

Building A Decentralized Application on the Ethereum Blockchain

Ruhi Taş
Computer Engineering
Ankara University
Ankara, Turkey
ruhitas@yahoo.com

Ömer Özgür Tanrıöver
Computer Engineering
Ankara University
Ankara, Turkey
ozgur.tanriover@ankara.edu.tr

Abstract—Blockchain technology is being used in many areas, during last few years. Furthermore, different application opportunities are still been investigated. Blockchain relies on and permits to implement the concept of Decentralized Application (DApps). This makes the applications more transparent, distributed and flexible. The complexity of blockchain and its integration problems require expertise that differs from traditional application development approaches. Within this context, this paper presents our experience in building a DApp with one of the most popular blockchain based platforms called Ethereum.

Keywords—Blockchain, Decentralized Application, Smart Contract, Ethereum

I. INTRODUCTION

Recently, blockchain has been gaining interest from both industry and researchers. The concept of blockchain was first introduced by Satoshi Nakamoto in 2008 [1]. Blockchain's first field of application has been lash coins. After the implementation of smart contracts, blockchain applications are now being implemented in all areas. Ethereum has recently become the most popular blockchain platform. Ethereum DApps and smart contracts are used together. The aim of Ethereum is to create an alternative protocol to create decentralized applications. It is claimed that it offers the advantage of rapid code development time with security features.

As more and more companies and developers try to discover new business models by using the blockchain infrastructure, decentralized applications and smart contracts are becoming more and more on the agenda [2]. A large amount of data was generated with the use of DApp. There are also some difficulties in creating an application based on the Blockchain platform. Many different development environment options are offered separately on the internet. In this paper, we aim to provide our experience to new comers who plan to develop applications with blockchain etherium.

Sites such as DApp review [3] and State of the DApps [4] keep statistics of various DApp applications. When the applications in these sites are examined, it is seen that many applications in different fields continue to be developed gradually.

Table 1 shows the variety of applications developed in different areas. The most developed application was found to be in game applications [3].

TABLE I. DAPP CATEGORIES IN ETHEREUM

Type of Dapp	# of Apps
Game	526
Casino	326
Exchange	95
High Risk	279
Finance	22
other	325
Social	40

It is argued that Blockchain will lead to emergence of a completely new decentralized class of applications. If a new collaboration and incentive ecosystem is developed truly distinctive, creative and useful DApps may emerge [3],[4],[7]. In this context, we focus on architectural components for blockchain oriented applications and present an example smart contract and application development environment for new comers.

II. DESIGN CONSIDERATIONS FOR DAPP DEVELOPERS

Modern web applications are based on an infrastructure in which single point of failure naturally exist. DApps aims to alleviate these issues by distributing critical components that store data or parts of infrastructure between various peers or nodes. That's why, when designing DApp, security, cost, usability features should be taken into consideration [19].

DApp applications are required to contain some features. Such as; [10]

- Better performance (low latency, high throughput),
- Reasonable low transaction fee,
- Flexible maintainability,

DApps should not store or replicate user data. The point of Blockchain security is based on not having any central vulnerability. Users should use the application as the sole administrator of their independent and unique identifiers. It reduces reliance on central authorities. The application should not be too complicated and should have a simple interface. Unnecessary coding should be avoided; it increases the cost and may create a security weakness.

It is noted by some researchers that the use of blockchain technology is not always useful in preparing a secure software system [8]. Developers should first learn the basic concepts of blockchain and current blockchain technologies

and applications. It is stated that it would be more appropriate to start working on application development [9].

In our context, a DApp is a "block chain enabled" web application running on a peer-to-peer computer network, rather than a single server. It includes both front end and back end and works independently on all nodes. Typically, other applications also use the same technology to create the front-end interface. The only critical difference is a smart contract that connects the application with a blockchain network. A smart contract is a self-executing contract that contains the conditions for writing the contract between the buyer and the seller directly in line with computer codes. Smart contracts allow reliable transactions and agreements to be made between different, anonymous parties, without the need for a central authority, a legal system, or an external implementation mechanism [7].

