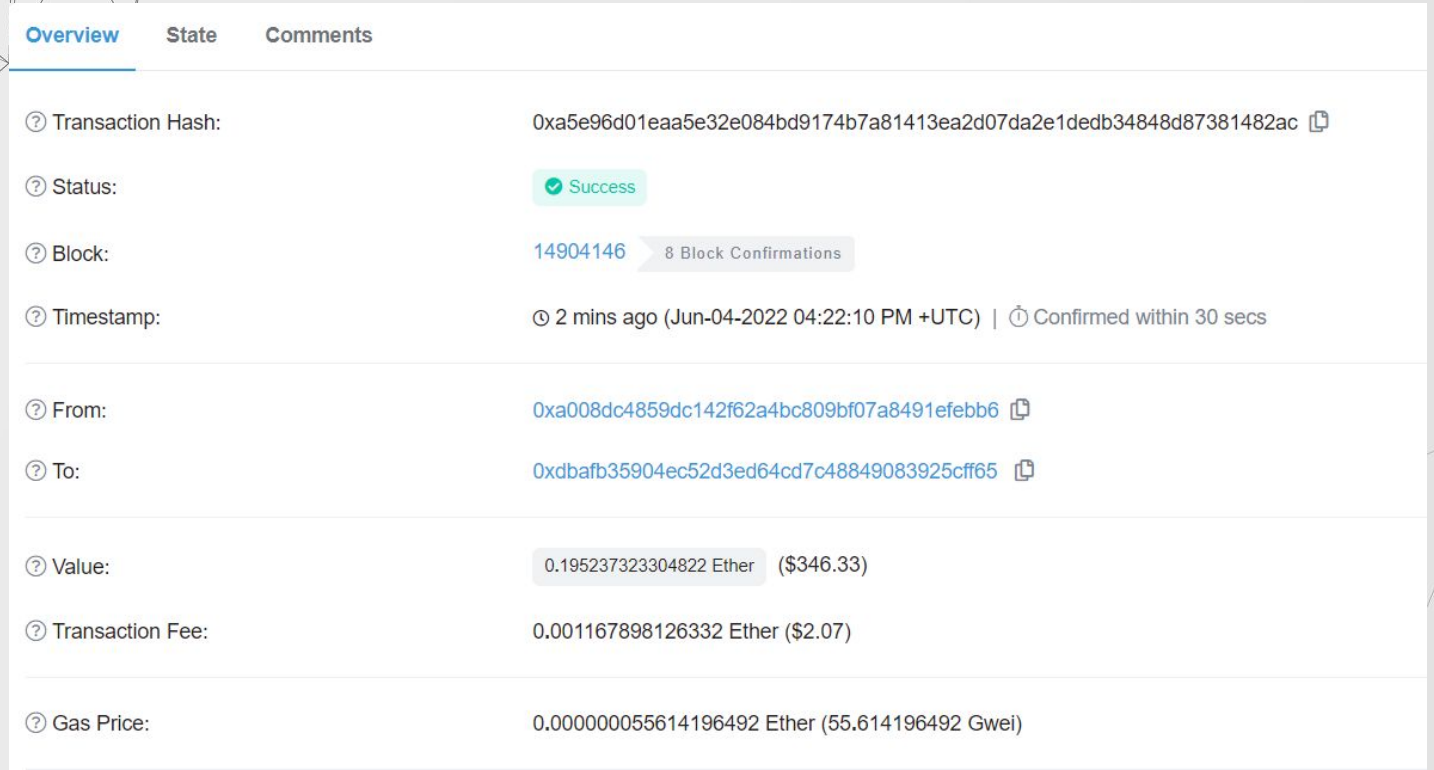
What is the Blockchain ?

- Chain of blocs
- Based on the system of cryptography



- Allows to leave a trace of any transaction anonymously

Example of transaction



What is an NFT ?

