2023 IEEE 18th International Conference on Computer Science and Information Technologies (CSIT) | 979-8-3503-6046-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/CSIT61576.2023.10324190

Modern approaches to interact Smart-Contracts in React.js development with ThirdWeb framework

Mykola Kolomoyets
PIT Department
Lviv Polytechnic National University
Lviv, Ukraine
mykola.kolomoiets @gmail.com

Yurii Kynash
PIT Department
Lviv Polytechnic National University
Lviv, Ukraine
https://orcid.org/0000-0002-3762-3215

Abstract — **Blockchain technology has gained significant popularity in recent years, with its ability to create secure, transparent and decentralized systems. React.js is a popular frontend development library, known for its performance and reusability. There are several common ways to create front-end dApps on blockchain networks, including using libraries such as Web3.js, web3-react, and ThirdWeb framework (later ThirdWeb). There are also several ways to create a complex dApp like NFT Marketplace, and the methods of architecture creating and implementing further projects have their own nuances and limitations that have to be discussed in this article.**

Keywords— React.js, Web3, ThirdWeb, smart-contract, NFT, MetaMask, BNB, BSC, minting.

I. Introduction

The range of applications for web is wide, including traffic monitoring [1], smart chef web applications [2], and service-oriented computing systems [3]. Blockchain is also used in web applications. In recent years, blockchain technology has become a revolutionary tool that provides secure, transparent, and decentralized systems for various industries [4,5]. Known for its flexibility, performance, and reusability, React.js, a popular front-end development library from Meta Corp [6], has become the best choice for modern web development. By combining the benefits of blockchain technology with React.js, developers can create efficient, secure, and easy to use decentralized applications (later called dApps) [7]. ThirdWeb is a comprehensive set of tools and utilities that allow developers to interact with various blockchain networks in their React.js projects, Easily integrate blockchain functionality into React.js applications.

II. Problem Statement

Blockchain technology is a decentralized distributed database that allows secure, transparent, and tamper-proof transactions without the need for a central authority. The technology is based on a network of nodes that maintain a shared ledger of transactions. Once transactions are recorded, they cannot be altered or deleted and are verified by a network of nodes, thus ensuring the integrity and authenticity of the data. Blockchain technology has a wide range of potential applications in modern web development, including:

- Decentralized applications (dApps): blockchain technology can be used to build decentralized applications running on a distributed network of nodes, providing high security and transparency [4].
- Identity management: blockchain technology can be used to build secure, decentralized identity management systems that allow users to manage and control their own identity data [8].
- Supply chain management: blockchain technology can be used to create transparent and secure supply chain management systems that can track and control products and deliveries in real-time [9].
- Digital payments: blockchain technology can be used to create secure digital payment systems that allow users to transact directly without intermediaries [10].

Common ways to build front-end applications on a blockchain network include using libraries such as web3.js, web3-react, and ThirdWeb:

1) Using Web3.js [11]. web3.js is a popular JavaScript library that provides a simple and convenient interface for interacting with the Ethereum network. To build a front-end dApp using web3.js, developers can use the injector to connect to the Ethereum network, interact with smart contracts, and use the library to manage user accounts. web3.js is a powerful tool for developing a contract wrapper, provider, injector, wallet, and front-end applications.

2) Using web3-react [12] web3-react is a library that provides a simple and flexible way to connect front-end applications to ethereum and other blockchain networks. web3-react is a web3 connector, web3-react also supports a number of authentication methods including MetaMask, WalletConnect, and other wallet providers.

3) Using ThirdWeb [13] ThirdWeb is a comprehensive set of tools and utilities for interacting with various blockchain networks, including Ethereum, in the React.js project ThirdWeb is a contract installer, provider, and provider that makes it easy to integrate blockchain functionality into React.js applications by offering a set of features, including a wallet. ThirdWeb also includes a set of pre-built user interface components, making it easy to build front-end dApps without extensive knowledge of blockchain technology.

All three solutions can be applied to build dApps like the NFT Marketplace [14]. However, the question arises. How comfortable is it for developers when using each of the above solutions and which lib is better? A typical NFT marketplace has blockchain-related features such as acquiring (owned) listings, selling listings, buying listings, bidding in auctions, approving/rejecting bids for listings, removing listings from sale, NFT minting, etc.