III. SYSTEM SETUP AND IMPLEMENTATION

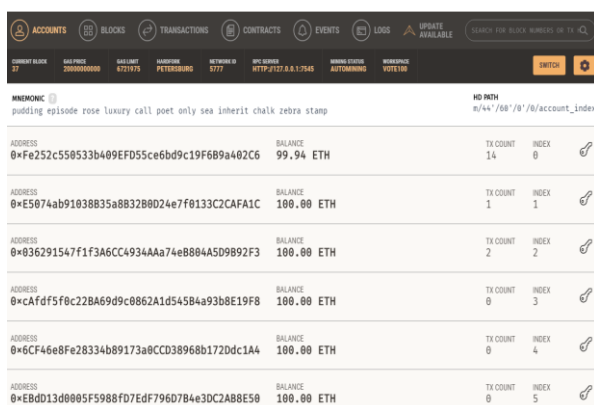
In this paper, we focus on architectural design for blockchain-oriented applications for beginners and propose an approach to decide which elements of local application architecture can benefit from the use of Ethereum. The first dependency we need is to set up is the Node Package Manager (NPM) which comes with Node.js [6]. Truffle, on the other hand, is a development environment that tests the framework and pipeline presence for Ethereum [5]. Truffle comes as a bundle with a local blockchain development server that launches automatically. Truffle includes:

- Built-in smart contract gathering, linking, deployment, and binary managing
- Interactive console for direct contract communication.
- The instant rebuilding of assets during development.
- Automated contract testing

Truffle installation code is made with the command:

```
npm install -g truffle
```

After the installation, we can use truffle either to compile, migrate and test. Also it can be used to compile contracts or arrange those contracts to the network and run their associated unit tests.



ACCOUNTS	BLOCKS	TRANSACTIONS	CONTRACTS	EVENTS	LOGS	UPDATE AVAILABLE
<p>HD PATH: m/44'/0'/0'/0'/0/account_index</p> <p>mnemonic: pudding episode rose luxury call poet only sea inherit chalk zebra stamp</p>						
ADDRESS	BALANCE	TX COUNT	INDEX			
0xFe252c558533b409EF055ce6bd9c19f689a402C6	99.94 ETH	14	0			
ADDRESS	BALANCE	TX COUNT	INDEX			
0xE5074ab91038835a88328024e7f01332CFA31C	100.00 ETH	1	1			
ADDRESS	BALANCE	TX COUNT	INDEX			
0x036291547f1f3A6CC4934AA74eB884A509892F3	100.00 ETH	2	2			
ADDRESS	BALANCE	TX COUNT	INDEX			
0xcAfd5f8c22BA69d9c8862A1d545B4a93b8E19F8	100.00 ETH	0	3			
ADDRESS	BALANCE	TX COUNT	INDEX			
0x6CF46e8Fe2834b89173a0CCD38968b172Ddc1A4	100.00 ETH	0	4			
ADDRESS	BALANCE	TX COUNT	INDEX			
0xEbDd13d0005F598fD7Edf796D7B4e3DC2AB8E50	100.00 ETH	0	5			

Fig. 1. Ganache Account address and balances

Secondly, Ganache is a personal blockchain for Ethereum development and this can be used to deploy contracts, develop applications, and run tests. A GUI for the server displays transaction history and chain state as shown in Fig. 1. Ganache comes with 100 accounts ready for testing. In fig. 1

- Workspace sets the workspace name and shows the currently linked Truffle projects. We can choose what we want from the loaded smart contract list Fig.1 show two workspace deployed in Ganache.
- The server shows details about the network connection, including hostname, port, network ID, and whether to automatically mine each transaction into a block. Default IP and port address localhost:7545 or 127.0.0.1:7545.
- Accounts & Keys set details about the number of accounts created, and whether to use a specific mnemonic or let Ganache generate its own.
- Chain sets details about the actual workings of the generated blockchain.
- The balance of the accounts can be seen. All transactions are examined in detail from the transaction section.

Thirdly, solidity is used to write smart contract in DApps applications [11], [14]. Solidity has object-oriented features and is primarily designed to write contracts at the Ethereum [19]. Remix and Visual Studio code are suitable platforms for testing and debugging of solidity programs. While using Remix in the web browser, VS code can be used offline on a computer with installation. The remix is an open-source tool that helps to write Solidity contracts straight from the browser as shown in figure 2. It facilitates writing and debugging Solidity contracts. Written in JavaScript, Remix maintains usage both in the browser and locally. Remix is used in testing, debugging and deploying smart contracts [12],[13].

Previously added solidity codes can be compiled by selecting them from the left-hand window. By selecting Remix environment, it is possible to test on different platforms. Microsoft Visual Studio Code has a plugin for solidity development. With this plugin, code writing and debugging operations are facilitated.

