Mobile Money - Blockchain Transactions

Internship Report

Filipe Daniel Ramos Morgado
(2019137625)

Advisers:

João Durães | ISEC

Jorge Sousa | WIT Software

João Sousa | WIT Software

Bachelor's Degree in Informatic Engineering

Specialization in Applications Development

Polytechnic Institute of Coimbra

Coimbra Institute of Engineering

July of 2022

AGRADECIMENTOS

Gostaria de começar por agradecer à WIT Software pela confiança, oportunidade e ajuda neste processo de evolução académico, profissional e pessoal. Ao João Sousa, Mylena Dias e Jorge Sousa por toda a ajuda, orientação, confiança e transmissão de conhecimento.

Agradeço ao meu orientador, professor João Durães, pela ajuda ao longo do estágio e acompanhamento no desenvolvimento do relatório.

Aos meus colegas de estágio na WIT, por toda a ajuda e companheirismo no decorrer deste estágio.

Aos meus amigos, pela enorme amizade, memórias partilhadas e apoio nos bons e maus momentos.

Aos meus pais e irmã, pelo apoio demonstrado em todos os meus passos nesta caminhada.

Por fim, mas não menos importante, agradeço à minha namorada pela força e apoio que sempre recebi, e por acreditar que seria capaz de atingir todos os meus objetivos durante todo o meu percurso académico.

RESUMO

A Bitcoin foi a primeira *cryptocurrency* e é a *Blockchain* mais antiga, tendo como principal foco a segurança e descentralização. A *Bitcoin* cresceu a um ponto em que o número de transações que é capaz de fazer por segundo é bastante baixo e a criação de novos blocos na cadeia é cada vez mais lenta, demorando em média 10 minutos, evidenciando cada vez mais o seu problema de escalabilidade. As taxas são também um problema, rondando entre $1 a $5 em 2022, tendo um pico de $63 em Abril de 2021, tornando-a ainda menos atrativa para micropagamentos e desenvolvimento de aplicações descentralizadas que se baseiem na *Blockchain* da *Bitcoin*.

O número de cadeias nível 2 tem vindo a crescer como uma tentativa de solucionar problemas existentes nas *Blockchains* de nível 1, inclusive na *Blockchain* da *Bitcoin*. A *Lightning Network (LN)* é uma dessas recentes cadeias de nível 2 que visa permitir transações quase instantâneas com taxas de envio de quase, ou até mesmo $0, tentando desta forma solucionar os dois problemas já mencionados na *Blockchain* da *Bitcoin*.

Empresas como a WIT Software (WIT) têm como ambição a procura de ideias inovadoras de modo a fornecer novas tecnologias aos seus clientes e parceiros. Com esta ambição, WIT tem cada vez mais apostado em *Blockchain*, apercebendo-se também que cadeias de nível 2 se enquadravam dentro da sua visão, missão e área de negócio.

Este estágio focou-se, primeiramente, na investigação das capacidades da *Lightning Network*, com o propósito de dar a conhecer à WIT as suas vantagens e limitações. De seguida, foi feito um levantamento do estado da arte das *wallets* existentes no mercado e respetivas implementações técnicas, possibilitando a criação de uma aplicação protótipo em Android que permite aos seus utilizadores gerir a sua *Lightning Network wallet*, para melhor demonstrar as suas potencialidades.

Palavras-chave: *Lightning Network, Blockchain, Bitcoin, Wallet, Scalability.*

ABSTRACT

Bitcoin was the first cryptocurrency and is the oldest Blockchain, with a main focus on security and decentralization. Bitcoin has grown to a point where the number of transactions it is able to do per second is quite low and the creation of new blocks on the chain is increasingly slower, averaging 10 minutes, highlighting its scalability problem. Fees are also an issue, ranging from $1 to $5 in 2022, peaking at $63 in April 2021, making it even less appealing for the creation of decentralized applications and micropayment solutions that use the Bitcoin Network.

The number of layer 2 chains has been growing in an attempt to solve existing problems in layer 1 Blockchains, including the Bitcoin Blockchain. The Lightning Network is among the most recent layer 2 networks that aims to support nearly instant transactions with almost, or even $0 fee rate, in an effort to address the two issues mentioned above with the Bitcoin Blockchain.

Companies like WIT Software (WIT) have as one of their goals to find innovative ideas in to provide new technologies to its customers and partners. With this ambition, WIT has increased its investments on Blockchain while also realizing that layer 2 networks fit within its vision, mission, and business domain.

This internship focused, firstly, on the investigation of the Lightning Network's capabilities, with the purpose of making WIT aware of its advantages and limitations. Then, a survey was made of the state of the art of the wallets on the market and their respective technical implementations, enabling the development of a prototype Android application that gives users the ability to manage their Lightning Network wallet, better showcasing the network potential.

Keywords: *Lightning Network, Blockchain, Bitcoin, Wallet, Scalability.*

INDEX

Agradecimentos i

Resumo iii

Abstract v

Figure Index xi

Table Index xiii

Acronyms xv

1 Introduction 1

1.1 WIT Software 1

1.2 Instituto Superior de Engenharia de Coimbra 2

1.3 Goals and work plan 2

1.4 Context and methodology 4

1.5 Report structure 4

2 Blockchain and Bitcoin 7

2.1 Blockchain 7

2.1.1 Distributed and decentralized databases 8

2.1.2 Proof of Work 9

2.1.3 Proof of Stake 10

2.1.4 Public-key cryptography 11

2.1.5 Advantages and problems solved by Blockchain 11

2.1.6 Disadvantages and problems yet to be solved by Blockchain 13

2.2 Bitcoin 14

2.2.1 Price and volatility 15

2.2.2 Connection with Blockchain 15

3 Lightning Network 17

3.1 Bidirectional payment channels 17
3.1.1 Use case 18
3.2 Nodes and channel types 19
3.3 Onion routing 21
3.4 Lightning invoice 22
3.5 Fees 22
3.6 Smart contracts 23
3.7 Hashed TimeLock Contract 24
3.8 BOLT - Basics Of Lightning Technology 25
3.9 Network growth 27
3.10 Limitations 30
4 Cryptocurrency wallets 31
4.1 Lightning wallet types 31
4.2 Custodial and non-custodial 32
4.2.1 Custodial wallet 32
4.2.2 Non-custodial wallet 32
4.2.3 Choosing the right wallet 33
5 State of art 35
5.1 Lightning Network architecture and characteristics 35
5.2 Lightning Network implementations and available tools 36
5.2.1 Lightning Network Daemon 36
5.2.2 Core Lighting 37
5.2.3 Lightning Dev Kit 37
5.2.4 LNbits 38
5.3 Comparing Lightning Network implementations 38
5.4 Existing wallets solutions and its features 39
5.4.1 Cash App 40

5.4.2 Muun 41
5.4.3 Chivo 41
5.4.4 Phoenix 43
5.4.5 Spark 44
5.5 Wallet features summary 45
6 Problem analysis and requirements 47
6.1 General goals 47
6.2 Requirements 48
6.2.1 Functional requirements 48
6.2.2 Non-functional requirements 49
6.2.2.1 NFR1 - Usability 50
6.2.2.2 NFR2 - Extensibility 50
6.2.2.3 NFR3 - Security 51
7 Technologies and tools 53
7.1 Technologies 53
7.1.1 Android 53
7.1.2 Kotlin 53
7.1.3 Retrofit 2 54
7.2 Development tools 55
7.2.1 Android Studio 55
7.3 Management tools 55
7.3.1 GitLab 55
7.3.2 Jira 55
7.3.3 Confluence 56
8 Architecture and UI design 57
8.1 Repository pattern 57

8.2 Wallet prototyped UI design 58
9 Implementation 61
9.1 Coingecko use case 61
9.2 Lightning Network use case 62
9.3 PIN number pad 63
9.4 PIN encryption 63
9.5 QR Codes 64
10 Validations 67
10.1 Functional requirements 67
10.2 Non-functional requirements 67
10.2.1 Usability 68
10.2.2 Security 70
11 Conclusions and future work 71
11.1 Personal growth and internship goals 71
11.2 Internship value 71
11.3 Future work 72
References 73
Appendix A: Internship proposal A-1
Appendix B: Requirements specification B-1
Appendix C: Jira tasks schedule C-1
Appendix D: Confluence UI prototype draft D-1

FIGURE INDEX

Figure 1 - Blocks linked together [13] 7
Figure 2 - Message encryption [25] 11
Figure 3 - Payment channel lifecycle [42] 18
Figure 4 - Coffee Shop use case [43] 19
Figure 5 - Payment path [48] 20
Figure 6 - Onion routing example [50] 21
Figure 7 - Routing fees example [48] 23
Figure 8 - How a smart contract works [53] 24
Figure 9 - HTLC use case example [48] 25
Figure 10 - Payment volume and payment count on the Lightning Network [48] 28
Figure 11 - Users with access to Lightning payments [48] 29
Figure 12 - Channel capacity in BTC and USD [59] 29
Figure 13 - Cash App demonstration [73] 40
Figure 14 - Muun demonstration [74] 41
Figure 15 - Chivo demonstration [79] 42
Figure 16 - Phoenix demonstration [81] 43
Figure 17 - Spark demonstration [83] 44
Figure 18 - Wallet prototype repository design pattern 58
Figure 19 - M-Pesa pin screen 59
Figure 20 - M-Pesa home page screen 59
Figure 21 - M-Pesa payment confirm screen 59
Figure 22 - Prototype pin screen 59
Figure 23 - Prototype home page screen 59
Figure 24 - Prototype payment confirm screen 59
Figure 25 - Coingecko use case 61
Figure 26 - Lightning Network use case 62
Figure 27 - Symmetric encryption example [103] 64
Figure 28 - Encryption and Decryption algorithm snippet 64
Figure 29 - Prototype QR code scan screen 65
Figure 30 - Prototype camera permissions 65
Figure 31 - Prototype payment details screen 65
Figure 32 - QR code generation algorithm snippet 65
Figure 33 - Prototype import screen 69
Figure 34 - Prototype scan invoice request 69

Figure 35 - Prototype invoice description.....69
Figure 36 - Prototype payment copied feedback 69
Figure 37 - Prototype create wallet empty name 69

TABLE INDEX

Table 1 - Tasks schedule.....3
Table 2 - Prototyped wallet function requirements prioritization.....48
Table 3 - Prototyped wallet non-function requirements prioritization 50
Table 4 - Prototyped implemented functional requirements.....67
Table 5 - Prototype implemented non-functional requirements 68
Table 6 - Prototype usability tests.....68
Table 7 - Prototype followed security requirements.....70

ACRONYMS

AES	Advanced Encryption Standard
API	Application Programming Interface
BOLT	Basics Of Lightning Network
BTC	Bitcoin
DNS	Domain Name System
EUR	Euro
HTTP	Hyper Text Transfer Protocol
IDE	Integrated Development Environment
ISEC	Instituto Superior de Engenharia de Coimbra
JSON	JavaScript Object Notation
KYC	Know Your Customer
LDK	Lightning Dev Kit
LN	Lightning Network
LND	Lightning Network Daemon
MVC	Model View Controller
MVP	Model View Presenter
MVVM	Model View View-Model
PIN	Postal Index Number
REST	Representational State Transfer
SSOT	Single Source Of Truth
TPS	Transactions Per Second
USD	United States Dollars
WIT	WIT Software
XML	Extensible Markup Language

1 INTRODUCTION

The purpose of this document is to report the work that was done during the internship related to the *Projeto ou Estágio* curricular unit. This curricular unit is integrated in the Informatics Engineering course [1] at Instituto Superior de Engenharia de Coimbra [2], in the 2nd semester of the 3rd year and aims to provide students with first-hand experience in the industry: students are placed in a company which assigns them a real-world project. This internship occurred in the academic year of 2021/2022 at WIT Software [3] and this document is mainly focused on the details of the project that was assigned by WIT to the intern. This internship was supervised by Professor João Durães (ISEC), by Head of Unit Jorge Sousa (WIT Software), by Business Analyst Mylena Dias (WIT Software), and by João Sousa, Lead Engineer (WIT Software).

The project assigned by WIT is *Mobile Money - Blockchain Transactions* and focus on the Lightning Network which aims to improve the scalability and transaction fee problem that exist in the Bitcoin Blockchain. The project has two main goals:

- to explore the features of the Lightning Network, identifying its advantages and limitations, and generate a report detailing the findings.
- to develop an android wallet application to demonstrate the main features of Lightning Network.

With a heavy focus on the first main goal, the company plans to expand its knowledge of this new technology to analyze the feasibility of possible future investments. Therefore, an in-depth analysis of the first goal, the exploration of the Lightning Network, will be given in this report. An application protype was developed to address the 2nd goal. An application that enables users to manage their Lightning Network wallet will allow for the evaluation of the solution with regards to user experience and its potential to be integrated in other projects within the financial world.

