applied sciences

MDPI

Article

Easing ĐApp Interaction for Non-Blockchain Users from a Conceptual Modelling Approach

Miguel A. Teruel [1,2,*] and Juan Trujillo [1,2]

[1] Department of Software and Computing Systems, University of Alicante, 03690 San Vicente del Raspeig, Spain; jtrujillo@dlsi.ua.es

[2] Lucentia Lab, University of Alicante ,03450 Alicante, Spain

* Correspondence: materuel@dlsi.ua.es; Tel.: +34-965-909-581

Received: 15 May 2020; Accepted: 19 June 2020; Published: 22 June 2020

Featured Application: Easing the interaction of non-blockchain users with a decentralized application (ĐApp) through a middleware created by using conceptual modeling in UML (Universal Modeling Language).

Abstract: Blockchain decentralized applications (ĐApp's) are applications which run on blockchains nodes. Thus, to interact directly with this sort of applications, users need to have a blockchain address, wallet, and knowledge about how to make transactions to interact with ĐApp's. Therefore, the knowledge required to use a ĐApp can easily make users to desist when trying to interact with them. To tackle this issue, we propose a software architecture that will be in the middle of the user and the ĐApp, thus making users initially unaware of the fact that they are interacting with a ĐApp. This is achieved by analyzing the relationship between ĐApps and Apps by using UML modelling. Next, based on the previous analysis, we created a middleware for users to interact with a ĐApp in the same manner they do with a traditional web app, i.e., by using usernames, passwords and user interface elements instead of addresses, private keys or transactions. To put the developed middleware into practice, we developed a ĐApp that makes use of it. This ĐApp registers the time control of workers from companies by using blockchain to store the data in a secure and non-modifiable manner. Finally, we performed an experiment, thus demonstrating that a ĐApp that implements the proposed middleware would improve its usability for users with no experience with blockchain.

Keywords: blockchain; ĐApp; UML; conceptual modelling; ethereum; smart contract; solidity; quorum; middleware; clockchain

1. Introduction

Back in 2008, a paper written by somebody under the pseudonym of Satoshi Nakamoto [1] revolutionized both the world economic system and the distributed management of data. Bitcoin was born as a distributed ledger that would enable online payments without the intervention of a third party (i.e., bank, financial services, payment gateway, etc.). Indeed, instead of trusting a third party as the transactions' validator, this process is conducted by a network of peers, using cryptographic techniques. Bitcoin has been adapted widely, having a market capitalization of 160 trillion $ in May 2020.

Nevertheless, not only were cryptocurrencies [2] born but also the mechanism used to store the transfers of Bitcoins, namely the blockchain [3]. A blockchain is a data structure whose information is stored in blocks. Hence, each block is hashed and connected with the previous blocks. Therefore, in order to alter any block of the chain, the whole chain would have to be rehashed, which would

require more than half the computing power of all the network's peers, which is known as a 51% attack [4,5]. Consequently, blockchain arises as an immutable data structure to anonymize and secure data in a distributed manner, regardless of its scope of application. As a matter of fact, Blockchain has already been applied to different domains such as healthcare [6] or education [7].

After the Bitcoin revolution, the next significant milestone arrived with Ethereum [8] and the smart contracts [9]. Hence, not only did the Ethereum blockchain store transactions of its cryptocurrency, namely Ether, but also it can execute rules written in smart contracts that can manage users' digital assets (i.e., tokens). Thanks to the capability of executing smart contracts, Ethereum enabled us to create decentralized applications that run in the blockchain (ĐApp's) [10,11] as well as decentralized autonomous organizations (DAO) [12], which are virtual organizations able to deal with digital assets without human intervention thanks to a set of rules codified in smart contracts.

We are talking about a world-wide revolution in the economic and distributed data storage paradigms that would enormously benefit users. However, such users are not entirely ready yet due to blockchain's learning curve. Indeed, in order to interact with a ĐApp, users must have a wallet per blockchain, as well as the knowledge about how to use their public and private keys to sign and deploy transactions to the blockchain that will interact with the smart contracts. Because of that reason, a ĐApp is prone to fail nowadays as far as the public is concerned.

In order to tackle this issue, we propose a middleware which is located within the smart contracts and the user interface of the ĐApp, that will ease the interaction with it by making it similar to the classic user web interaction based on usernames, password and visual web controls (e.g., button, forms, and so on). In this sense, users will be able to use a ĐApp in a comfortable manner until they get used to interacting directly with smart contracts. Moreover, to develop such middleware, we rely on the conceptual modeling analysis of the Ethereum blockchain architecture.

This paper is structured as follows. After this introduction, Section 2 presents different works related to blockchain modeling and ĐApps. Next, Section 3 widens the explanation of why users have trouble when interacting with ĐApp's. Section 4 will explain the relationship between ĐApps and Apps from a UML (Universal Modeling Language). modelling point of view. Section 5 will present out middleware proposal which will be put into practice in Section 6 by means of a running example. Next, Section 7 will present the experiment we conducted to assess if the implementation of the proposed middleware would improve the usability of a ĐApp. Finally, Section 8 will finish with our conclusions and future work.

2. Related Work

Despite being quite a recent field of research, we can find several works related to the development of ĐApp's. For instance, the authors of [13] presented a ĐApp to share everyday objects. By using this ĐApp, which was deployed on the Ethereum blockchain, users can register and rent objects by scanning QR codes. Besides, a different scenario was considered in [14], whose authors presented a ĐApp for real-time ride-sharing services. By means of this ĐApp, they propose a blockchain version of the Uber platform, consisting in "decentralized, community-owned-and-managed transportation network without unsatisfactory centralized decision-making or risks". In a different vein, a ĐApp for the healthcare domain is presented in [15]. In this work, they presented FHIRChain, a ĐApp to store and share clinical data, thus benefiting from blockchain's security. Therefore, they use digital health identities (blockchain addresses) to authenticate patients in this ĐApp, thus maintaining data ownership and privacy.

As far as conceptual modeling and software engineering for blockchain is concerned, we can find only a handful of works. Indeed, Rocha and Ducasse started the discussion regarding the need for applying software engineering to the development of smart contracts in [16]. They propose using UML's class diagrams to model the structure of smart contracts since they are class-based artifacts. In this sense, the authors of [17] applied several UML diagrams to model an Ethereum-based ĐApp for managing workers contracts. In this work, they applied the blockchain oriented software engineering approach [18], which in turn, acknowledges the need for applying software engineering to blockchain-based applications. Despite of the fact of not being directly related to UML, in [19] it is

proposed a fine-grained approach to identify which are the elements of an application architecture that could be implemented by using blockchain. They illustrate such approach by means of a case study based on a system for the coordination and payment of craftsmen constructing buildings.

As far as we know, there is no work aimed at easing the interaction of non-blockchain expert users with a ĐApp. Therefore, this constitutes the main motivation for this work: to conceptually model and analyze the ĐApp structure, with the aim of creating a middleware. It would help users to interact with ĐApps without the need of being used to blockchain concepts which could be cumbersome and overwhelming for non-expert users.

3. Motivation

When dealing with a ĐApp for the first time, it is quite usual that a user does not have the faintest idea about how to do it. That is because ĐApp interaction is based on making calls and transactions according to smart contract methods. Such transactions, which typically involve the exchange of tokens, must be made from the user's wallet, with is univocally identified along the blockchain by means of its public addresses. Hence, to perform a transaction, the user would have to expend some tokens to pay for the transaction to be executed (known as gas on the Ethereum network). Afterwards, the user would have to sign the transaction by using his/her private key and to relay it through a blockchain node for the transaction to be mined and executed. Finally, the user would have to follow the transaction execution through the transactions TX receipt, which can or cannot be successfully executed depending on several aspects such as the smart contract rules or the shortage of funds added to the transaction in order to pay for it.

In addition, the user's wallet is typically installed on just one device. Because of that reason, if the user needed to interact with the ĐApp by using a different device, the wallet would have to be restored by means of the private key or 12-word mnemonic [20], a process most users are unaware of. Furthermore, for the time being, there is little support for mobile wallets featuring ĐApp transactions. Therefore, using a ĐApp with a mobile device could not be always possible.