- Non Fongible Token
- Digital asset

Nyan cat : sold
300ETH ≈ 470 000$

- The blockchain allows to keep a trace of the transactions of this asset and introduce the notion of ownership

Completion Steps :

Customer 1

Possessions

1. Creating a digital wallet
2. Scanning the object as an NFT
3. Creating a smart contract
4. Storing the object in a warehouse
5. Visualization of the object in a DAPP (Decentralized Application)

Completion Steps :

Customer 1

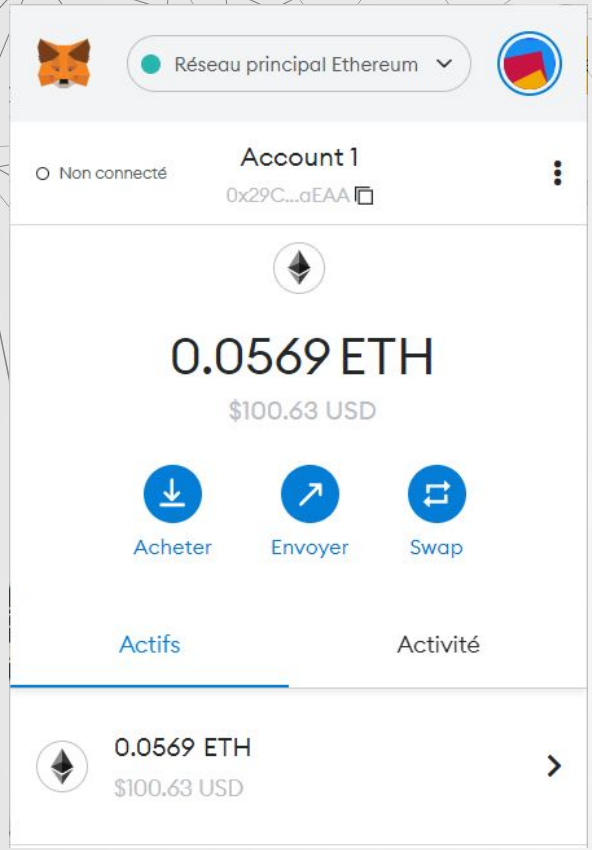
Possessions

Smart contract

1. Creating a digital wallet
2. Scanning the object as an NFT
3. Creating a smart contract
4. Storing the object in a warehouse
5. Visualization of the object in a DAPP (Decentralized Application)

Digital Wallet :

Secure and anonymous



Customer 1 become 0x29C...aEAA

Smart contract :

- Language used :

Dedicated to write smart contracts on the ethereum blockchain

- Libraries required :

Communicate with metamask wallet

- Tools : Hashlips art engine

Facilitate the creation of NFT’s

Smart contract :

Function Mint

```
32    function mint(address _to, uint256 _mintAmount) public payable {
33      uint256 supply = totalSupply();
34      require(!paused);
35      require(_mintAmount > 0);
36      require(_mintAmount <= maxMintAmount);
37      require(supply + _mintAmount <= maxSupply);
```

Creation of the NFT

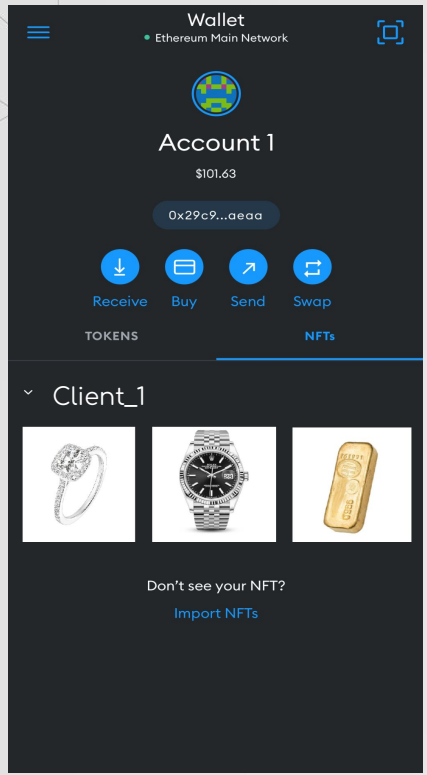
```
1   const basePath = process.cwd();
2   const { MODE } = require(`${basePath}/constants/blend_mode.js`);
3   const { NETWORK } = require(`${basePath}/constants/network.js`);
4
5   const network = NETWORK.eth;
6
7
8   const namePrefix = "Client_1";
9   const description = "Posséssions du client_1";
10  const baseUri = "ipfs://QmPXNBfDgS6WF1KGoRJWqHHio3ptcjxGF8fYrLNCaWg5ec";
```

