BCDV1011 Design Patterns for Blockchain

Designing applications with tokens

Tokens and token factories

- Token types
- Where to use tokens
- Example utility token
- Example from use case

Tokens types

- Token types
  - Utility
  - Security
  - Native coin
  - Governance
  - Asset

Utility Tokens

- Needs to be an integral part of a system - not just a stand-in for a security
- Based on ERC20
- Added extra functions to provide utility
- Needs to be a functional part of a platform

Security Tokens

- Could be modified ERC20 or another standard
- Represents securities - shares in a company
- Should be securities regulators compliant
- Howie test
- Reg D 504c
- Qualified investors

Native coin

- Built-in for blockchain
- Used for transfer fees, mining or staking
- Bitcoin, Ether, XRP, Lumens
- Utility/Currency

Governance

- To fund/control a decentralized system
- Represents voting rights
- Paid dues
- Funds projects

Token uses - precision

- Used when current financial divisions are not enough
- Micro transactions
- Fractional ownership

Token uses - disintermediation

- Remove middle men
- Smart contracts to handle trust
- Holding funds in escrow
- Transparency for voting
- Explicit rules for processes

Token uses - risk reduction

- Trade risk for security
- Derivatives for hedging
- Fungible liquidity

Token uses - complimentary services

- Behaviour rewards
- Frequent customer
- Sell your info
- Watch ads

Token uses - crowdsourcing

- Collective funding of projects
- Voting mechanisms
- Escrow
- Payouts
- Prediction markets

Token uses - marketplace

- Decentralized markets
- Confederation of competitors
- Repository of goods for curation

Token uses - fractional ownership

- Large asset purchase funding
- Group ownership
- Securtize alternative assets
- Create liquidity in new markets

Token uses - digital representation

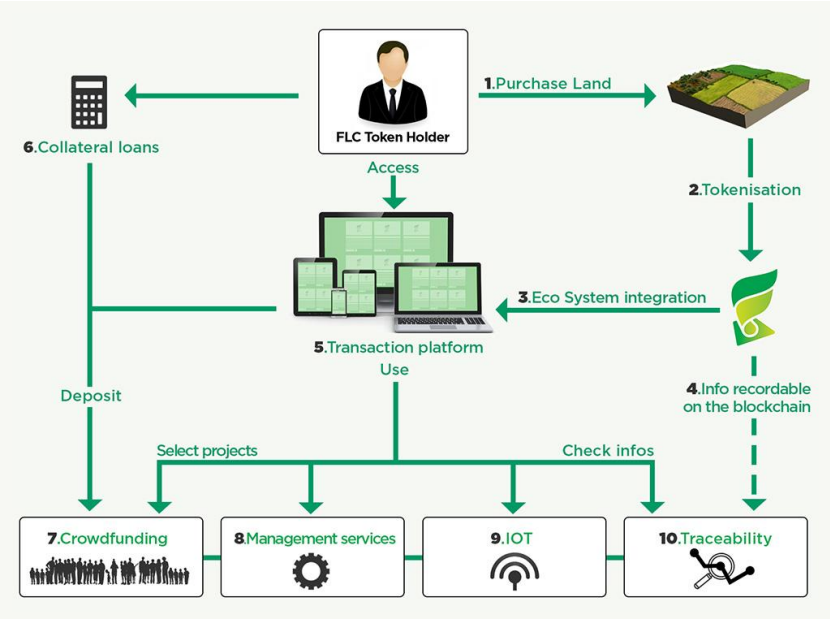
- Digital clone to real world goods
- Supply chain
- Industrial processes
- Stand in

Class examples

- Arts provenance
- Micro loans
- Real estate bidding
- Perishable goods supply chain
- Group Project Bidding
- Timeshare reservation
- Community content rewards
- Green energy tracker
- License plate registry

Fieldcoin - example

- Crowdfunding for agricultural land purchases



Fieldcoin - example

- https://github.com/Fieldcoin/Fieldcoin-ERC20
- Two tokens
  - FieldCoinSale - pre-sale token for special enticements
  - FieldCoin - token for working with the FieldCoin system
- FieldCoin is mintable, burnable
- FieldCoinSale is crowdsale, pausable
- Derived from StandardToken and ERC20

Token factory - use case

- Art provenance
- The goal
  - To show a clear line of ownership for a work of art from artist to current owner.
- People involved
  - Artist - originator of the art work
  - Owner - current owner of the artwork
- Requirements
  - Only one owner at a time
  - Artist can create new works of art

Token factory - use case

- The artist can have a collection of artworks that they have created and each artwork can have multiple limited edition prints

```
contract Artist {
    // Collection of artworks by this Artist
    mapping(uint => ArtWork) artworks;
    address artist;

    constructor() public {
        artist = msg.sender;
    }
```

Token factory - use case

- The artist can make new artworks and add them to their collection

```
function createArtwork(uint hashIPFS, string memory Name) public returns (ArtWork) {
   ArtWork artContract = new ArtWork(hashIPFS, Name);
   artworks[hashIPFS] = artContract;
   return artContract;
}
```

- Check to see if the artist is the originator of an artwork

```
function checkArtwork(uint hashIPFS) public view returns(bool) {
     if(artworks[hashIPFS] == ArtWork(0x0)) {
         return true;
     }
     return false;
}
```

Token factory - use case

- An artwork is stored on IPFS and we keep the hash and name of the artwork on the blockchain

```
contract ArtWork {
    // Detail of artwork
    address artist;
    string  name;
    uint  hashIPFS;
    address owner;

    constructor(uint ipfsHash, string memory artName) public {
        artist = msg.sender;
        name = artName;
        hashIPFS = ipfsHash;
        owner = artist;
    }
```

Token factory - use case

- An artwork is stored on IPFS and we keep the hash and name of the artwork on the blockchain

```
contract ArtWork {
    // Detail of artwork
    address artist;
    string  name;
    uint  hashIPFS;
    address owner;

    constructor(uint ipfsHash, string memory artName) public {
        artist = msg.sender;
        name = artName;
        hashIPFS = ipfsHash;
        owner = artist;
    }
```

Token factory - use case

- An artwork can change ownership

```
function setOwner(address newOwner) public {
    if(owner == msg.sender) {
        owner = newOwner;
    }
}
```