III. Software Solution Architecture Design

We need to investigate the process of creating the complex system of smart-contracts (later SC) interacting and managing on the client side. Before looking at library solutions, it is necessary to define a common system that needs to be created to interact with the SC system from the browser API (e.g. Metamask [15] browser extension). These are the client side (browser API, front-end JS/TS code, browser extensions), the blockchain (chain provider or Metamask API), and the chain network (currency network

BSC (mainnet and testnet) [16]). The current common layer management scheme is described in Fig 1.

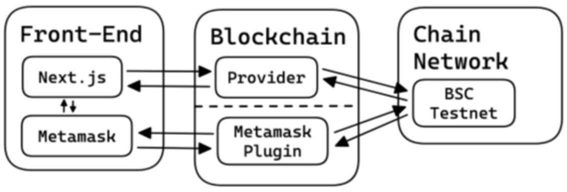


Fig. 1. Common dApp architecture layers.

Since SC is distributed and stored on the blockchain network (chain network layer), the only task of the front-end developer is to create a client-side SC interaction system (later SCIS). Since the React.js library has the large community of all front-end (then FE) tools, there are also solutions specifically for this library, such as web3-react and ThirdWeb. Web3.js is a basic JavaScript solution, this library can be implemented with any FE tool (jQuery, Vue, Solid, etc.). You can create a class instance, extract a contract instance from it, and perform any action using the contract, its ABI, and parameters. The web3.js SCIS algorithm is shown in Fig. 2.

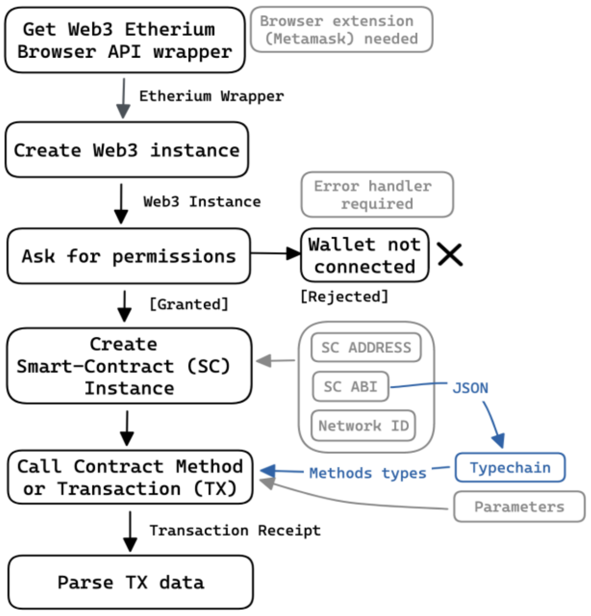


Fig. 2. Web3.js scheme to implement Web3.js SCIS on FE side.

Proper Web3 development needs creating the type-safe SCIS - the system where every SC method, its payload definitions, and response declaration are defined as Typescript type declarations or as an ABI-like JSON. Creating a type-safe SCIS requires a few additional steps:

- Adding the SC ABI JSON file to the project assets;
- Adding a typechain library to convert SC ABIs to Typescript types;
- Creating your own microservices (regular classes) [17] on the FE side to manage SC methods in a type-safe and correct way.

An ABI (Application Binary Interface) is a JSON file that defines all public methods, input parameter types, and output type data. When implementing such a system for the first time, it takes time to set up microservice endpoints for SC ABI methods. They need to be created from scratch for each new project, and code generation is not currently possible. When the ThirdWeb framework is used in all blockchain projects, this procedure will not make sense because ThirdWeb already has its own SDK which might be used as a regular JavaScript library, or as a React hooks kit. An almost similar way to integrate a custom SCIS is the web3-react library. The diagram in Fig. 3 [15] describes the implementation flow of such an approach.

