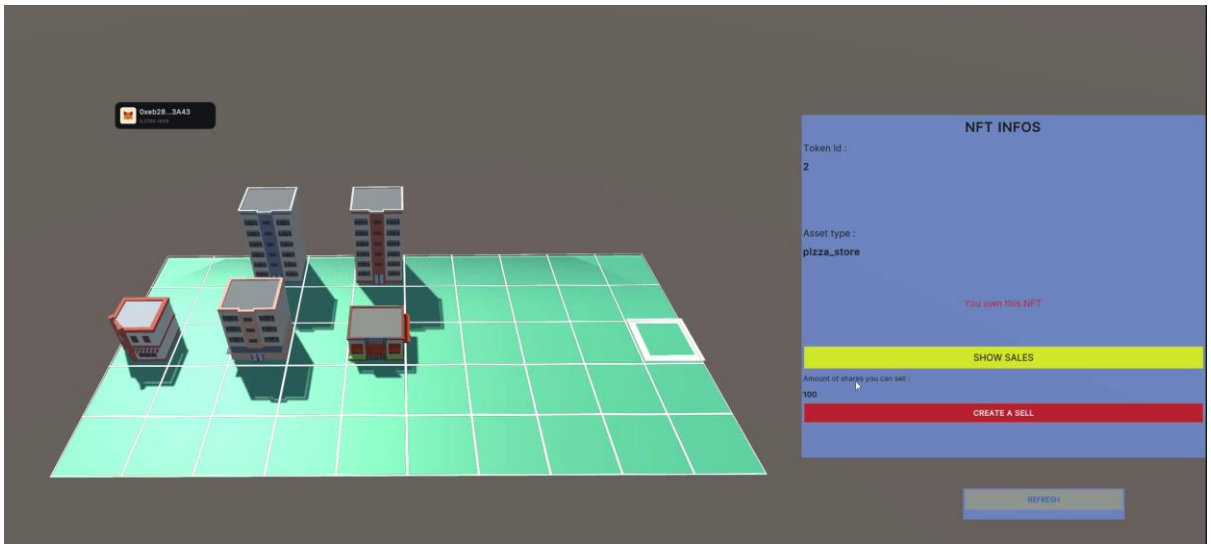
SwapAssets conception document

Swap Assets description



SwapAssets is a web3 **tokenisation** project of real estate assets.

In the backend, it's using the technologie of **blockchain** and a smart contract in it, to tokenize the real estate assets in the form of **NFTs**.

The smart contract manage **the life cycle of these NFTs**, such as :

- **buy**

- **sell**

- **fractionation**

On the frontend, the project use an **3D engine** to gives at users an overview of all the actual assets, in a given area.

In SwapAssets, users can :

- select an NFTs to get their infos.

- create a sale ticket to list their NFTs on the exchange with the conditions of their choice : **price** and the **shares** to sell.

- buy a sale ticket of an NFT to become the or one of the owners.

SwapAssets purposes and user stories

The purpose of SwapAssets and projects tokenizations in general, is to give more freedom to the two sides of a market : the buyers and sellers.

By putting the assets on the blockchain, they get more visibility than their local origins. On the DeFi, they are exposed to all the market liquidity.

Also, by using the fractionization of the assets, the base price of an asset is divided into multiple parts. The possibility to become an owner and invest, become more accessible.

At the end, on the seller's side, the sell can be completed faster with more competitives prices, and on the buyer side, the entry point to invest become more low.

Manage the lifecycle of an NFT :

UserStory 1: As a user, I want to be able to see existing NFTs, in order to be able to consult their information

UserStory 2: As an owner user, I want to be able to consult the NFTs that I own, in order to be able to create sales proposals for them

UserStory 3: As an owner user, I want to be able to choose to put up for sale part of an NFT that I own, so that I can only sell part of it and remain the owner

UserStory 4: As a purchasing user, I want to be able to purchase an NFT available for sale, in order to become the owner of this NFT

General functionals and technicals specifications

User Story 1

As a user, I want to be able to see existing NFTs, so I can view their information

Context : The smart contract to which swapAssets is linked acts as an NFT exchange. In this smart contract, it is necessary to define the information that describes each NFT.

Functionality :

As a user, when I view an NFT, I want to be able to view the information that describes it: its unique number, its asset type, its spatial coordinates.

Technical requirements:

R1 : Form of data

In the smart contract, the data linked to an NFT takes the form of a "TokenInfos" struct composed of the variables: - "tokenId" of type uint - "assetType" of type uint - "xPosition" of type int8 - "yPosition " of type int8

R2 : Gas optimization and safety

In order to find each "TokenInfos" struct, efficiently in terms of computation and therefore gas consumption, but also to avoid making the smart-contract unusable, an 'allTokensInfos' mapping is used with: - for key: the token id, and for value: the “TokenInfos” struct

R3 : Form of data

In the smart contract, it is necessary that an NFT has one or more owners. Each owner having a number of shares of the NFT and a number of shares that can be put up for sale. This information takes the form of the "OwnerInfos" struct composed of the variables: - "isOwner" of type bool - "owner" of type address - "shares" of type uint - "sharesForSelling" of type uint - "numTicket" of type uint

User Story 2

As an owner user, I want to be able to view the NFTs I own, so I can recognize the NFTs I own

Context : The smart contract to which swapAssets is linked acts as an NFT exchange. In this smart contract, it is necessary to define the information that describes the owners of each NFT.

Functionalities :

1 : As an owner user, when I view an NFT, I want to be able to know if I own it, so that I can recognize which NFTs I own.

