Freelancing blockchain: A practical case-study of trust-driven applications development

Milan Radosavljevic, Aleksandar Pesic, Nenad Petrovic, Milorad Tosic

***Abstract*—Nowadays, a large amount of work is done by freelancers across various areas – from graphical design and music composition to data input and software development. However, many issues appear due to participation of several third parties together with different rules and policies imposed by different platforms. On the other side, the emerging blockchain technology provides the execution of transactions in a trustable, decentralized, but still transparent manner. In this paper, we demonstrate a case-study where blockchain is adopted to eliminate the barriers and make freelancing more convenient and profitable at the same time. As an outcome, a proof-of-concept implementation of blockchain-based freelancing platform relying on Ethereum and Solidity smart contracts is presented that provides practical pointers for trust0driven applications development.**

***Index Terms*— Blockchain, Ethereum, Solidity, Freelancers**

I. Introduction

In today's business world, everything is based on trust. Any monetary transaction, ownership or arrangement. This trust, however, is provided in a very specific way - by the role of a third party, i.e. an institution of trust. In money transactions these are banks, in ownership relations there are cadastres and similar state institutions, in the case of any type of contract, there are courts. The positive side of these institutions, i.e. third parties, is that all parties to all these agreements trust them and expect protection in case of any unexpected occurrences. On the other hand, the appearance of third parties brings with it a lot of negative effects, so you often end up in a waiting list in order to make payments or get a certificate of ownership of a real estate and the like. Mistakes made by these institutions themselves are also very common, and as a rule they fall on the common man as a burden. There is bureaucracy, inefficiency, mistakes, enormous costs that at some point completely make the role of these intermediaries meaningless. The question is, is it possible to exclude third parties from future business, and still preserve that positive factor that they brought with them. In the last few years, there has been a development of a technology called *blockchain*, whose primary idea is just this. Decentralized system with minimal costs, efficient, transparent and yet safe enough to take on the role of e.g. banks. [1]

The share of freelancers in today's business world is large. A large amount of work is completed by freelancers, and as a freelancer, everyone has the opportunity to work in a huge number of domains as an independent. So, people who are engaged in graphic design, writing stories, teaching foreign languages and, of course, programming, earn their living by freelancing. Current platforms for freelancers are safe and the most popular places where freelancers can find work are [2] [3] [4]. The problem that arises is the possession of a third party and some rules that must be followed on certain platforms. Blockchain provides an opportunity to eliminate these problems and as a new solution sets some new boundaries and presents some new problems that are obtained by introducing this solution.

In this paper, it is explored how the blockchain technology can be leveraged to eliminate barriers when it comes to freelancing. As main outcome of this research, we introduce prototype implementation of freelancing platform based on Ethereum blockchain technology and Solidity smart contracts.

II. Background

A. Blockchain

Blockchain is a "cryptographically secure transactional singleton machine with a shared state". Cryptographically secure means that the creation of digital currency is provided by complex mathematical algorithms that are practically impossible to break. With the help of these algorithms it is almost impossible to cheat the system (e.g. creating fake transactions, deleting transactions etc.). A transactional singleton machine means that there is one canonical instance of the machine responsible for all transactions created in the system. In other words, there is one global truth that everyone believes in. Shared status means that the status stored on this machine is shared and available to everyone.

B. Ethereum blockchain platform

Ethereum blockchain is practically a transaction-based state machine. As it is known, the state machine receives inputs and, based on the current state, passes into new states. With Ethereum's state machine, we start from the "initial state". This is practically the state before any transaction occurred on the network. When transactions are executed, the initial state passes to some final state. At any time, the final state represents the current state in the Ethereum network. The state of Ethereum has millions of transactions. These transactions are grouped into "blocks". Each block

Milan Radosavljevic is student at Faculty of Electrical Engineering, University of Niš, Aleksandra Medvedeva 14, 18000 Niš, Serbia (e-mail: milan.radosavljevic@elfak.rs).