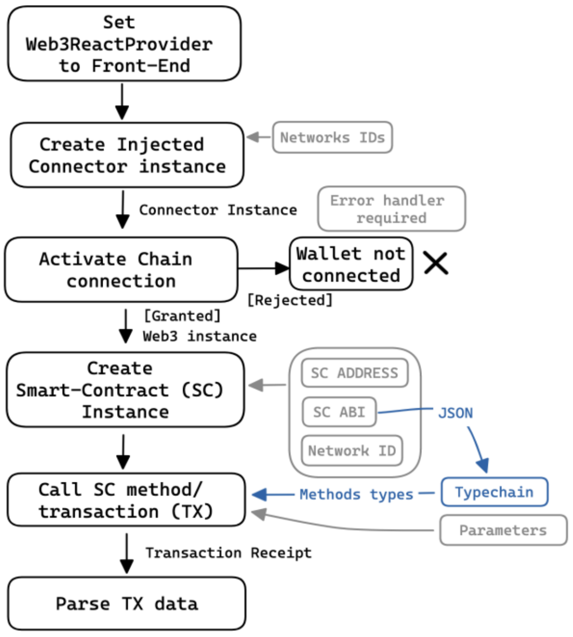


Fig. 3. Web3-react scheme to implement SCIS on FE side.

This approach is more efficient and reliable because some moments from the previous approach are already done:

- Built-in React-Context provider.
- Built-in hooks for all actions like activation, disconnection, getting an instance of the web3 library, etc.

However, a Typechain needs to be configured, SC ABI JSON data imported and a custom microservice created.

The most efficient and simplest way to create a SCIS is the ThirdWeb framework approach (see Fig. 4) [15]. It has context providers under the hood and allows hooks to be connected (and meta-masked) for each provider. The core functionality of such an approach is a built-in use contract hook that only requires SC addresses, and by setting the SC type, the data needed later (type of SC entity, method with type parameter, transaction receipt) can be analyzed and obtain the data needed later (type of SC entity, method with type parameter, transaction receipt). Also, other built-in hooks can be used to perform marketplace and NFT transactions. Each transaction has its own hook that requires an SC instance from the useContract hook. The main disadvantage is that built-in hooks cannot perform mint transactions. The SC is built according to Web3 core SC standards: ERC720 and ERC721 [18] are the main standards defined directly in the SC Solidity code. In particular, the mint is one of the operations that can be written according to the client's request and from which different types of input data can be obtained.

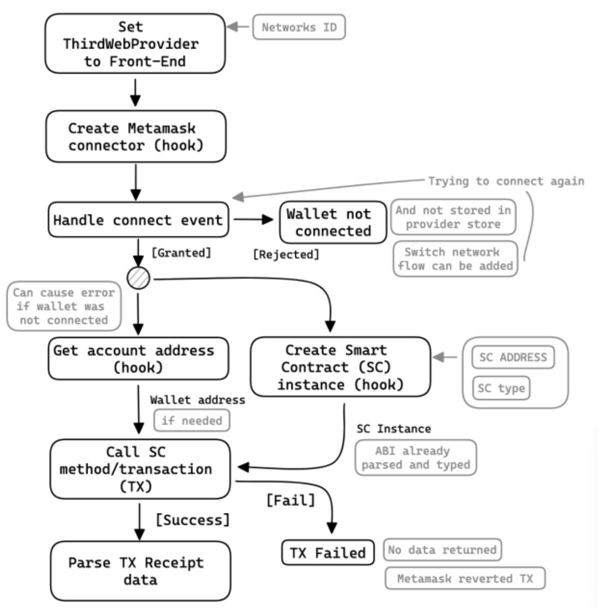

Set ThirdWebProvider to Front-End
Networks ID
Create Metamask connector (hook)
Trying to connect again
Handle connect event
Wallet not connected
And not stored in provider store
Switch network flow can be added
[Granted]
[Rejected]
Can cause error if wallet was not connected
Get account address (hook)
Create Smart Contract (SC) instance (hook)
SC ADDRESS
SC type
Wallet address
if needed
SC Instance
ABI already parsed and typed
Call SC method/transaction (TX)
[Fail]
[Success]
Parse TX Receipt data
TX Failed
No data returned
Metamask reverted TX

Fig. 4. ThirdWeb scheme to implement SCIS on the FE side.