Due to this level of complexity when executing a single blockchain transaction, it is typical that a non-blockchain user would not be able to interact with a ĐApp without previous training. What is more, even though a user were able to finally interact with a ĐApp in a blockchain, it would not mean that he/she will be able to interact with another ĐApp deployed in a different blockchain. That is because each blockchain requires its own wallet and has unique interaction mechanisms. By way of example, making transactions to a smart contract deployed on Ethereum [8] will significatively differ from a similar contract deployed on Ripple [21], Corda [22], HyperLedger [23], or the upcoming Libra [24] due to the differences among such blockchains [25]. Therefore, users will have to learn the specifics of each blockchain to interact with different ĐApp's.

If we bear in mind examples of real ĐApp's, the most popular ones can be found in the website State of the ĐApps [26]. Looking at the top 10 ĐApp's by number of users in the last 24 h, they range from 10,248 to 1693 users as of today. That these figures cannot be compared with non-distributed apps. For instance, 3 out of these 10 ĐApp's are games, whose number of players is far from those from current non-ĐApp-based games (ranging from 630K to 46K [27]). Therefore, just considering the figures, ĐApp's have not reached the public.

Because of the reasons, if we need our ĐApp's to be used by every user regardless of their blockchain expertise, a mechanism that eases ĐApp interaction is necessary. Based on that need, we formulated the following research questions:

- RQ1: Which is the correspondence between the typical app interaction elements and ĐApp ones?
- RQ2: How can ĐApp interaction be improved for users without blockchain experience?
- RQ3: Does the inclusion of typical app interaction features in a ĐApp improve its usability for users without blockchain experience?

Next, Section 4 will analyze the conceptual model of a blockchain, and the Section 5 will present our proposed middleware to simplify the interaction with a ĐApp, thus partially camouflaging it into a common web app for users with no prior blockchain knowledge to interact with it.

4. From ĐApp to App: A UML Modeling Analysis

To identify how to ease the interaction with ĐApp's, we opted for a Model Driven Development approach. Therefore, aiming at answering RQ1, we will identify the relationship among ĐApp's and Apps elements by modeling an Ethereum-based blockchain architecture and a common App. Figure 1 shows such modeling. Hence, at the top of the figure we can see the UML class architecture of a ĐApp interacting with a blockchain, as suggested in [16]. Moreover, at the bottom it can be seen the equivalent App architecture where its classes have been related to each blockchain element they will have to access in order to expose the blockchain functionality to a non-expert user in an App. For the sake of model comprehension, the relationship among ĐApp and App elements will be presented in the following through the model entities:

- User: The proposed App model features a collection of users (UsersDB). In this sense, the user entity will be associated with the blockchain address for the user to be able to interact with the blockchain without needing to be aware of its address' public and private keys. Further, the user entity will also be associated to the blockchain token's balances. Therefore, it will represent the number of tokens the user will have. Note that for the sake of the model understandability, we will consider only ERC20 (fungible) tokens [28]. Nevertheless, the model could be easily adapted to working with ERC721 [29] (non-fungible) tokens.
- Login: Since we are modeling an App, it is common to have a login functionality. Therefore, our model will have an action called login. Hence, when a user logs in with a correct email/password combination, the wallet containing its Address will be unlocked, thus enabling such user to make transactions towards the blockchain.
- Recovery: One of the issues of blockchain is that if the user loses the private key of an address, he/she will never be able to access it since blockchain does not provide any private key recovery mechanism. It is worth noting that users are used for counting with password recovery systems in almost every App. Therefore, this functionality will be enclosed within the Recovery entity. Therefore, if requested, this entity can access the UserDB to provide password restoration mechanisms. Note that what will be recovered will be the User's password, not the private key. That is because, to facilitate the interaction, we would provide a non-custodial system, where the user is not directly responsible for his/her private key or wallet, being those custodied by the App database. It is worth noting that a password recovery procedure would imply the creation of a new password. Therefore, this entity will make use of the WalletGateway entity. Such an entity will communicate directly with de node (not using Web3 [30] since it cannot re-encrypt wallets, but accessing Geth directly [32]) and re-encrypt the user's wallet with the new password.
- PasswordChange: Another issue of blockchain is that the user cannot change his/her private key. Indeed, getting a new private key would imply also getting a new public address. Then, the user would have to move his/her assets to the new address, task that would require gas to be performed. To overcome this issue, we propose the *PasswordRecovery* entity. Since the authentication is made by using an email/password combination, we will offer the possibility of changing the passwords. It is done though this entity that, in turn, will make use of WalletGateway. Therefore, the password will be changed by re-encrypting the user's wallet by using the new password.
- Movement: This entity will represent the transfer of tokens among users in a straightforward manner. In that sense, if a user wanted to send funds to a different App user, the only required data would be the number of tokens to be transferred and the recipient's email. Therefore, the recipient's address will never be required, thus making it more difficult to choose the wrong recipient when making token transferences.
- Interaction: The emission of payable blockchain transactions (invocation to payable Methods) will be translated into interactions. It is worth noting that we are referring to operations that

alter the current blockchain status. Therefore, such transactions require gas (they need to be paid for by using Ether in the case of Ethereum) for the miner to confirm the validity of such transaction. Nevertheless, to make the model easier to understand, we will consider a private Ethereum blockchain whose transactions, despite modifying the blockchain states, require no gas.

- Query: If an interaction represented a payable Method call, Query would correspond to a view one (a read-only method). Therefore, View methods are methods that, despite reading data from the blockchain, do no change its status. Therefore, since no modifications are involved, they do not need to be mined, not requiring any gas to be processed. Hence, we will represent all the blockchain data requests into Queries that will invoke view methods to retrieve information (e.g., token balances, smart contact status, etc.).
- App: Needles to say that a ĐApp interacts with one or more smart contract. Therefore, our App entity will also have an association with the smart contracts it has access to. Therefore, our users do not have to be aware neither of the addresses of the smart contracts they are interacting with nor the addresses of smart contacts that define the tokens they treasure.

