Blogchain

Blogchain makes your content unstoppable. Transform your blogs into smart contracts and posts into NFTs.

Mintbase Templates This is part of theMintbase Templates , a collection of templates that you can use to scaffold your own project

Project Walkthrough

Within the framework of blogchain, every blog manifests as an nft contract deployed from the Mintbase contract factory, while each individual blog post is uniquely represented as a non-fungible token (NFT).

NOTE: As a standard on Mintbase as we use the latest versions of Next.js we recommend using pnpm, but the package manager is up to your personal choice.

Run the project

install

```
pnpm i
```

run project

```
pnpm run dev
```

Create a Blog (deploy contract)

Step 1: check if the contract (blog) name already exists

Using@mintbase-js/data checkStoreName method we can check if the store already exists.

```
const

{ data : checkStore }

=

await

checkStoreName ( data . name , NEAR_NETWORKS . TESTNET ) ;

if

( checkStore ?. nft_contracts . length

===

0 )

{ ( ... ) }
```

Step 2: if contract name doesn't exist execute the deploy contract action with the instantiated wallet

Create deploy contract args usingmintbase-js/sdk deployContract method. This will deploy an NFT contract from themintbase contract factory

```
const wallet =

await selector . wallet ( ) ;

const deployArgs =

deployContract ( { name : data . name , ownerId : activeAccountId , factoryContractId :

MINTBASE_CONTRACTS . testnet , metadata :
```

{ symbol :

"" , } , }) ; We can then execute the deploy contract by passing in the wallet. If you wan't to learn about wallet connection check out the[wallet starter template]

await

execute ({ wallet } , deployArgs) ;

Create a Blog Post (mint an NFT)

Step 1: call storage method to upload file inserted by the user to arweave

Using[@mintbase-js/storage] uploadReference method we upload the nft image to arweave.

const metadata =

{ title : data . title , media : data . media , } ; const referenceJson =

await

uploadReference (metadata) ; const reference = referenceJson . id ;

Step 2: mint the nft in the contract (blog)

Create mint args using[mintbase-js/sdk] mint method.

const wallet =

await selector . wallet ( ) ;

const mintCall =

mint ({ noMedia :

true , metadata :

{ reference : reference , title : data . title , description : postContent , extra :

"blogpost" , } , contractAddress : data . contract , ownerId : activeAccountId , }) ; We can then execute the mint nft method

await

execute ({ wallet } , mintCall) ; note We populate the 'extra' field with the value 'blogpost' to subsequently filter the displayed NFTs and blogs in blogchain, ensuring that only blogs are included.

Get Data

Get blog posts (nfts) from a blog (smart contract)

Using[Mintbase GraphQL Indexer] we can fetch the nfts from a specific smart contract - to filter by blog post we use 'blogpost' as an extra field as explained in the previous step.

export

const

GET_BLOG_POSTS

= query GET_BLOG_POSTS(contractId: String!) { mb_views_nft_tokens(where: {extra: {_eq: "blogpost"}, _and: {nft_contract_id: {_eq: contractId}}}) { metadata_id title description media minted_timestamp } } ;

Get user blog posts (nfts)

export

const

GET_USER_POSTS

= query GET_USER_POSTS(accountId: String!) { mb_views_nft_tokens(where: {extra: {_eq: "blogpost"}, _and: {nft_contract_owner_id: {_eq: accountId}}}) { metadata_id title description media minted_timestamp } } ;

Get user blogs (smart contracts)

export

const

GET_USER_BLOGS

= query GET_USER_BLOGS(accountId: String!) { nft_contracts(where: {owner_id: {_eq: accountId}}) { id } } ;

Get latest blogs (smart contracts)

export

const

GET_LATEST_UPDATED_BLOGS

= query GET_LATEST_UPDATED_BLOGS { mb_views_nft_metadata(where: {extra: {_eq: "blogpost"}} distinct_on: [nft_contract_id, nft_contract_created_at] limit: 6 order_by: [{nft_contract_created_at: desc}, {nft_contract_id: desc}]) { nft_contract_id nft_contract_owner_id } } ;

Get latest blog posts (nfts)

export

const

GET_LATEST_POSTS

= query GET_LATEST_POSTS { mb_views_nft_tokens(where: {extra: {_eq: "blogpost"}} limit: 10 order_by: {minted_timestamp: desc}) { metadata_id title description media minted_timestamp minter nft_contract_id } } ;

Get blog post (nft) data

export

const

GET_POST_METADATA

= query GET_POST_METADATA(metadataId: String!) { mb_views_nft_tokens(where: {metadata_id: {_eq: metadataId}}) { metadata_id title description media minted_timestamp minter nft_contract_id } } ; Presently, this template exclusively functions within the testnet environment. To transition to a different network the configuration must be changed in and every 'testnet' instance.

Extending

This project is setup using Next.js + @mintbase/js You can use this project as a reference to build your own, and use or remove any library you think it would suit your needs.

Get in touch You can get in touch with the mintbase team using the following channels:

- Support:Join the Telegram
- Twitter:@mintbase Edit this page Last updatedonApr 29, 2024 byGuille Was this page helpful? Yes No Need some help?Chat with us or check ourDev Resources ! Twitter Telegram Discord Zulip

Previous Marketplace Frontend Next BOS Loader