1.1 WIT Software

WIT is a software company with over 21 years of solid expertise that creates products and develops projects for the Telecom industry [4]. Starting its activity in 2001 in Coimbra, WIT has strong expertise [5] in voice-video-messaging over IP, mobile money, mobile commerce, IPTV, secure software, and core telecom services. It is deployed in more than 40 countries with clients such as Vodafone, Verizon, AT&T, Deutsche Telekom and SoftBank. The company has offices in Coimbra, Porto, Lisbon, Aveiro, Leiria and United Kingdom. WIT is founded on strong human values, and it differentiates by giving the customers the full journey of the software. It is also innovation driven and maintains strong processes of software quality following the best practices of agile development

methodologies, including SAFE, Scrum and Kanban. They have now three available products for the public: WIT RCS+ [6], WIT Buzz Platform [7] and WIT Conversation Platform [8]. WIT Software has a strong engagement in humanitarian actions such as supporting the homeless by volunteering in street actions, blood donations, supporting 10 children in the province of Nampula, and much more.

1.2 Instituto Superior de Engenharia de Coimbra

The Instituto Superior de Engenharia de Coimbra, is a higher education polytechnic institution of engineering, based in Coimbra, with the mission of creation, transmission and diffusion of culture, science and technology [9].

Currently it has 3357 students distributed over 39 courses, 12 being bachelor's degrees, 9 master's degrees and 18 professional higher technical courses [10]. In terms of employability, ISEC has 70% of its students employed before the conclusion of the course and 98% of them are employed in the following 6 months after ending the course.

1.3 Goals and work plan

To accomplish the projects' goals, a solid understanding of the concepts and technologies involved is necessary. Thus, an introductory chapter about Blockchain and Bitcoin is given, to better contextualize the Lightning Network analyzed in the following chapter. An overview is then given of the Lightning Network, to have a solid understanding of how it allows to improve the scalability and transaction fees problem in the Bitcoin Blockchain. Both chapters, chapter 2 and 3, aim to fulfill the first goal of the project.

To demonstrate the usability and feasibility of these same technologies, especially the Lightning Network, an Android prototype was developed that allows a user to execute activities necessary in their daily life. These include activities such as: sending money to other users and receiving money from other users through the Lightning Network, checking the current balance, and viewing the transaction payment history. Thus, the prototype must be able to:

- Create/register a wallet
- User authentication
- Check the current balance
- Start a transaction
- Receive a payment
- Consult transaction history

To accomplish these prototype features as well as to create the detailed report, the project was organized with the following tasks:

- **T1 - Acquisition of Know-how** - Learning about Blockchain, Bitcoin, and the Lightning Network
- **T2 – State of Art research** - Research currently available solutions for decentralized Mobile Money management using Blockchain technology.
- **T3 - Requirements Analysis and Functional Specification** - Perform a functional requirements analysis and specification.
- **T4 - UI Specification** – Prototype specification of the functionality's visual interface with the help of WIT's UI/UX (user interface/user experience) team.
- **T5 - Phase 1 Implementation** - Initial implementation of the prototype, in which the user can register their wallet and view their current balance.
- **T6 - Phase 2 Implementation** - The final features will be implemented. enabling the user to conduct transactions, receive transactions, and check their transaction history.
- **T7 - Internship report** - Preparation of an internship report detailing the findings and accomplishments.

The following table displays the duration of each task for the internship, organized chronologically as follows:

	Months									
Tasks	N		N+1		N+2		N+3		N+4	
T1										
T2										
T3										
T4										
T5										
T6										
T7										

Table 1 - Tasks schedule

In Table 1 – Tasks schedule, each cell represents two weeks. Since T3 and T4 are related to each other, they were done simultaneously. The internship began on March 14, 2022.

Because Phase 1 and Phase 2 are more complex tasks, they were broken into smaller sections, which included the following:

- Phase 1 Implementation
    - Create a New Wallet

    - Check Current Balance
    - Wallet Authentication
- Phase 2 Implementation
    - Send Lightning Payment
    - Receive Lightning Payment
    - Transaction History

This being the initial work plan, it has suffered minor changes along the way in the development of the prototype, having additional features. A more detailed report is given on which features were implemented in chapter 10.

1.4 Context and methodology

The internship took place in a business setting, at WIT Software, following their standard methodology of work, SCRUM [11]. This methodology does not have a sequential approach, the products are improved and developed in an iterative and gradual manner under the SCRUM framework, rather than attempting to deliver a complete solution all at once. In this methodology, the iterations made through the development of the project are called sprints, each sprint has a duration of 1 to 4 weeks. Daily or weekly meetings with the team are held to support project progress, when the previous day's or week's work, future work, and potential impediments are discussed. At the end of each sprint, the work completed is presented and evaluated, and the project is adjusted as needed. A SCRUM team consists of three roles: the Scrum Master, the product owner, and the development team.

In the context of this internship, the development period was divided in sprints of one week with weekly meeting (20 minutes to 30 minutes), and daily meetings, when necessary (5 minutes to 10 minutes), where all of the work completed during the week was presented, along with a demonstration when appropriate. There is also a monthly status meeting with company management so they can keep up with the work being done as well as future planning in the internships. This weekly development planning can be seen in appendix C. When it comes to the scrum team roles, João Sousa was the Scrum Master, Mylena Dias the Product Owner and the intern the development team.

Despite having the option of working in the office or remotely, most of the work was completed remotely.

1.5 Report structure

This report was arranged into chapters addressing the main topics of the project in order to provide a comprehensive view about the internship. The remainder of this report is organized as follows:

- **Chapter 2 - Blockchain and Bitcoin:** This chapter provides the key concepts of this technologies.
- **Chapter 3 - Lightning Network:** This chapter will provide a more extensive introduction to this technology, as it is the project's main focus.
- **Chapter 4 - Cryptocurrency wallets:** This chapter introduces the existing lightning wallet types with a description of one of their main differences, being custodial or non-custodial.
- **Chapter 5 - State of Art:** This chapter includes a comprehensive analysis of competing applications as well as current Lightning Network implementation and tools options, existing architectures, and characteristics in Wallets.
- **Chapter 6 - Problem analysis and requirements:** This chapter offers firstly an analysis of the problem followed by the functional and non-functional requirements for the prototype.
- **Chapter 7 - Technologies and tools:** This chapter presents all the technologies and tools used along the project development.
- **Chapter 8 - Architecture and UI design:** This chapter describes the architecture used in the prototyped application and decisions made regarding its UI design.
- **Chapter 9 - Implementation:** This chapter presents technical implementation details about the construction of a Lightning Network wallet.
- **Chapter 10 - Validations:** This chapter presents all the validations of the functional and non-functional requirement.
- **Chapter 11 - Conclusion and future work:** Gives an overview of the work and perspective on possible follow up directions.

2 BLOCKCHAIN AND BITCOIN

The material presented in this chapter is important for the understanding of some of the main concepts and architectures used in this project. It is also an essential starting point to understand the Lightning Network presented later, in chapter 3.

2.1 Blockchain

A Blockchain is a distributed ledger technology that targets decentralization as a security measure [12]. It's a shared and distributed database of a growing list of records, known as blocks, that are encrypted and securely linked together. Blocks[1] are added to the Blockchain in a linear and chronological fashion. Each block contains a cryptographic hash[2] of the previous block (Figure 1), a timestamp, and transaction data. The timestamp is a short piece of data that is uniquely serialized and preserved in each block. Its primary purpose is to establish the precise time when the block was created. A timestamp is accepted as valid if it is greater than the median timestamp of previous 11 blocks, and less than the network-adjusted time plus 2 hours. Every block in a chain reinforces the ones before it because each one includes information about the one before it. As a result, Blockchains are resistant to data tampering since the data in any given block, once recorded, cannot be changed retroactively without affecting all subsequent blocks.

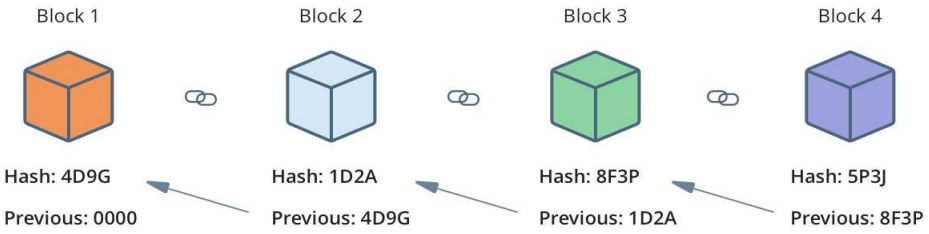


Figure 1 - Blocks linked together [13]

Blockchains are managed by a peer-to-peer network, where nodes[3] collectively adhere to a protocol to communicate with other nodes and validate new blocks. These validators, or so-called miners, are rewarded for performing this link. Because the Blockchain needs a means to prevent a rogue individual or organization from controlling the majority of the

[1] Contain data about transactions.

[2] Can be seen as a fingerprint.

[3] Can be seen as computers.

validation, two main consensus methods are used: Proof of Work and Proof of Stake. Both consensus methods are given a more detailed explanation in the next subsections.

Being decentralized means that there is no single point of failure in the system. If one node leaves the network, other nodes already have an exact copy of all shared information stored. Conversely, if a node joins the network, the initial nodes immediately create copies of their information for the new member. No centralized "official" copy exists, and no user is "trusted" more than any other.

A person (or collection of persons) going by the name Satoshi Nakamoto popularized the Blockchain concept in 2008 to act as the public transaction ledger for the cryptocurrency Bitcoin, based on work done by W. Scott Stornetta, Stuart Haber and Dave Bayer [14]. The identity of Satoshi Nakamoto is still a mystery. The implementation of the Blockchain within Bitcoin made it the first digital currency to solve the double-spending problem, thus not allowing the same digital token[4] to be spent twice, without requiring a central server or a trusted authority.

2.1.1 Distributed and decentralized databases

Currently, there are at least four types of Blockchain networks: public Blockchains, private Blockchains, hybrid Blockchains and sidechains [15].

Public Blockchains: A public Blockchain has no access restrictions. Anyone with Internet access can use it to send transactions and register as a validator. Such networks typically provide financial rewards to those who protect them using a Proof of Stake or Proof of Work algorithm. The Bitcoin Blockchain and the Ethereum Blockchain are two of the most well-known public Blockchains.

Private Blockchains: A private Blockchain is permissioned. Unless the network administrators invite you, you cannot join. Access is limited for participants and validators. To distinguish between open Blockchains and other peer-to-peer decentralized database applications, the terminology Distributed Ledger is normally used for private Blockchains.

Hybrid Blockchains: Both centralized and decentralized features can be found in a hybrid Blockchain. The chain's exact workings depend on which parts of centralization and decentralization are used. Organizations can establish both a private, permission-based system and a public permissionless system, enabling them to administer who can access data stored in the Blockchain, and which data will be made public.

[4] Represents the total or partial ownership of any asset such as Bitcoin.

Sidechains: It is a Blockchain ledger that operates side by side with a main Blockchain. Entries from the primary Blockchain (said entries typically represent digital assets) can be linked to and from the sidechain. This allows the sidechain to operate independently of the primary Blockchain (by using alternate consensus algorithm for example) improving its privacy and security. This ability to exchange assets between the primary chain and the side chain smoothly allows the ecosystem to expand in a decentralized manner [16].

Layer 2 chain: Layer 2 refers to a secondary framework or protocol that is built on top of an existing Blockchain system. The main goal of these protocols is to try and solve the transaction speed and scaling difficulties that are being faced by the major cryptocurrency networks. Although sidechains and Layer 2 chains are frequently confused, their main shared feature is that they both operate as separate networks with a route to Layer 1 [17].

2.1.2 Proof of Work

Proof of work is the first crypto consensus mechanism, used in projects such as Bitcoin and Ethereum 1.0 [18]. In this type of cryptographic proof, the prover demonstrates to the verifiers that a specified amount of a certain computing effort has been used. In essence, a proof of work is a solution to a complex mathematical problem. It is difficult to produce (thus the term), but simple for others to validate.

A computer software used by miners offers each miner an equal chance, based on their computing capacity, of finding the answer to the next block's problem. They compete to determine the hash value (solution) to the parameters for that block's Proof of Work. This hash is a 64-character long answer, and is based upon the inputs to the block, such as the transactions it contains. Bitcoin Miners search for this hash by combining this input with a random number known as a *nonce*, until someone finds the correct answer. This solution is then broadcast to, and verified by, other miners. Once confirmed, it is added to the Blockchain by the other miners, which then use this new block as the input for the hash needed for the next correct block. This chain forms a canonical ledger of all transactions from Bitcoin’s inception.

As a result, Proof of Work solves the Byzantine Generals Problem[5] [19] as it achieves a majority agreement without any central authority, in spite of the presence of unknown/potentially untrustworthy parties and despite the network taking its time to register transactions.

Consequently, whoever has more computational power, has a better chance of figuring out the solution to the next block, approving it and getting the reward forming a big flaw. Miners have formed *mining pools* [20] to boost their chances of mining blocks even more.

