applied sciences

MDPI

Article

Blockchain: A Tale of Two Applications

Micael Ferreira [1], Sven Rodrigues [1], Catarina I. Reis [2,*] and Marisa Maximiano [2]

[1] School of Technology and Management, Polytechnic Institute of Leiria, 2411-901 Leiria, Portugal; micaelsfer@gmail.com (M.F.); sven.nogueira@gmail.com (S.R.)

[2] School of Technology and Management, Computer Science and Communication Research Centre (CIIC), Polytechnic Institute of Leiria, 2411-901 Leiria, Portugal; marisa.maximiano@ipleiria.pt

* Correspondence: catarina.reis@ipleiria.pt; Tel.: +351-244-820-300

Received: 31 July 2018; Accepted: 24 August 2018; Published: 1 September 2018

Abstract: Bitcoin continues to get more and more attention from the media, mainly because of the volatility of its value and insignificantly associated with the technological innovation. This cryptocurrency is supported by an immutable database and is distributed throughout a network of thousands of nodes, known as Blockchain. One way to ensure that all the concepts behind the Blockchain technology and infrastructure are seized is to conduct the development of one of the most popular context applications for it: a wallet for well-known cryptocurrencies. Yet Another Bitcoin Wallet (YABW) is a hybrid application available for both Android and iOS, which was developed with the Ionic and Angular frameworks. This application communicates with Bitcoin Blockchain to send, receive and store bitcoins; provides a set of features focused on security and user experience, and is available on the Play Store and Apple Store. A rather relevant issue that is becoming a major subject of current research is the application of the Blockchain infrastructure to other contexts that are neither directly connected to cryptocurrencies, nor are finance related. The implementation of a proof-of-concept application proposes the use of a blockchain for a specific case study: the exchange of meal vouchers of an institution amongst students. This is achieved using the decentralized platform Ethereum, which allows us to create a Smart Contract using the Solidity programming language to create a token that follows the Ethereum Request for Comment (ERC), the ERC-20 standard and represents the meal vouchers. This second application uses the architecture defined for YABW, reusing major components and custom developing specific modules to enhance the required features. There is still a lot of research to be done on the non-financial applicability of the Blockchain infrastructure and technology, but for the moment, we have left further evidence that it is possible and is a relative straight-forward process to accomplish from the technological perspective.

Keywords: Android; Bitcoin; Blockchain; cryptocurrency; decentralization; ERC-20; Ethereum; Ionic; iOS; mobile; proof-of-work; Smart Contracts; Solidity; transactions; wallets

1. Introduction

Bitcoin was published in 2008 as the first decentralized cryptocurrency and was proposed by Satoshi Nakamoto whose identity still remains unknown [1]. This cryptocurrency is backed by the Blockchain technology that ensures the validity of a transaction between two entities without a need to use a third/intermediary entity. Through the usage of a distributed ledger spread across thousands of nodes from the Bitcoin network, it is possible to ensure the decentralized validity of a transaction between two peers. The strength of this technology, as well as the success of Bitcoin, has led to the current state of outnumbered cryptocurrencies available in the market [2,3].

Furthermore, distinct frameworks, such as Colored Coins [4] and Ethereum [5], have evolved to push metadata into the initial proposal for Blockchain, establishing the ground for the development of improved and powerful applications.

There is a growing demand for the use of Bitcoin, especially for the applicability of Blockchain technology in contexts other than the financial one [3]. It is, therefore, of great importance to explore possible solutions.

Therefore, two main objectives are defined: the development of a Bitcoin Wallet application [6] that allows to send, receive and save bitcoins; and a case study of which main purpose is to obtain a proof of concept, which could potentially lead to the deepening of theoretical concepts in an attempt to solve a proposed challenge: a non-financial application using Blockchain technology [7] not directly associated to the cryptocurrency context. The development of the case study is expected to shed some light on the advantages and disadvantages that arise in the use of alternative frameworks, such as the Ethereum Request for Comment (ERC), the ERC-20 standard [8], for this type of projects.

For the development of Yet Another Bitcoin Wallet (YABW), we conducted a rather extensive research and review study that allowed us to identify the major features available in similar application wallets. We used the results to prioritize the development of the features of YABW. Our aim was to accomplish a highly scalable architecture that could be easily (re)used for other contexts. Section 2 provides a summary of the initial research, a description of the proposed architecture and the actual flow of the YABW application.

We envision a real implementation of the second application ("CriptoSenhas") in a concrete setting, but the scope and all the details that must be addressed for this to happen are beyond our work. Nevertheless, we believe that our contribution is an initial small enhancement that can actually be the beginning of a new era for highly bureaucratic environments, while promoting transparency and shared responsibilities, as we explain in Section 3.

Finally, in Section 4 we present the conclusions and major achievements of our work while leaving an extended list of topics to extend the work.

2. Yet Another Bitcoin Wallet (YABW)

The Yet Another Bitcoin Wallet (YABW) application is a multiplatform application, which allows the user to create Hierarchical Deterministic (HD) wallets [9] for Bitcoin and provides a set of features for the created wallets that can be used with a simple graphical user interface (GUI).

Before the development of the application, the requirements were elicited, and several choices were made regarding the functionalities to implement:

- Create or import Bitcoin wallets;
- Show the balance and transactions of a wallet;
- Show the details of a specified transaction;
- Show wallet addresses to receive bitcoins;
- Send bitcoins to other addresses;
- Show pending transactions and cancel or complete the sending of them;
- Backup a wallet;
- Remove a wallet from the application;
- Enable or disable PIN code protection;
- Change the Bitcoin unit;
- Change the currency used to convert Bitcoin's value;
- Change Bitcoin Wallet Service Uniform Resource Locator (URL).

Some of these features' implementation will be further detailed with the help of Navigation Diagrams.

For the development of the application, the framework Ionic [10] was used, with which a single code base in Javascript can compile to applications for the Android and iOS platforms. The code that was developed was as agnostic and modular as possible. The final architecture of the application is presented below.

2.1. Related Work

Prior to the development of the YABW application, we conducted a rather extensive research and review study that allowed us to identify the major features available in similar applications.

Mobile application wallets have a high level of demand mainly because of the fact that mobile phones are personal devices usually not shared amongst untrusted parties, thus providing a secure environment to hold a wallet. Furthermore, mobile phones allow an "always online" experience through their network (3G, 4G, Wifi, et al.) connections to the Internet. Thus, researching these mobile wallets was considered fundamental to proceed with the work here described. After the conclusion of the research process, we reviewed the most used (downloaded) applications.

An analysis of the applications reviewed allowed us to consider the following basic operations: check wallet balance, send and receive cryptocurrencies, and check the transactions list. All the wallets reviewed provide this basic set of operations [11–21].

In order to allow users to store their wallet information, backup and restore features are required. Most applications provide these two features and also ensure an extra level of protection though the usage of a PIN code that restricts access to functionalities that have a cost or endanger a wallet.

In order to allow users to move between several wallet applications and transfer their wallet information amongst them, exporting and importing a private key is required.

While Copay [11], Bitcoin Wallet [12], Breadwallet [13], Mycelium [14] and ArcBit [21] allow users to import their private key, only Copay, Mycelium and Electrum [17] allow users to export their private key.

Finally, only Copay and Mycelium allow users to create a new wallet. This allowed us to conclude that both Copay and Mycelium are the most complete applications on the market, backed by the number of actual downloads each application has.

We used these results to elicit and prioritize the requirements of the features of YABW. We started by including the basic set of operations, ensured that the backup and restore features were achieved, and then implemented the remaining features, including the creation of new wallets.

In order to achieve such a development, we also conducted a research on existing Application Programming Interfaces (APIs) that already deal with most of the implementation issues regarding the access to the blockchains that support cryptocurrencies, and this allowed us to focus on the development of the core generic architecture for YABW. We decided to use the Bitcore Wallet Service (BWS) [22] that provides a specific client library in Javascript.

2.2. Architecture

The architecture of the application is represented in different levels of abstraction according to the C4 model [23], allowing a simplified visualization by each level with the information of the interactions between different components.

Figure 1 shows the system context in a global way. It describes the interaction of the user with the application and the application interaction with two external services—whether to obtain data or perform actions.

Figure 2 shows some details allowing the visualization of the interactions within the structure of the application (within the dashed area) and also the external dependencies on it.

Internally, the application makes use of two providers. The *StorageProvider* stores the data on the device, and the *WalletProvider* provides all the methods necessary to interact with Blockchain and to get the currency conversion rates. The use of providers decreases code repetition and consequently the risk of bugs in the application. It facilitates its corrective and evolutive maintenance, since the methods used are in a single place, allowing to reuse the functionalities for different components of the application. Several variables have been created in the *WalletProvider*, including the *wallet* array that contains all the user's wallets (initialized when the application is started). All subsequent accesses to the wallets will be made to this variable, thus avoiding calls to the *StorageProvider* that have a higher computational cost, saving computational resources and consequently device battery.

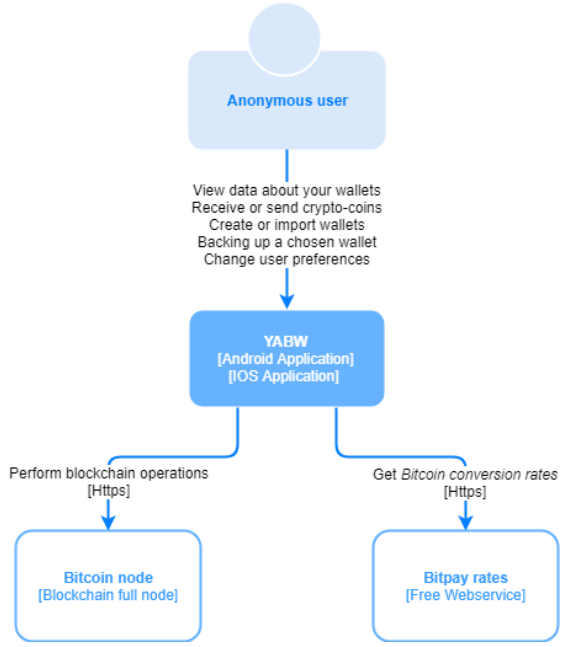

Figure 1. System context diagram [C4–Level 1].