Aleksandar Pesic is student at Faculty of Electrical Engineering, University of Niš, Aleksandra Medvedeva 14, 18000 Niš, Serbia (e-mail: alpesh@elfak.rs)

Nenad Petrovic is teaching assistant at Faculty of Electrical Engineering, University of Niš, Aleksandra Medvedeva 14, 18000 Niš, Serbia (e-mail: nenad.petrovic@elfak.ni.ac.rs)

Milorad Tosic is full professor at Faculty of Electrical Engineering, University of Niš, Aleksandra Medvedeva 14, 18000 Niš, Serbia (e-mail: milorad.tosic@elfak.ni.ac.rs)

contains some set of transactions and each block is cryptographically chained together with the previous blocks, which can be seen in Figure 1.

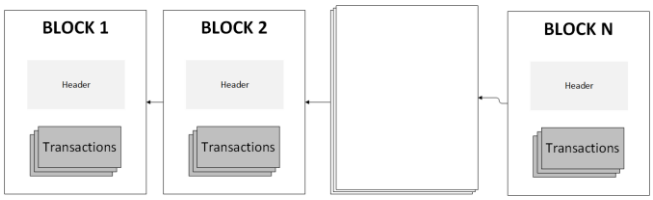

Fig. 1. Preview of Ethereum blockchain.

To cause a transition from one state to another, the transaction must be valid. For a transaction to be considered valid, it must go through a validation process known as mining. Mining is the process when a group of Ethereum nodes (more precisely computers) spend their computing resources to create a block of valid transactions. Any Ethereum network node that declares itself a miner can try to create and validate the block. All miners try to create and check blocks at the same time. Every miner provides mathematical "proof" when submitting a block, and this proof acts as a guarantee: if the proof exists, the block must be valid. The process of validation of each block by the miner who is supposed to provide mathematical proof is called *proof of work.*

C. *Concept of fees*

One very important concept in Ethereum is the concept of fees. Any calculation that occurs as a result of a transaction on the Ethereum network is charged a fee. The fee is paid in denominations called "gas". Gas is a unit used to measure the fees required for a particular calculation. Two factors determine how much it takes to pay for an action: the gas price, and how much gas that action requires. The important part is that Ethereum gas prices aren't fixed. Gas prices are determined by supply and demand. The busier the Ethereum network, the higher the gas price. The amount of gas required for each transaction depends on how complex the transaction is. Gas prices are denoted in gwei, which itself is a denomination of ETH. Gwei is 10^{-9} ETH. It is possible to set a gas limit for each transaction. Gas limit refers to the maximum amount of gas you are willing to consume on a transaction.

D. *Solidity smart contracts*

A smart contract is a contract that is performed by itself together with the terms of the contract to which the parties have agreed. The terms of the contract are written directly in the code of that smart contract. And the contract itself and the 'consent' of the participants exists throughout the Ethereum network. Practically smart contracts are programs that are immutable and deterministic. They depend on the context of the Ethereum Virtual Machine and the decentralized global network. The contract controls the execution of transactions, which are public and non-refundable. In essence, contracts are reduced to programs, which are modeled on traditional contracts, '*if it happens, then do it*'. The contract is executed on many computers to ensure reliability and trust. Smart contracts provide autonomy, trust, speed, security and money savings.

III. Related Work

The blockchain was invented in 2008 by a group or an individual, and this information is still unknown to the public. Therefore, we can consider that blockchain, and at the same time Ethereum, is a newer technology that poses some new challenges and problems in front of us. Much research has been conducted in academia as well as in industry to explore the benefits of smart contracts as well as the worlds in which they are applicable. There are many smart contract platforms on the market with different features that suit certain applications. In [5], the authors focus on the technique of using blockchain to store vaccination records, which is secure and efficient and is based on smart contracts found on the Ethereum platform. In [6], a system based on blockchain was proposed, which refers to workers who are temporarily employed in companies. In that way, employees are provided with a fair and legal salary for their work obligations, as well as protection if the employer becomes a debtor. The author's work in [7] gives a focus on climate change and proposes a solution which, with the use of blockchain, would reduce global warming while keeping records of the crown impression of the product. In [8], the authors provided an overview of all the challenges that smart contracts would face in the future.