The next section examines the software usage scenario of these technologies.

IV. System Realization

When building the web3-react implementation of SCIS, first configure the provider and add the web3-react context provider wrapper to the Next.js application [19]. Then, activate the binding: the connector reads the global Ethereum wrapper embedded in the browser API with the Metamask extension installed; retrieves the ABI data of the SC parsed by web3.eth, and integrates the service instance. Finally, it creates a transaction and invokes the method. Alternatively, SCIS can be used with the ThirdWeb approach [13]. ThirdWeb and web3-react providers each have their own approach to handle wallet connections, although they want to do ThirdWeb-based translation, If the wallet connection is web3-react based, SCIS will fail in the first phase of the wallet connection.

If there is no valid user wallet address or metamask data from the browser, the system will be blind – the system has no possible ways for the wallet to be connected. It could be due to several reasons: by the time the wallet becomes deleted, the user has the proper browser extension, but has no wallet signed in or the user has no proper browser extension. web3-react connectors can be enabled on the page or before the actual action is required. In the case of Mint, there can be a special hook that receives the contract data and parses the ABI; if there is a ThirdWeb-based wallet address and the component uses this useMint hook link, the activation action is executed when the hook is called. The NFT can then be minted without any impediment on the FE side [6].

In the system example, all approaches are implemented in separate parts of the final application, the PunkPanda NFT Marketplace [20]. The end user can connect to the Metamask crypto wallet if the Metamask extension for browsers is installed; the ThirWeb library has its own components for connection, but you can also create your own layout. Metamask then offers to select the wallet element to connect to the app. Immediately after confirmation, the app processes the wallet connection success event by reading the message from the browser extension.

Using only the wallet address, the app can get information about the current user's current status in the marketplace. The app also displays the user's wallet address next to the profile button in the header. Any user can view the NFTs available in the market. However, the Own NFTs feature is only available to logged-in users: The list of NFT items is displayed in a grid view with the minimum data required to be visible at a glance: ID, NFT image, and a brief description of the NFT. Each user can be directed to a separate page to view the NFT in more detail. The NFT image and treatment are also displayed. In addition, they can observe the current transaction status of the NFT (e.g. whether it is for sale, whether it has been sold, or whether it is the highest bidder in an auction). If the user is logged in and on the page of the NFT they own, they can put the NFT up for sale/auction or approve someone else's bid. The selling logic is also implemented in the ThirdWeb hook (Fig. 5).

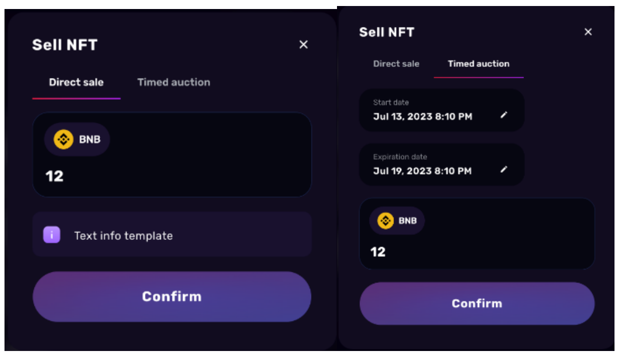
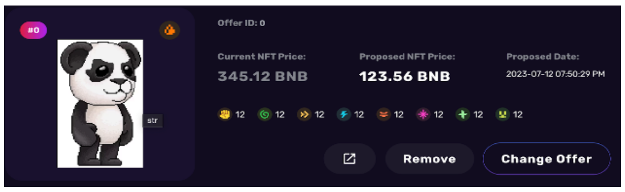


Fig. 5. Putting NFT on sale and on the auction.

Logged-in users can also view existing offers. There are two types of offers: received (offers submitted by other users to their NFT) and proposed (offers submitted by the current user to other NFTs) (Fig. 6).