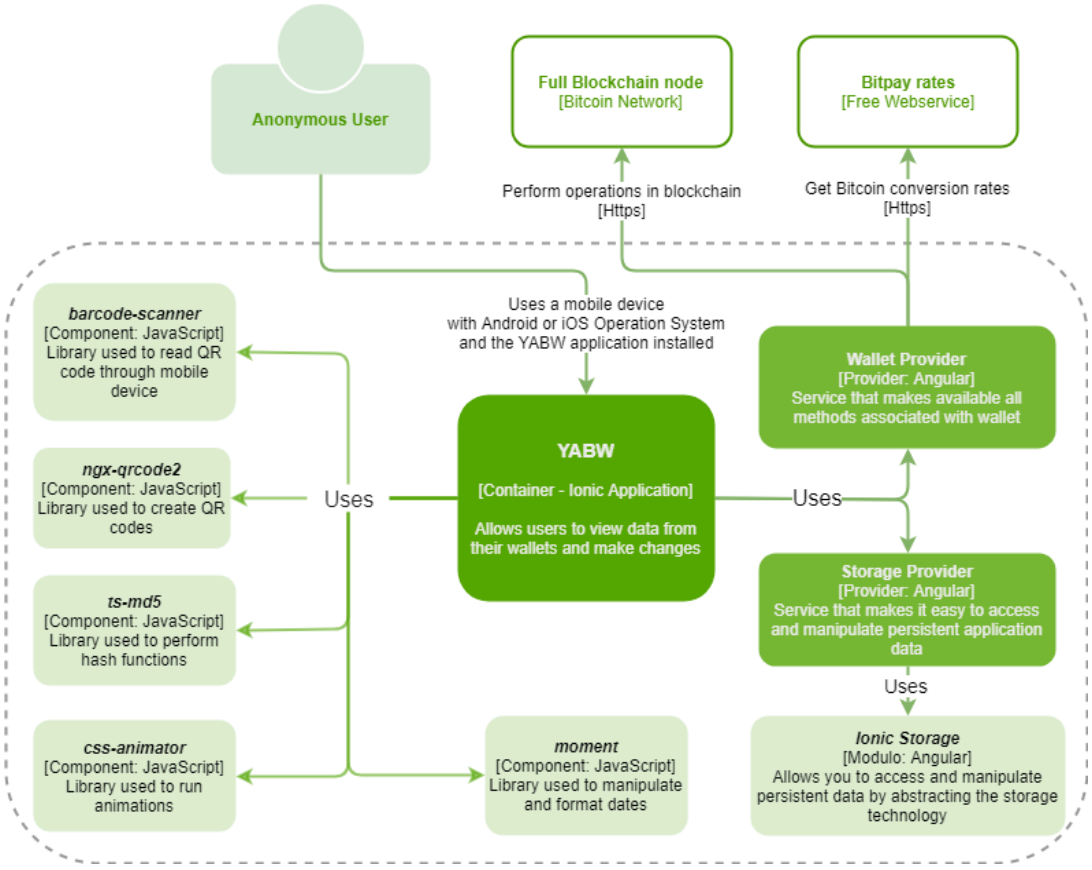

Figure 2. Containers diagram [C4–Level 2].

A large part of the application's functionality works asynchronously, such as calls to the BWC library methods, calls to *StorageProvider* to get data stored on the device, and checking the Internet connection. In order to ensure the correct treatment of these asynchronous calls, Javascript Promise

was used that allows listening for asynchronous data or possible errors and manages these cases without compromising the main flow of the application [24].

In order to inform the user of the lack of connection to the network, a functionality has been implemented that receives operating system events from the mobile device, when it is connected to or disconnected from the network. In order to continuously listen for these events, Observables and Subscribers of the *RxJS* library [25] were used, and an Angular component was developed to allow the display of real-time connection failures in every view of the application.

The application further uses 6 open source libraries (light green) that complement the application itself–YABW (dark green).

Figure 3 represents an example of the interactions of a specific component of the application: the *WalletPage* component. This component makes use of the developed *StorageProvider* and *WalletProvider* providers and only uses the external library *moment* to format the dates that are presented to the user.

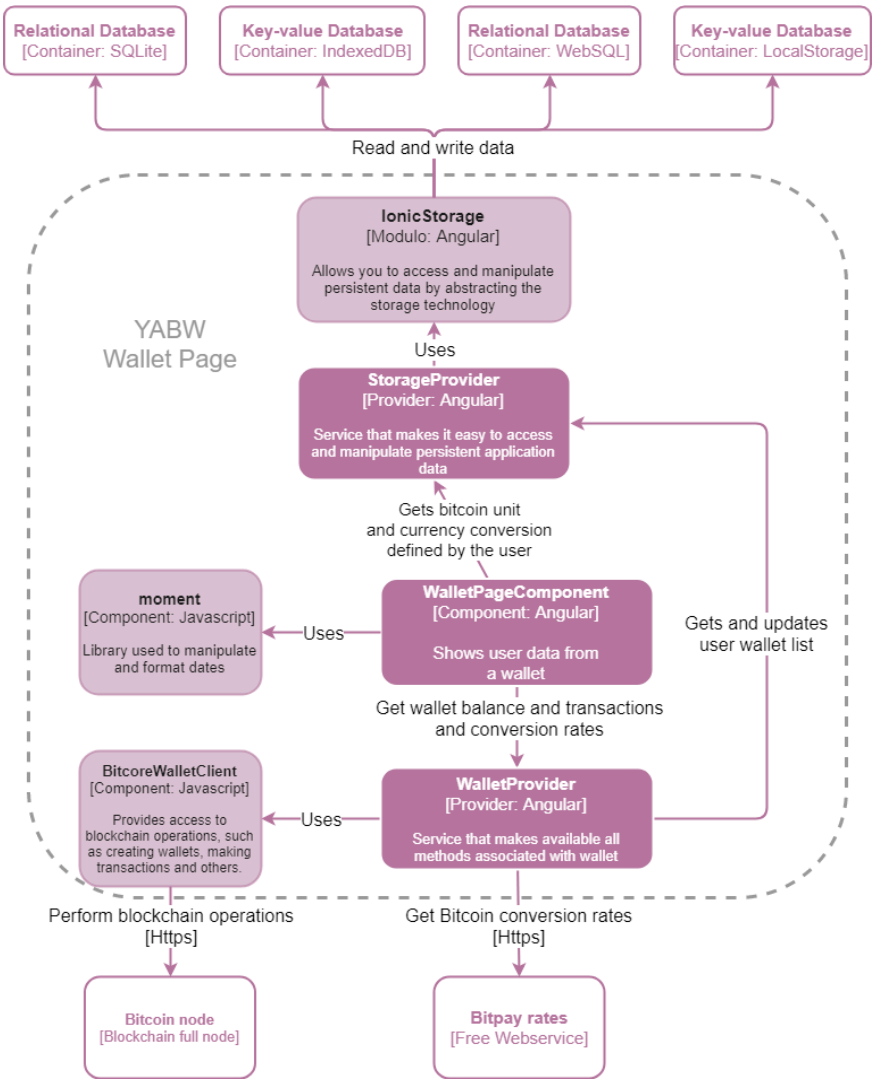

Figure 3. Yet Another Bitcoin Wallet (YABW) components—an example [C4–Level 3].

2.3. Implementation Details

Some of the features developed have high security considerations and implementation details that we consider to be relevant and will depict below.

2.3.1. "IntroPage" View

When the user uses the application for the first time after installing it, a sequence of initial steps is presented for an introduction to the application. This is accomplished by setting up a flag/key *intro-shown* stored in the *StorageProvider* that avoids this introduction to occur more than once.

As shown in Figure 4, the user can activate the PIN code to increase the security of their wallets and create his/her first wallet.

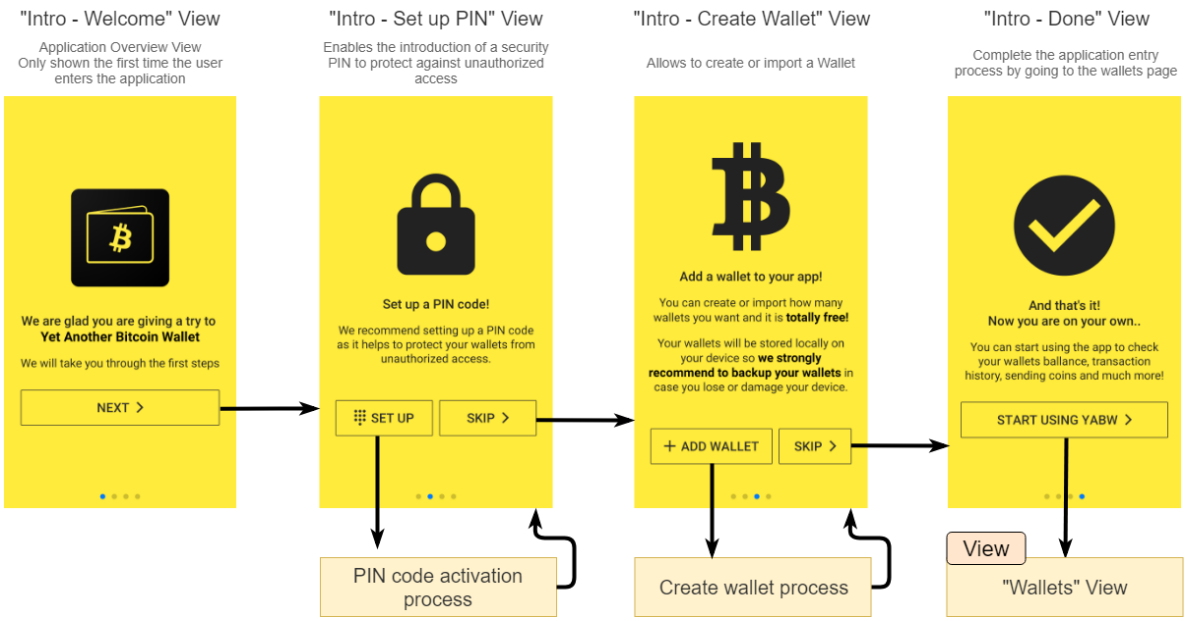

Figure 4. Navigability diagram of the application introduction views.

2.3.2. General Navigation for the Options: "Wallets", "Receive" and "Send"

The application provides a navigation bar that gives access to the "Wallets", "Receive" and "Send" views, as presented in Figure 5.