NFT name

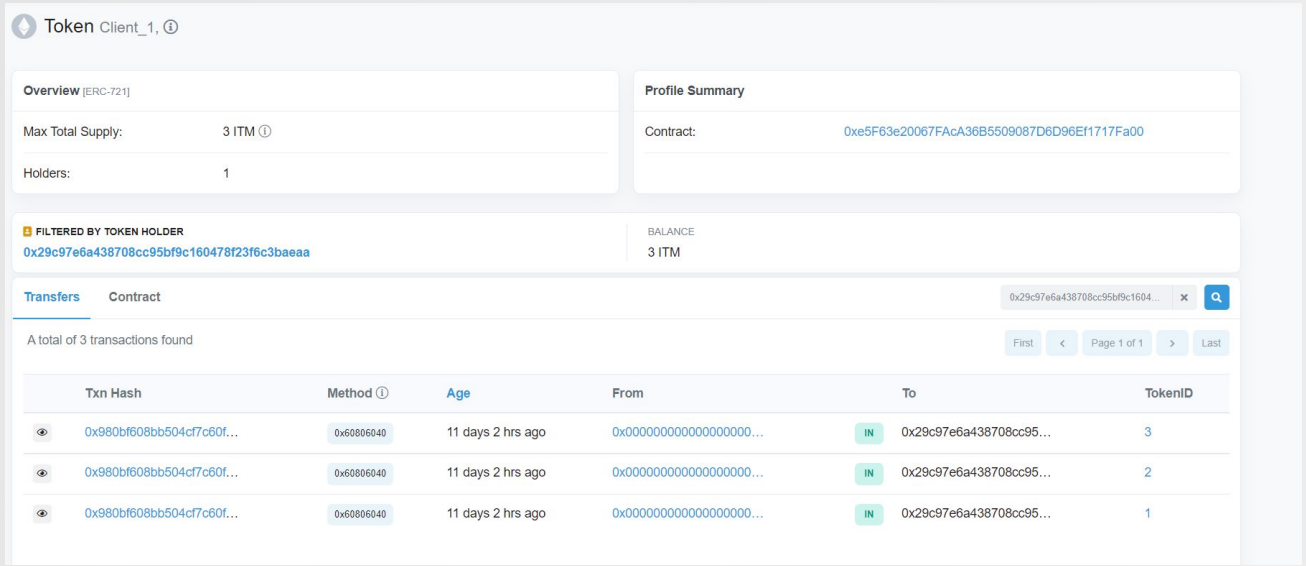
Description

Smart contract :

On Metamask



On the Ethereum blockchain



Creation of a cryptocurrency :

Services in the Blockchain → Use of crypto-assets

Choice of name+ symbol

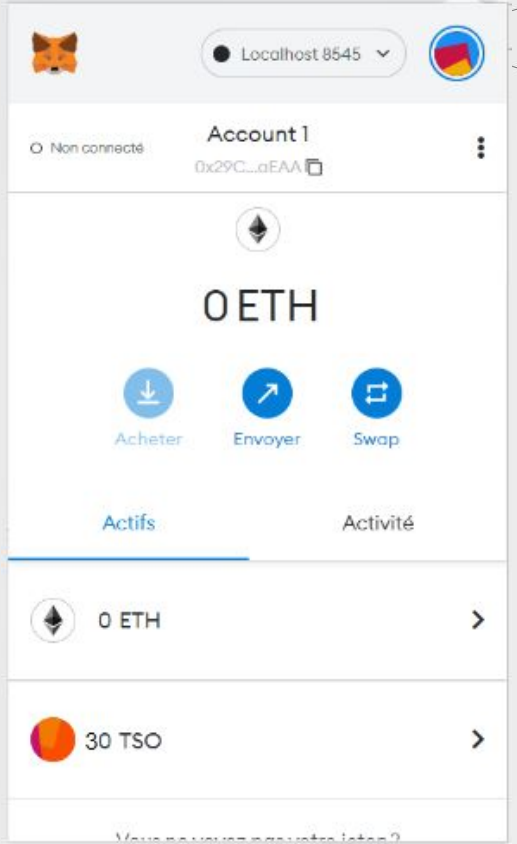
```
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract Token is ERC20 {
    constructor() ERC20("Token Source", "TSO") {
        _mint(msg.sender, 30 * (10 ** 18));
    }
}
```