[5] Describes the difficulty decentralized systems have in agreeing on a single truth.

Using this method, users can combine their mining power with others and distribute the payout evenly, increasing their potential earnings, motivating this way people to join *mining pools*. This results in an increase in energy consumption for mining, as well as a further centralization of the Blockchain, which is the opposite of the goal. Due to this tendency, a 51% attack [21] can occur. A group of miners who control more than 50% of the mining hash rate on the network can launch this attack. Attackers with a majority on the network can stop other miners from finishing blocks, which will stop the recording of new blocks.

Despite these increasingly well-known drawbacks, fraudulent transactions are nearly impossible to carry out since mining a new block is a complex, time-consuming process that requires a significant computing effort and must also be confirmed by other block miners before being accepted.

Because proof of work is well-tested and used in many cryptocurrency projects, such as Bitcoin, even with its drawbacks, it is still viewed as a viable solution when choosing the consensus method.

2.1.3 Proof of Stake

Proof of Stake protocols are a class of consensus mechanisms for Blockchains that select validators at random while considering their holdings in the related cryptocurrency [22]. The greater the user's participation in the network, the greater the chance that a validator will be chosen to confirm the next block. This approach is used to avoid Proof of Work schemes' computing costs.

Proof of Stake prevents network attacks, such as validation of fraudulent transactions, mainly by forcing malicious attackers to obtain a substantial percentage of the tokens on the Blockchain before launching an attack [23]. Also, to validate a block, the validator has to provide a security deposit. This disincentivizes potential attackers as they would have to be willing not only to forego the security deposit but also their position as a validator and, eventually, even their entire stake in the network. Since there is so much to lose, it also reduces the probability of malicious validators to try and double-spend coins.

On the other hand, it is possible to buy most of the coins in the network, become the validator with more probability of being chosen, and validate wrong transactions as part of an attack. However, the market economy has a natural safety valve for this, because the price of the coin will rise significantly when someone tries to buy such a massive amount of coins, the attackers' job will become far more difficult. A good Proof of Stake system requires much more complex code than a proof of work system. More bugs and vulnerabilities might result from that.

Because Proof of Stake does not require expensive hardware to validate transactions, it encourages more people to create new nodes since they will receive transaction fees for

validating transactions. This makes the network more secure and decentralized providing a more scalable Blockchain with higher transaction throughput, being this the main factors when choosing between proof of stake and proof of work.

2.1.4 Public-key cryptography

Public-key cryptography is a cryptographic system that uses pairs of keys [24]. Each pair consists of a public key and a private key. A public key (a long, random-looking string of numbers) can be viewed as the user public address on the Blockchain. A private key is like a password that gives its owner access to their digital assets or the means to otherwise interact with the capabilities of the Blockchain. Value tokens, such as Bitcoin, sent across the network are recorded as belonging to the user's public address.

The generation of such key pairs depends on cryptographic algorithms which are based on mathematical problems. Effective security requires keeping the private key private, while the public key can be openly distributed without compromising security.

In this system, any person can encrypt a message using the intended receiver's public key, but that encrypted message can only be decrypted with the receiver's private key. The image below, Figure 2, explains how the process works.

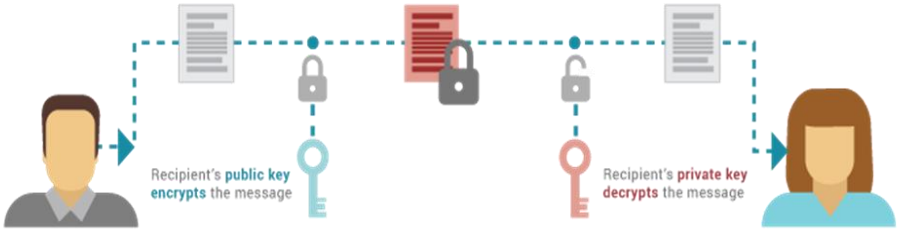


Figure 2 - Message encryption [25]

2.1.5 Advantages and problems solved by Blockchain

Blockchain can solve real-world problems and offer a better business model and economic structure for organizations and individuals. Overall, Blockchain technology has the following characteristics and advantages:

- Decentralization
- No third-party involvement
- Transparency
- Immutability
- Traceability
- High level of security and integrity

In contrast with Blockchain, traditional database, such as Hierarchical or Network databases, is that the entity that created the database owns all rights to it. They have control over who gets access to it and who does not, as well as what is stored, erased, and

preserved. There are at least two potential flaws with this design: centralization and intermediaries.

Centralization: If a system relies on a central database, there is a single point of failure. If the central authority is compromised, such as through cyber-attacks, the database becomes vulnerable. Through decentralization, Blockchain solves the information integrity problem since even if the information on one node would be compromised, the adulterated information won't be accepted by the other nodes in the network

Intermediaries: Banks, for example, must verify available cash and approve loans when it comes to mortgages. Each of these independent and historically significant central authorities wields a unique power that imposes significant fees on mortgage transactions. With Blockchain, it is possible to avoid middlemen, removing the need to trust in such entities while simultaneously reducing transaction fees, possibility of human error and other types of attacks.

Other improvements are also evident with Blockchain technology [26], some of those are:

- **Cross-border payments**
  - Blockchain helps smooth the entire process by cutting out all middlemen and lengthy procedures, thereby reducing the burden of unnecessary time delays and fees.
- **Supply Chain Management**
  - Everyone, from manufacturers to retailers, vendors, and even contractors, are on the same page thanks to the transparency in data sharing. This reduces potential conflict and delays in business operations.
- **Identity Theft**
  - Municipalities in India have already turned towards Blockchain technology to simplify the issuance of birth certificates [27]. The system works via an application and speeds up verifying identity. Once the user's identity is authenticated, a unique QR code is generated that provides access to the verified data. The application then allows the user to prove their identity without the need to physically hand in identity documents or stand in front of an agent to apply for a birth certificate.
- **Charity**
  - The UN has been testing the use of Blockchain technology for its refugee aid system, with the aim of facilitating the effective delivery of humanitarian aid while reducing fraud and inefficiency [28]. Skimming and bribes are a common problem, but Blockchain technology is helping the UN track how every penny is spent, from donation to aid distribution, maximizing how much aid reaches the intended recipients.

- **Voting**
    - Blockchain authenticated voting would record all the data from the election process on a publicly verifiable ledger that maintains the anonymity of voters and allows anybody with access permission to audit the Blockchain. To ensure that votes are really counted, it combines the simplicity of digital voting with the immutability of Blockchain technology.

The main uniqueness of this technology is that algorithms enforce the stipulated conditions replacing many existing flaws and vulnerabilities in traditional mechanisms and human trust. With this method, authority over decisions can be spread among the participants, removing the risks and challenges of centralized authority. The security and integrity of all types of transactions are also improved by Blockchain technology. These characteristics make Blockchain technology useful in many application scenarios.

2.1.6 Disadvantages and problems yet to be solved by Blockchain

Despite the enhancements of this technology, there are still some challenges to be solved [29]. They are mainly the following:

- **Energy consumption**
    - Blockchains that use a proof-of-work system can become extremely energy-demanding. As the network grows, the number of validators increases, and there's a fight for more computer power, which consumes energy. Due to the fact that only one node will ultimately be awarded the right to confirm the following block, the energy consumption is incredibly inefficient.
- **Unstable regulatory state**
    - The current market is suffering from high fragmentation, where different companies, consortia and products operate using different rules and protocols. This means developers can't learn from the mistakes and vulnerabilities of others. For example, in November 2019, Bitcoin sank to an all-time low when China accelerated a crackdown on cryptocurrency businesses, mirroring what happened in 2017 when South Korea made a move to regulate cryptocurrency trading.
- **The human risk factors**
    - Recent Blockchain attacks haven't focused so much on the technology, but on basic human vulnerabilities. For example, stolen cryptographic keys[6]

[6] Private digital signatures.

were the likely cause of crypto exchange[7] *Bitfinex's* $73 million breach in 2016 [30].

Because Blockchains are used to manage money, hackers are attracted to this technology. Decentralized finance-related breaches constituted 76% of all major hacks in 2021, with over $1 billion lost in the third quarter alone, according to *Atlas VPN* [31].

A new class of cyber threats is emerging, involving tactics that are unique to Blockchain networks, while others adapted to the technology. These include the following:

- **The 51% attack**
  - As mentioned before, a 51% attack takes place when a person or group takes control of more than half of all the computing power or validation authority of a cryptocurrency network. A group or individual who controls a majority of a cryptocurrency's Blockchain can create and tamper with transactions.
- **Cryptojacking**
  - It's a type of cybercrime where a criminal secretly uses a victim's computing power to generate cryptocurrency [32]. Typically, this happens when a victim unknowingly installs a program that contains malicious scripts that give the hacker access to their computer or another Internet-connected device, such as by clicking on an unknown link in an email or going to a website that has been compromised. The criminal then uses applications referred to as "coin miners" to produce, or "mine," cryptocurrencies.
- **Rug pulls**
  - Rug pulls are a lucrative scam in which a crypto developer promotes a new project[8] to investors, and then disappears with tens of millions or even hundreds of millions of dollars [33]. This fraud accounted for $2.8 billion in lost money for victims, or 37% of all cryptocurrency scam revenue in 2021 [34].

2.2 Bitcoin

Bitcoin was the first decentralized cryptocurrency made available in 2009 as open-source software. A decentralized cryptocurrency is a digital currency that can be used as a medium of exchange over a computer network and is not supported or maintained by any central authority, such as a bank or government.

[7] Platform to buy and sell cryptocurrency.

[8] Usually a new token.

At the heart of the Bitcoin proposal, was an attempt to solve the double-spending problem [35]. If a digital currency can simply be copied, it will. Therefore, it wouldn't be viable as money. Satoshi solved this problem with Blockchain registration. Because Bitcoin is an open-source project, anyone can inspect, change, and improve it. This decision has proven to be crucial in the development of various cryptocurrencies based on the Bitcoin code.

2.2.1 Price and volatility

The main factors influencing the price of Bitcoin are its supply, market demand, accessibility, other cryptocurrencies, and investor sentiment. Its price has experienced cycles of appreciation and depreciation known as bubbles and busts. In 2011, the price of one Bitcoin quickly increased from about $0.30 to $32 before dropping back to $2 [36]. The price of Bitcoin started to rise again in the second half of 2012 [37] and peaked on April 10, 2013, at $266 before falling to about $50. Since then, its price has been growing steadily until the most recent crypto crash in 2022, losing up to 70% of its value in half a year [38].

The volatility for trades between BTC-USD and USD-EUR in November 2021 was 12.6% and 0.95%, respectively [39]. The difference between the two is generally quite large, with Bitcoin showing, on average, about 12 times the volatility of the most traded fiat pair. Compared to the USD price of gold, gold is about two times more volatile than USD-EUR, but still only a sixth of the range of Bitcoin.

2.2.2 Connection with Blockchain

Bitcoin enabled digital money to be used directly from person to person (peer-to-peer) without the involvement of intermediaries, which was the opposite of all previous currencies. Blockchain technology allowed this. Both terms, Blockchain and Bitcoin, have a close connection when talking about one or the other, because the first Blockchain was the database on which every Bitcoin transaction was stored. When Blockchain was first implemented in 2009, it wasn't known as such.

3 LIGHTNING NETWORK

Bitcoin Blockchain is extremely secure, but this initially came at a trade-off: it is slow. On its base layer, there's no getting around the immense computing power and 10-minute average block time [40], increasing concerns regarding its scalability issue. The fees are also a big problem, making it almost impossible to use Bitcoin for micropayments, having an average fee in 2022 ranging from $1 to $5, surging to over $63 in April 2021 [2], displaying its high volatility fees.

The Lightning Network was proposed in a white paper by Joseph Poon and Thaddeus Dryja in 2016 [41]. It overcomes this scalability and fee problem without sacrificing security or trust.

The Lightning Network is a layer 2 payment protocol created on top of the Bitcoin Blockchain. It is intended to enable near-instant transactions among participating nodes[9] and has been proposed as a solution to the Bitcoin scalability and fee problem. It features a peer-to-peer system for making micropayments of cryptocurrency through a network of bidirectional payment channels[10] with low cost transaction fees.

Following this brief explanation, a more detailed view about the key concepts of the network and how it works, will be given in the next subchapters.

3.1 Bidirectional payment channels

On the Lightning Network, a payment channel is a direct, bidirectional payment connection between two nodes. A payment channel can also be viewed as a *multisig* wallet that always needs two signatures to update the balance.

The Lightning Network's main component is its payment channels, which enable quick, low-priced transactions. Payment channels use the underlying Blockchain's security to provide trustless payments between two parties through on-chain[11] opening and closing transactions. Based on the balance and capacity of the channel, users exchange off-chain[12] commitments transactions with each other. Payments made through a channel are off-chain and don't need to be verified by or communicated to the rest of the network. This

[9] Entity on the Lightning Network that can connect to other nodes by establishing a payment channel.