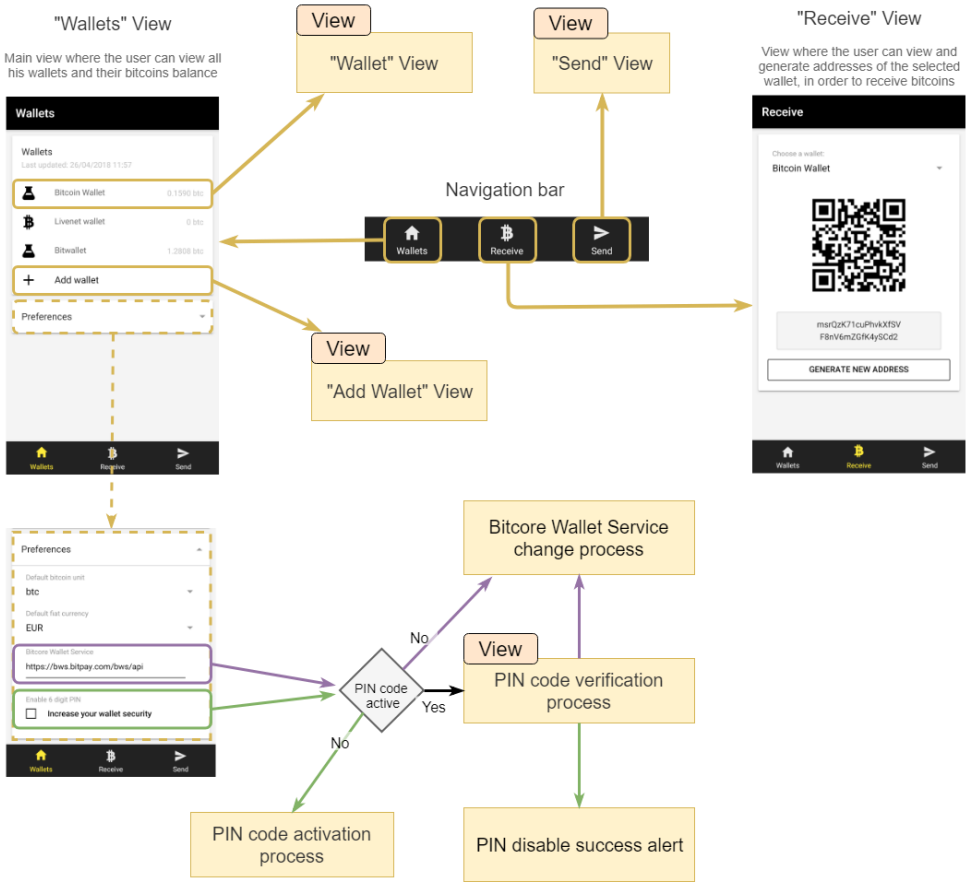

Figure 5. Navigability diagram of the navigation bar and the "Wallets", "Receive" and "Send" views.

In the "Wallets" view, all the user's wallets, available in the *WalletProvider*, are listed, and the user can access each wallet individually. It is possible to change the Bitcoin unit, the conversion currency, the BWS URL and enable/disable the PIN code protection. All of these preferences are managed by the *StorageProvider* and are used throughout the application.

When the BWS URL is changed, only valid APIs are accepted, ignoring any attempt to enter an incompatible URL. This validation is done by verifying the valid instantiation of the Bitcore Wallet Client (BWC) library with the provided URL (*WalletProvider).*

The "Receive" view shows the current Bitcoin address of the selected wallet. It is also possible to generate new Bitcoin addresses, as it is recommended that for each transaction made, a new Bitcoin address should be generated to increase the anonymity and security of the transactions. A feature for copying the Bitcoin address, shown in the view, to the clipboard was also implemented, so the user can share his/her address. The Bitcoin address can be copied by pressing it for 1 s.

Handling Wallet's Bitcoin Addresses with the WalletProvider

A Bitcoin Wallet should normally allow you to generate multiple addresses, as it is not recommended to reuse addresses to receive bitcoins. However, BIP44 [26] defines a limit of 20 consecutively generated addresses without any transaction, and beyond this limit, the user is unable to generate more addresses.

To improve the user experience, an algorithm is created, that is, once it reaches the limit of the 20 generated addresses, it shows the previously generated addresses that have not yet been used. For this, it is necessary to use two *Map* type variables: the variable *limit Reached Wallets Map* that has the identifier of the wallet as a key, and a Boolean that indicates if the wallet already has already reached the limit or not as a value; the variable *wallet Not Used Addresses* that has the wallet identifier as a key, and an array of all the unused addresses as values in that wallet. The unused addresses are

obtained through a recursive search that results from the difference between all the addresses already generated from the corresponding wallet and from all the addresses contained in transactions already made from the same wallet (addresses used). Figure 6 shows a sequence diagram that presents this algorithm in a more simplified way.

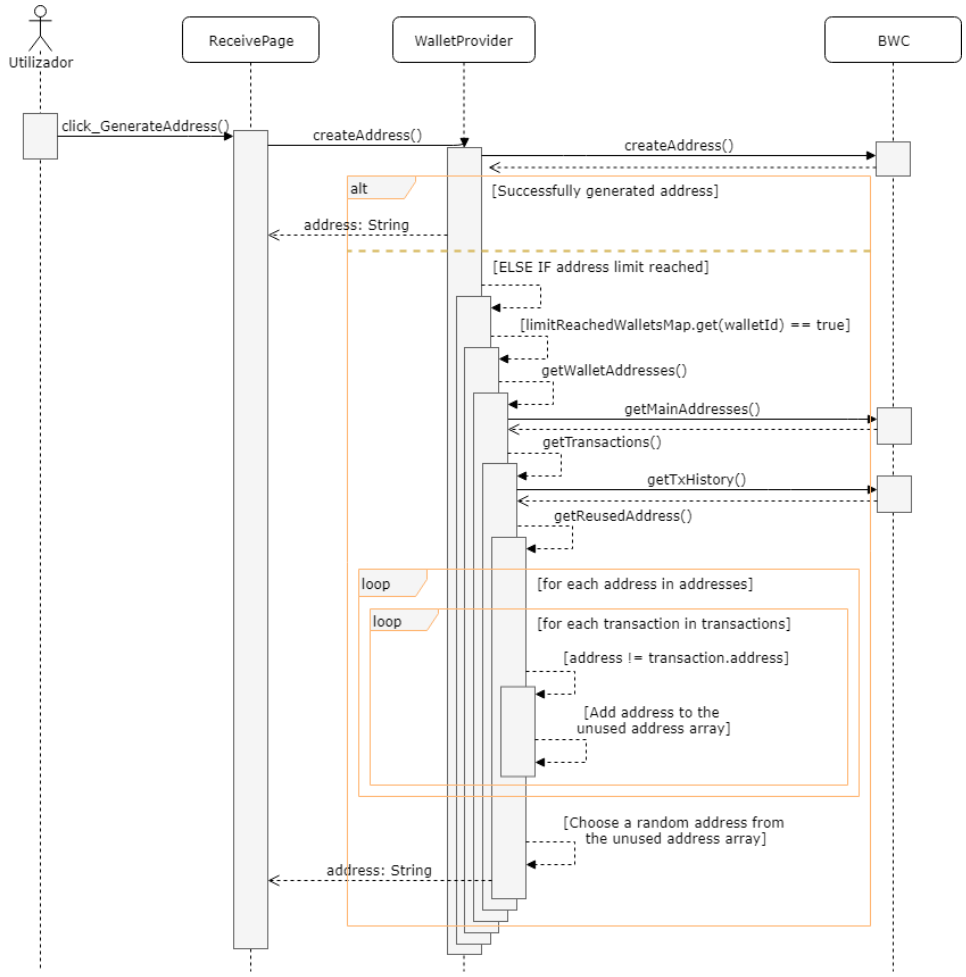

Figure 6. Sequence diagram of the algorithm for address reuse.

2.3.3. PIN Code

The PIN code consists of entering and confirming a 6-digit code that is asked for, to execute certain actions of the application. When a valid PIN code is inserted, a hash function is applied to it and it is stored in the *StorageProvider*.

PIN code verification corresponds to a major security feature implemented in the application. The PIN code aims to minimize the inconvenience caused in situations of unauthorized access to the user' mobile device and that could somehow compromise their wallets.

When the PIN code is active, the following actions are restricted:

- Disabling the PIN code: without this restriction, a malicious person with access to the phone could easily disable the feature, and will be able to access the rest of the application;
- Changing the BWS URL: this action has been restricted since this is the service that handles all wallet operations. Changing this URL to a malicious one could totally compromise the correct functioning and security of the application and its wallets;
- Adding wallets: this action is restricted to prevent unauthorized creation of wallets, which could create inconvenient confusions;

- Removing a wallet: removing a wallet has serious consequences, and there is even the possibility of completely losing access to it and all its bitcoins if your owner does not have a backup of the wallet;
- Backing up a wallet: during the backup process, the mnemonic phrase is displayed for the user to write on a safe place. Access to this mnemonic phrase allows full access to the wallet by importing it on a Bitcoin Wallet app;
- Sending bitcoins or accessing pending transactions: sending bitcoins is an irreversible action, after publishing a transaction to the Bitcoin network, you cannot "claim" the amount back.

2.3.4. "Add Wallet" View

In Figure 7, one can see that in the "Add Wallet" view, the user can choose to either create a new wallet or import an existing one.

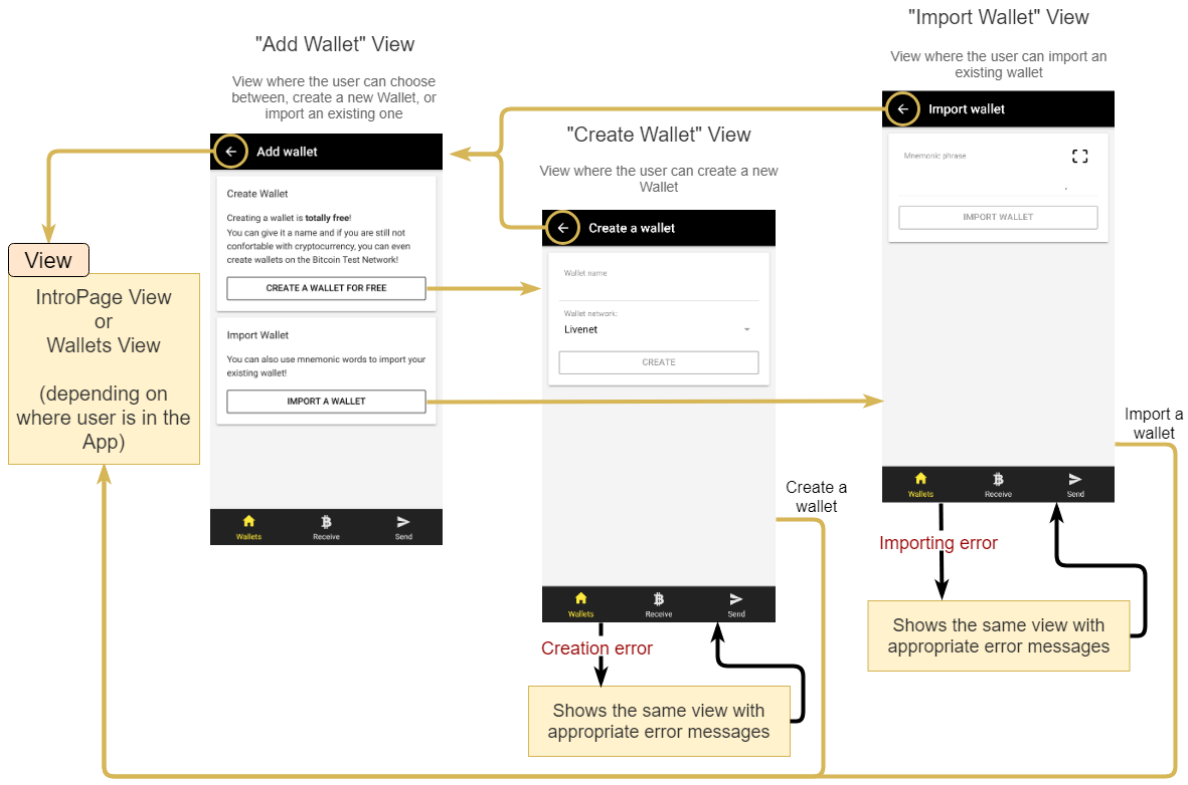

Figure 7. Navigability diagram of the "Add Wallet" view.