Creation of a cryptocurrency :

Services in the Blockchain → Use of crypto-assets

Choice of name+ symbol

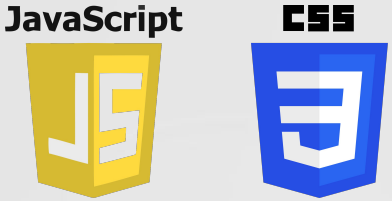
```
1  pragma solidity ^0.8.0;
2
3  import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
4
5  contract Token is ERC20 {
6      constructor() ERC20("Token Source", "TSO") {
7          _mint(msg.sender, 30 * (10 ** 18));
8      }
9  }
```



Number of TSO Obtained ←

Decentralized application :

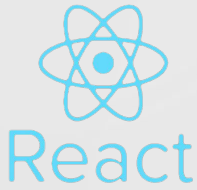
- Language used :



Front-end

Interaction entre la dapp et le smart contract

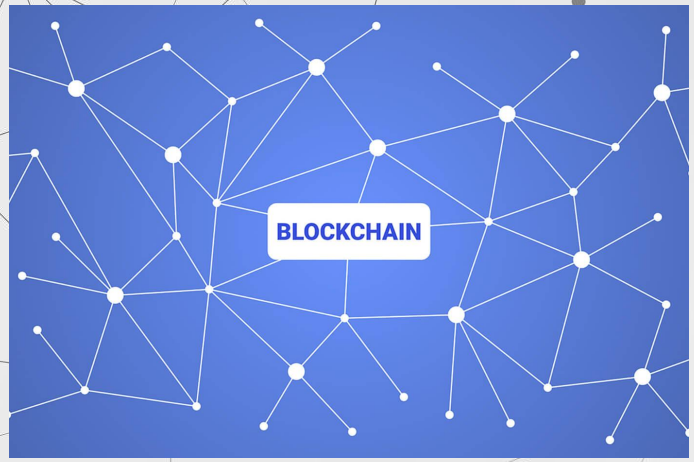
- Libraries required :



Framework pour faciliter la mise en oeuvre de la dapp

Decentralized application :

Conclusion :



- Possibility to make pawnbroks in cryptocurrency
- Exchange your euros for cryptocurrency
- Create the smart contract in the form of NFT → Leave a trace on the blockchain

Annexes : (smart contract)

Client_1.sol

```sol
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract Client_1 is ERC721Enumerable, Ownable {
  using Strings for uint256;

  string public baseURI;
  string public baseExtension = ".json";
  uint256 public cost = 0 ether;
  uint256 public maxSupply = 3;
  uint256 public maxMintAmount = 20;
  bool public paused = false;
  mapping(address => bool) public whitelisted;

  constructor(
    string memory _name,
    string memory _symbol,
    string memory _initBaseURI
  ) ERC721(_name, _symbol) {
    setBaseURI(_initBaseURI);
    mint(msg.sender, 3);
  }

  // internal
  function _baseURI() internal view virtual override returns (string memory) {
    return baseURI;
  }

  // public
  function mint(address _to, uint256 _mintAmount) public payable {
    uint256 supply = totalSupply();
    require(!paused);
    require(_mintAmount > 0);
    require(_mintAmount <= maxMintAmount);
    require(supply + _mintAmount <= maxSupply);
```

Client_1.sol

```sol
    if (msg.sender != owner()) {
        if(whitelisted[msg.sender] != true) {
          require(msg.value >= cost * _mintAmount);
        }
    }

    for (uint256 i = 1; i <= _mintAmount; i++) {
      _safeMint(_to, supply + i);
    }
  }

  function walletOfOwner(address _owner)
    public
    view
    returns (uint256[] memory)
  {
    uint256 ownerTokenCount = balanceOf(_owner);
    uint256[] memory tokenIds = new uint256[](ownerTokenCount);
    for (uint256 i; i < ownerTokenCount; i++) {
      tokenIds[i] = tokenOfOwnerByIndex(_owner, i);
    }
    return tokenIds;
  }

  function tokenURI(uint256 tokenId)
    public
    view
    virtual
    override
    returns (string memory)
  {
    require(
      _exists(tokenId),
      "ERC721Metadata: URI query for nonexistent token"
    );

    string memory currentBaseURI = _baseURI();
```

Annexes : (smart contract)