[10] Connection between two nodes.

[11] Transactions that are registered in the Blockchain.

[12] Transactions that are registered outside the Blockchain.

means that there is no set limit on the number of payments that can be shared within a channel and that there is no explicit cost associated with each payment.

A payment channel can be closed by any of the parties by broadcasting an on-chain transaction agreed upon by both parties in the last made commitment transaction, that finalizes the net balance transferred over channel's lifetime.

The lifecycle of a bidirectional channel explained before, can be visualized in Figure 3.

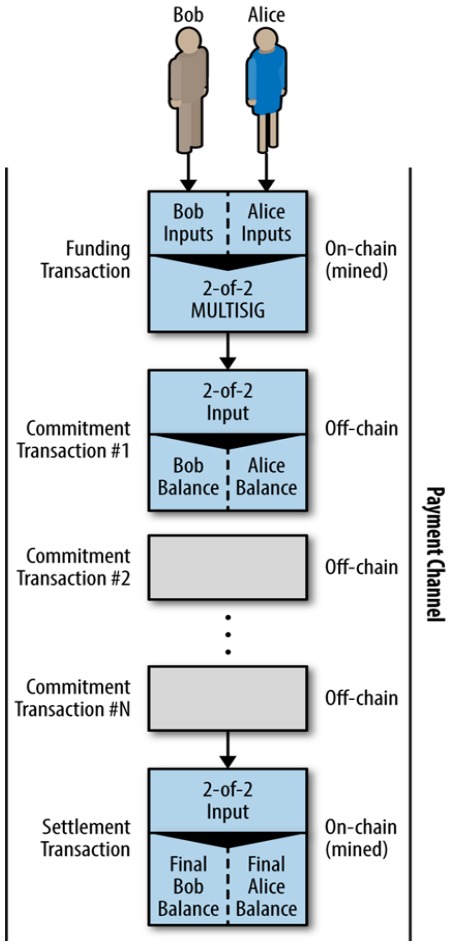


Figure 3 - Payment channel lifecycle [42]

3.1.1 Use case

To better understand how it works, a use case where Adam visits a Coffee Shop every day is a good example, displayed in Figure 4. The following steps are the lifecycle of the payment channel presented in the Coffee Shop use case, adding a possible ending to the channel.

1. **Payment Channel Creation:** Both users open a new payment channel and deposit the adequate funds, signing it. This deposit is registered in the Blockchain.

2. **First Coffee Payment:** The Coffee shop creates an invoice, much like a receipt, so Adam can make the payment and update the channel balance.
3. **Adam Confirms:** Adam signs the receipt (pays the invoice), confirming the coffee value is correct. Balance is updated in the channel.
4. **Second Coffee Payment:** The Coffee shop creates another invoice to the Adam, so Adam can make the payment of another coffee.
5. **Adam Confirm:** Adam signs the receipt, confirming the coffee value is correct. Balance is updated in the channel.
6. **Payment Channel Closing:** Adam decides he needs the funds existing in the channel. The last balance update is distributed for both accordingly.

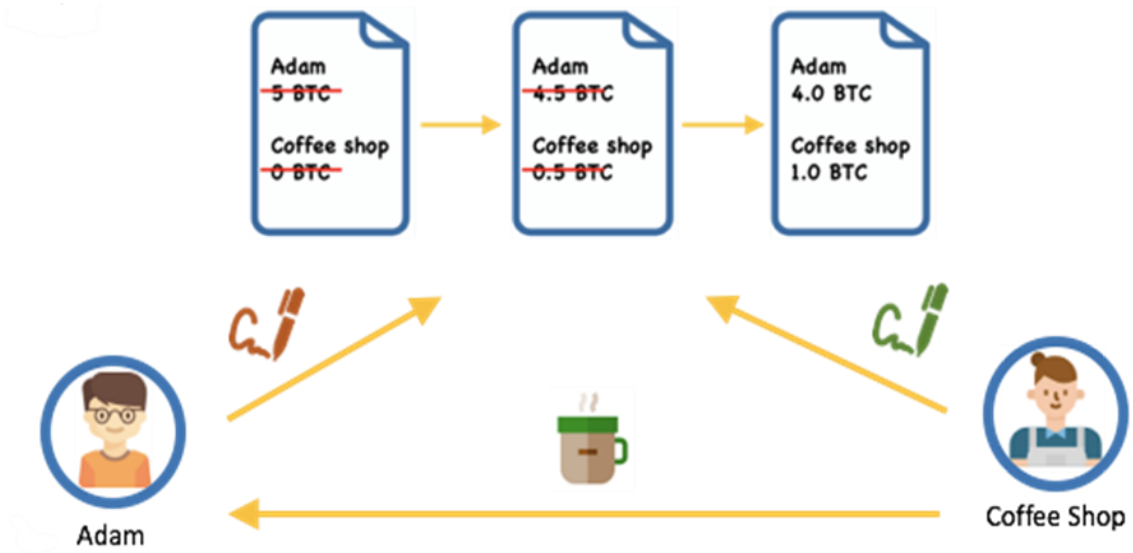


Figure 4 - Coffee Shop use case [43]

3.2 Nodes and channel types

A node is an entity on the Lightning Network that can connect to other nodes by establishing a payment channel. Users can create a node by running an implementation of Lightning Network software, either directly or as part of pre-packaged wallet software. This should not be confused with a Bitcoin full node that validates Bitcoin blocks.

There are different types of nodes with different purposes [44], the most used ones in lightning are:

- **Full Node**
  - Full Nodes are fully independent on the network and don't need another node or server's assistance. Users private and public keys are held by the node so they're in full control of the node.

- **Partial Node**
    - Partial nodes don't store the whole Blockchain, only a tiny part of it, making it more efficient. They always require an external server to delegate tasks, like routing (explained in the following section).
- **Routing/Bridge Node**
    - Routing node [45], or Bridge node, is another term for a *hub*. A functional routing node requires high uptime and strong connectivity to other nodes to facilitate consistent, reliable, and economic routing. Routing nodes can charge fees for passing traffic through to the remainder of the route.
- **Gateway Node**
    - These nodes will directly serve end users. They will serve a relatively small number of users (likely in the hundreds) and will have modest hardware, bandwidth, and capital requirements.
- **Trampoline Node**
    - These nodes contain the full network graph and will be in charge of finding a route to the final destination instead of the sender [46]. In the future, as the number of channels and nodes increases, this will be especially helpful for mobile applications that use the Lightning Network by allowing them to run light clients rather than complete Lightning Network nodes.

There are two main types of lightning channels: **advertised** and **non-advertised** channels [47]. The majority of nodes and channels will not be available for routing and will not be visible in the network graph. End user nodes (smartphones, laptops, etc.) will not "advertise" channels by default since they aren't online all the time. Bridge channels connect two routing nodes. Figure 5 shows a possible path of a lightning payment passing through different types of nodes, showing their uses, until it arrives to the recipient.

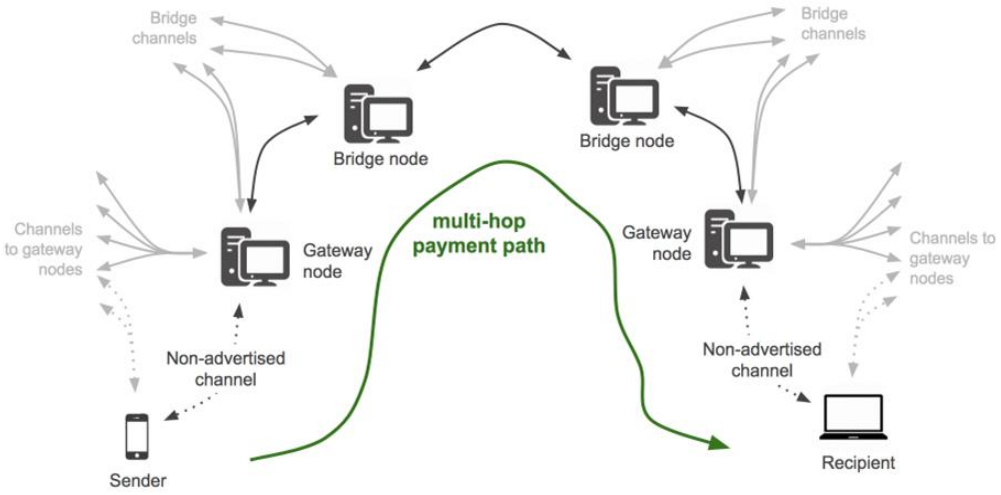


Figure 5 - Payment path [48]

To combat one of the disadvantages of the lightning network, that being, the need to wait for the funding channel transaction to be registered in the Blockchain when creating a new payment channel between users, **turbo channels** [49] were created.

Turbo channels allow users to immediately receive and send funds over the Lightning Network without having to wait for a channel to confirm on the Blockchain. While the user waits for the approval of the on-chain transaction to open the payment channel, the wallet provider publishes that an opening transaction of the required amount has been carried out, even if it has not been confirmed, and whoever sent the channel opening transaction retains the value associated with it in the channel. The obvious disadvantage of this type of channel is trust. The user must trust the wallet provider since there is nothing on-chain that proves the existence of the channel.

3.3 Onion routing

Onion routing is a technique for anonymous routing of payments used by the Lightning Network. This routing can be calculated by the user node (source routing) or by a specific node to do so, the routing nodes. This routing is calculated using an algorithm specified in the Lightning Network implementation. Routing messages are encapsulated in layers of encryption, analogous to layers of an onion, as showed in Figure 6. The encrypted data is sent through (routed) network nodes, each of which "peels" away a single layer to reveal the data's subsequent location. When the final layer of routing information is decrypted, the message arrives at its destination. Since each middleman only knows the locations of the nodes that are immediately before and after it, the sender's identity is kept private.

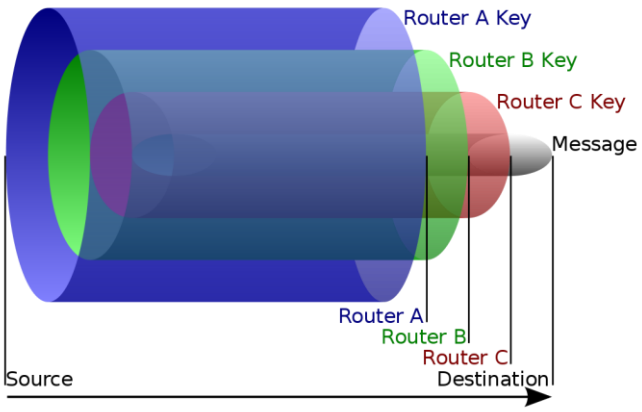


Figure 6 - Onion routing example [50]

Onion routing means that each node only sees the immediate hop[13] before it and the immediate hop after that. It is called onion routing because the routing information is

[13] A jump between two nodes/routers.

wrapped in layers. So, a node receives an encrypted package from the node one hop before it and doesn't know where the final destination is.

3.4 Lightning invoice

A Lightning Network invoice is a request for a payment. Invoices contain the data required to complete a payment on the network, such as the payment amount, the Blockchain to which the invoice applies, its expiration date, and the payee public key. The Lightning Network does not use Bitcoin-style addresses. Instead, payments are made using invoices. Invoices are commonly presented as alpha-numerical strings or QR codes.

3.5 Fees

The Lightning Network utilizes several types of transaction that have different possible fees [51]. The Lightning Network's nearly free internal payment transfers are one of its most prominently advertised benefits.

Fee types include:

- **Singles channel transactions**
    - Fees are never charged during transactions between parties in an established channel. As transactions occur between the two parties, the channel balance is updated without the need for fee allocation.
- **Routed transactions**
    - Transactions are routed between network participants through multiple channels, subject to routing fees from intermediate nodes. These fees could be set at zero.
- **On-chain transactions**
    - Channel opening and closing transactions are recorded on the Bitcoin Blockchain, and as a result, they are subject to fees based on the current Bitcoin fee rate.

Figure 7 presents a simple scenario to exemplify and explain how fees work on the Lightning Network. The use case describes the routing fees necessary to make a payment from Alice to Daniel. Since Alice and Daniel don’t have a direct payment channel, a routing path was created that hops through Bob and Charlie. Since Bob and Charlie will have to do some work to hop the transaction to the next node, they will take a fee for it, that being, 0.001 Bitcoin. After considering the fees of every hop needed until the final destination, the total fee amount, 0.002 Bitcoin, is added to intended amount that Alice wants to send to Daniel, 0.2 Bitcoin, making a total of 0.2002 Bitcoin. So, when hoping though Bob and Charlie, the necessary fee can be paid while Daniel will still receive the intended amount.