While creating a new wallet, in addition to entering a name for it, the user also has the possibility to choose the network of Bitcoin, to which the wallet will belong. The network "testnet" may be a good initial option that provides an opportunity for users to test the application using bitcoins without any associated value [27]. The "livenet" network is the main Bitcoin network, where this cryptocurrency has a value. When creating a new wallet, a random mnemonic phrase is generated with the BWC library, and from this, it obtains the private and public keys that are then stored in the *StorageProvider* wallet list.

To import a wallet, the user can enter the mnemonic words manually in the available field or use the Scan option if he/she has access to the mnemonic Quick Response (QR) code to fill the field automatically. The Scan feature is useful if you want to, for example, directly import a wallet from another application. When importing, the wallet is obtained through the BWC library with the given mnemonic phrase and the wallet is added to the *StorageProvider* wallet list.

2.3.5. "Wallet"—Details

In this view, the user has access to the data and actions of the selected wallet, as shown in Figure 8. When the wallet has not been backed up yet, a warning message and a button that forwards the user to the "Backup Wallet" view also appear.

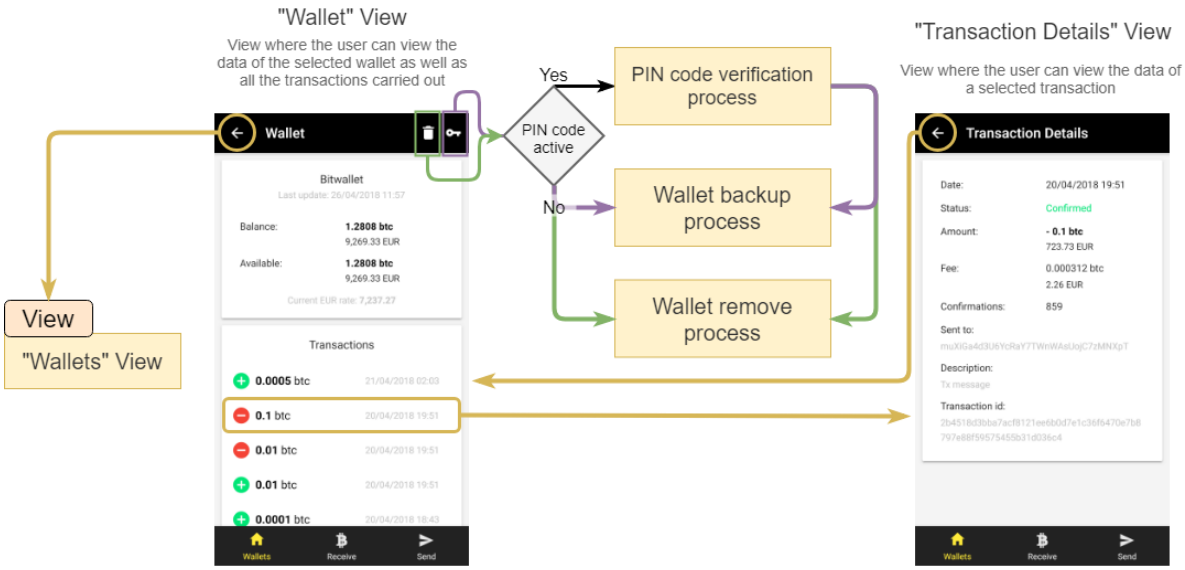

Figure 8. Navigability diagram of the "Wallet" view.

It is possible to check the "Balance": the total amount associated with the wallet—the sum of all bitcoins of all their addresses; and the amount "Available": the total amount of available bitcoins that a user can use. When a transaction is made in addition to the value sent to the destination address, the remaining value in the original address is sent to a new address of the origin wallet, which also has to wait for the confirmation of the transaction.

The view also shows a list of transactions with the value and the date of the transaction. Initial 5 transactions are obtained through the BWS, and if the user has more transactions, each time the user scrolls to the end of the view, another 15 transactions are obtained. Each transaction can be accessed individually to obtain the following information: date, confirmation status, value and its transaction rate, number of confirmations, destination Bitcoin address, transaction description and finally the identification code.

2.3.6. Wallet Backup

Figure 9 shows that the backup process is divided into three steps: a detailed information about the process, the presentation of the mnemonic phrase and its QR Code, and finally a confirmation of the mnemonic phrase to ensure that it has been correctly registered.

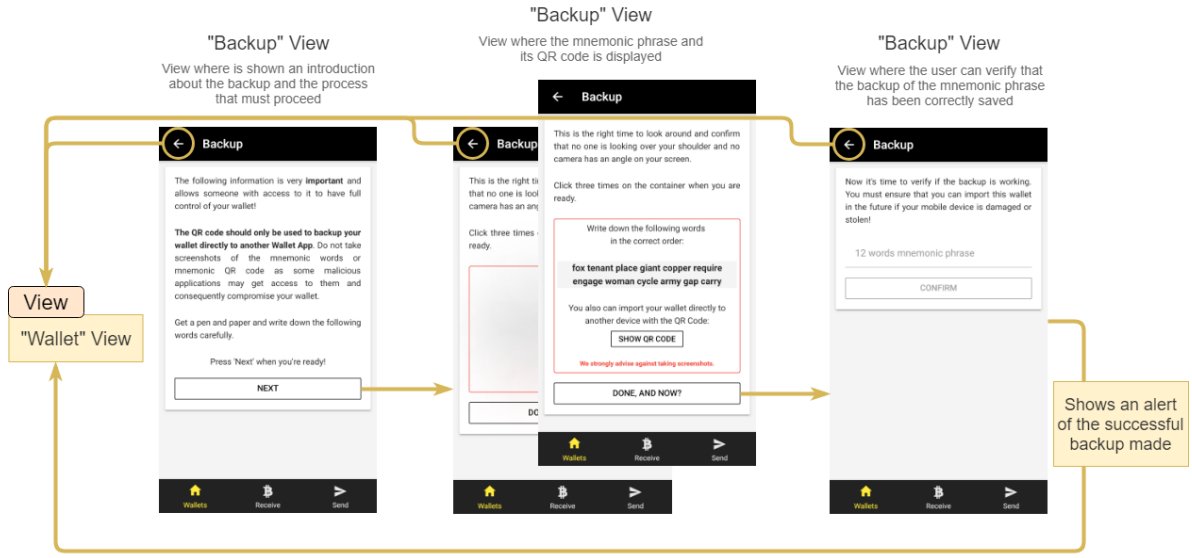

Figure 9. Navigability diagram of the process of backing up a wallet.

The mnemonic phrase is very sensitive information, because having access to it grants full control of the wallet and its cryptocurrencies. Therefore, in the presentation phase, it is hidden and only after three clicks in the sentence box. When the user confirms the mnemonic in the last step, the inserted mnemonic is compared to one of the wallet, and when everything is valid, the wallet is considered as backed up.

2.3.7. Removing a Wallet

As shown in Figure 10, before removing a wallet, the user is advised to back it up, if he/she has not done it yet. The user is warned about the consequences of this process. If the wallet balance is positive, it is presented to the user and the confirmation button requires 3 clicks to complete the removal. Removing a wallet removes it from the *StorageProvider* wallet list.

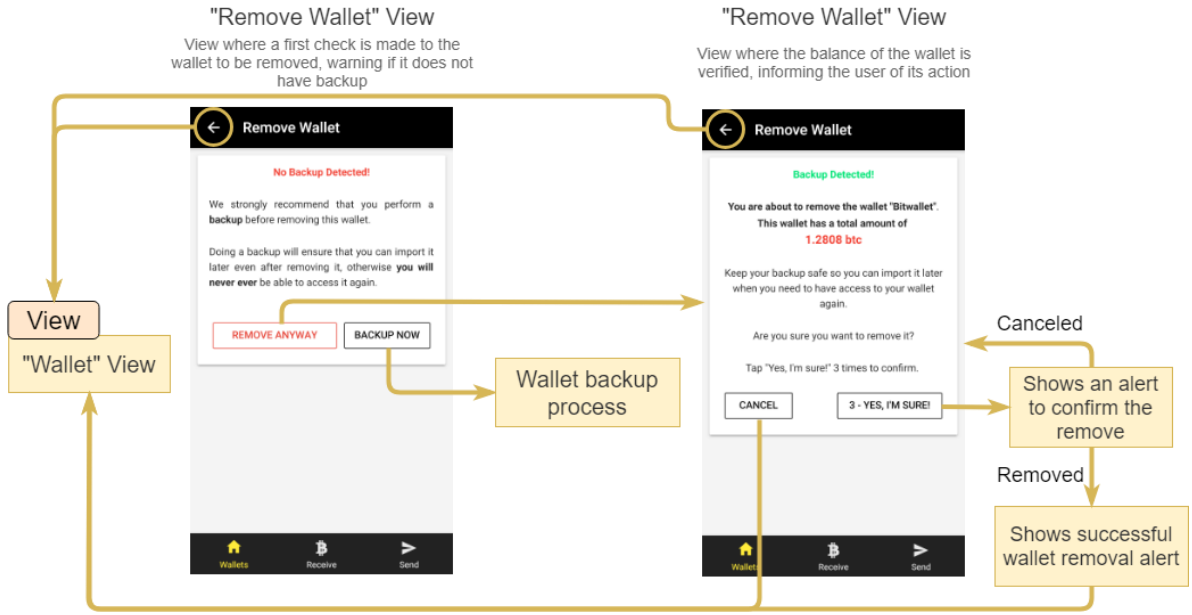

Figure 10. Navigability diagram of the process of removing a wallet.

2.3.8. Sending Bitcoins

Figure 11 shows the "Send" view, where the user can send a number of bitcoins to a Bitcoin address of the same network (testnet or livenet) or can also access the pending transactions.