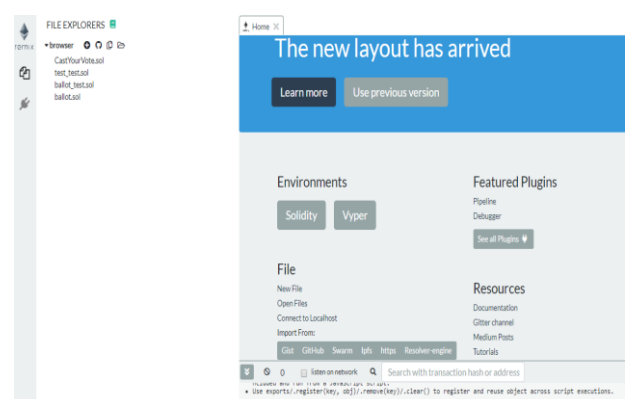


Fig. 2. Solidity Remix Web page [12]

The helloworld.sol code prepared as an example can be compiled and checked in remix and VS code.

```
Helloworld.sol

pragma solidity ^0.4.18;

contract HelloWorld {
    enum LANGUAGE {EN, TR, DE}

    string english = "Hello World";
    string turkish = "Merhaba Dünya";
    string german = "Hallo Welt";

    function HelloWorld() public {

    }

    function sayHello(LANGUAGE lang) public view returns(string) {
        if (lang == LANGUAGE.EN) {
            return english;
        } else if (lang == LANGUAGE.TR) {
            return turkish;
        } else {
            return german;
        }
    }
}
```

First statement declares the Solidity compiler version, which is supported by our smart contract. The structure used here enforces that version 0.4.18 is needed as the minimum Solidity compiler [20].

In the second part, we define the equivalents of the phrase “Hello World” in all three languages. Here we will use the data type “string.” In this way, the variables defined in the contract are called state variables, the values of which are permanently kept in the storage area of the contract on the blockchain.

Function definitions start with the keyword “function”. This is followed by the function name and the values required for its operation (function parameters). Here, we also specify the definition of “public”, indicating that this function can be called inside and outside the block.

Before distributing the smart contract, we need to compile the robustness code into the Bytecode for Ethereum Virtual Machine (EVM) with the following;

```
truffle compile
```

After compiling the solidity code, we obtain two objects:

- Byte code / EVM code
- ABI (Application Binary Interface)

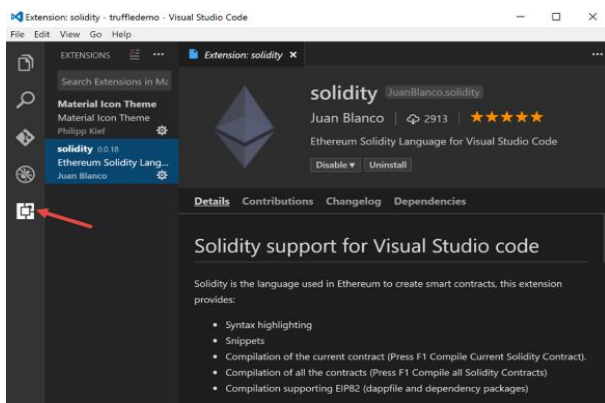


Fig. 3. Visual Studio Solidity Adds on

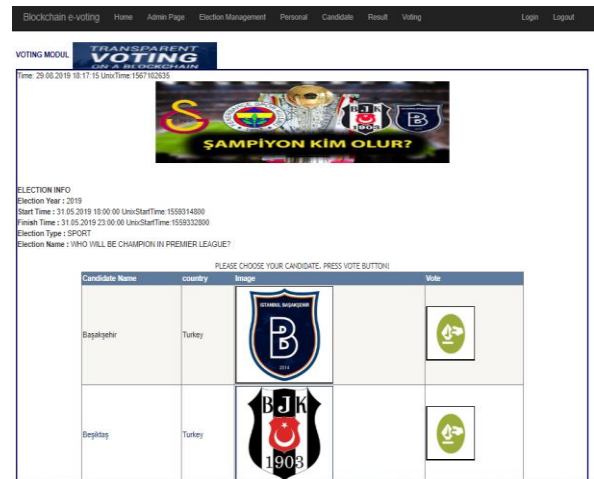


Fig. 4. Blockchain Web Application

The byte code is the EVM code in the blockchain network. ABI identifies the actions that we interact with the functions of the smart contract. Locally compiled solidity can be loaded into the Ganache test platform. The compiled application can be sent to the Ganache server with,

```
truffle deploy --reset --network ganache

command.
```

After compiling the smart contract, the resulting ABI must be integrated into the desired platform. ABI defines an interface between object components for a particular architecture between software components. ABI is a set of low-level and detailed technical rules to make separately compiled modules work together. As long as this compliance is maintained, the background of the software components interacting with them changes, but they continue to be used without the need for recompilation. The aim of this interface was to make machine code portable across the blockchain.