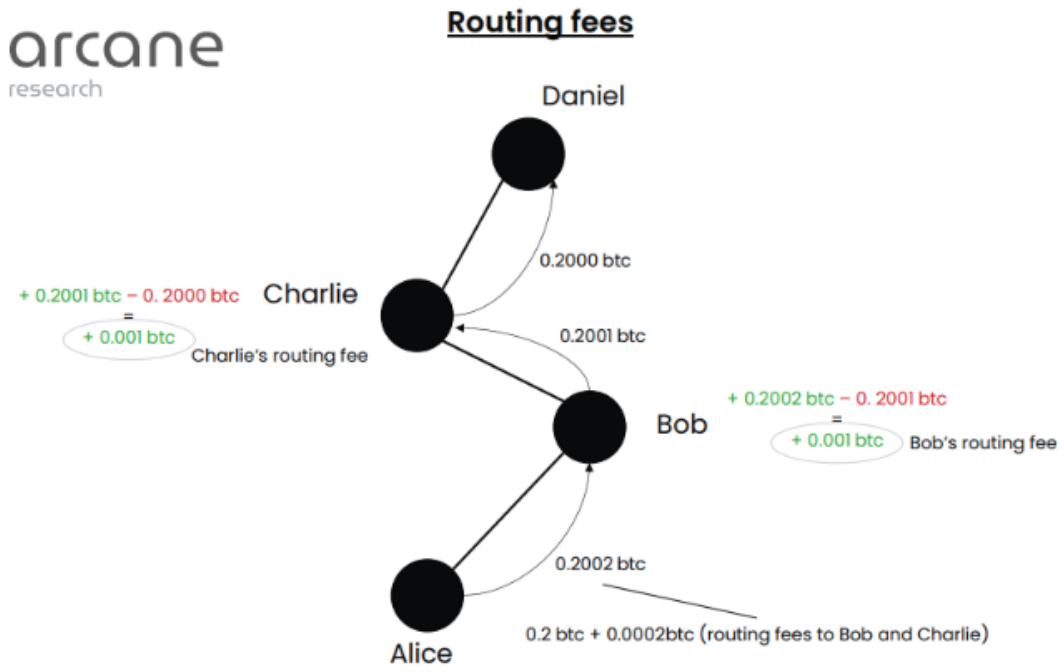


Figure 7 - Routing fees example [48]

3.6 Smart contracts

A smart contract is a computer program or transaction protocol created to automatically manage, record legally, or carry out significant events and actions in accordance with the terms of a contract or an agreement [52]. It is self-executed and can be used to automate workflows, triggering the next action when specific conditions are met as showed in Figure 8. The agreements and code are distributed across a decentralized Blockchain network. They enable the execution of trusted agreements and transactions between various, anonymous parties.

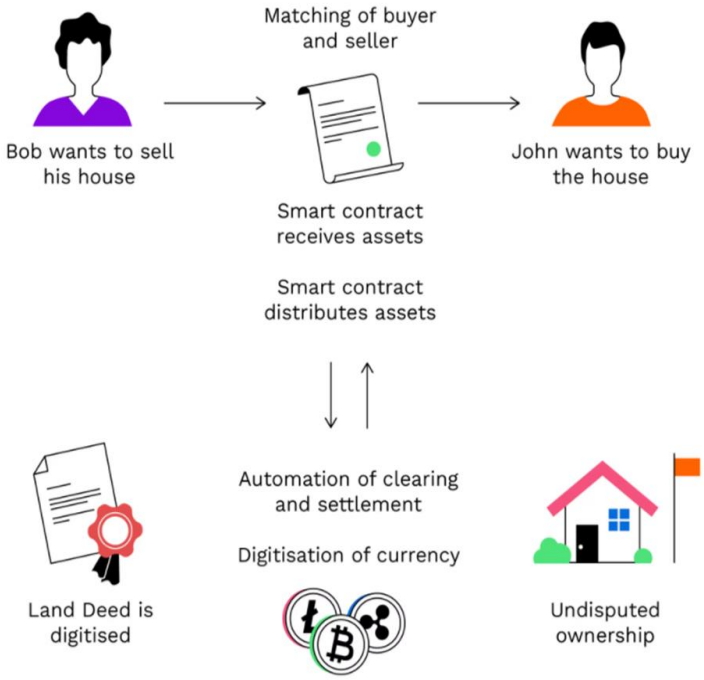


Figure 8 - How a smart contract works [53]

Despite its advantages, there are still some drawbacks to be considered, those being:

- Inability to adjust smart contracts work since they are immutable
- Weak legal regulation of smart contracts work
- High dependence on programmers and exposure to bugs

Smart contracts operate with simple "if/when ... then" statements written into code on a Blockchain [54]. The computer's network acts when certain conditions are met and verified. These actions might involve paying out money someone, registering a car, sending out notifications, or writing a ticket. The Blockchain is then updated following the completion of the transaction. That means the transaction cannot be changed as mentioned before, and only parties who have been granted permission can see the results.

3.7 Hashed TimeLock Contract

A Hashed TimeLock Contract or HTLC is a smart contract that allows transactions to be sent between parties who do not have a direct channel on the Lightning Network. HTLCs rely on a person's ability to structure a payment so that a different party can only accept it if the other party is aware of the preimage, also called secret[14] of that HTLC.

[14] Data string, working like a password key to the HTLC, that allows the user to accept the payment.

HTLCs employ hashlocks and timelocks to guarantee payment security. A Hashlock is a claim against a property by a party that isn't the owner of said property, that restricts the spending of an asset (e.g., an amount of Bitcoin) until a specified piece of data is publicly revealed, the preimage. The timelock is a feature of HTLCs that mandates that the recipient of a payment acknowledge receipt of the payment before a deadline by producing cryptographic proof of payment, failing to do that, the recipient loses the right to claim the payment and must return it to the payer. This concept can be expanded to allow a series of payments because each receipt of money prompts the creation of a new hash. With the proper conditionality, payments can be safely routed through a number of users. Given the complexity of this process, an illustration of a transaction utilizing multiple HTLC's is displayed detailed in Figure 9 below, giving a step-by-step explanation.

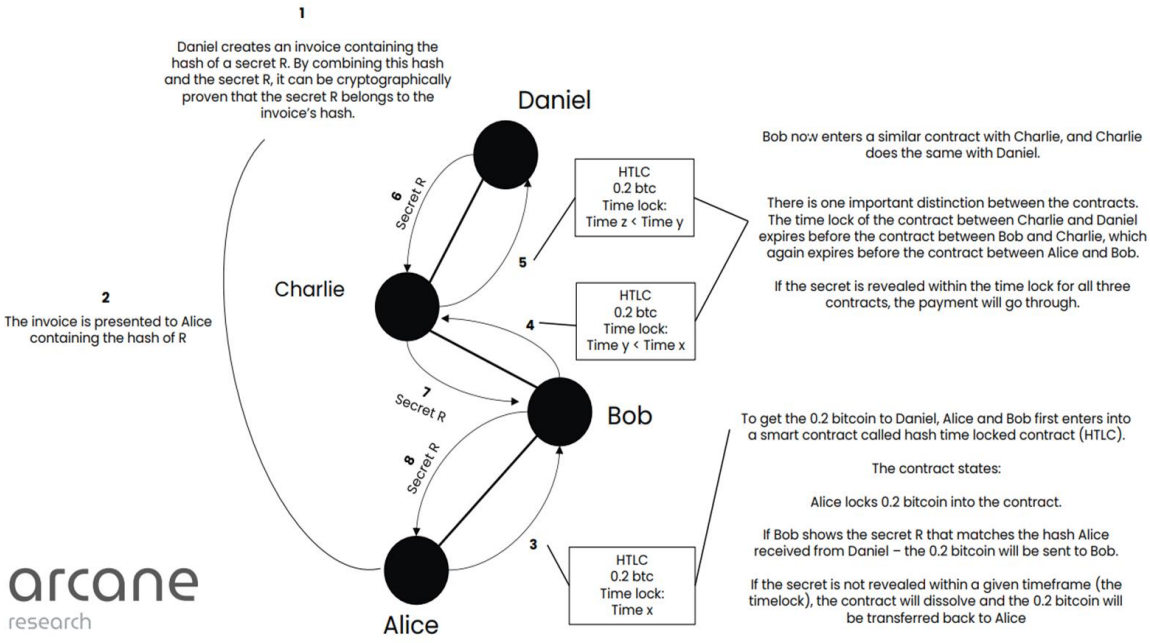


Figure 9 - HTLC use case example [48]

3.8 BOLT - Basics Of Lightning Technology

The Basis Of Lightning Technology [55] is a standardized technical specification for the Lightning Network's implementation. The standards specify how different implementations can communicate with one another and build the same network. The specifications are currently a work-in-progress and are continually being iterated upon as projects like LND and Core Lightning are built in parallel.

Below is the list of the current BOLTs to be followed by the lightning implementations:

- **BOLT #1 - Base Protocol**

    - This BOLT describes the basic LN Protocol. Each LN Node is connected to the Bitcoin Network, monitoring the Blockchain and connected to Lightning Network.
- **BOLT #2 - Peer Protocol for Channel Management**
    - The peer channel protocol has three phases: establishment where the channel creation request and channel funding transaction are made, normal operation where the transactions are made between both users and closing the channel phase where the balance is distributed adequately to each user.
- **BOLT #3 - Bitcoin Transaction and Script Formats**
    - This details the exact format of on-chain transactions, which both sides need to agree to ensure signatures are valid. Consists of the funding transaction output script, the commitment and HTLC transactions.
- **BOLT #4 - Onion Routing Protocol**
    - This BOLT describes the construction of an onion routed packet that is used to route a payment from an origin node to a final node. The packet is routed through several intermediate nodes, called hops and described in subchapter 3.3.
- **BOLT #5 - Recommendations for On-chain Transaction Handling**
    - Lightning enables two parties to conduct transactions off-chain by providing each party with a cross-signed commitment transaction that summarizes the channel's current state (the current balance of each).
- ~~**BOLT #6 - IRC Bootstrap Protocol**~~
    - Describes the use of IRC to exchange network announcements.
- **BOLT #7 - P2P Node and Channel Discovery**
    - This specification describes simple node discovery, channel discovery, and channel update mechanisms that do not rely on a third-party to disseminate the information.
- **BOLT #8 - Encrypted and Authenticated Transport**
    - All communications between Lightning nodes are encrypted in order to provide confidentiality for all transcripts between nodes and are authenticated in order to avoid malicious interference.
- **BOLT #9 - Assigned Feature Flags**
    - This BOLT tracks the assignment of feature flags for different bolts like BOLT #1 as well as BOLT #7. These flags are intended to help the developer with the Lightning Network implementation, being necessary to use them in channel and node announcement messages for example.
- **BOLT #10 - DNS Bootstrap and Assisted Node Location**
    - This specification describes a node discovery mechanism based on the Domain Name System (DNS). Its two main purposes are to provide the initial node discovery for nodes that have no contacts in the network and

supporting nodes in discovery of the current network address of previously known peers.

- **BOLT #11 - Invoice Protocol for Lightning Payments**
    - A QR-code-ready, extendable protocol for requesting payments over Lightning. This protocol uses some technicalities of the Bitcoin invoices, such as bech32 encoding[15] making it easier for developers with Bitcoin implementation experience.

As listed above, there are currently 10 BOLTs in effect, with BOLT #6 being superseded by BOLT #7, allowing for a more efficient way of transmitting information on the network than all the information dissemination regarding node discovery, channel discovery, and channel.

The BOLTs are in constant update, and the most recent proposal, "BOLT #12: Offer Protocols for Lightning Payments", was already made by the creator of BOLT #11, Rusty Russel, a Linux developer and lead developer in the Core Lightning project [56]. He identifies some limits to BOLT #11: "Some are simple things, like it's not easily extensible, the encoding is weird, and the bech32 encoding is tied closely to the format" [57]. With BOLT #12 it will also be possible to generate a lightning invoice to prompt the user to pay certain amount, but the same invoice can be reused to later receive a different amount from a different user.

3.9 Network growth

By observing actual transaction data to estimate the use of the Lightning Network, it's easy to conclude that adoption is rising quickly.

An analysis of the number and volume of Lightning Network transactions indicates that the use of this network is rapidly rising (Figure 10). The number of payments has roughly doubled over the last year, while the value of the payments has increased by more than 480%, measured in US dollars. In November 2021, the data is distorted by a large number of deposits and withdrawals of Bitcoin because of the record high Bitcoin value.

[15] Encoding scheme used to encode Lightning and Bitcoin invoices.