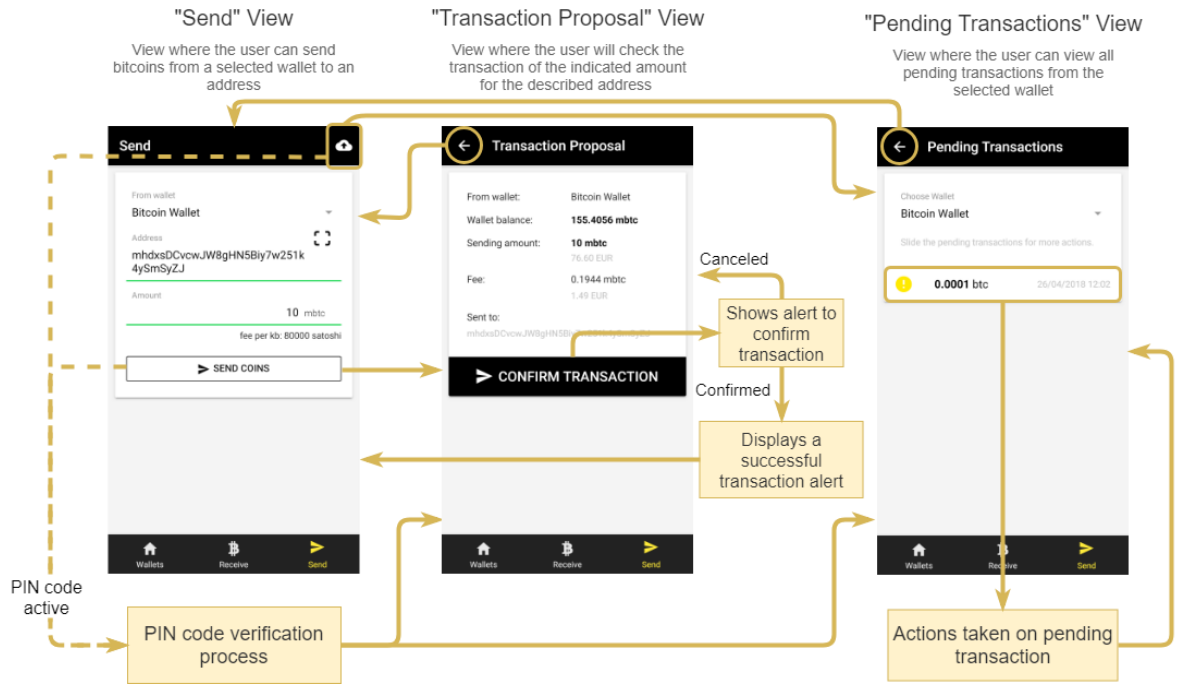

Figure 11. Navigability diagram of the "Send" view.

Internally, the functionality of sending cryptocurrencies goes through a 3-step sequence: (1) creating a transaction proposal, (2) publishing and signing the transaction, and (3) broadcasting the transaction, which are transparent to the user.

Whether manually or by scanning a QR Code, the user only has to enter the destination address and the "Amount" field and press the "SEND COINS". If everything goes as expected, a transaction proposal is created with the data entered by the user (step 1). When confirming the transaction, the user is requested to provide an additional confirmation with information about the effect of this action, and if agreed, another of the internal steps is performed: the publication and signature of the transaction (step 2). After executing this step, the addresses needed to send the number of bitcoins are zeroed, thus blocking the remaining balance and the bitcoins associated with these addresses. Next, a final confirmation request is presented, informing the user that after the confirmation, the transaction can no longer be canceled. If the user agrees to proceed, the third and last internal steps will be performed, which consist of broadcasting the transaction to the Bitcoin network. If the user cancels the process in this last phase, the transaction goes to a list of pending transactions.

In the list of pending transactions, the user can choose to cancel the transaction, or end it (see Figure 12).

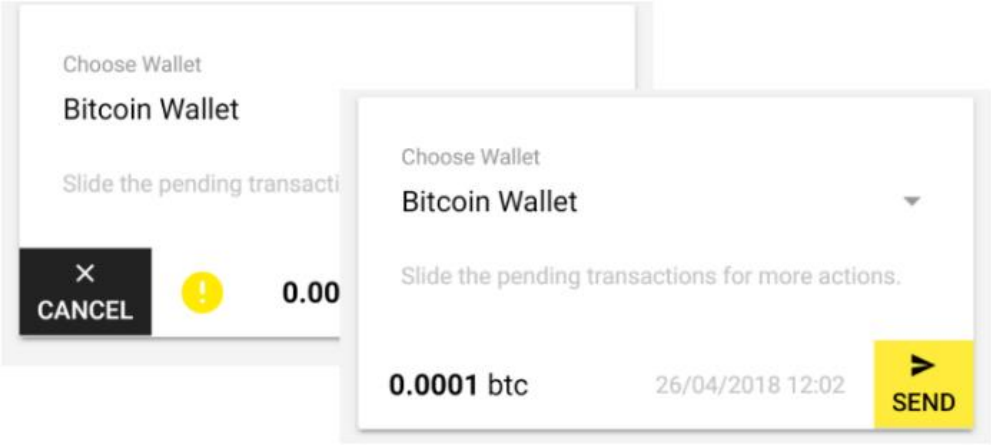

Figure 12. Demonstration of two possible functionalities to perform in a pending transaction.

2.4. Acceptance Testing

Tests are important in the development of a software project, as there is a need to validate and verify the code to facilitate maintenance and even understanding.

The YABW application was implemented according to Behavioral-Driven Development [28], and to automate the process, we used the *jasmine* [29] and *karma* [30] libraries, which provide us with the methods necessary to develop the tests and the browser environment support to run them, respectively. This allowed a better definition of the requirements for the application and the usage of mock data instead of the actual real data.

Besides the automatic testing ensured, the application was also presented to actual users in a specific workshop, where the direct observation of the users and the feedback collected allowed us to improve the application that is available in both stores [31,32].

3. Case Study—The "CriptoSenhas" Scenario

The case study was carried out with the main purpose of applying the Blockchain technology for the transactions of any type of assets in an indirect financial context. The case study should respond to a need for any educational institution that provides meals, implementing a meal voucher mechanism that allows vouchers to be exchanged not only between the institution and users, but also among users.

We envision a real implementation of "CriptoSenhas" in a concrete setting and believe it to be a small contribution to a smart city environment [33].

At the institutional level, community well-being can be enhanced by enabling students to use the institutional services, promoting the quality of the services and the satisfaction of the students. Canteens and cafeterias can have a bad reputation, mainly due to the lack of quality of the food they serve. The quality of the service can be easily affected by the fact that the people responsible for the meals are unable to predict the amount of meals required. This can lead to one of two possible outcomes: a reduced number of meals that promotes a "fast food" cycle upon the arrival of unexpected students or a large amount of wasted meals upon the absence of students that already bring their vouchers but do not show up. Furthermore, public educational institutions receive governmental financial support for the meal vouchers that they must offer to the students. Thus, each meal that is sold to a student has a higher effective cost than what is actually charged. The measurable expenses from the wasted meals or the ingredients, required to be in stock to provide an effective answer to serve the unplanned meals, are enough to engage in a deeper process to control and monitor the meal vouchers being sold and used. Options, such as lowering the price of the meal if the voucher is bought a couple of days before the day it will be used, have proven to be very ineffective per se. There is no staff member allocated to control and monitor the meal vouchers. This is a highly intensive administrative job that could be easily automated, which we believe.

"CriptoSenhas" can also be transformed into a rewarding system, where students will be rewarded by the correct usage of the meal vouchers and penalized if there is an expired unused meal voucher, for example. This would promote the shared responsibility for the measurable expenses between students and the institution. The fact that this is not contemplated in the development is due to the fact that we want to ensure that all that could lead to be read as "money" should not be a part of the application. The same applies to the fact that we do not propose the solution for the financial transactions that occur upon the trading of the meal vouchers. We assumed it could be addressed offline and should continue as long as the context is not totally closed.

In fact, "CriptoSenhas" could be one of the modules of a bigger project that was responsible for ensuring the identity, the academic curriculum and several other services promoted inside a campus. For now, it is the first step that can actually be the beginning of a new era for highly bureaucratic environments, while promoting transparency and shared responsibilities.

Initially, the idea was to use the ColoredCoins framework [4], a framework that adds metadata to Bitcoin transactions, thus registering additional information to Blockchain, which, in our case, would be the exchange of meal vouchers from one entity to another. The idea turned out not to go forward. As we progressed in the research and implementation, we came across technical issues concerning the inactive community of the ColoredCoins framework and its unstable test servers.

As an alternative and an obvious next step, we have redirected the implementation efforts for the decentralized platform Ethereum [5], which will be explained in more detail in the following topic.

The use of the technology analyzed and applied to the case study led to the development of a mobile application named CriptoSenhas, which will also be discussed in this section.

3.1. Related Work

The existing literature revealed that there are similar approaches to solving challenges from both academia and university scenarios, and out of the financial scope. The approaches we found are proposals for solving obscure issues, addressing them through the usage of the Blockchain technology.

A solution to a token-based peer-review payment system in Information System journals is proposed in [34]. It appears as an opportunity to address the archaic peer-review process currently installed and as a way to address the "tragedy of the commons" that reflects the limited capacity of academic peer-review, in an era of rising demand for reviewers. This proposal does not refer to an actual implementation and real-life application.

Shelper et al. propose a cryptocurrency loyalty program named Unify Rewards [35]. The program was tested at the University of New South Wales (UNSW) Sydney campus and defines a strategy for the engagement of participants (students and staff) in transacting with a choice of 12 retailers (in campus). Participants receive an initial bonus (in Ether–Ethereum cryptocurrency) upon joining the program. They are able to easily earn cryptocurrency either by executing a business transaction or positively answering marketing inquiries and upon their wish, they can also easily redeem the amount they earned. Their pilot project with an actual and real application allowed the authors to conclude that loyalty programs that use cryptocurrencies are extremely engaging. While they have limitations, they can be overcome through the effective definition of the strategy to include the cryptocurrency option. This work, despite all the non-financial context that is implied in the strategy definition, actually recurs to money (Ether) to achieve the engagement. It is the definition of the strategy that we find extremely inspirational for our work.

NTUSToken is the name of the token of a rewarding system created within the Smart Campus Project of the National Taiwan University of Science and Technology [36]. The program defines a strategy for the engagement of students that involves them in a gamifying context that rewards them with "good behavior" while on campus. Despite the conceptual similarities to the work in [35], their goal was to create a decentralized rewarding system capable of awarding and retrieving tokens to the university students, depending on their behavior, but without the direct inclusion of the cryptocurrency

option. In order to accomplish their proposal, they created a private network to avoid paying the fees to the miners, recurring to the Ethereum platform and using Smart Contracts [36].