Client_1.sol

```
76      return bytes(currentBaseURI).length > 0
77          ? string(abi.encodePacked(currentBaseURI, tokenId.toString(), baseExtension))
78          : "";
79    }
80
81    //only owner
82    function setCost(uint256 _newCost) public onlyOwner {
83      cost = _newCost;
84    }
85
86    function setmaxMintAmount(uint256 _newmaxMintAmount) public onlyOwner {
87      maxMintAmount = _newmaxMintAmount;
88    }
89
90    function setBaseURI(string memory _newBaseURI) public onlyOwner {
91      baseURI = _newBaseURI;
92    }
93
94    function setBaseExtension(string memory _newBaseExtension) public onlyOwner {
95      baseExtension = _newBaseExtension;
96    }
97
98    function pause(bool _state) public onlyOwner {
99      paused = _state;
100   }
101
102   function whitelistUser(address _user) public onlyOwner {
103     whitelisted[_user] = true;
104   }
105
106   function removeWhitelistUser(address _user) public onlyOwner {
107     whitelisted[_user] = false;
108   }
109
110   function withdraw() public payable onlyOwner {
111     require(payable(msg.sender).send(address(this).balance));
112   }
```

Annexes : (smart contract)

Config.js

```
const basePath = process.cwd();
const { MODE } = require(`${basePath}/constants/blend_mode.js`);
const { NETWORK } = require(`${basePath}/constants/network.js`);

const network = NETWORK.eth;


const namePrefix = "Client_1";
const description = "Possessions du client_1";
const baseUri = "ipfs://QmPXNBfDgS6WF1KGoRJWqHHio3ptcjxGF8fYrLNCaWg5ec";

const solanaMetadata = {
  symbol: "YC",
  seller_fee_basis_points: 1000,
  external_url: "https://www.youtube.com/c/hashlipsnft",
  creators: [
    {
      address: "7fXNuer5sbZtaTEPhtJ5g5gNtuyRoKkvxdjEjEnPN4mC",
      share: 100,
    },
  ],
};


const layerConfigurations = [
  {
    growEditionSizeTo: 3,
    layersOrder: [
      { name: "Items" },
    ],
  },
];

const shuffleLayerConfigurations = false;

const debugLogs = false;

```

Config.js

```
const format = {
  width: 546,
  height: 1000,
  smoothing: false,
};

const gif = {
  export: false,
  repeat: 0,
  quality: 100,
  delay: 500,
};

const text = {
  only: false,
  color: "#ffffff",
  size: 20,
  xGap: 40,
  yGap: 40,
  align: "left",
  baseline: "top",
  weight: "regular",
  family: "Courier",
  spacer: " => ",
};

const pixelFormat = {
  ratio: 2 / 128,
};

const background = {
  generate: true,
  brightness: "80%",
  static: false,
  default: "#000000",
};

```

Annexes : (smart contract)

Config.js

```
const preview = {
  thumbPerRow: 5,
  thumbWidth: 50,
  imageRatio: format.height / format.width,
  imageName: "preview.png",
};

const preview_gif = {
  numberOfImages: 5,
  order: "ASC", // ASC, DESC, MIXED
  repeat: 0,
  quality: 100,
  delay: 500,
  imageName: "preview.gif",
};

module.exports = {
  format,
  baseUri,
  description,
  background,
  uniqueDnaTorrance,
  layerConfigurations,
  rarityDelimiter,
  preview,
  shuffleLayerConfigurations,
  debugLogs,
  extraMetadata,
  pixelFormat,
  text,
  namePrefix,
  network,
  solanaMetadata,
  gif,
  preview_gif,
};

```

Annexes : (smart contract)

Hardhatconfig.js

```
require("@nomiclabs/hardhat-waffle");


task("accounts", "Prints the list of accounts", async (taskArgs, hre) => {
  const accounts = await hre.ethers.getSigners();

  for (const account of accounts) {
    console.log(account.address);
  }
});

module.exports = {
  solidity: "0.8.4",
  paths: {
    artifacts: './src/artifacts'
  },
  networks: {
    ropsten : {
      url: "https://ropsten.infura.io/v3/255bfa87f9e74104878c308d8bd96b44",
      accounts: ['insérer la clé privé']
    }
  }
};