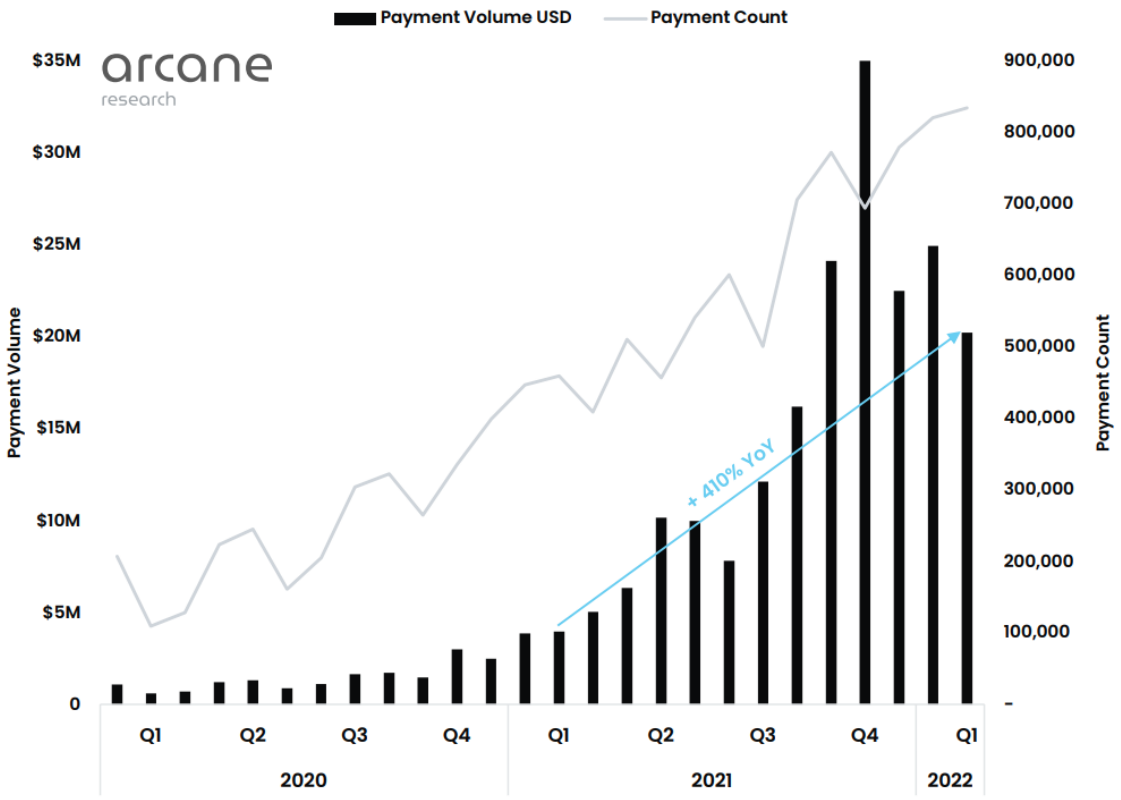


Figure 10 - Payment volume and payment count on the Lightning Network [48]

As seen in Figure 11, estimations show that just over 100.000 users had access to Lightning payments globally as of summer of 2021. In March 2022, estimations show that more than 80 million people had access to Lightning payments. More and more big companies Are expected to enter the ecosystem in the near future the ecosystem with names such as: Shopify, NCR and Blackhawk Network [58].

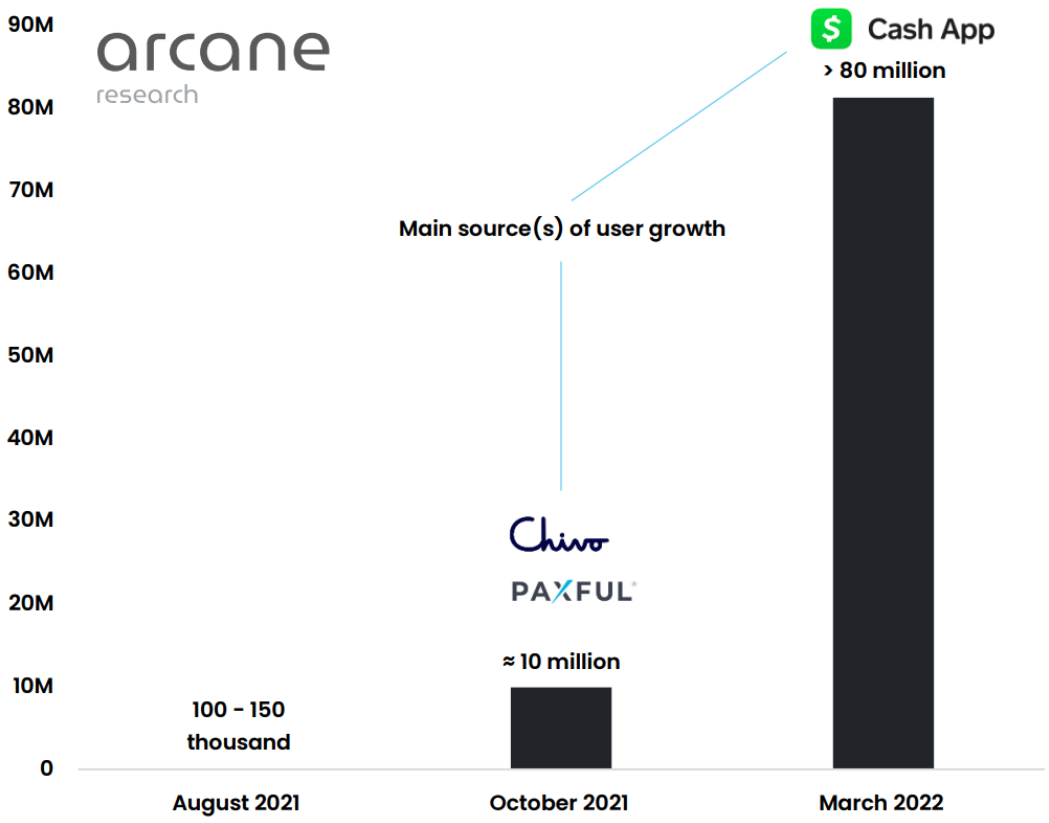


Figure 11 - Users with access to Lightning payments [48]

Given the recent crash of the Bitcoin value, it is relevant to understand how the use of the Lightning Network is affected by it. The cumulative Bitcoin capacity across all channels [59] shows that the use of the Lightning Network was not affected by it (Figure 12). In fact, the Lightning Network Bitcoin capacity has steadily grown, suggesting that this technology is still a key factor for Bitcoin-related applications. The difference observed is that the amount held in the channels dropped due to this devaluation.

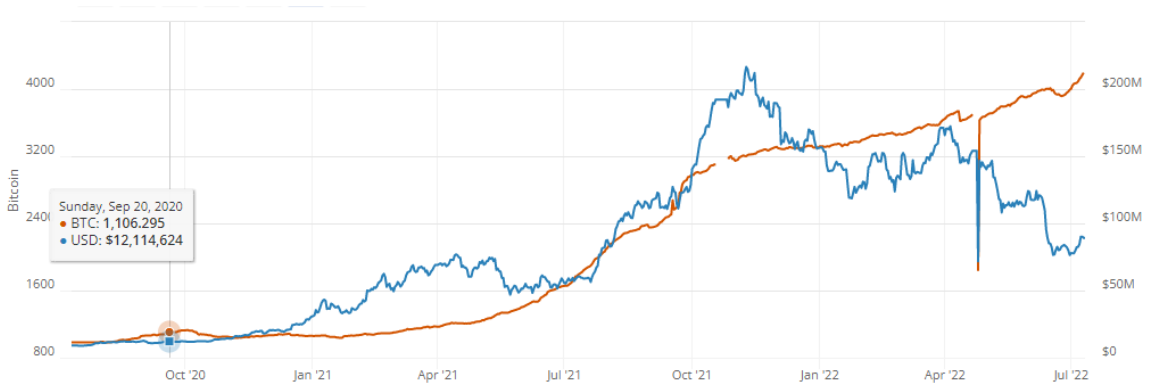


Figure 12 - Channel capacity in BTC and USD [59]

3.10 Limitations

The Lightning Network is an ever-evolving concept that is likely to make a significant difference to Bitcoin's Blockchain in the long run. However, the network might not be the solution to all the challenges facing Bitcoin due to some of its limitations [60]. The most prevalent are the following:

- **Locked funds**
  - Users have to close the channel to be able to use the funds on it for other purposes.
- **Opening and closing channel costs**
  - Since opening and closing channel transactions are registered on the Bitcoin Blockchain, fees can be high.
- **Channel capacity**
  - Each payment channel has a maximum capacity set with the funding transaction, to send a higher amount, a new channel has to be created.
- **No offline support**
  - Both sides need to be online (Node) to make a successful transaction. These encourages centralization by using a third-party node that runs 24 hours a day, since running a node 24 hours a day can be hard.

Another possible problem are attacks on the network. On example is the congestion attack. Congestion attacks are made on HTLC-Base payment channel networks [61], which is the Lightning Network case. In short, if a malicious party creates numerous channels and forces them to expire at the same time, which would broadcast to the Blockchain, the congestion caused could overwhelm the capacity of the block. A malicious attacker might use the congestion to steal funds from parties who are unable to withdraw their funds due to the congestion.

4 CRYPTOCURRENCY WALLETS

A cryptocurrency wallet is an application or program that enables users to manage their cryptocurrency. Just like bank accounts, to use cryptocurrencies it is necessary to have an account, in this case, a wallet, to "store" the virtual currencies. "Store" because technically each user's cryptocurrencies are registered on the cryptocurrency Blockchain in question, where only the user with the respective private key can access them.

Wallets manage public and private keys, track balances, and create and sign transactions. Keys are often stored in the wallet on each user's computer or smartphone. Possession of the key that can sign a transaction is the only prerequisite to spending cryptocurrency, placing the control entirely in the hands of each user.

4.1 Lightning wallet types

Wallets are one of the most actively developed applications in the Bitcoin and Lightning ecosystem with an existing intense competition between solutions. Choosing a wallet is highly subjective and depends on the use that will be given to the wallet and user experience.

There are multiple types of wallets [62], the most known are:

Desktop wallet: A desktop wallet is supported by macOS, Windows, or Linux. Desktop wallets, which frequently operate a full Bitcoin and Lightning node, are common in the Lightning Ecosystem because the Lightning Network needs continuous uptime to maintain channels since they must be current with any channel modifications.

Mobile wallet: Numerous mobile wallets for Lightning are currently in development and have not yet matured due to the requirement of hosting a full Bitcoin node in order to interact with the Lightning Network. It is currently inefficient for the user phone to host a wallet with such resource requirements.

Hardware wallet: The hardware wallet runs self-hosted Bitcoin and Lightning nodes on dedicated hardware. Hardware wallets can operate with high uptimes and enable high levels of security because they handle all Bitcoin and Lightning-related operations on the hardware.

Web wallet: Web-based wallets are operational Bitcoin or Lightning wallets that are entirely based on web services. They enable straightforward interaction with the Lightning Network or Bitcoin protocol and frequently have few features.

4.2 Custodial and non-custodial

In addition to the different types of wallets, there is also a talking point about wallets and one difference that distinguishes the wallets when choosing one [63]. That being if the wallet is custodial or non-custodial.

One of the big differences is that with custodial wallets, the entity that provides the wallet holds the keys. With non-custodial wallets, the user has full control of its own keys. Some other advantages and disadvantages of each can be seen in the following subchapters.

4.2.1 Custodial wallet

If the user chooses a custodial wallet, the main advantages are:

- Account recovery is possible in case of lost passwords
- Network fees are often lower, or even none, when transferring within the same ecosystem
- Custodial wallets usually have a more friendly user interface. The wallet entity often hides some complexity when managing keys

Users must also consider a few potential drawbacks:

- User has technically no control or security over their own funds, not governing his own wallet keys
- Wallet provider can be hacked by malicious actors
- Need to abide by all KYC regulations (mandatory process of identifying and verifying the client's identity)

4.2.2 Non-custodial wallet

The benefits of selecting a non-custodial wallet are:

- More secure since users have control of their own keys
- User retains full control of the funds
- No need for KYC

Non-custodial wallets have the following disadvantages:

- Lost keys mean irrecoverable funds
- Transactions fees borne fully by the user
- Usually requires more expertise from the user, being necessary more caution when dealing with the user keys

4.2.3 Choosing the right wallet

Choosing between custodial and non-custodial wallets depends largely on what the user wants to do with it. For example, if it's a new user, a beginner in crypto, custodial wallets are usually recommended since it's easier to try out how crypto trading works having generally a more user-friendly interface that requires less management and low responsibility when it comes to worrying about securing their own keys.

On the other hand, non-custodial wallets are a great way to ensure user assets are not subject to confiscation by a third party, usually preferred when storing greater amounts of cryptocurrency since they are more secure since the user controls his own keys.

Each type of crypto wallet has its own advantages and setbacks, and the comparison made between them presents clarity for making decisions.

5 STATE OF ART

The purpose of this chapter is to introduce readers to the main Lightning Network architectures and characteristics, some of the existing implementations and tools as well as several of the existing wallets in the market.