#0

Offer ID: 0

Current NFT Price: 345.12 BNB

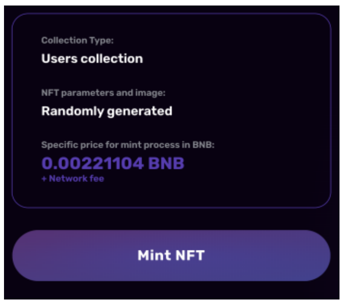
Proposed NFT Price: 123.56 BNB

Proposed Date: 2023-07-12 07:50:29 PM

12 12 12 12 12 12 12 12

Fig. 6. The offer item.

If due to user actions, Metamask platform issues, or the user tries to view the offer page without the wallet attached, an error screen pops up: NFT minting (Fig. 7) uses the regular web3 library for the browser environment, not the ThirdWeb hook.

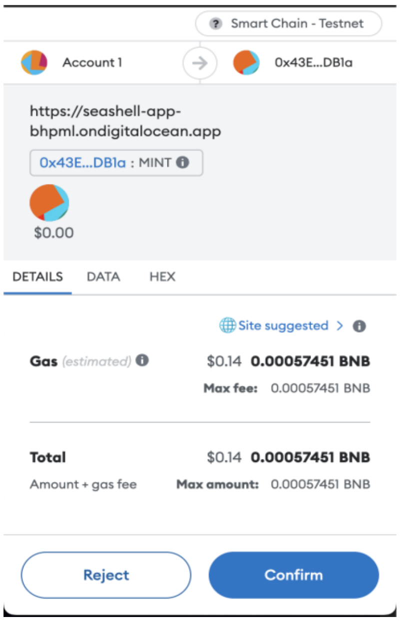

Collection Type:
Users collection

NFT parameters and image:
Randomly generated

Specific price for mint process in BNB:
0.00221104 BNB
+ Network fee

Fig. 7. Mint NFT block.

When the button is clicked, the user has the option to approve or cancel the mint (Fig. 8). If approved, the amount of gas and additional currency will be transferred from the user's wallet to the NFT Mint smart contract and the generated NFT will belong to the current user.


Smart Chain - Testnet

Account 1 → 0x43E...DB1a

https://seashell-app-bhpml.ondigitalocean.app

0x43E...DB1a : MINT

$0.00

DETAILS DATA HEX

Site suggested >

Gas (estimated) $0.14 **0.00057451 BNB**

Max fee: 0.00057451 BNB

Total $0.14 **0.00057451 BNB**

Amount + gas fee **Max amount:** 0.00057451 BNB

Fig. 8. Metamask pop-up window to confirm NFT Mint by the Smart Contract.

Blockchain can be used to build various web applications [21, 22]. As we saw in the current example, it is possible to implement full Create-Read-Update-Delete (CRUD) solutions that use SCs as the data exchange interface.

V. Conclusion

Blockchain has become very fashionable in the last few years and is a trending infrastructure technology in today's world. web3-react and ThirdWeb are considered excellent libraries for building NFT marketplaces in Next.js. Each library has its own unique features and advantages for specific needs and requirements. web3-react provides a simple and intuitive way to manage Ethereum accounts, interact with smart contracts, and process transactions. Its modular architecture and support for multiple Ethereum providers make it a popular choice for developers looking to build decentralized applications quickly and efficiently. ThirdWeb, on the other hand, offers a more comprehensive solution with web3.js integration as well as a number of blockchain-related tools and features such as IPFS and DID support, and ThirdWeb focuses on usability and ease of use, a big advantage for developers who are new to blockchain technology or want to simplify the development process. In conclusion, the choice between web3-react and ThirdWeb depends on the specific use case and requirements. If the system is not implemented under the reactive framework - you should use Web3.js. However, while using reactive frameworks like Next.js and Astro - it would be more suitable to use ThirdWeb solutions that can provide everything the development process needs, instead of writing your own system implementation and configuring methods from scratch.

References

[1] O. Petrushynskyi, Y. Kynash, Y. Miyushkovych, R. Martsyshyn and N. Kustra, "Web-Oriented Information System for Lviv Transport Data Monitoring," *2022 IEEE 17th International Conference on Computer Sciences and Information Technologies (CSIT)*, Lviv, Ukraine, 2022, pp. 450-453, doi: 10.1109/CSIT56902.2022.10000771.