2 : As the owner of an NFT, when I create a sale, I list my NFT partially or completely in the exchange, so I transfer my NFT to the smart-contract so that it can manage the sale.

Technical requirements :

R1 : Gas optimization and safety

In order to find each "OwnerInfos" struct, efficiently in terms of computation and therefore gas consumption, but also to avoid making the smart-contract unusable, a double mapping "nftsOwners" is used with: - key of the first mapping: the token id, value: next mapping - key of the second mapping: address of the owner, value: struct "OwnerInfos"

R2 : Find/Iterate the owners of an NFT

Since it is necessary to know the key of a mapping in advance to access its value, in the context of the double "nftsOwners" mapping, to access the information of an owner, you must know his address. To meet this requirement: creation of an “ownersList” mapping which contains an array listing the owner addresses of an NFT: - mapping key: tokenId, value: array

R3 : Find/Iterate NFTs owned by an owner

Use of a mapping which contains a table listing the tokenIds of the NFT held: - mapping key: owner address, value: table

User Story 3

As an owner user, I want to be able to choose to put up for sale part of an NFT that I own, so that I can only sell part of it and remain the owner

Context : The smart contract to which swapAssets is linked acts as an NFT exchange. In this smart contract, it must be possible to split an NFT so that it has several owners. When an owner wants to sell shares, he creates a sales ticket.

Functionalities :

1 : As an owner user, I can choose how much of an NFT I want to sell, in order to create a sales ticket that will allow me to remain owner.

It is possible for an owner to create as many sales tickets as he or she has a share in an NFT.

2 : As an owner user, I can cancel a sales ticket, in order to withdraw the shares from sale.

When a ticket is canceled, the theoretical shares initially put up for sale return to the owners.

3 : As the owner of an NFT, when I create a sale, I list my NFT partially or completely in the exchange, so I transfer my NFT to the smart-contract so that it can manage the sale.

Technical requirements :

R1 :

For security requirements, the sum of shares to be sold in an owner's sales tickets cannot exceed the number of actual shares he holds of an NFT. For this, in the 'OwnerInfos' struct, there are two variables linked to this case: - "shares": the shares held by an owner, they only change when a sale or purchase is concluded - "sharesForSelling": the theoretical shares available for sale, its value changes when a ticket is created/cancelled, when a sale or purchase is concluded.

R2 : Form of data

The data that describes a sales ticket is present in the “SaleInfos” struct. This struct is composed of the following variables: - "owner" of type address - "saleTicketNumber" of type uint - "price" of type uint - "sharesToSell" of type uint - "currentlyForSale" of type bool

R3 : Link between the tickets / its owner / the NFT

In order to link several tickets to its owner, himself linked to his NFT, the use of a triple “salesListing” mapping is required. This also to meet the security and performance requirements of the smart contract.

This triple mapping consists of: - key of the first mapping: tokenId, value: next mapping - key of the second mapping: the address of the owner, value: following mapping - key of the third mapping: the ticket number, value: struct " SaleInfos"

R4 : Find/Iterate sales tickets

Since it is necessary to know the key of a mapping in advance to access its value, within the framework of the "salesListing" tripkle mapping, to access the information of a sales ticket, you must know its ticket number. To meet this requirement: creation of a double mapping "ownersSalesTicketsList" which contains a table listing the ticket numbers of an NFT owner: - key of the first mapping: tokenId, value: next mapping - key of the second mapping: address of the owner, value: table containing sales receipts

R5 : Security requirements

When a user wishes to list an NFT for a sale then, the smart-contract must check: - if the NFT exists - if the user who calls the sale function is indeed one of the users of the NFT - if the number of shares available for sale that the owner holds is less than or equal to the number of shares he wants to sell in this ticket - if the sale price is greater than 0

R6 : When a sale is canceled, the owner's "sharesForSelling" shares: - are incremented by the value of the shares of the "sharesToSell" tickets of the "SaleInfos" struct

User Story 4

As a purchasing user, I want to be able to purchase an NFT available for sale, in order to become the owner of this NFT

Context: The smart contract to which swapAssets is linked acts as an NFT exchange. In this smart contract, it must be possible to buy an NFT from a sales ticket in order to become the owner of this NFT

Functionality :

1 : As a user, When I buy an NFT from a sales ticket, then I become the owner of this NFT

2 : When a purchase is made, then the shares put up for sale in the ticket are transferred from the seller's shares to the buyer's shares

3 : When a purchase is made, there can be four scenarios: - the seller is no longer the owner and the buyer is a new owner - the seller is no longer the owner and the buyer is not a new owner - the seller is still the owner and the buyer is a new owner - the seller is still the owner and the buyer is not a new owner

Requirements :

R1 : Security requirement

When a user wishes to buy an NFT, they call a purchase function in which they must provide: - the address of the seller - the token ID of the NFT - the sales receipt number

It is necessary to check in advance: - if the NFT exists - if the ticket number exists - if the user has sent enough funds to buy shares of the NFT

R2 : When a sale or purchase is concluded, the shares for sale "sharesToSell" of the struct "SaleInfos": - are incremented to the shares of the buyer, in the variable "shares" of the struct "OwnerInfos" - are incremented to the shares available for the sale, of the buyer, in the variable "sharesForSelling" of the struct "OwnerInfos" - are decremented to the seller's shares, in the variable "shares" of the struct "OwnerInfos" - are decremented to the seller's shares, in the variable " shares" of the struct "OwnerInfos"

Architecture