Answers to questions like what architectures and characteristics are currently in use (Section 5.1), what type of implementations and tools are being used nowadays (Section 5.2), what wallets that make use of the Lightning Network and its characteristics (Section 5.4) can be found here. A conclusion on features that are expected to exist in a Lightning wallet is also given in (Section 5.5)

5.1 Lightning Network architecture and characteristics

One way to categorize Lightning Network wallets is by their degree of autonomy [62], being one of their most important architecture characteristics, which details how they interact with the Bitcoin Network:

Full-node client: A full client, also known as a "full node," is a client that handles user wallets, records the whole history of all Bitcoin transactions (every transaction made by every user, ever), and can start transactions on the Bitcoin and Lightning networks. A full node manages every component of the protocol and is capable of independently validating any transaction and the whole Blockchain.

Lightweight client: A lightweight client connects to Bitcoin full nodes or utilizes protocols like neutrino[16] for access to the Bitcoin transaction data, but saves the user wallet locally and independently validates, creates, and transmits transactions. Lightweight clients communicate directly and without a middleman with the Bitcoin Network.

Third-party API client: A third-party application programming interface (API) client is one that communicates with Bitcoin or Lightning through a system of application APIs provided by a third party as opposed to connecting directly to the Bitcoin Network. All transactions go through a third party even though the wallet may be stored on the user's computer or on servers owned by a third party.

After mentioning the different types of node connections, a node difference is noticeable, if a node is remote or not. The differences are similar to the custodial vs non-custodial wallet differences. With a remote node, the user has the following advantages:

[16] Bitcoin light client with the main objective to minimize bandwidth and storage use on the client side.

- Easier to manage with little responsibility and usually more convenient
- These services usually have a better user experience, requiring less technical knowledge from the users to make transactions, making it a better choice for less experienced users
- Higher backup possibility

The disadvantages with the remote node are:

- The user does not hold their own private keys
- Data breach threat due to attacks on company servers
- Need for KYC

With the expansion and evolution of the Lightning Network, more solutions are being developed to upgrade this new technology with new solutions to existing problems. Three examples of this are specific nodes just to calculate routes between peers, submarine swaps servers and turbo channels.

A submarine swap [64] is a transaction involving both off-chain and on-chain digital assets (i.e., between Bitcoin held on the Lightning Network and Bitcoin on-chain). Without counterparty risk[17], submarine swaps can be carried out without additional restrictions like contractual arrangements or background checks. Either the two parties successfully swap their assets, or the swap is unsuccessful. But neither side is ever able to take the money belonging to the other side with them.

5.2 Lightning Network implementations and available tools

The landscape of Lightning implementations and tools is steadily evolving, providing improved coverage of multiple use cases and better meeting requirements of differently skilled users. Productization routes are becoming more obvious as the major developing organizations improve their individual strengths and value propositions.

5.2.1 Lightning Network Daemon

Lightning Network Daemon (LND) [65] is developed by Lightning Labs, and it is known for its extensive documentation. Developers can more easily experiment with the software and create programs that interact with the implementation and expand its capabilities as a result.

[17] Probability that the other party in an investment, credit, or trading transaction may not fulfill its part of the deal.

The LND implementation has experienced the highest level of community involvement due to its emphasis on developer integration, the creation of applications on top of it, as well as an easier setup process.

LND also has the largest full-time development team. As a result, the team has managed to build an abundance of value-added services around LND, such as the autopilot [66] and the liquidity service Lightning Loop [67]. Autopilot partially automates the task of opening channels based on heuristic rules. The main objective is to help the user connect with the best channels when creating new ones. Lightning loop is a non-custodial service offered by Lightning Labs that makes it easy to move Bitcoin into and out of the Lightning Network using submarine swaps.

5.2.2 Core Lighting

Core Lightning [56], previously known as c-lightning, developed by Blockstream, is a versatile and light-weight Lightning Network implementation that can operate on low-end hardware. Blockstream's approach aims to give users the tools they need to customize Core Lightning by letting them add plugins to the node's solid foundation. BOLTs are heavily valued by Core Lighting.

Core Lightning implementation was built from the ground up, leveraging the base specification documents of Lightning technology to create a fully compliant software. This implementation focuses on doing the basics with security and efficiency, leaving it up to the user to add what they need on top with the plugins. The fact that Core Lightning currently requires some user effort to get up and running is a drawback of this strategy.

The usage of the Tor Network and BOLT 12 are two additional new features that have been developed for Blockstream's implementation that increase the capability of the underlying specifications. With the Tor Network[18] [68] there is the possibility to run the user node behind it and even generate an address in the network for others to connect to open channels.

5.2.3 Lightning Dev Kit

Lightning Dev Kit (LDK) [69], created by Spiral, Jack Dorsey's[19] company [70], on the other hand, takes an abstraction-based approach, relieving developers of the burden of low-level Lightning technology specifics so they can rapidly and simply integrate the Lightning Network into their existing applications. Instead of offering a complete node, LDK aims to provide a toolkit to assist those who are developing on Lightning. It offers code for every isolated component of a Lightning node, including the routing logic,

[18] Open-source software for enabling anonymous communication.

[19] Entrepreneur, programmer, and philanthropist who is the co-founder and former CEO of Twitter.

channel management, logic for tracking the Blockchain's current status to determine whether channels are open, and more.

The LDK team is undertaking a very distinct endeavor in contrast to other Lightning implementations. As a result, it isn't giving any feature set a higher priority than any other. The objective of LDK is to broadly support all Lightning protocol standard capabilities and to provide developers the freedom to utilize any specified features in their own applications whenever they see fit.

5.2.4 LNbits

LNbits is a free and open-source lightning-network wallet/accounts system, that facilitates the use of the Lightning Network to developers [71]. LNbits can run on top of any lightning-network funding source, currently there is support for LND, Core Lightning, Spark, LNpay, OpenNode and lntxbot.

LNbits offers a variety of interesting Lightning applications through the extension feature, which allows to create and customize the service to the desired needs quickly and easily.

Currently, LNbits can be used as:

- Accounts system to mitigate the risk of exposing applications to the user's full balance, via unique API keys for each wallet
- Extendable platform for exploring lightning-network functionality via LNbits extension framework
- Stream Alerts, Paywalls, Split Payments and more
- Instant wallet for LN demonstrations
- Part of a development stack via LNbits API

5.3 Comparing Lightning Network implementations

One of the drawbacks when using LND is that it isn't as performant as Core Lightning, a flexible and lightweight implementation of the Lightning Network using C programming language. Another growing concern is that Lightning Labs reduced their contributions to the BOLT and instead produced features without going the open-source protocol/BOLT way. Recent releases also show preference for centralized solutions.

Core Lightning has its features and advantages compared to other implementations, but, overall, it has fewer features, and it requires a lot more work from the user to get up and running, with the necessity, for example, to manage their own payment channels. Given its efficiency and lightweight footprint, Core Lightning is better suitable for low-specification devices.

LDK has its focus on the customization of the node and developer friendly integration, having less features than LND and Core Lightning while also still receiving constant updates, causing it to still receive considerable changes from time to time.

When it comes to LNbits, it is a good choice for developers who want to avoid the complexity of LND and Core Lightning implementations. It is also easier to create Lightning demonstrations or a Lightning implementation via LNbits API. Considering this, removing complexity for the developer also means that the implementation will be less customizable, being the main disadvantage comparing to the previous implementations.

5.4 Existing wallets solutions and its features

This section presents some of the most used lightning wallets with its features and implementations of choice. This study is aimed at obtaining an overview about these wallets and to identify the most relevant features of each wallet to help develop a wallet prototype.

The analyzed wallets were:

- Cash App
- Muun
- Chivo
- Phoenix
- Spark

The choice was made considering the downloads made for each and their type of Lightning Network implementation mainly, making it a diverse pool of wallets in term of implementations.

When studying all the wallets mentioned, a group of core features were identified in almost all of them, those being:

- Create/register wallet
- User authentication
- Check current balance
- Initiate a lightning transaction
- Receive a lightning transaction (Except Cash App)
- Check transaction history

Some additional features and drawbacks were also analyzed to create a more complete analysis.

5.4.1 Cash App

Cash App, created by Block, Inc., a separate Jack Dorsey company, is a mobile payment service that enables users to send money to one another via a mobile app [72]. Recently it has introduced the possibility to make Bitcoin payments using the Lightning Network. This implementation is made using LDK.

It also has some additional features, such as:

- Pay any Lightning invoice for free
    - Cash app covers any fee cost.
- Easy to buy Bitcoin with the application

Some of the downsides:

- Only available in USA and UK
- At the moment, the user can only pay invoices using Lightning Network and not receive

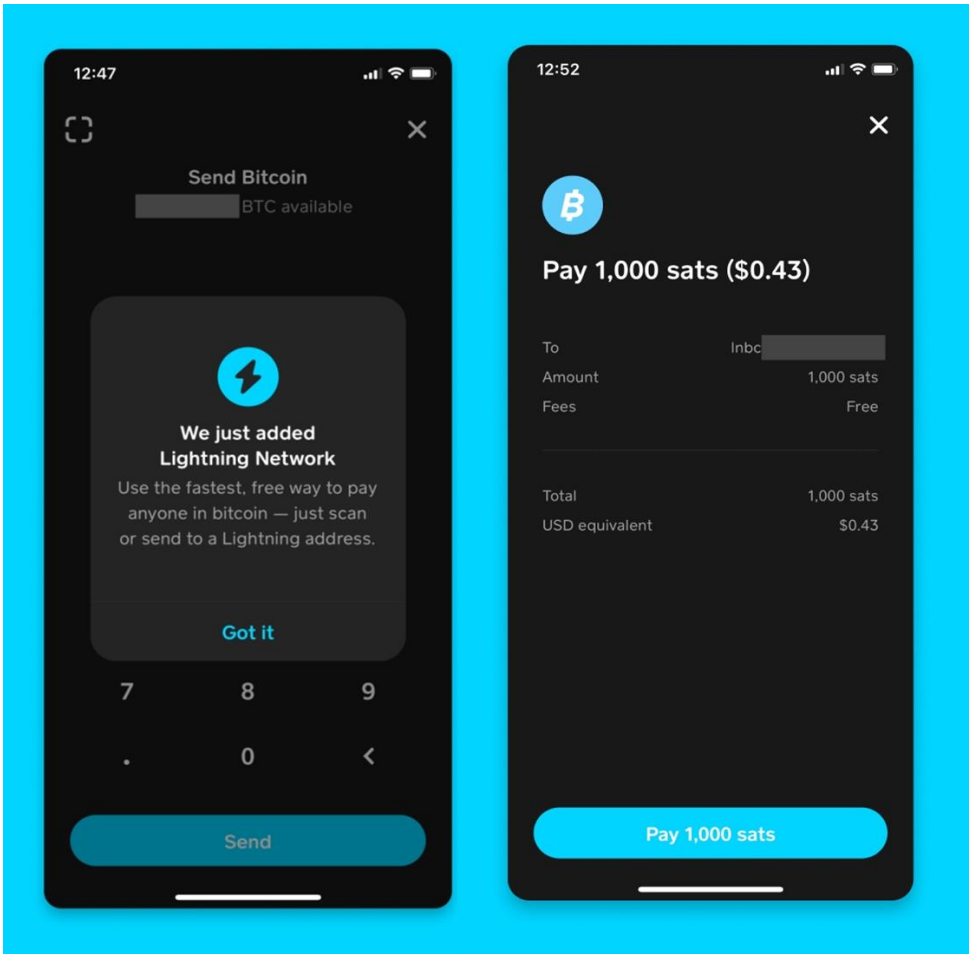


Figure 13 - Cash App demonstration [73]

5.4.2 Muun

Muun is one of the most popular among the non-custodial wallets, having more than 100.000 download in the Google Play Store alone [74]. It is a Bitcoin and Lightning wallet that has found a way to integrate a Lightning and on-chain wallet into the same user interface with just a single Bitcoin balance that users need to consider. The lightning implementation is made using LND.

Some of the Muun additional features:

- Protected with a 2-of-2 multisig
    - Two keys are needed to spend the wallet balance, one is on the user phone and the provider has the other one, preventing from malicious attacks when the attacker has access to one of the keys.
- Turbo channels
    - Allow users to immediately receive and send funds over the Lightning Network when creating a new channel.
- Emergency kit
    - PDF document with the information to recover lost funds.

One drawback is that Muun fees fluctuate to the high end compared to other wallets.