IV. Solution Overview

A. *System Architecture*

The application architecture consists of three parts: client side, Web3 interface and server side. The server side is located on the Ethereum blockchain network and uses Solidity smart contracts which are accessed via Web3 interface on the client side. The client side is located in the browser and uses HTML, CSS and Javascript programming language. Also, the client side contains the Web3.js Javascript library through which it communicates with Solidity smart contracts as can be seen in Figure 2.

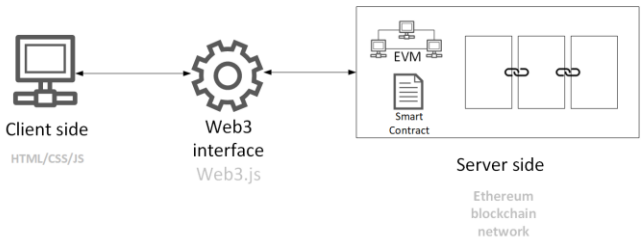

Fig. 2. Representation of system architecture.

B. *Tools used for system implementation*

1) *Ganache*

Ganache is a local Ethereum blockchain that runs on a local computer. Intended for the development and testing of smart contracts and decentralized applications in a secure and deterministic environment. Provides ten externally owned accounts for testing purposes. The application contains a graphical interface and can also be used as a console application.

2) *Metamask*

Metamask is a software that allows you to own a

cryptocurrency wallet and allows you to interact with the Ethereum blockchain. Metamask provides the ability to store and manage account addresses on the browser. It also allows us to connect securely to decentralized applications. Using this software enables multi-user browser behavior.

3) Remix IDE

Remix IDE is an open source web and desktop application. It enables the rapid development of Solidity smart contracts and contains a large number of plugins as well as a graphical interface. The Remix IDE is used for contract development, but also as a platform for learning programming on the Ethereum platform. The Remix IDE is part of a Remix project that develops a handful of tools related to Solidity smart contracts. It is written in the Javascript programming language and allows you to run and test contracts in a web browser. It also allows you to test, debug and deploy contracts as well as many other useful options. [9]

V. Implementation

As indicated in the System Architecture chapter, the system consists of a server and a client side. The server side consists of smart contracts written in the Solidity programming language. The UML diagram in Figure 3 shows the organization of the contracts and the structures used in the system. The main component is a *FreelancerContract* contract that uses *Service*, *FreelancerStructure* and *Offer* data structures. This component represents one Freelancer who is registered in the system. While *PlatformContract* acts as a repository and it contains all registered freelancers in the system. In addition to the data structure that stores basic information about the freelancer, there are also structures for services and offers. *Service* is what a freelancer offers, while an *Offer* acts like a real job offer.

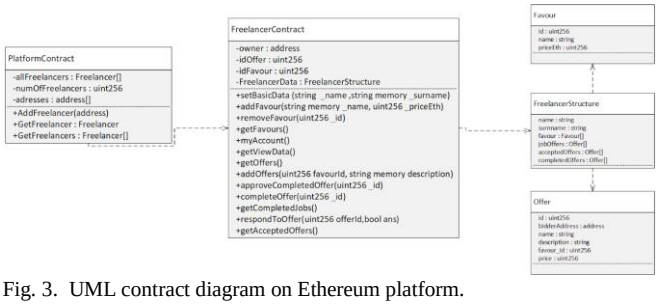

Fig. 3. UML contract diagram on Ethereum platform.

The client side of the application is written in the Javascript programming language. On the client side there are two proxy classes that correspond to the contracts on the server side, and also uses the Web3.js library to communicate with the server side. Practically one function on the client side calls one function in the contract. There are also functions that are handlers for the events which are broadcasted in contracts. The UML class diagram of the client side can be seen in Figure 4.

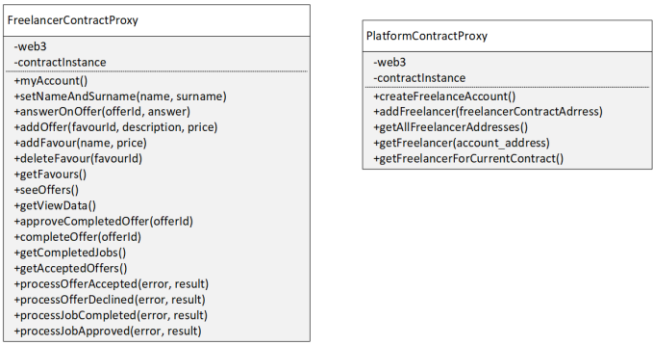

Fig. 4. UML class diagram on the client side.