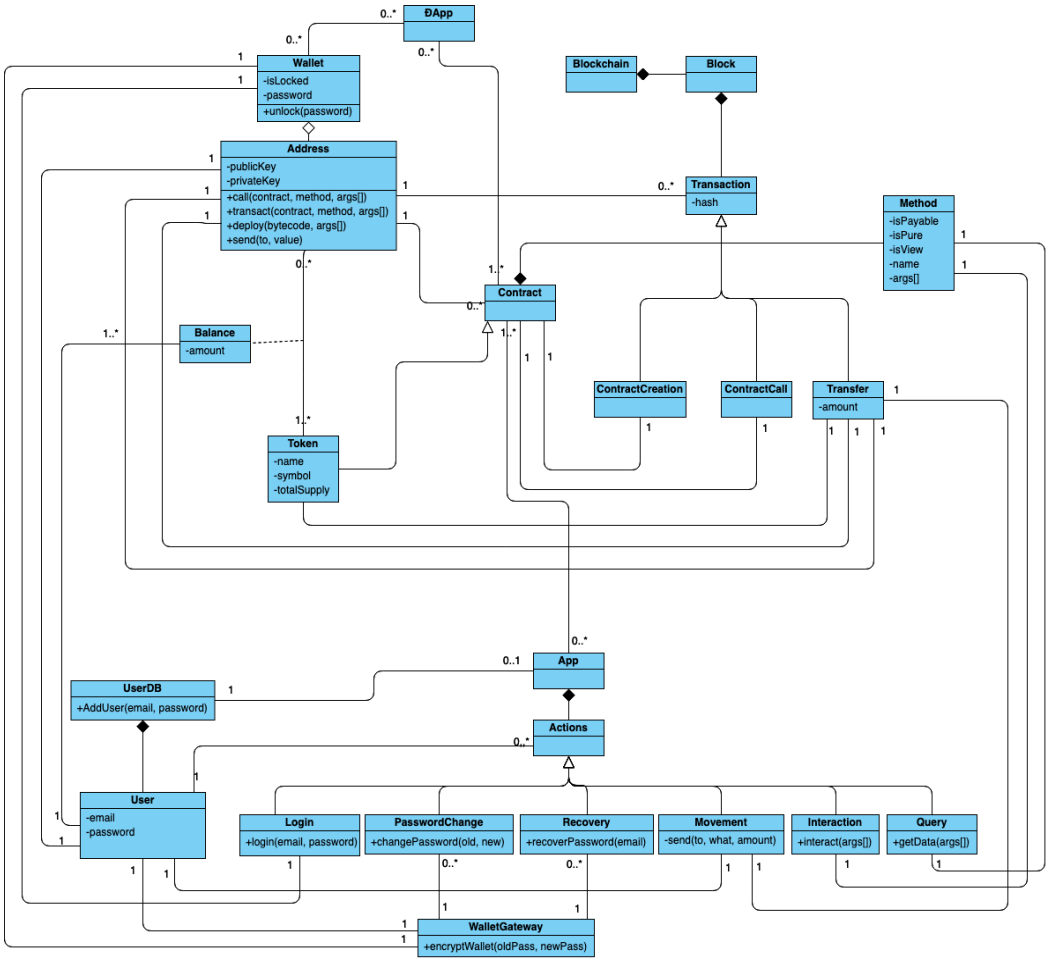


Figure 1. Relationship among blockchain decentralized applications (ĐApp's) and application (App) elements.

Once the model elements have been explained, we will present in the next section how to implement them. The relationship between models will be implemented by using a middleware that will transform user interaction through an App into Blockchain calls and transactions.

5. Middleware to Ease ĐApp User Interaction

With the aim of tackling the ĐApp interaction problem questioned by RQ2, we propose a technique based on a middleware that will "translate" the blockchain's mechanisms into concepts users are familiar with. In other words, the middleware will implement the required functionality to transform the ĐApp model shown on Figure 1 to the App-equivalent one. From now on, we will exemplify the proposed middleware by using Ethereum since it currently is the most used blockchain to deploy smart contracts.

In a common Ethereum ĐApp, the software architecture is distributed between a JavaScript-based web application featuring the Web3 library and a wallet [30] as frontend and a set of Ethereum's smart contracts as backend (top of Figure 2). This architecture, for starters, makes users unable to interact if they do not have an Ethereum wallet (e.g., Metamask [31]) embedded into their browsers. Furthermore, since the wallet is running and properly installed, they will be able to log in and interact with the ĐApp by signing transactions with their private keys. What is more, for those transactions involving the modification of the blockchain, the user will have to put some gas into the transaction (i.e., pay for it) in order for it to be successfully committed into the blockchain (Figure 3). Finally, users will also have to take another decision before releasing the transaction: deciding how much gas they are willing to put on in. Hence, putting not enough gas could make the transaction unsuccessful, since no miner will mine the transaction into a block due to the low reward to be received. This process gets even more complicated if the user deals with a private Ethereum-based blockchain, such as Quorum, since the transaction will have to be manually adjusted to use no gas.

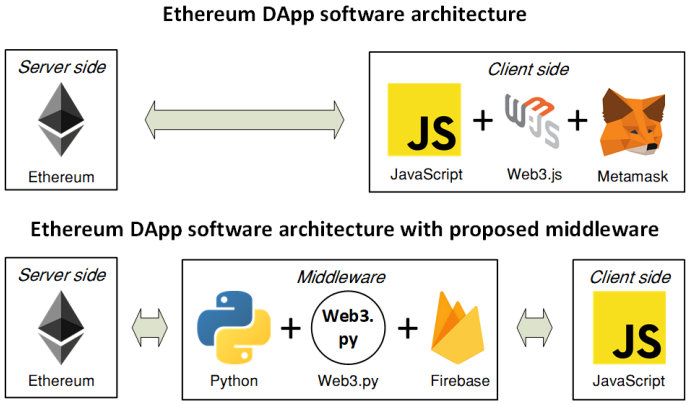
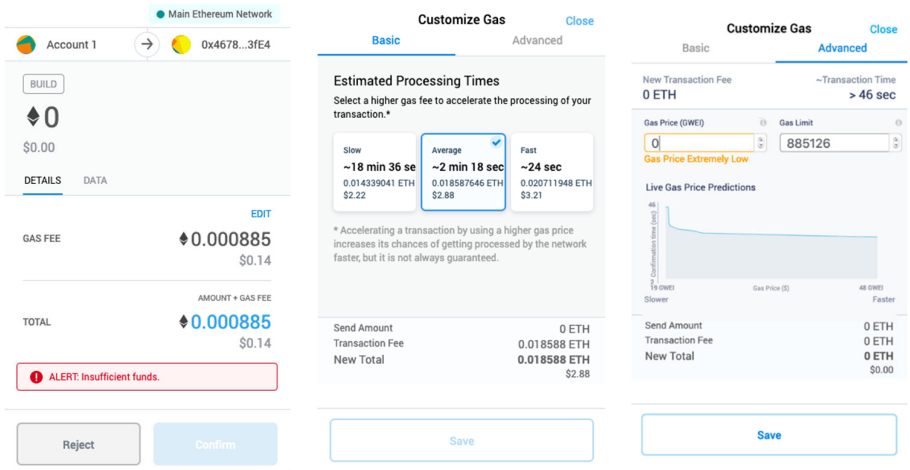


Figure 2. Ethereum software architecture comparison.



(**a**) (**b**) (**c**)

Figure 3. Sending a transaction to an Ethereum Smart Contract (**a**) and deciding how much gas to spend (**b**) and customizing it for a private Blockchain (**c**).

Putting everything together leads us to a straightforward conclusion: A non-expert user interaction with a ĐApp will probable fail due to the considerable number of concepts he / she needs to know. Therefore, we propose using a middleware (bottom of Figure 2) that will relieve the user of the need of mastering blockchain concepts and technologies when interacting with a ĐApp for the first time. To demonstrate our proposal, we will use a Python implementation and a Firebase database. Nevertheless, any technology could be used provided it has the required libraries (e.g., Web3 for the Ethereum) to interact with the blockchain. In the following, the different components of our proposed middleware will be shown.

5.1. Login into the ĐApp

In a normal ĐApp, a user logs in by using his/her blockchain public address, typically by linking it to the ĐApp through the wallet's address. That in the event the user has no wallet, this operation will not succeed. To tackle this first issue, we propose the use of a mechanism every user is familiar with emails and passwords. Therefore, to use this mechanism, we need to establish a correspondence between users' emails and their public addresses. With this aim, we used a Firebase database. It is worth noting that this database will never store private keys. Indeed, to log in, the user must provide the password to unlock the Geth wallet present on the node (see Section 5.3 for more details), but the private key is never required nor stored in the database or local storage.

Thus, when a user signs up into the ĐApp, a register form will be shown requiring the email and password. In this very moment, our middleware will create an Ethereum wallet by using the Web3 package web3-eth-personal. Hence, such wallet will be encrypted by using the provided password. At this point, the user will receive a confirmation email with his/her public and private address since he/she could also interact directly with the smart contract as soon as he/she mastered Ethereum procedures. In parallel, we will store the user's email, public address, and a hash of the password. This will enable the user to log in in a traditional manner without having to be aware of the blockchain keys.

5.2. Sending Transactions to the Smart Contract

Once we have tackled the issue related to the user's authentication, the devious transaction mechanism will be eased. To do that, we will extremely simplify the transaction bust just making it transparent to the user. Hence, sending a transaction will be as easy as clicking a button in the user interface.

This procedure is possible by encapsulating the transaction invocation into a RESTful API POST call that will receive the parameters required by the transaction. Then, it will invoke the send transaction method from the Web3 web-eth package. Nevertheless, if we are dealing with the Ethereum main network, gas is required to relay the transaction. For that reason, the middleware developer could provide the users' wallet with gas to run transactions and refuel it in the event it was low enough. Moreover, the Ethereum calls (i.e., queries to the blockchain which do not require gas since they are read-only) are encapsulated into RESTful GET calls accordingly. It is worth noting that this mechanism will also benefit the frontend developer, who will be able to develop the web interface without having to interact with Web3 directly.