Three generations of blockchains are often referred to in the literature, namely, "Blockchain 1.0 for digital currency, Blockchain 2.0 for digital finance, and Blockchain 3.0 for digital society" [37,38]. Nowadays, these generations coexist, but there is a maturity adoption model already proposed by Wang et al. to guide institutions to make blockchain adoption decisions more systematically [39].

Hence, while YABW is a common example of an application for Blockchain 1.0, the works described in [34–36] are evidence of Blockchain 2.0 and good entry points into Blockchain 3.0. As a matter of fact, adopting the usage of smart contracts as described by Diaz et al. allows organizations to be autonomous based on the autoexecution of code. Our case study application, "CriptoSenhas", is definitely a Blockchain 2.0 application and also a proposal for an entry point in Blockchain 3.0.

"The essence of Blockchain lies in its ability to support trustworthy transactions via networked computation in place of human monitor and control." is stated by Zhao et al. [37], and while Blockchain is not a "magic bullet", evaluating its potential can bring innovation to a highly bureaucratic environment. In order to proceed with the evaluation of the potential of innovation that the application of Blockchain will bring into our specific context, we will answer the key questions posed by Gatteschi et al. [38].

"CriptoSenhas, To or Not To, Blockchain?":

- Is it necessary to have a shared database? Yes, every student needs to check the status of his/her meal vouchers and the institution also needs to check the global status.
- Is it necessary to have multiple parties writing data? Yes, both the institution and each student need to sell/transfer meal vouchers.
- Are potential writers untrusted (should writers be prevented from modifying others' previous entries)? Yes, using the application token as the meal voucher would also prevent fraud and some other types of attacks into the system. A brief side note is added that although trust and security issues are tackled from the core of the Blockchain infrastructure, there is still a considerable amount of ongoing research on this topic [40–42]. Inclusively, some relevant work has already been published that presents a trust management, reputation model, with important gains in speed and accuracy in peer-to-peer (P2P) environments, using the fuzzy theory [43] and a proposal to defeat side channel threats that expose privacy and could be exploited [44].
- Is disintermediation needed (is it necessary to remove trusted intermediaries verifying or authenticating transactions)? Yes, bureaucracy is a major concern for educational institutions, and human-intensive administrative processes can be replaced by Smart Contracts, for instance.
- Is it necessary to see how transactions are linked to each other (should different actors independently write transactions concerning a single user)? Yes, for users to be held accountable for not using the meal voucher, it should be possible to see the flow of the meal voucher.

Answering the above questions allowed us to assess that introduction of Blockchain to our case study application will likely introduce some overheads but will also bring true benefits.

3.2. Ethereum

Ethereum is a decentralized platform supported by a blockchain and capable of running Smart Contracts [45]. This platform provides its own cryptocurrency called Ether, which is used as a fuel to perform the functionality of these contracts. If the applications are computationally complex, the execution cost is higher, thus preventing network congestion [46].

Smart Contracts

Smart Contracts are, in a simple way, applications that run exactly the way they are programmed within Blockchain without interference from intermediary entities, or other problems associated with centralization. This allows the creation of decentralized applications of all kinds, from the creation of markets, registration of ideas, debts or promises, among many other ideas yet to be "discovered", such as this case study. After the publication of a Smart Contract, it can never be changed.

With Smart Contracts, it is possible to create tokens that can represent any type of asset, such as a meal voucher in our case study. In 2015, the ERC-20 standard was published and defines the implementation and basic functionalities that the created tokens should follow [8]. This standard allows all new tokens to be compatible with existing applications, from wallets to decentralized exchanges.

> *"All Smart Contracts implementing this standard, by default can be listed to crypto exchanges without any extra technical work."* [47]

To program the Smart Contracts, the creators of Ethereum have created a programming language named Solidity, inspired by C++, Javascript and Python [48]. When deploying a Smart Contract, the bytecode resulting from its compilation is stored in Blockchain, as shown in Figure 13.

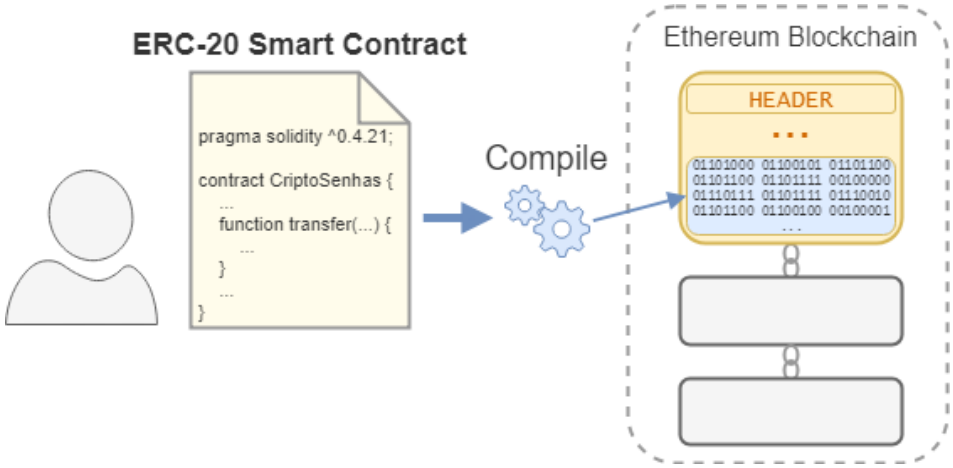

Figure 13. Smart Contract compilation and bytecode registration at Blockchain.

When a transaction invokes a feature of a Smart Contract, as viewed in Figure 14, the corresponding bytecode in Blockchain is executed by the Ethereum Virtual Machine (EVM) at the price of a gas-denominated fee that is paid in Ether [49].

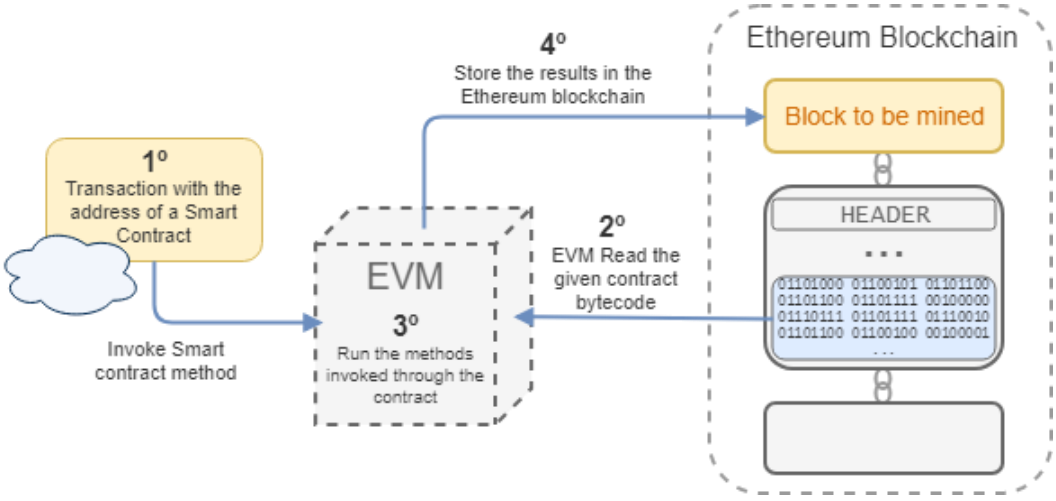

Figure 14. Invocation of a Smart Contract method.

3.3. CriptoSenhas Mobile Application

The CriptoSenhas application is a decentralized application, since its information is stored in the Ethereum Blockchain, developed for mobile devices, and has the main purpose of demonstrating

the possibility of transaction of meal vouchers (token named "CriptoSenhas") between different users, or between a user and the institution. Initially, the institution is the owner of all tokens (Smart Contract creator).

3.3.1. The Implementation of the CriptoSenhas Token

For the creation of the CriptoSenhas token, a Smart Contract that follows the ERC-20 standard was implemented. It was programmed so that the resulting token had the following characteristics, as seen in Figure 15:

- Name: CriptoSenhas;
- Symbol: XCS;
- Decimals (divisibility): 0 (not divisible)—it does not make sense for a meal voucher to be divisible;
- Total tokens: 1,000,000,000,000 (one billion). As this value cannot be changed, it should be used a value that is large enough to reduce the possibility of all tokens to be spent and never used.

```
12  pragma solidity ^0.4.21;
13
14  import "./EIP20Interface.sol";
15
16
17  contract CriptoSenha is EIP20Interface {
18
19      uint256 constant private MAX_UINT256 = 2**256 - 1;
20      mapping (address => uint256) public balances;
21      mapping (address => mapping (address => uint256)) public allowed;
22
23      uint256 totalSupply     = 1000000000000;
24      string public name      = "CriptoSenha";
25      string public symbol    = "XCS";
26      uint8 public decimals   = 0;
27
28      constructor(···
33      ) public {···
39      }
40
41      function transfer(address _to, uint256 _value) public returns (bool success) {
42          require(balances[msg.sender] >= _value);
43          balances[msg.sender] -= _value;
44          balances[_to] += _value;
45          emit Transfer(msg.sender, _to, _value);
46          return true;
47      }
48
49      function transferFrom(address _from, address _to, uint256 _value) public returns (bool success) {···
59      }
60
61      function balanceOf(address _owner) public view returns (uint256 balance) {···
63      }
64
65      function approve(address _spender, uint256 _value) public returns (bool success) {···
69      }
70
71      function allowance(address _owner, address _spender) public view returns (uint256 remaining) {···
73      }
74  }
75
```

Figure 15. The Smart Contract developed for CriptoSenhas.

3.3.2. Architecture

The CriptoSenhas application used the architecture defined for YABW, reusing major components and custom developing specific modules to enhance the required features. The *WalletProvider* component was rewritten to be compatible with the Ethereum network, and consequently different libraries were used, namely:

- *web3*—a library that provides methods that communicate directly with the Remote Procedure Calls (RPCs) of an Ethereum network node. This allows operations on Blockchain and Smart Contracts;

- *eth-lightwallet*—a library for experimental use that provides operations to create and manage Ethereum wallets. It also uses the *web3* library internally to perform some of its functionalities.

3.3.3. Implementation Details

The CriptoSenhas application provides the following features:

- Show user account data (wallet)—account address, number of tokens (CriptoSenhas), details of the transactions that were sent;
- User Preferences—PIN code protection and the Ethereum network endpoint;
- Option to backup and remove an account (wallet);
- Show the account address through the "Receive" tab;
- Send CriptoSenhas through the "Send" tab.

The following restrictions apply: only one account can be used per device; and only sent transactions can be seen. The Ethereum network endpoints do not provide the transaction history, and to avoid large processing in the client device, we have chosen to list only the transactions that were sent from the app.