VI. Results

Figure 5 shows the initial view of the client application. The application offers the ability to create a new account and navigation that allows to navigate through the entire application.

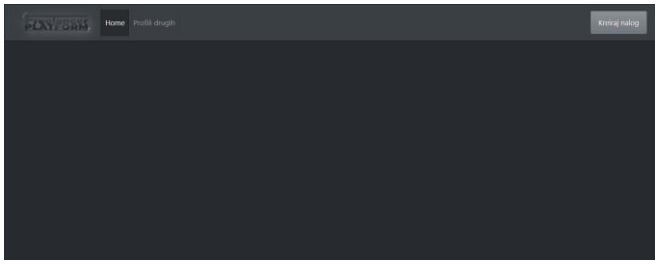

Fig. 5. Initial view of the application.

After creating the account, you get the view as in Figure 6 and the application allows the user to set the name and last name, add services he is offering and have an insight into the job offers that are offered to him, offers accepted by the user and offers waiting for client's approval.

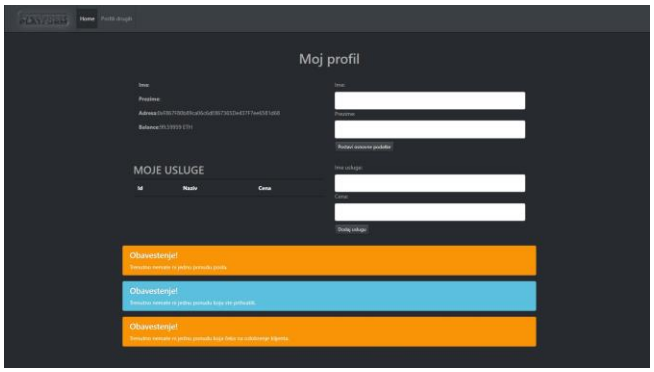

Fig. 6. View of the application after creating the account.

Clients can see the services of all freelancers on the Profiles of Others page, which can be selected from the navigation. The view of the page can be seen in Figure 7. The application provides the ability to search all offered freelancer services that are in the system and the ability to send a service offer to a specific freelancer that will be displayed on the homepage of the freelancer for whom the offer is intended.

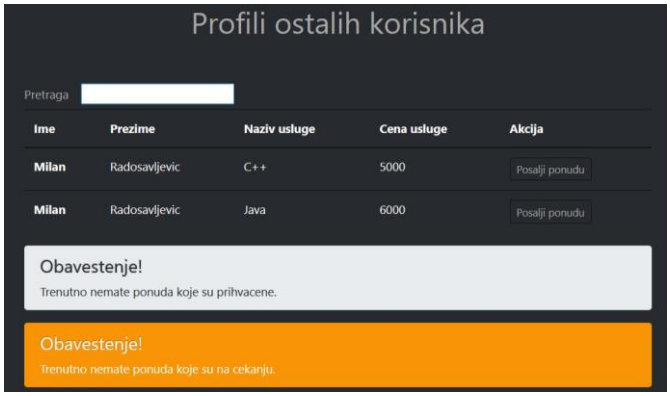

Fig. 7. View of page *Profiles of other users*.

When submitting an offer it is necessary to add a description and offer a price for a particular service. Based on the submitted offer, the freelancer will decide whether to accept the offer for the job. Form for sending the offer can be seen in Figure 8. After the offer is sent, the offered price is transferred from the client's account to the smart contract account. In case the freelancer rejects the offer, Ether will be returned to the customer account.