5.3. Enabling Users to Change their Passwords

As aforementioned, the authentication mechanism of a blockchain is the public and private key pair. This fact makes it impossible to change the private key of an address. Moreover, if the user wanted a different identification id, (i.e., a new address), the funds on the old wallet would have to be manually transferred to the new one. This action, that would have a cost in gas, would generate a new private key. Moreover, all the data in the smart contract related to the old address would remain assigned to it. Hence, these data could only be migrated to the new address if the smart contract implemented a transfer data mechanism, which would probably result expensive as far as gas is concerned.

However, thanks to our middleware, a user does not have to deal with this problem since an email can always be changed in the Firebase database, thus generating a new login id that is independent of the address and, consequently, independent of its funds and stored data.

The problem comes when dealing with password changes. The user's wallets are physically located in the Ethereum node when using our middleware and encrypted by using the users' password. Nevertheless, the Web3 library does not provide the functionality to re-encrypt such wallets by using a new password. Because of that reason, we propose the direct interaction with Geth, the command line interface for Ethereum nodes implemented in Go [32]. Thus, when a user wants his/her password to be changed, he/she will ask the ĐApp for it, thus receiving an email with a link which will enable the user to do so. Therefore, when the user provides the new password, the ĐApp will invoke a middleware RESTful method which, in turn, will connect to Geth via Secure Shell (SSH) and change the password accordingly. Finally, the new password will be re-hashed and stored in the database to enable the user to log in again by using the new credentials.

5.4. Tokens Management

One of the main benefits of blockchain is the tokenization of assets, thus providing value to physical or virtual entities. Consequently, our middleware must provide a double functionality. On the one hand, since the users are not required to control their wallets (despite they can do so by importing their private keys into a normal Ethereum wallet) they require a mechanism to transfer tokens among users. On the other hand, as the ĐApp can provide the user with tokens under certain conditions (DAO), its manager, who is not required to be blockchain-savvy, needs to have a way to create new tokens (also known as minting tokens).

In this sense, the middleware provides functionality to both users and managers to query token balances, transfer tokens among users and mint tokens (only for managers). However, this operation will benefit from our middleware due to the email-address translation mechanism. Hence, a user could transfer tokens to a different one by only knowing his/her email. Further, they will also be able to transfer tokens to the contract to exchange them for whatever asset the manager decides, without knowing the contract address. In this sense, a user will never send funds to the wrong contract or user.

5.5. Security Concerns

One of the key features of blockchain is its security. Therefore, our ĐApp will have to keep up this by implementing several security procedures. Hence, we propose the use of JSON Web Tokens (JWT), an open, industry standard method for representing claims securely between two parties [33] as credentials for the users as an extra layer over the blockchain security.

When a user logs in successfully, a JWT will be created by the middleware and sent to the frontend for the user interface to be able to invoke the RESTful API. Moreover, the user's wallet will be unlocked for several seconds equal to the JWT expiration time. Hence, if a user logs out, the JWT will be blacklisted and the wallet locked. Nevertheless, if a user does not log out, the JWT will automatically expire and the wallet will be locked, thus requiring a new log in to keep on using the ĐApp.

Moreover, we propose implementing an additional security layer regarding IP addresses whitelisting. With this aim, our Ethereum node will only be accessible to users whose IP addresses are whitelisted. Therefore, if an unauthorized user tries to unlock a wallet without signing in properly, the node will automatically reject the operation. For that reason, when a user logs in, his/her IP address will be whitelisted. Finally, the log out of a user will remove the IP address from the whitelist. It is worth noting that using this procedure will disable users to interact directly with the blockchain through our node. However, this issue will be only applicable to private Ethereum-based blockchains where only a few (or even one) nodes can access the blockchain.

6. Running Example: Employee Schedule Tracker ĐApp over Quorum Blockchain

Once the proposed middleware has been explained, this section will put it into practice. To do so, we developed a ĐApp aiming at tracking employees schedule over an Ethereum-based blockchain, namely Quorum. More specifically, we deployed our smart contracts over the Quorum network of Alastria [34], a Spanish national blockchain consortium consisting of the main companies in Spain (banks, telecommunication, public entities, transportation, IT, and so on). This semipublic permissioned blockchain will enable us to run our ĐApp in a secure environment as well as maintaining the application expenses constant since no gas is required to run transactions on the blockchain.

Regarding the developed ĐApp, it is named Clockchain [35] after its functionality and underlying technology. This ĐApp arises due to a new Spanish law, the Royal Decree-Law August 2019, of 8th March, on urgent social protection measures and the fight against job insecurity in the workday. This law states that the workers' daily entries and exits must be safely stored for a minimum of four years in a secure and non-modifiable storage medium. Indeed, the security, durability, and non-modifiability requirements for the data to be stored make this a perfect scenario to apply blockchain. Nevertheless, the main issue in this scenario is that users are not used at all to interacting with a ĐApp. Therefore, we applied the proposed middleware to this scenario to make Clockchain easy to be used by anyone.

Figure 4 shows the main interface of Clockchain after logging in. It is worth noting that, to perform such log in, neither users need a wallet, nor they must know they are working with a blockchain-based system. On the contrary, they only use a combination of email and password to log in. Once users are logged in, they will see all their current entries and exits during the current day, as well as the elapsed working time (Figure 4). Such information is retrieved from the blockchain in a transparent manner, thus corresponding to a query action (see Section 4) of the middleware. Indeed, what these actions will perform in the middleware is a call to a view function inside the smart contract.

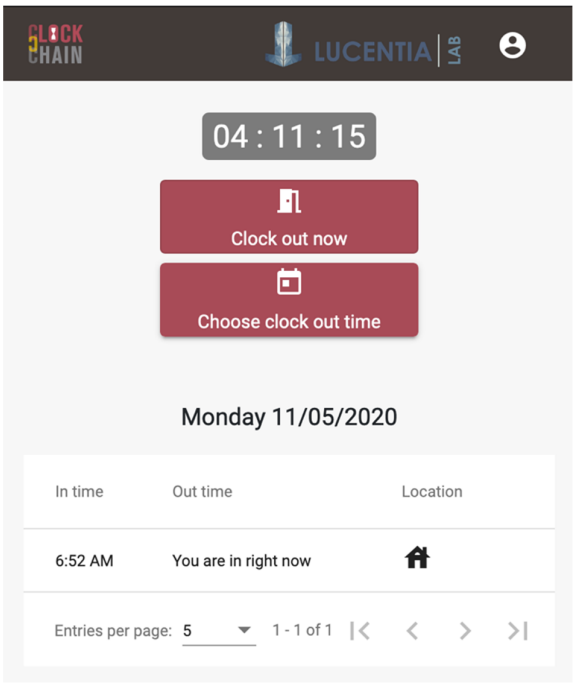


Figure 4. Clockchain interface for a user when clocking out.

Besides, Figure 4 shows the two interaction actions the user can perform at that moment. More specifically, we can see the interface of a worker who has already clocked in around four hours ago from home. Therefore, the interactions with the blockchain that could be performed at that moment