In Figure 16, we show the navigability diagram of the main structural change made to the previous structure described for the YABW application (Section 2.2):

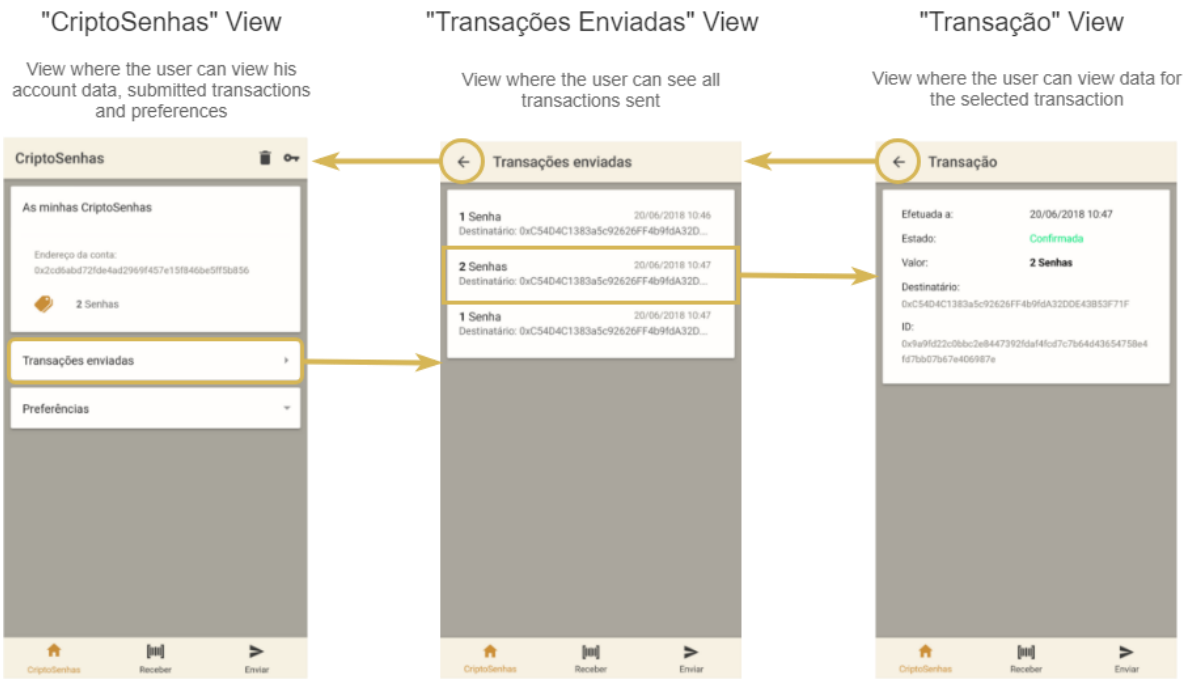

Figure 16. Navigation Diagram of the initial view "CriptoSenhas" and of the transactions sent.

In Figure 16, we can see that there is no option to create more accounts, as in the context of this case study, the application was intended to be used by students from an educational institution, where each student would only have one account (wallet).

Invoking Smart Contract Methods

The Smart Contracts bytecode is stored in the Ethereum Blockchain in a binary format. In order to be able to use the Smart Contract CriptoSenhas, it is necessary to instantiate the contract with the Application Binary Interface (ABI) that describes the contract interface (as exemplified in Figure 17) and the contract address at the blockchain.

```
{
    constant: false,
    inputs: [
        { name: '_to', type: 'address' },
        { name: '_value', type: 'uint256' }
    ],
    name: 'transfer',
    outputs: [{ name: 'success', type: 'bool' }],
    payable: false,
    stateMutability: 'nonpayable',
    type: 'function'
},
```

Figure 17. Application Binary Interface (ABI) example of the "transfer" method.

Having access to the contract instance, it is possible to invoke the methods of this contract. However, there are two types of distinct operations: (1) transactions that require registration in Blockchain, such as a transaction of a token between two entities; and (2) operations that are not registered in Blockchain, such as checking the token amount of an address.

The latter operations, which do not need to be registered in Blockchain, are simple and straightforward: the method is invoked, and a response is returned by an Ethereum network node. However, the operations that are registered in Blockchain are more complex, and, beyond invoking the method, it is necessary to:

1. Estimate the gas limit to perform the method and the current average gas price in Ether.
2. Get the number of transactions already done;
3. Create the transaction object with the following parameters:
    a. from: source address;
    b. nonce: the number of transactions already done;
    c. gasPrice: price per gas in Ether;
    d. to: Smart Contract address;
    e. value: amount of Ether to send to the contract address, which should always be 0 on token transactions;
    f. data: encoding of the contract method together with its parameters;
    g. chainId: identifier code of the Ethereum network [50].
4. Sign the created transaction object;
5. Send the signed transaction to the Ethereum network.

The implementation of the transaction process (shown in Figure 18) was not straightforward, since we found that the *eth-lightwallet* library was no longer compatible with the latest version of *web3*, so it was necessary to reimplement the *signTransaction* method, which signs a transaction object.

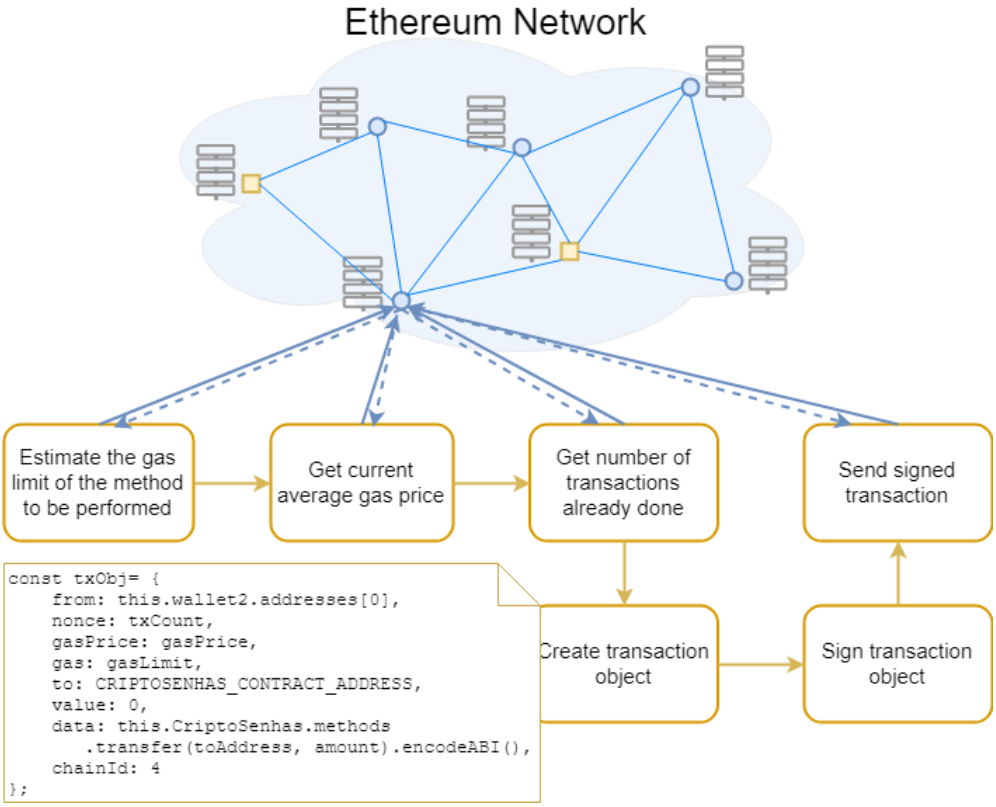

Figure 18. Execution flow to perform a transaction of meal vouchers.

List of Transactions from an Account

Unlike the YABW application, the CriptoSenhas application only has a list of transactions sent through the application itself. This is due to the fact that the Ethereum RPCs do not provide a specific method to obtain all the transactions. In addition, it requires that they are obtained from block to block. Although this is a possible solution, it would spend significant computing resources on the users' devices, which is not a good practice in the context of mobile devices.

Finally, the result of some of the views of the CriptoSenhas application can be seen in Figure 19. The functional requirements of this application led to changes at the interface level, the implementation of an account (a single wallet) and transactions (exchange of meal vouchers) between users or between users and the institution.

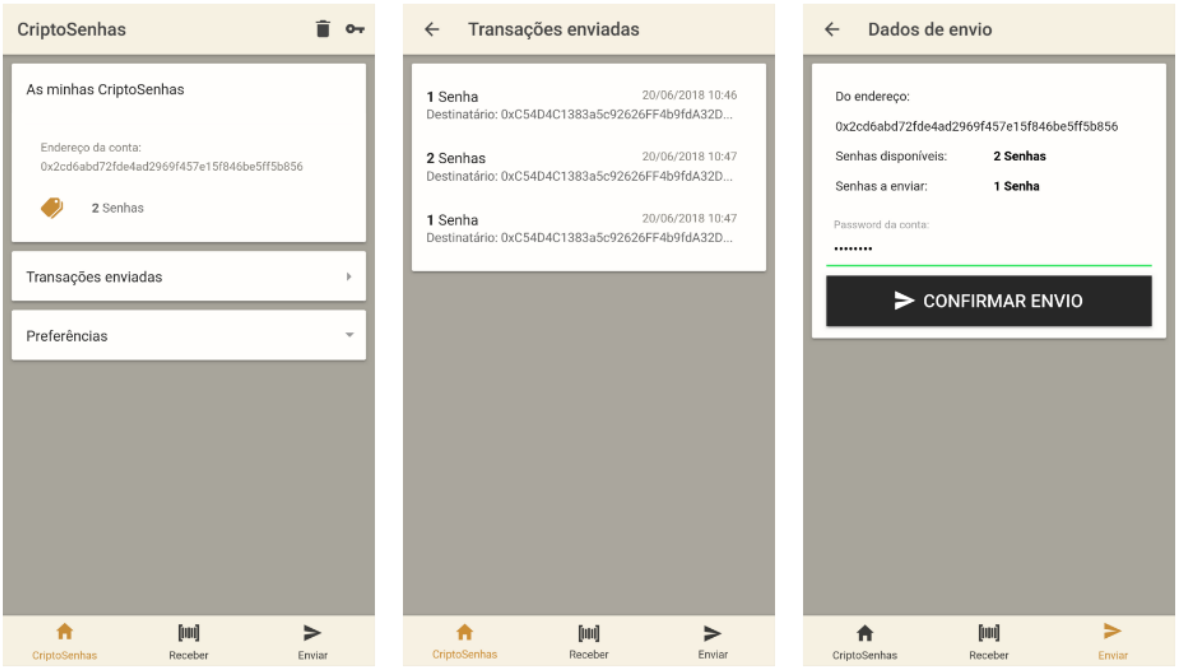

Figure 19. Results of some of the views of the CriptoSenhas application.

