SOLIDITY

Solidity - Cours 3

Explorateur de blockchain

Maintenant que nous avons déployé notre premier contrat sur le testnet on va pouvoir aller l'inspecteur sur BSCscan. L'explorateur de blockchain est l'outil le plus utile pour un développeur blockchain, il va nous permettre d'aller visionner toutes les informations disponibles de la blockchain :

- Adresses
- Transactions
- Contrats
- Blocs, frais...etc.

Et maintenant c'est moi au tableau qui montre des trucs

On va s'amuser un peu

Je vais maintenant déployer un contrat de token et on va s'amuser à échanger ce token entre nous afin d'aller ensuite inspecter le contrat et essayer de comprendre ce qu'on peut lire.

Notre premier NFT

Un NFT c'est comme un token ERC20 sauf que chaque token est indivisible et unique, ensuite on peut rajouter plein de fonctions utiles.

Pour notre premier NFT nous allons utiliser le standard ERC721 mais quasiment rien coder.

Et là GitHub

```
pragma solidity ^0.8.0;

interface ERC721 {
    event Transfer(address indexed _from, address indexed _to, uint256 indexed
_tokenId);
    event Approval(address indexed _owner, address indexed _approved, uint256
indexed _tokenId);
    event ApprovalForAll(address indexed _owner, address indexed _operator, bool
_approved);

    function balanceOf(address _owner) external view returns (uint256);
    function ownerOf(uint256 _tokenId) external view returns (address);
    function safeTransferFrom(address _from, address _to, uint256 _tokenId, bytes
calldata data) external payable;
    function safeTransferFrom(address _from, address _to, uint256 _tokenId)
external payable;
    function transferFrom(address _from, address _to, uint256 _tokenId) external
payable;
    function approve(address _approved, uint256 _tokenId) external payable;
    function setApprovalForAll(address _operator, bool _approved) external;
    function getApproved(uint256 _tokenId) external view returns (address);
    function isApprovedForAll(address _owner, address _operator) external view
returns (bool);
}

interface ERC721Metadata {
    function name() external view returns (string _name);
    function symbol() external view returns (string _symbol);
    function tokenURI(uint256 _tokenId) external view returns (string);
}

interface ERC165 {
    function supportsInterface(bytes4 interfaceID) external view returns (bool);
}

interface ERC721TokenReceiver {
    function onERC721Received(address _operator, address _from, uint256 _tokenId,
bytes _data) external returns(bytes4);
}
```

InterPlanetary File System - IPFS

L'IPFS est un projet global de cloud décentralisé, on peut y faire héberger ou héberger un fichier gratuitement.

Tout le monde va installer IPFS mais pour le bien de l'exercice on va utiliser des liens déjà opérationnels, vous allez voir.

Suivez ce lien

InterPlanetary File System - IPFS

Les commandes à réaliser une fois IPFS installé :

- Ouvrir 2 terminaux
- 1T : ipfs init
- 1T : ipfs daemon
- 2T : ipfs add image.png

```
added QmTUr8zb51X3wHyYPUDPDSb3TuFqNdGC3p7CPGeJAhSw51 UMLA.jpg
```

Ensuite on obtient un hash et il nous suffit de taper https://ipfs.io/ipfs/<hash> pour retrouver notre image

InterPlanetary File System - IPFS

Nous utiliserons ces liens ci pour nos premiers NFTs :

- Lien 1
- Lien 2
- Lien 3

Notre premier NFT

Maintenant place au code, vous avez normalement téléchargé un fichier nommé MesNFT.sol. Dans ce fichier vous avez toutes les bases pour commencer une collection
NFT.

On se retrouve dans le contrat Collection où on va devoir définir toutes les variables et fonctions dont nous aurons besoin pour notre collection.

Sources :

- https://soliditydeveloper.com/erc-721
- https://www.quicknode.com/guides/solidity/how-to-create-and-deploy-an-erc-721-nft
- https://github.com/OpenZeppelin/openzeppelin-contracts/tree/release-v3.4/contracts
- https://betterprogramming.pub/generate-your-nft-metadata-11a878c082b9
- https://docs.opensea.io/docs/metadata-standards