are to clock out at that very moment, or to choose a different clock-out time (in case the worker forgot to clock out at a time in the past). By way of example, Figure 5 shows the corresponding smart contract methods such interactions will correspond to. For this example, the button *Clock out now* will correspond to the public function *exit ().* In addition, the button *Choose to clock out time* will correspond in turn to the function *exit (uint256 time)* since the user could choose the clocking out time. As it can be seen in Figure 5, both public functions will call the private *_exit (uint256 time)* function. This function, after checking the requirements for the blockchain transaction to be performed, will store the clocking out time of the worker represented by his/her blockchain public address (*msg.sender*). Moreover, our smart contract will also reward the worker's time with tokens, that will be transferred to his/her address by means of the *_transferTokens (uint256 workTime)* function. Indeed, each worker will receive one ERC20 token for each worked hour. Hence, the user will be able to query and transfer the received tokens by using the ĐApp's web interface, not requiring any external wallet such as Metamask.

```
function exit() public {
    _exit(now);
}

function exit(uint256 time) public{
    require(time <= now, "You cannot exit in the future");
    _exit(time);
}

function _exit(uint256 time) private {
    require(enabled[msg.sender], "You are not enabled");
    require(companies[employer[msg.sender]].license >= now, "The company license has expired");
    require(entries[msg.sender].length != 0, "You never entered");
    require(entries[msg.sender][entries[msg.sender].length-1].in_time!=0, "You must be in");
    require(entries[msg.sender][entries[msg.sender].length-1].out_time==0, "You must not be out");
    require(entries[msg.sender][entries[msg.sender].length-1].in_time < time, "You cannot exit before entering");
    entries[msg.sender][entries[msg.sender].length-1].out_time = time;

    uint256 workTime = time - entries[msg.sender][entries[msg.sender].length-1].in_time;
    _transferTokens(workTime);
}

function _transferTokens(uint256 workTime) private {
    uint256 per_second = (uint256(10) ** tokenContract.decimals()).div(3600);
    // emit test_value(per_second);
    uint256 tokensToReceive = workTime.mul(per_second);
    // emit test_value(tokensToReceive);
    require(tokenContract.balanceOf(address(this)) >= tokensToReceive, "The smart contract has not enough tokens");

    require(tokenContract.transfer(msg.sender, tokensToReceive), "The tokens have not been transfered to you account");
}
```

Figure 5. Smart contact methods for clocking out.

Finally, in order to provide our ĐApp with the required scalability to deal with an unpredictable number of users, it was decided to deploy it in the Google Cloud Platform (GCP) as shown in Figure 6. In that sense, we will create the blockchain node by using a GCP Compute Engine instance. In the event the node got overloaded, we will be able to scale it to improve its power. This can happen due to either a vast number of users using our ĐApp concurrently or a growth of the blockchain. Therefore, we could increase the computational power or the disk size accordingly.

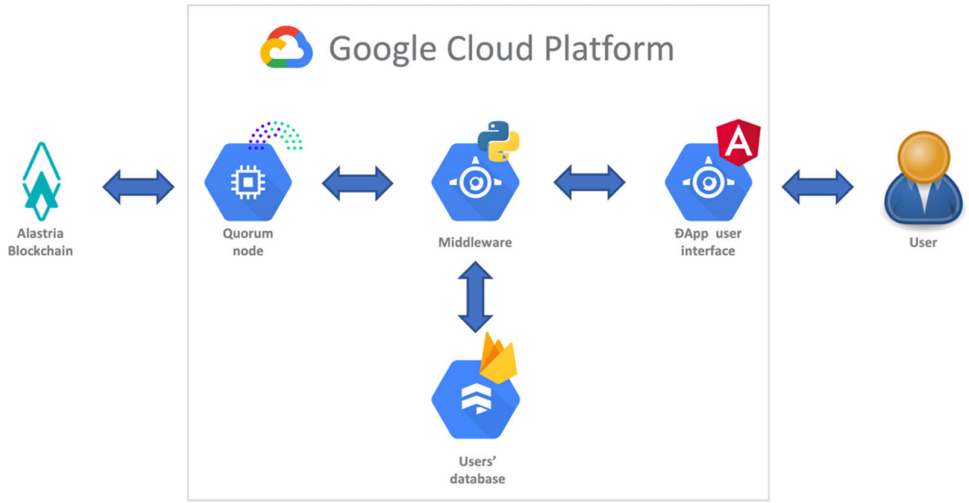


Figure 6. Clockchain ĐApp cloud architecture.

The middleware and the frontend are deployed as GCP AppEngine instances. In this case, this will provide us with total flexibility regarding the number of simultaneous users. Hence, more AppEngine instances will be created dynamically to deal with the users' demand. Opposingly, if the ĐApp is not been used, all AppEngine will be shut down, thus reducing the ĐApp computation expenses to the cost of just maintaining the node active. Finally, the database will act in the same manner, thus generating expenses only when it is accessed.

7. Evaluation

To evaluate our proposal, an experiment was performed aiming at assessing if implementing the proposed framework would improve the usability of a ĐApp. We compared two versions of the Clockchain ĐApp, with and without implementing the proposed middleware, aiming at answering RQ3.

7.1. Experimental Context

We define the main goal of this experiment by using goal question metric [36] as follows: analyzing the implementation of the middleware for the purpose of evaluating the usability of a ĐApp for blockchain researchers in the context of employees from different software development companies working from home due to the COVID-19 containment. With this aim, Table 1 presents the hypothesis that this experiment tried to demonstrate.

Table 1. Main Features of the Experiment.

Feature	Description
Null Hypothesis	H_{0A}: The Clockchain ĐApp has the Same Score Form Usability Regardless of the Implementation of the Middleware. H_{1A}: ¬H_{0A}
Dependent Variable	Usability Score
Independent Variable	Whether the Middleware Was Implemented or Not
Location	Multiple Locations (Experimental Subjects Participated from Home Due to Public Regulations Regarding the COVID-19 Pandemic)
Date	July 2020
Subjects	35 Employees from Different Software Development SMEs from Alicante (Spain)

It is worth noting that, given the containment measures taken by the Spanish government due to the COVID-19 pandemic, we were unable to perform a controlled experiment properly speaking.

Instead, we enrolled as experimental subjects, several software developers working for three different companies who performed the experimental task from home. Such participants were required not to have previous experience with blockchain nor with clockchain to avoid any bias.

7.2. Experimental Design

The experiment consisted in logging into Clockchain and then clocking in or out (depending on the Clockchain version the participants were using). To avoid the learning effect, a 2 × 2 factorial design with confounded interaction [37] was used, as shown in Table 2.

Table 2. Experiment 2 × 2 Factorial Design with Confounded Interaction.

		Task	
		Log in and Clock in	**Log in and Clock Out**
Clockchain Version	**With Middleware**	Group 1	Group 2
	Without Middleware	Group 2	Group 1

It was decided to use as experiment tasks the clock in and out since they are the most common actions that users are supposed to perform when using clockchain. In addition, participants must log in into clockchain before performing such actions. Regarding the clock in action, clockchain enables us to specify both the location (workplace or home) and the time (at that very moment or select a time in the past in the event the user forgot to clock in or out) clock in and out. Besides, a clock out does not require to specify the location since it will be the same as the corresponding clock in. Hence, the participants were required to perform a clock in from home at a time in the past and to clock out at that very time.

To analyze the usability [38], we used the system usability scale (SUS) questionnaire [39], a usability test widely accepted. Thereby, the SUS questionnaire consisted of the following set of questions:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

Each question has to be answered by using a 5-point Likert scale [40], varying from “strongly disagree” to “strongly agree”. Then, for odd questions, the “strongly disagree” answer will score 0 points while “strongly agree” will score 4 points. This score will be reversed for even questions. Finally, each score will be multiplied by 2.5. The sum of the scores of que questions will provide us with the usability score, which can be interpreted according to Table 3.

Table 3. The system usability scale (SUS) Grading Scale [39].

The System Usability Scale (SUS) Score	Letter Grade	Adjective Rating
Above 80.3	A	Excellent
Between 68 and 80.3	B	Good
68	C	OK
Between 51 and 67	D	Poor
Below 51	F	Awful

As far as experimental material is concerned, we provided the participants with two URLs pointing to the versions of clockchain, with and without middleware. Moreover, a printable version of the SUS questionnaire translated into Spanish according to the guidelines presented on [41] is given to the participants in order to gather their answers.

7.3. Running the Experiment

As mentioned before, the experiment was carried out remotely due to public health constraints. Nevertheless, we always had direct contact with the participants since a group video conference tool was used (see Figure 7). Since the experimental subjects belonged to three different companies, we performed three experimental sessions. In each one of the sessions, we started with an introduction, where we explained the experiment procedure, introduced the participants to blockchain and helped them to install metamask in order to perform the tasks related to the clockchain version without middleware. These sessions took around 45 min each.