4. Conclusions and Future Work

The world of cryptocurrencies is expanding, and the deeper the research, the more technology is discovered to study, which becomes a cycle of research after research on a theme that provides feelings of both interest and achievement.

The basic objectives of the work described here were to explore and apply the concepts related to cryptocurrencies and Blockchain. We believe to have accomplished the goals that allowed us to achieve the following: (1) exploration and grasp of the concepts inherent to Blockchain; (2) specification, design, implementation and publication (in the Apple Store and Google Play Store) of the YABW application—Yet Another Bitcoin Wallet; (3) deepening of theoretical concepts with the purpose of applying them to a concrete non-financial case study; (4) specification, design and implementation of the case study application—CriptoSenhas.

The development of YABW required some research and study that allowed us to identify the major features available in similar application wallets and then used the results to prioritize the development of the features for YABW. Besides the actual deployment of the wallet into the two stores, we also accomplished a highly scalable architecture that could be easily (re)used for other contexts.

We validated the reusability of the architecture by developing the "CriptoSenhas" application, using major components, and custom developing specific modules to enhance its specific features. "CriptoSenhas" allows meal vouchers provided by an educational institution to be exchanged not only between the institution and users, but also among users. We hope that "CriptoSenhas" will be, in the future, one of the modules of a bigger project responsible for ensuring the identity, the academic curriculum and several other services promoted inside a campus. For now, we believe it to be the first step in the beginning of a new era for highly bureaucratic environments that promotes transparency and shared responsibilities.

There is always work that can be added or improved in the future, and in the case of this work, the list can become quite extensive and ambitious, but we will mention these that seem to be the main opportunities for improvement and future developments. Regarding the YABW Application, the following would be of great interest: allowing the purchase of traditional currency cryptocurrencies using existing platforms; adding individual wallet encryption through passwords; allowing sending

cryptocurrencies to multiple addresses in a single transaction; reading and sharing addresses with Near-Field Communication (NFC) technology; sharing address via email, Short Message Service (SMS), social networks or other means of communication of mobile devices; and adding a limit of PIN code insertion attempts, with a possible retrieval of the code via email.

In the CriptoSenhas application, it would be interesting to investigate other Blockchain-based technologies, including some of the Linux Foundation's Hyperledger frameworks [51], which may provide more flexibility and control over the technology; and to create a solution for fees paid in Ether for sending meal vouchers, which make this case study less practical at the user experience level.

In addition to the cryptocurrencies and the case study presented in this work, we believe that there are many more research areas and business opportunities that have emerged with the use of Blockchain technology, many of which are yet to be explored. This technology has the potential to change the world as we know it and it is therefore worth investing in studying and researching it to explore this potential.

Author Contributions: Conceptualization, C.I.R. and M.M.; investigation, M.F. and S.R.; software, M.F. and S.R.; supervision, C.I.R. and M.M.; writing of the original draft, M.F. and S.R.; writing of review and editing, C.I.R. and M.M.

Funding: This research was funded by the FCT-Foundation for Science and Technology, I.P. grant number UID/CEC/04524/2016 And The APC was funded by the FCT-Foundation for Science and Technology, I.P.

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

References

1. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 1 April 2018).
2. Coinranking.com. Coinranking. 2018. Available online: https://coinranking.com/ (accessed on 1 June 2018).
3. Tapscott, A.; Tapscott, D. How Blockchain Is Changing Finance, Harvard Business Review. Available online: https://hbr.org/2017/03/how-blockchain-is-changing-finance (accessed on 11 August 2018).
4. Colored Coins. Available online: http://coloredcoins.org/ (accessed on 1 June 2018).
5. Foundation, E. Ethereum–Blockchain App Platform 2018. Available online: https://www.ethereum.org (accessed on 1 June 2018).
6. Kurve, A. 4 Different Types of Bitcoin Wallets You Can Use. Available online: https://beebom.com/types-of-bitcoin-wallets/ (accessed on 1 April 2018).
7. Vaidya, K. Bitcoin's Implementation of Blockchain. Available online: https://medium.com/all-things-ledger/bitcoins-implementation-of-blockchain-2be713f662c2 (accessed on 1 April 2018).
8. Vogelsteller, F.; Buterin, V. ERC-20 Token Standard. Available online: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md (accessed on 1 June 2018).
9. Bitcoin. Master Key Generation 2018. Available online: https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki#master-key-generation (accessed on 1 June 2018).
10. Ionic. Ionic Framework. Available online: https://ionicframework.com/ (accessed on 1 May 2018).
11. Copay. Available online: https://copay.io (accessed on 11 August 2018).
12. Bitcoin Wallet. Available online: https://wallet.bitcoin.com (accessed on 11 August 2018).
13. Breadwallet. Available online: https://brd.com (accessed on 11 August 2018).
14. Mycelium Wallet. Available online: https://wallet.mycelium.com (accessed on 11 August 2018).
15. Bither. Available online: https://bither.net (accessed on 11 August 2018).
16. Green Bits. Available online: https://bitcoin.org/en/wallets/mobile/android/greenbits/ (accessed on 11 August 2018).
17. Electrum. Available online: https://electrum.org (accessed on 11 August 2018).
18. Green Address. Available online: https://greenaddress.it (accessed on 11 August 2018).
19. Coin.Space. Available online: https://coin.space (accessed on 11 August 2018).
20. Simple Bitcoin. Available online: https://btcontract.com (accessed on 11 August 2018).
21. ArcBit. Available online: https://bitcoin.org/en/wallets/desktop/windows/arcbit/ (accessed on 11 August 2018).

22. Bitcore Wallet Service. Available online: https://github.com/bitpay/bitcore-wallet-service (accessed on 11 August 2018).
23. Brown, S. The C4 Model for Software Architecture. Available online: https://c4model.com (accessed on 11 April 2018).
24. Promise. Available online: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise (accessed on 1 June 2018).
25. "Observable". Available online: https://rxjsdev.firebaseapp.com/api/index/Observable (accessed on 1 June 2018).
26. BIP44. Bitcoin/Bips/Bip44. Available online: https://github.com/bitcoin/bips/blob/master/bip-0044.mediawiki#address-gaplimit (accessed on 1 June 2018).
27. Coinbase.com. What Is the TestNet? Available online: https://support.coinbase.com/customer/en/portal/articles/1973566-what-is-thetestnet (accessed on 1 June 2018).
28. Smart, J.F. *BDD in Action*; Manning Publications: New York, NY, USA, 2014.
29. Jasmine. Available online: https://jasmine.github.io/ (accessed on 30 July 2018).
30. Karma. Available online: https://karma-runner.github.io/2.0/index.html (accessed on 30 July 2018).
31. YABW iOS–Apple Store. Available online: https://itunes.apple.com/pt/app/yabw/id1384445290?l=en&mt=8 (accessed on 30 July 2018).
32. YABW Android–Google Play. Available online: https://play.google.com/store/apps/details?id=pt.ipleiria.estg.dei.yabw (accessed on 30 July 2018).
33. Sun, J.; Yan, J.; Zhang, K.Z. Blockchain-based sharing services: What blockchain technology can contribute to smart cities. *Financ. Innov.* **2016**, *2*, 26. [CrossRef]
34. Avital, M. Peer Review: Toward a Blockchain-enabled Market-based Ecosystem. *Commun. Assoc. Inf. Syst.* **2018**, *42*, 646–653. [CrossRef]
35. Shelper, P.; Lowe, A.; Kanhere, S.S. Experiences from the Field: Unify Rewards-A Cryptocurrency Loyalty Program. In Proceedings of the Symposium on Foundations and Applications of Blockchain 2018, Los Angeles, CA, USA, 9 March 2018.
36. Montiel, M.D.; Pérez-Villegas, A.; Sagalés, A. Smart Token for the Campus Using Blockchain. Bachelor's Thesis, National Taiwan University of Science and Technology, Taipei City, Taiwam, June 2017.
37. Zhao, J.L.; Fan, S.; Yan, J. Overview of business innovations and research opportunities in blockchain and introduction to the special issue. *Financ. Innov.* **2016**, *2*, 28. [CrossRef]
38. Gatteschi, V.; Lamberti, F.; Demartini, C.; Pranteda, C.; Santamaría, V. To Blockchain or Not to Blockchain: That Is the Question. *IT Prof.* **2018**, *20*, 62–74. [CrossRef]
39. Wang, H.; Chen, K.; Xu, D. A maturity model for blockchain adoption. *Financ. Innov.* **2016**, *2*, 12. [CrossRef]
40. Xu, J.J. Are blockchains immune to all malicious attacks? *Financ. Innov.* **2016**, *2*, 25. [CrossRef]
41. Cai, Y.; Zhu, D. Fraud detections for online businesses: a perspective from blockchain technology. *Financ. Innov.* **2016**, *2*, 20. [CrossRef]
42. Shi, N. A new proof-of-work mechanism for bitcoin. *Financ. Inn.* **2016**, *2*, 31. [CrossRef]
43. Javanmardi, S.; Shojafar, M.; Shariatmadari, S.; Ahrabi, S.S. Fr trust: a fuzzy reputation–based model for trust management in semantic p2p grids. *Int. J. Grid Util. Comput.* **2014**, *6*, 57–66. [CrossRef]
44. Pooranian, Z.; Chen, K.C.; Yu, C.M.; Conti, M. RARE: Defeating side channels based on data-deduplication in cloud storage. In Proceedings of the IEEE International Conference on Computer Communications, Honolulu, HI, USA, 15–19 April 2018.
45. Ethereum Wiki. A Next-Generation Smart Contract and Decentralized Application Platform. Available online: https://github.com/ethereum/wiki/wiki/White-Paper (accessed on 30 July 2018).
46. Foundation, E. ETHER. Available online: https://www.ethereum.org/ether (accessed on 1 June 2018).
47. Lukas, K. ERC Standards to Move Ethereum Forward? ERC-20, ERC-223, ERC-721. Available online: https://medium.com/wepower/erc-standards-to-moveethereum-forward-erc-20-erc-223-erc-721-e1712456449d#941f (accessed on 1 June 2018).
48. Foundation, E. Solidity. Available online: https://solidity.readthedocs.io/en/v0.4.24/ (accessed on 1 June 2018).
49. Ethereum Development Tutorial. Available online: https://github.com/ethereum/wiki/wiki/Ethereum-Development-Tutorial (accessed on 30 July 2018).

50. Ethereum. Available online: https://ethereum.stackexchange.com/questions/17051/how-to-select-a-networkid-or-is-there-a-list-of-network-ids (accessed on 1 June 2018).
51. The Linux Foundation–Business Blockchain Frameworks Hosted with Hyperledger. Available online: https://www.hyperledger.org/ (accessed on 30 July 2018).