Then the web or mobile interfaces that will communicate with this application can be developed in the desired languages. In web development, general web programming languages such as ASP.NET, PHP can be preferred.

Functions called through the development and generated transactions can be debugged via ganache. Information about the account calls which function, and other logs can be checked via Ganache GUI as shown in Figure 1.

The blockchain application we have developed with Asp.net is designed for one-time voting for each account. "Who will become the Champion?" as shown in figure 4. The results of the voting were held on the Ganache blockchain. Block time is used to run the vote in a specific time period. Each account is allowed to vote only once. As votes are cast, the falls in the balance of the account can also be controlled from the Ganache interface. To release our app to the end user, we can install it on Heroku [16], Microsoft Azure [18], and Amazon Blockchain services [17], which also support various types of application development environments.

IV. CONCLUSION AND FUTURE WORK

In different fields, developers who are considering developing applications on blockchain platforms have recently turned interest to Ethereum infrastructure. We have presented the desirable characteristics of DApps. Then, a possible combination of tools required to easily develop

applications with the Ethereum infrastructure is shown. The sample smart contract example, step by step is examined and a sample application development environment is explained. In future studies, we aim to further improve the developed application for scalability problems in compliance with the general accepted security criteria in the e-voting domain.

REFERENCES

- [1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," White Paper, [Online]. Available: <https://bitcoin.org/bitcoin.pdf>, 2008.
- [2] V. Buterin, "Ethereum white paper: a next generation smart contract & decentralized application platform," [Online]. Available: <https://github.com/ethereum/wiki/wiki/White-Paper>, 2013.
- [3] Dappreview, [Online]. Available: <https://analytics.dapp.review/>, 2019.
- [4] Stateofthedapps, [Online]. Available: <https://www.stateofthedapps.com/stats/platform/ethereum#new>, 2019.
- [5] Truffle, [Online]. Available: <https://www.trufflesuite.com/docs>, 2019.
- [6] Nodejs, [Online]. Available: <https://nodejs.org/en/>, 2019.
- [7] K. Wu, "An Empirical Study of Blockchain-based Decentralized Applications", [Online]. Available: <https://arxiv.org/pdf/1902.04969.pdf>, 2019.
- [8] F. Wessling, C. Ehmke, M. Hesenius, and V. Gruhn, "How Much Blockchain Do You Need? Towards a Concept for Building Hybrid DApp Architectures", 2018 ACM/IEEE 1st International Workshop on Emerging Trends in Software Engineering for Blockchain, Sweden, 2018.
- [9] C. Cai, H. Duan, and C. Wang, "Tutorial: Building Secure and Trustworthy Blockchain Applications", 2018 IEEE Cybersecurity Development (SecDev), USA, 2018.
- [10] W. Cai, Z. Wang, J. B. Ernst, Z. Hong, C. Feng, and V. C. M. Leung, "Decentralized Applications: The Blockchain-Empowered Software System", IEEE Access, vol. 6, pp. 53019-53033, 2018.
- [11] Q. Xu, Z. He, Z. Li, and M. Xiao, "Building an Ethereum-Based Decentralized Smart Home System", 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS), Singapore, 2018.
- [12] Remix, [Online]. Available: <https://remix.ethereum.org>, 2019.
- [13] Visual Studio Code, [Online]. Available: <https://code.visualstudio.com/>, 2019.
- [14] Solidity, [Online]. Available: <https://solidity.readthedocs.io/en/v0.5.11/>, 2019.
- [15] Techbrij, "Hello World Smart Contract In Solidity", [Online]. Available: <https://techbrij.com/hello-world-smart-contract-solidity-ethereum-dapp-part-1>, 2019.
- [16] Heroku, [Online]. Available: <https://devcenter.heroku.com/>, 2019.
- [17] Amazon Web Services, [Online]. Available: <https://aws.amazon.com/tr/blockchain/>, 2019.
- [18] Azure Blockchain, [Online]. Available: <https://azure.microsoft.com/tr-tr/services/blockchain-service/>, 2019.
- [19] P. Zhang, J. White, D. C. Schmidt, and G. Lenz, "Design of Blockchain-Based Apps Using Familiar Software Patterns to Address Interoperability Challenges in Healthcare", the 24th Pattern Languages of Programming conference, Canada, 2017.
- [20] M. Wohrer and U. Zdun, "Design Patterns for Smart Contracts in the Ethereum Ecosystem", 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Canada, 2018.