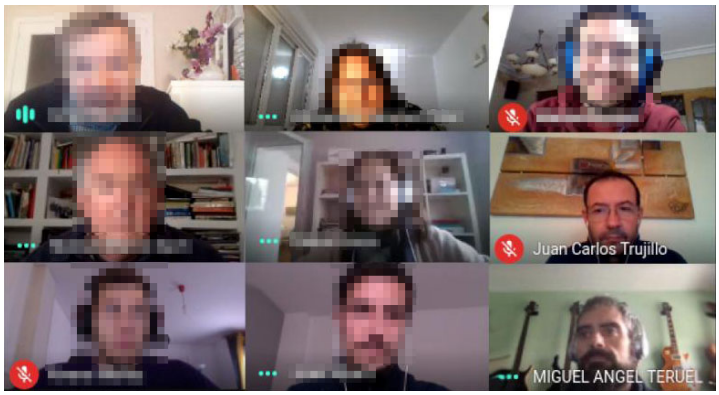


Figure 7. Participants and researchers during a remote experiment session.

After these introductory sessions, the participants were given the experimental material. It is worth highlighting that we decided to use a printable version of the SUS questionnaire to avoid interruptions during the experimental tasks' performance caused by switching between applications. Finally, they were asked to send us a photo of the filled-in questionnaire. During the sessions, there were no dropouts. Table 4 shows some statistics about the experimental sessions.

Table 4. Statistics About the Experiment.

	Company A	Company B	Company C	Total
Number of Participants	7	14	17	38
Percentage of Female Participants	14.29%	30.77%	11.76%	18.42%
Average Age	36.26	33.14	34.47	34.32
Average Elapsed Time in Seconds	234	234	228	231

7.4. Results

After manually recording the participants' questionnaires, which were received in the form of photos, we obtained the results shown in Tables 5 and 6. In the following, such results will be analyzed.

Table 5. Questionnaire Results per Clockchain Version and Question.

				Questions									
Company	**Group**	**Size**	**Version**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
A	1	4	Middleware	8.8	8.8	8.8	8.1	6.3	6.3	9.4	9.4	7.5	6.9
			No Middleware	4.4	0	6.3	4.4	1.9	8.1	2.5	5.6	1.3	1.3
	2	3	No Middleware	5.8	2.5	4.2	5	0.8	7.5	0.8	3.3	3.3	0
			Middleware	6.7	7.5	9.2	8.3	7.5	9.2	8.3	6.7	7.5	9.2
B	1	7	Middleware	8.4	10	9.7	6.9	6.6	7.5	8.4	9.4	7.2	7.2
			No Middleware	3.8	0.6	6.6	5	3.1	6.9	0.3	3.8	2.5	1.3
	2	7	No Middleware	5.4	1.3	5.4	5	2.5	4.2	2.5	2.1	4.2	0.8
			Middleware	7.5	7.9	8.3	9.6	6.3	10	8.8	7.5	7.1	9.2
C	1	8	Middleware	9.1	8.8	8.8	6.9	5.9	6.3	8.8	9.4	7.2	6.6
			No Middleware	5.6	0.9	5.3	5	2.8	6.3	1.3	5.9	2.2	1.3
	2	9	No Middleware	5.8	0.8	4.7	5.3	3.1	4.7	1.7	1.9	4.4	0.6
			Middleware	8.3	5.6	8.9	9.2	7.8	8.9	9.4	6.1	6.7	8.6
All	1	19	Middleware	8.8	9.3	9.1	7.1	6.3	6.8	8.8	9.4	7.3	6.9
			No Middleware	4.6	0.6	6	4.9	2.8	6.9	1.1	5	2.1	1.3
	2	19	No Middleware	5.7	1.3	4.9	5.1	2.5	5	1.8	2.2	4.2	0.6
			Middleware	7.8	6.7	8.8	9.2	7.2	9.3	9	6.7	6.9	8.9

Table 6. Experiment Results per Clockchain Version.

Company	**Version**	**Average SUS Score**	**Average Elapsed Time**
A	Middleware	80.00	73.14 s.
	No Middleware	34.58	168.04 s.
B	Middleware	81.61	75.65 s.
	No Middleware	33.57	157.54 s.
C	Middleware	79.80	73.37 s.
	No Middleware	34.02	157.80 s.
All	Middleware	79.93	72.57 s.
	No Middleware	34.02	158.64 s.

As it can be seen in Table 6 and Figure 8, the clockchain version with middleware obtained better results for usability regardless of the company the subjects belonged to. These results were computed as the average score for all the questionnaire questions. Hence, the middleware version would be classified as SUS-grade A whereas the no-middleware one would obtain an F.

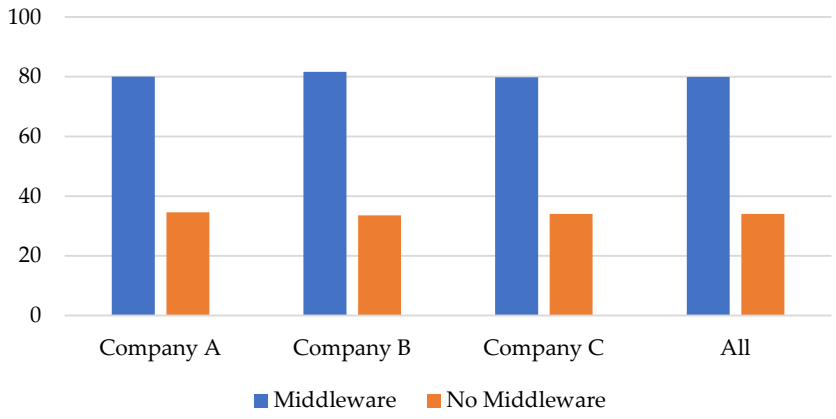


Figure 8. Usability results.

To accept or reject the null hypothesis H_{0A}, a 2-Sample t Test was performed with an alpha of 0.05. Thanks to this test, we could conclude that the means for usability differ at the 0.05 level of significance, with a *p*-value of 1.93e-48. Therefore, with a 95% confidence level, that we rejected the null hypothesis H_{0A}, meaning that the version of Clockchain with the implemented middleware does not have the same score for usability.

Moreover, t Test requires that those data follow a normal distribution. With this aim we performed normality tests for the results for SUS score of both Clockchain versions, considering as null hypotheses the normality of both series of data. It was obtained a *p*-value of 0.63 for the middleware version and 0.59 for the no-middle version. Therefore, the normality of our data is confirmed. This fact is also represented in the histograms shown on Figure 9.

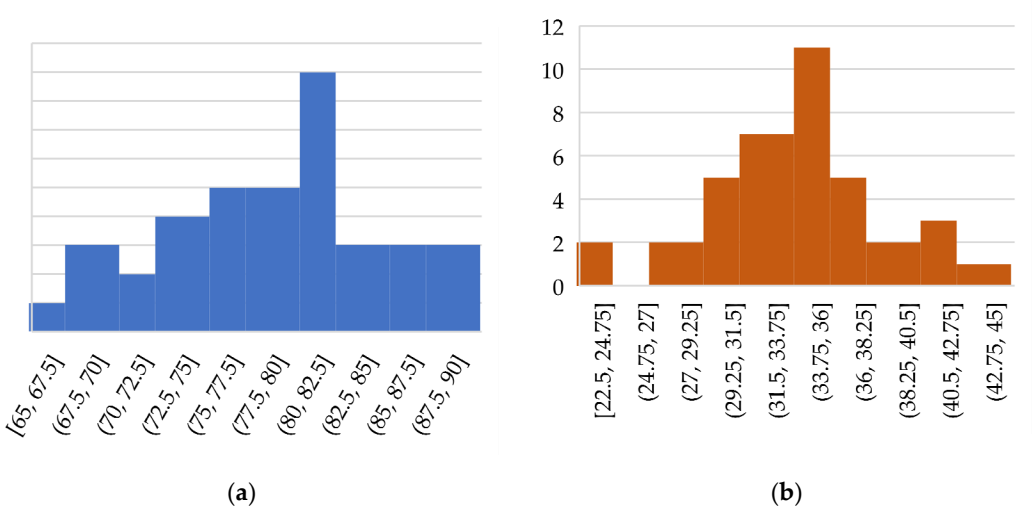


Figure 9. Histogram for the system usability scale (SUS) score of Clockchain (**a**) with middleware and (**b**) without middleware.