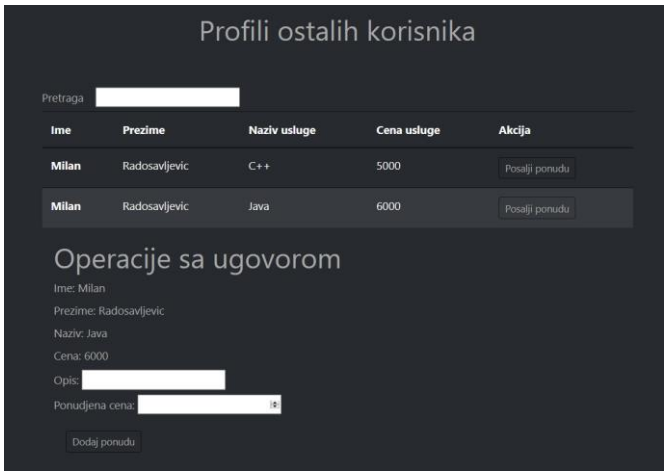

Fig. 8. Appearance of the offer submission form.

After the accepted offer, freelancer completes the offer and sends it to the client for approval, after the approved offer, the client accepts the completed work and only then the offered price is transferred to the freelancer's account. The layout of the offer table can be seen in Figure 9.

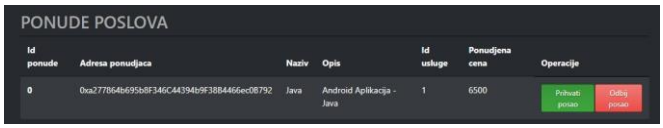

Fig. 9. Appearance of the table with job offers.

VII. Conclusions and Future Work

A platform for supporting a freelancing work community is developed using Solidity smart contracts on the Ethereum platform. The implementations process is used as a case-study for learning practical aspects of trust-driven applications development. Practical experiences and results are presented in this paper.

This paper gives our first experiences and more systematic approach is needed particularly validation and evaluation that are planned for future work. Regardless of absence of a full scientific rigor, we present our experiences that could be useful for future work in the field of trust-based applications development.

The good side of the solution presented in this paper is that it exploits advantages of the blockchain such as reliability, security and speed. Due to the fact that since its launch, the Ethereum platform has not had downtime due to consensus and decentralized approach. Hence the fact that this is another advantage, more precisely the advantage that the platform is always online and working. This further implies that the applications running on the blockchain do not have a downtime, including this one. The solution has an intuitive user interface that is capable of expansion. Even though the application was developed as a proof-of-concept, it shows high potential for commissioning in a real environment.

One of disadvantages of the approach is that every action that changes the state of the blockchain uses gas that requires compensation from the user, as stated in the chapter *Background*. Hence, some actions performed in the application are not free. In the current prototype, freelancer can not deliver product to the client who hired him. In future work this shortcoming could be fixed by connecting accounts with Github service, for example. It is possible to further optimize the speed of the application, on both client and server side, and to minimize the fee spent for performing actions on the server side. The obtained solution proves that it is possible to reach a satisfactory solution at an acceptable price.

Acknowledgment

This work has been supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia.

References

[1] Uvod u blockchain [online], Blog, Đorđe Ivanović, 2018. Source: https://blog.itkonekt.com/2018/07/30/uvod-u-blockchain/
[2] Freelancer [online], freelance recruitment platform, https://www.freelancer.com
[3] Upwork [online], freelance recruitment platform, https://www.upwork.com
[4] Fiverr, freelance recruitment platform [online], https://www.fiverr.com
[5] S. K. Deka, S. Goswami, A. Anand, "A Blockchain Based Technique for Storing Vaccination Records", IEEE Bombay Section Signature Conference (IBSSC), pp. 135-139, 2020.
[6] A. Pinna, S. Ibba, "A blockchain-based Decentralized System for proper handling of temporary Employment contracts", Proceedings of the 2018 Computing Conference vol. 2, pp. 1-6, 2019.
[7] A. M. Rosado da Cruz, F. Santos, P. Mendes, E. Ferreira Cruz, "Blockchain-based Traceability of Carbon Footprint" Proceedings of the 22nd International Conference on Enterprise Information Systems (ICEIS), pp. 1-10, 2020.
[8] Sh. Wang, Y. Yuan, X. Wang, J, Li, R. Qin, F. Wang, "An Overview of Smart Contract: Architecture, Applications, and Future Trends", IEEE Intelligent Vehicles Symposium, pp. 108-113, 2018.
[9] Remix IDE documentation [online]. Source: https://remix-ide.readthedocs.io/en/latest/