[2] S. Chaudhari, R. Aparna, V. G. Tekkur, G. L. Pavan and S. R. Karki, "Ingredient/Recipe Algorithm using Web Mining and Web Scraping for Smart Chef," *2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, Bangalore, India, 2020, pp. 1-4, doi: 10.1109/CONECCT50063.2020.9198450.

[3] R. A. Elghondakly, S. M. Moussa and N. L. Badr, "The DSW Model: An Efficient Approach for Single Web Services Modeling," *2021 Tenth International Conference on Intelligent Computing and Information Systems (ICICIS)*, Cairo, Egypt, 2021, pp. 500-505, doi: 10.1109/ICICIS52592.2021.9694204.

[4] J. P. de Brito Gonçalves, G. Spelta, R. da Silva Villaça and R. L. Gomes, "IoT Data Storage on a Blockchain Using Smart Contracts and IPFS," *2022 IEEE International Conference on Blockchain (Blockchain)*, Espoo, Finland, 2022, pp. 508-511, doi: 10.1109/Blockchain55522.2022.00078.

[5] X. Wang *et al.*, "Application of data storage management system in blockchain-based technology," *2023 IEEE 2nd International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA)*, Changchun, China, 2023, pp. 1437-1440, doi: 10.1109/EEBDA56825.2023.10090564.

[6] React.js documentation, [online] Available: https://reactjs.org/

[7] R. Dange, A. Sawant, A. Chavan, P. Bhardwaj and A. Bundele, "Decentralized Fundraising Application Using Blockchain," *2022 IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS)*, Pune, India, 2022, pp. 1-5, doi: 10.1109/ICBDS53701.2022.9935998.

[8] S. R. Niya, J. Willems and B. Stiller, "A Case Study of a Blockchain-GDPR Adaptation," *2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, Shanghai, China, 2022, pp. 1-3, doi: 10.1109/ICBC54727.2022.9805553.

[9] M. Sigwart, P. Frauenthaler, C. Spanring, M. Sober and S. Schulte, "Decentralized Cross-Blockchain Asset Transfers," *2021 Third International Conference on Blockchain Computing and Applications (BCCA)*, Tartu, Estonia, 2021, pp. 34-41, doi: 10.1109/BCCA53669.2021.9657007.

[10] T. Duan, S. Sun, C. Yang, Y. Sun and X. Zhu, "Sharded Blockchain Architecture Oriented to Multilateral Collaboration of Source-Grid-Load- Storage," *2023 IEEE International Conference on Control, Electronics and Computer Technology (ICCECT)*, Jilin, China, 2023, pp. 1050-1054, doi: 10.1109/ICCECT57938.2023.10140213.

[11] Web3.js, [online] Available: https://web3js.readthedocs.io/en/v1.8.2/

[12] React Tutorial, [online] Available: https://www.w3schools.com/react/

[13] ThirdWeb: the complete web3 development framework, [online] Available: https://thirdweb.com/

[14] NFT, [online] Available: https://docs.nft.com/

[15] Metamask, [online] Available: https://metamask.io/

[16] BSC, [online] Available: https://docs.bnbchain.org/docs/BSCtestnet

[17] Microservice, [online] Available: https://microservices.io/

[18] ERC720, ERC721, ERC1155 - Differences, [online] Available: https://www.leewayhertz.com/erc-20-vs-erc-721-vs-erc-1155/

[19] Next.js, [online] Available: https://nextjs.org/docs/getting-started

[20] PunkPanda NFT Marketplace[In development], [online] Available: https://seashell-app-bhpml.ondigitalocean.app/

[21] P. M. Mohan, V. Balachandran, O. Z. Quan, J. P. Z. Xin and D. M. Divakaran, "NFT-Merit: An NFT-based Module Credit Management System on Ethereum Blockchain," *2022 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*, Hung Hom, Hong Kong, 2022, pp. 472-476, doi: 10.1109/TALE54877.2022.00083.

[22] R. J. Anandhi, A. Bhakta, A. Purswani, K. Karan and A. Mishra, "NFT Club – A NFT Marketplace," *2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, India, 2023, pp. 157-161, doi: 10.1109/ICACCS57279.2023.10112704.