Finally, despite not being the main focus of the experiment, it is worth noting the difference on the time the participants took to complete the experimental tasks depending on the Clockchain version they used (see Figure 10). Therefore, not only is the Clockchain version with middleware more usable, but it can also reduce task performance time.

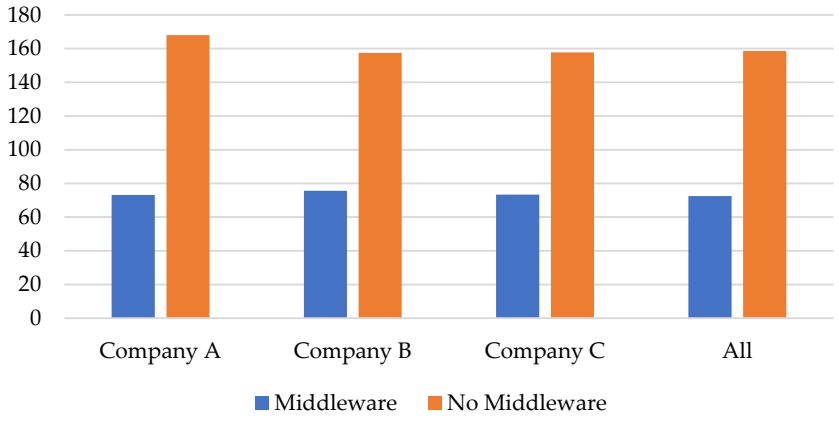


Figure 10. Elapsed time results (in seconds).

7.5. Threats to the Validity

As suggested by Wohlin et al. [42], in the following the most relevant threats to the validity of the experiment described above are analyzed. Such threats are classified into internal, external, construct, and conclusion validity.

- Internal validity: It is related to the influences on the independent variable [42]. The different subjects that participated in the experiment were not informed previously, avoiding social threats. A 2 × 2 factorial design was applied, to avoid learning effect. The subjects were randomly assigned within the groups to cancel out fatigue effects. However, the experiment could not be fully controlled due to the COVID-19 containment measures. Nevertheless, we tried to overcome this shortcoming by using a real-time remote meeting tool.
- External validity: According to Wohlin et al [42], external validity threats are related to the generalization of the experiment. The experimental subjects in the experiment had enough maturity level because the tasks to be carried out were not highly demanding. Moreover, it should also be noted that the practitioners who participated in the experiment had similar results for the experimental tasks. However, a threat we must highlight is that this experiment is only generalizable for participants without blockchain experience. If such participants had experience with this technology, the results could have changed drastically.
- Construct validity: The method used to evaluate the outcome of the experimental task may threaten the Construct validity [42] of the experiment. To avoid this threat, a widely accepted questionnaire was used to evaluate its usability. Moreover, the two clockchain versions used were identical as far as the user interface in concerned, thus only differing on the requirement of metamask to log in and to send transactions.
- Conclusion validity: These threats are related to the statistical relationship between the independent and dependent variables [42]. The statistical power can be considered high, since 38 subjects participated in the experiment, being enough according to the central limit theory. Moreover, we assessed the initial null hypothesis as well as the normality of the data by using statistical tests. Finally, the experiment was not balanced, that is, the number of participants per company were different as shown in Table 4, Company A participation being lower than the other ones.

8. Conclusions and Future Work

Blockchain supposes a new paradigm that not only has revolutionized the world digital economic system but also the foundations of distributed data management. Thus, beyond cryptocurrencies such as Bitcoin, blockchain provides us with the required tools to create smart contracts that will run in a distributed environment. Thanks to these smart contracts, we can develop

blockchain distributed applications, or ĐApp's. These are applications that will run in a distributed manner on the blockchain by obeying the rules codified on the smart contracts.

The problem with a relatively new-flanged technology such as blockchain is that users need to be aware of many concepts before interacting with an application of this sort (e.g., addresses, private keys, wallets, gas, gas price, transactions, signatures, etc.). Therefore, it is very common that a user confronting a ĐApp for the first time fails, due to the pronounced learning curve required to interact with blockchains.

To tackle this issue, which is related to RQ1 (Which is the correspondence between the typical app interaction elements and ĐApp ones?) we analyzed and related the elements of both ĐApps and Apps by using conceptual modeling. Hence, this analysis leads us to the identification of how a user would be able to interact with a blockchain through an App in an easy manner, thus not requiring any knowledge of blockchain concepts.

After this analysis, we proposed the use of a middleware that will ease the interaction with a ĐApp by making users feel like they were using a typical web application, thus answering RQ2 (How can ĐApp interaction be improved for users without blockchain experience?). Indeed, instead of using wallets, public addresses, private keys, and signed transactions, they will interact by using concepts they are familiar with, namely emails, passwords, and web interfaces. In other words, this middleware will make users able to interact with a ĐApp without any blockchain knowledge. Therefore, our middleware consists of five approaches that will facilitate the interaction with a ĐApp: enabling login with email and passwords, sending transactions, decoupling users' credentials from blockchain data, tokens management and security concerns.

To demonstrate how to put the middleware into practice, we presented a ĐApp that tracks employees' schedules by using an Ethereum-based permissioned blockchain. Hence, this ĐApp implements all the approaches of the proposed middleware, thus easing the user interaction without losing the security and reliability of blockchains.

Finally, to answer RQ3 (Does the inclusion of typical app interaction features in a ĐApp improve its usability for users without blockchain experience?) an experiment was performed. With this aim, we enrolled 38 participants from three deferent software development companies with no experience with Blockchain. Then, they performed several tasks with two versions the ĐApp, one implementing the proposed middleware and the other one without it. We conclude that implementing the middleware widely improved the ĐApp usability.

Despite the positive results obtained by this experiment, our work has several limitations. First, the experiment was conducted by using participants with no blockchain experience. Therefore, our results are not generalizable for blockchain advanced users who could consider our proposal not necessary. Therefore, further experimentation could be performed to broaden the usability of ĐApps featuring the proposed middleware. Second, our proposal is Ethereum-based, therefore, we cannot assure that it could be applicable to different blockchains. Because of this reason, additional middlewares are required to fulfill the needs of non-exert users to deal with other blockchains. Finally, the inclusion of Firebase in our proposal makes the ĐApps that implements this middleware not fully decentralized. Consequently, different alternatives need to be considered to foster the proposal's decentralization.

As a future work, we will work towards the model-based automatization of the middleware generation. In this sense, an initial version of the middleware would be automatically generated by taking the smart contracts behind it as an input. Finally, a scaffolding of the ĐApp could also be generated, thus providing web developers with a basic functionality to start building the ĐApp.

Regarding clockchain, it will be analyzed how to provide automatic scalability features to the node without shutting it down. Hence, by providing the node (or nodes) with the required scalability, we will achieve a fully scalable environment able to grow depending on the number of concurrent users.

Author Contributions: Conceptualization, methodology, investigation, software, experimentation, writing–original draft and writing–review& editing, M.T.; supervision, experimentation, and fund acquisition, J.T. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been funded by the ECLIPSE project (RTI2018-094283-B-C32) from the Spanish Ministry of Science, Innovation and Universities.