```

Deploy.js

```
const hre = require("hardhat");

async function main() {

  const Client_1 = await hre.ethers.getContractFactory("Client_1");
  const client_1 = await Client_1.deploy("Client_1,", "ITM", "ipfs://QmPQukyw1wUGT9iS8AGV5i5LtZ3DyGV6h4hra4JMt9kVBK/");

  await client_1.deployed();

  console.log("Client_1 deployed to:", client_1.address);
}



main()
  .then(() => process.exit(0))
  .catch((error) => {
    console.error(error);
    process.exit(1);
  });

```

Annexes : (cryptocurrency)

Token.sol

```sol
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract Token is ERC20 {
    constructor() ERC20("Token Source", "TSO") {
        _mint(msg.sender, 30 * (10**18));
    }
}
```

Hardhatconfig.js

```js
require("@nomiclabs/hardhat-waffle");


task("accounts", "Prints the list of accounts", async (taskArgs, hre) => {
  const accounts = await hre.ethers.getSigners();

  for (const account of accounts) {
    console.log(account.address);
  }
});


module.exports = {
  solidity: "0.8.4",
  paths: {
    artifacts: './src/artifacts',
  },
  networks: {
    hardhat: {
      chainId: 1337
    }
  }
};
```

Annexes : (cryptocurrency)

deploy.js

```
const hre = require("hardhat");

async function main() {
  const Token = await hre.ethers.getContractFactory("Token");
  const token = await Token.deploy();

  await token.deployed();

  console.log("Token deployed to:", token.address);
}

main()
  .then(() => process.exit(0))
  .catch((error) => {
    console.error(error);
    process.exit(1);
  });

```

Annexes : (dapp)

App.js

```
import {useState, useEffect } from 'react';
import { ethers } from 'ethers';
import Token from './artifacts/contracts/Token.sol/Token.json';
import './App.css';
import nft1 from './Images/4.png';
import nft2 from './Images/5.png';
import nft3 from './Images/6.png';

const tokenAddress = "0xe7f1725E7734CE288F8367e1Bb143E90bb3F0512";
const Client_address= "0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266"

function App() {

  const [balance, setBalance] = useState();

  useEffect(() => {
    getBalance();
  }, [])

  async function getBalance() {
    if(typeof window.ethereum !== 'undefined') {
      const accounts = await window.ethereum.request({method: 'eth_requestAccounts'});
      const provider = new ethers.providers.Web3Provider(window.ethereum);
      const contract = new ethers.Contract(tokenAddress, Token.abi, provider);
      const balance = await contract.balanceOf(accounts[0]);
      setBalance(balance / (10**18));
    }
  }
  return (
    <div className="App">
      <p className="txt4">Client_1: {Client_address}</p>
      <p className="txt1">Vous avez {balance} TSO</p>
      <div>
        <p className="txt2">Vos posséssions :</p>
        <img className="NFT" src={nft1} alt=""/>
        <img className="NFT" src={nft2} alt=""/>
        <img className="NFT" src={nft3} alt=""/>
```

App.js

```
        </div>
        <div>
          <p className="txt3">Objet 1</p>
          <p className="txt3">Objet 2</p>
          <p className="txt3">Objet 3</p>
        </div>
      </div>
    );
  }

  export default App;

```

Annexes : (dapp)

App.css

```css
.App {
  background-color: whitesmoke;
  width: 1600px;
  height: 1080px;
  background-size: 100% auto;
  padding: 50px;
}

.App-logo {
  height: 40vmin;
  pointer-events: none;
  background-color: #87CEFA;
}

@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}

.App-header {
  background-color: #87CEFA;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

.App-link {
  color: #87CEFA;
}

.NFT {
  width: 160px;
```

App.css

```css
  height: 150px;
  padding: 10px;
}

.txt1 {
  padding-top: 0px;
  font-family: Verdana;
  font-size: 20px;
}

.txt2 {
  padding-top: 60px;
  padding-bottom: 0px;
  font-family: Verdana;
  font-size: 20px;
}

.txt3 {
  padding: 65px;
  float: left;
  padding-top: 0px;
  font-family: Verdana;
  font-size: 13px;
}

.txt4 {
  padding-left: 400px;
  padding-top: 0px;
  padding-bottom: 60px;
  font-family: Verdana;
  font-size: 20px;
}

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
```