Acknowledgments: We would like to thank the development team of Lucentia Lab for developing the web interface of Clockchain. Thanks to Laura Ramos for reviewing the grammar of this manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System 2008. Available online: https://bitcoin.org/bitcoin. Pdf (accessed on 5 June 2020).
2. Luther, W.J. Cryptocurrencies, network effects, and switching costs. *Contemp. Econ. Policy* **2016**, *34*, 553–571, doi:10.1111/coep.12151.
3. Swan, M. *Blockchain: Blueprint for a New Economy*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2015.
4. Bradbury, D. The problem with Bitcoin. *Comput. Fraud Secur.* **2013**, *2013*, 5–8, doi:10.1016/S1361-3723(13)70101-5.
5. Sayeed, S.; Marco-Gisbert, H. Assessing Blockchain Consensus and Security Mechanisms against the 51% Attack. *Appl. Sci.* **2019**, *9*, 1788, doi:10.3390/app9091788.
6. Khezr, S.; Moniruzzaman, M.; Yassine, A.; Benlamri, R. Blockchain Technology in Healthcare: A Comprehensive Review and Directions for Future Research. *Appl. Sci.* **2019**, *9*, 1736, doi:10.3390/app9091736.
7. Alammary, A.; Alhazmi, S.; Almasri, M.; Gillani, S. Blockchain-Based Applications in Education: A Systematic Review. *Appl. Sci.* **2019**, *9*, 2400, doi:10.3390/app9122400.
8. Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
9. Christidis, K.; Devetsikiotis, M. Blockchains and Smart Contracts for the Internet of Things. *IEEE Access* **2016**, *4*, 2292–2303, doi:10.1109/ACCESS.2016.2566339.
10. Cai, W.; Wang, Z.; Ernst, J.B.; Hong, Z.; Feng, C.; Leung, V.C.M. Decentralized Applications: The Blockchain-Empowered Software System. *IEEE Access* **2018**, *6*, 53019–53033, doi:10.1109/ACCESS.2018.2870644.
11. Raval, S. *Decentralized Applications: Harnessing Bitcoin's Blockchain Technology*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2016.
12. Mehar, M.I.; Shier, C.L.; Giambattista, A.; Gong, E.; Fletcher, G.; Sanayhie, R.; Kim, H.M.; Laskowski, M. Understanding a Revolutionary and Flawed Grand Experiment in Blockchain. *J. Cases Inf. Technol.* **2019**, *21*, 19–32, doi:10.4018/JCIT.2019010102.
13. Bogner, A.; Chanson, M.; Meeuw, A. A Decentralised Sharing App running a Smart Contract on the Ethereum Blockchain. In Proceedings of the 6th International Conference on the Internet of Things (IoT'16), Stuttgart, Germany, 7–9 November 2016; ACM Press: New York, NY, USA, 2016; pp. 177–178.
14. Yuan, Y.; Wang, F.-Y. Towards blockchain-based intelligent transportation systems. In Proceedings of the IEEE 19th International Conference on Intelligent Transportation Systems (ITSC'16), Rio de Janeiro, Brazil, 1–4 November 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 2663–2668.
15. Zhang, P.; White, J.; Schmidt, D.C.; Lenz, G.; Rosenbloom, S.T. FHIRChain: Applying Blockchain to Securely and Scalably Share Clinical Data. *Comput. Struct. Biotechnol. J.* **2018**, *16*, 267–278, doi:10.1016/j.csbj.2018.07.004.
16. Rocha, H.; Ducasse, S. Preliminary steps towards modeling blockchain oriented software. In Proceedings of the 2018 IEEE/ACM 1st International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB), Gothenburg, Sweden, 27 May–3 June 2018; pp. 52–57.
17. Lallai, G.; Pinna, A.; Marchesi, M.; Tonelli, R. Software Engineering for DApp Smart Contracts managing workers Contracts. In Proceedings of the Distributed Ledger Technology (DLT'20), Ancona, Italy, 4 February 2020.
18. Porru, S.; Pinna, A.; Marchesi, M.; Tonelli, R. Blockchain-Oriented Software Engineering: Challenges and New Directions. In Proceedings of the IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C'17), Buenos Aires, Argentina, 20–28 May 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 169–171.

19. Wessling, F.; Ehmke, C.; Hesenius, M.; Gruhn, V. How much blockchain do you need? In Proceedings of the 1st International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB '18), Gothenburg, Sweden, 27 May–3 June 2018; ACM Press: New York, NY, USA, 2018; pp. 44–47.
20. Pillai, A.; Saraswat, V.; Arunkumar, V.R. Smart Wallets on Blockchain—Attacks and Their Costs. In Proceedings of the International Conference on Smart City and Informatization, Guangzhou, China, 12–15 November 2019; pp. 649–660.
21. Armknecht, F.; Karame, G.O.; Mandal, A.; Youssef, F.; Zenner, E. Ripple: Overview and outlook. In Proceedings of the International Conference on Trust and Trustworthy Computing, Vienna, Austria, 29–30 August 2015; pp. 163–180.
22. Brown, R.G.; Carlyle, J.; Grigg, I.; Hearn, M. Corda: An introduction. *R3 CEV August* **2016**, *1*, 15.
23. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, Y.; et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. In Proceedings of the Thirteenth EuroSys Conference, Porto, Portugal, 23–26 April 2018; p. 30.
24. Taskinsoy, J. Facebook's Project Libra: Will Libra Sputter Out or Spur Central Banks to Introduce Their Own Unique Cryptocurrency Projects? *SSRN Electron. J.* **2019**, doi:10.2139/ssrn.3423453.
25. Valenta, M.; Sandner, P. *Comparison of Ethereum, Hyperledger Fabric and Corda*; Frankfurt School Blockchain Center: Frankfurt, Germany, 2017.
26. State of the DApps State of the DApps—Explore Decentralized Applications. Available online: https://www.stateofthedapps.com/ (accessed on 5 June 2020).
27. Valve Corporation: Steam & Game Stats Available online: https://store.steampowered.com/stats/ (accessed on Jun 5, 2020).
28. Victor, F.; Lüders, B.K. Measuring Ethereum-Based ERC20 Token Networks. In *International Conference on Financial Cryptography and Data Security*; Goldberg, I., Moore, T., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 113–129.
29. Entriken, W.; Shirley, D.; Evans, J.; Sachs, N. ERC-721 non-fungible token standard. Available online: https://eips.ethereum.org/EIPS/eip-721 (accessed on May 11, 2020).
30. Pustišek, M.; Kos, A. Approaches to front-end iot application development for the ethereum blockchain. *Procedia Comput. Sci.* **2018**, *129*, 410–419.
31. Lee, W.-M. Using the MetaMask Chrome Extension. In *Beginning Ethereum Smart Contracts Programming*; Springer: Cham, Switzerland, 2019; pp. 93–126.
32. Iyer, K.; Dannen, C. The ethereum development environment. In *Building Games with Ethereum Smart Contracts*; Springer: Cham, Switzerlan, 2018; pp. 19–36.
33. Jones, M.; Tarjan, P.; Goland, Y.; Sakimura, N.; Bradley, J.; Panzer, J.; Balfanz, D. JSON Web Token (JWT). Available online: https://tools.ietf.org/html/draft-ietf-oauth-json-web-token-06 (accessed on 11 May 2020).
34. Alastria Alastria Spanish National Blockchain Consortium. Available online: https://alastria.io/ (accessed on 22 November 2019).
35. Lucentia Lab Clockchain, Available online: https://www.clockchain.es (accessed on 11 May 2020).
36. Basili, V.R.; Caldiera, G.; Rombach, H.D. The Goal Question Metric Approach. In *Encyclopedia of Software Engineering*; Wiley: Hoboken, NJ, USA, 1994; Volume 2, pp. 528–532. ISBN 9780471028956.
37. Winer, B.J.; Brown, D.R.; Michels, K.M. *Statistical Principles in Experimental Design*, 3rd ed.; McGraw-Hill Humanities/Social Sciences/Languages: New York, NY, USA, 1991; ISBN 978-0070709829.
38. Nielsen, J. *Usability engineering*; Morgan Kaufmann: Burlington, MA, USA, 1994.
39. Brooke, J. SUS: A "quick and dirty" usability scale. In *Usability Evaluation in Industry*; Jordan, P.W., Thomas, B., Weerdmeester, B.A., McClelland, A.L., Eds.; Taylor and Francis: London, UK, 1996; pp. 189–194.
40. Vagias, W.M. Likert-Type Scale Response Anchors. Clemson University: Clemson, SC, USA, 2006.
41. Finstad, K. The System Usability Scale and Non-Native English Speakers. *J. Usability Stud.* **2006**, *1*, 185–188.
42. Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M.C.; Regnell, B.; Wesslén, A. *Experimentation in Software Engineering*; Springer: Berlin/Heidelberg, Germany, 2012; ISBN 978-3-642-29043-5.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).