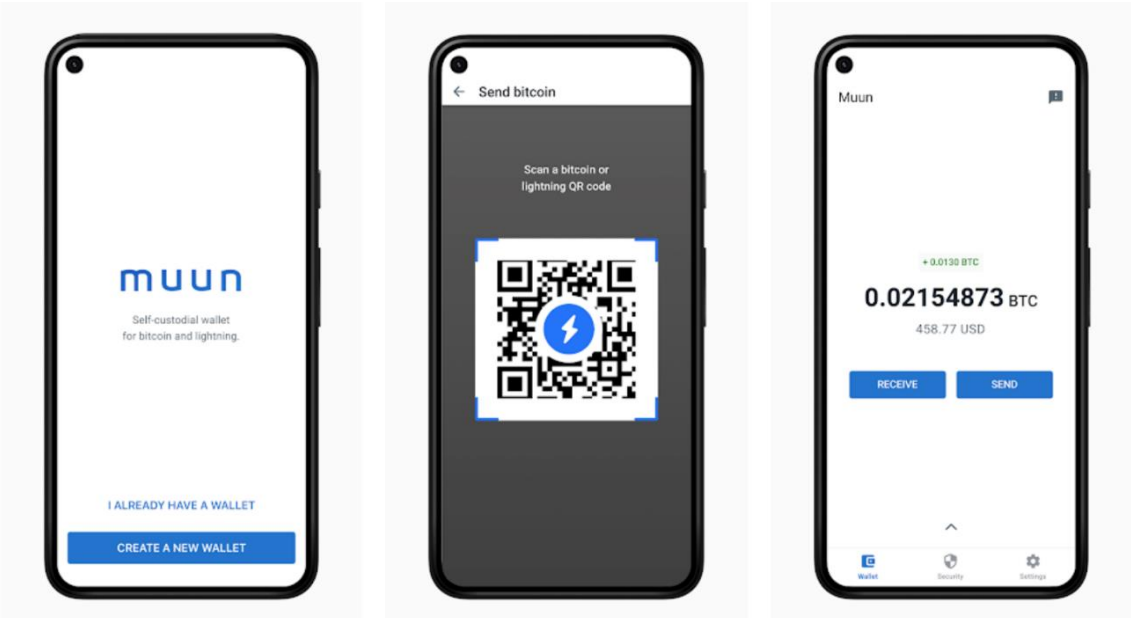


Figure 14 - Muun demonstration [74]

5.4.3 Chivo

Chivo wallet [75] is a state-endorsed Bitcoin wallet for El Salvador that became available on September 7, 2021. Chivo is slang for “cool” in El Salvador. Since both Bitcoin and the USD have been recognized as legal tender in the country, the Chivo wallet app keeps

track of both balances. Transaction fees are also nonexistent. Chivo Lightning implementation is private.

The country's reliance on remittances[20] is the primary factor contributing to the Lightning Network's success in El Salvador. According to a World Bank Report [76], personal remittances make up 24% of El Salvador's gross domestic product, which is roughly $6 billion.

Chivo additional features:

- It has no fees associated with any buy in El Salvador
- Every user receives 30$ for registering

Since Chivo was launched, it has had a large record of problems [77]. Problems like: Blocked accounts, unauthorized charges, failed transactions, personal data leaked and more.

The wallet has improved a lot since then, contracting recently *AlphaPoint* [78] to provide the technology behind Chivo's mobile application, mobile point-of-sale processing, merchant website portal, call-center support software and administrative console.

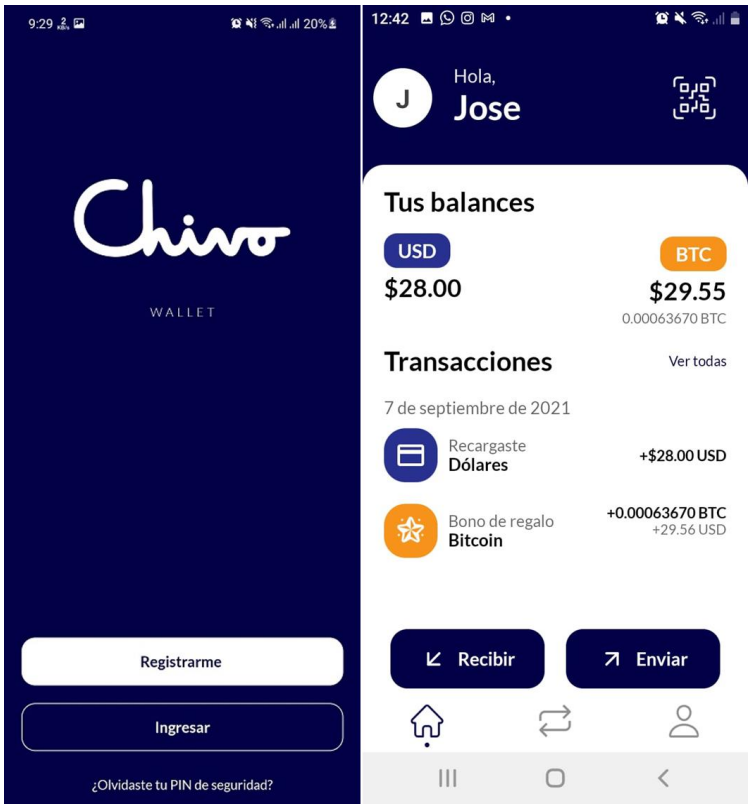


Figure 15 - Chivo demonstration [79]

[20] When migrants send home part of their earnings in the form of either cash or goods to support their families.

5.4.4 Phoenix

Phoenix [80] is a non-custodial Bitcoin wallet developed by ACINQ. It provides a simple and clean user experience. Thanks to native Lightning support provided by their own Lightning Network implementation, payments are even faster. This is possible since they only allow payments through their own ACINQ nodes, generating less routing meaning faster transactions, having some drawbacks mentioned below.

Additional Phoenix features:

- Trampoline payments
    - Phoenix has specific nodes just to find the optimal path from the sender to the receiver.
- Turbo channels
- Back up by providing the email address

On the downside, along with user complaints about high fees, Phoenix in April of 2022, introduced a 1% fee on channel creations. Another drawback is that connecting to their own nodes means that the node management is always dictated by ACINQ and there is no competition against other nodes. Their justification is to put it briefly, trust. With their own nodes they can detect cheating attempts with more ease, giving them also more trust when opening turbo channels for example.

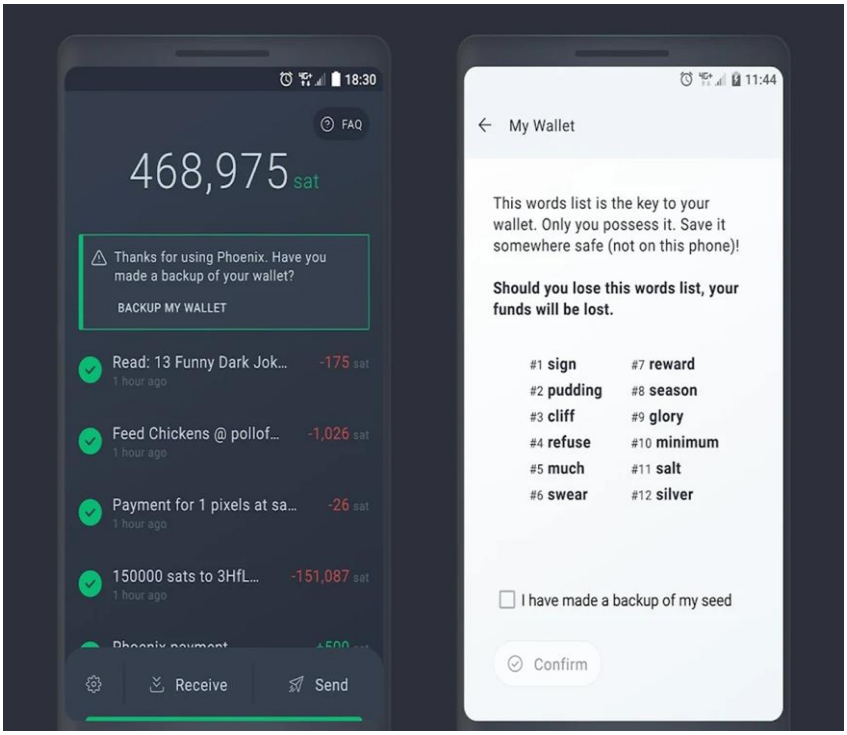


Figure 16 - Phoenix demonstration [81]

5.4.5 Spark

Spark [82] is a minimalistic wallet interface for Core Lightning, accessible over the web or through mobile and desktop apps (for Android, Linux, macOS and Windows). It is currently oriented for technically advanced users and is not an all-in-one package, but rather a "remote control" interface for a Core Lightning node that has to be managed separately.

Spark is a purely off-chain wallet, with no on-chain payments. This allows Spark to fully realize the user experience enabled by Lightning, without worrying about the complications and friction of on-chain.

Additional features:

- BOLT #12 Implementation: Offer Protocols for Lightning Payments
- Tor Hidden Service
    - Allows the user to run their node behind Tor Network and even generate an address for others to connect to open channels.
- Manual Channel Management
    - Users can create channels, close channels and more.

One clear drawback of this wallet is that it is targeted for experienced users since the user has already to have a node hosted somewhere else, to then be able to manage it using the Spark wallet.

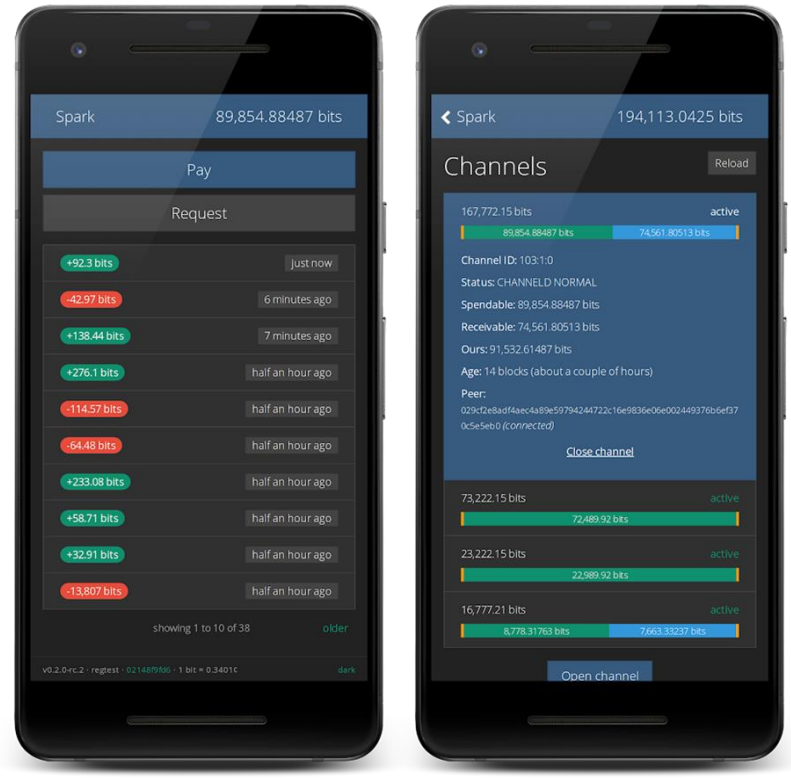


Figure 17 - Spark demonstration [83]

5.5 Wallet features summary

As evidenced before, there are a group of core features that exist in almost every wallet making them essential to exist in the application prototype to be developed. In addition to these features, some features can also be added to increase the value of the application. These features can go from the possibility to backup the wallet of the user to a more technical feature like manual management of payment channels.

The wallet that will be prototyped will cover all core features. Some additional features that will be detailed later, two of them being: the possibility to backup the wallet so it can be imported when needed and an introductory slide of screens into the application to familiarize the user. The combination of all these features will give the prototype the ability to compete with existing wallets in the market.

6 PROBLEM ANALYSIS AND REQUIREMENTS

The absence of native support for smart contracts means that the Bitcoin Blockchain, by itself, is not able to compete with the rich ecosystem of decentralized applications based on Ethereum or Solana (decentralized Blockchains with smart contract functionality). Furthermore, the Bitcoin Blockchain has grown to a point where the average time of transaction registration in the Blockchain is around 10 minutes due to its low number of transactions per second (TPS) registered, 5 to 10, compared to Solana 65.000 TPS [84]. Due to this and the rising number of Bitcoin transactions, the scalability problem grows by the day. High fees are also a big problem making Bitcoin impractical for microtransactions.

Similar to what happened with other Blockchains, Bitcoin has also seen its growth expanded with the creation of layer 2 chains, namely the Lightning Network, in order to fix this problem.

Considering this problem, mainly introduced in chapter 2, and the proposed solution, explained in detail in chapter 3, general goals and requirements were defined in the following subchapters that will allow the development of a prototype that can make microtransactions with Bitcoin almost instantly with low to no fees.

6.1 General goals

To address the problems described above, general goals were defined. By developing the prototype application using the Lightning Network, it will be possible to make Bitcoin transactions in near-instant time and with extremely low or non-existent fees.

While the transaction speed and low transaction fees are extremely important, the complexity associated with Blockchain wallets requires additional features to ensure a pleasant experience for the end-user:

- Create/register wallet
- User authentication
- Check current balance
- Check transaction history

Steps will be taken to improve the user experience, including the security and usability of the wallet. Other features like wallet PIN security and user onboarding will be incorporated to set this prototype apart from others now available on the market.

Functional and non-functional requirements were defined by the intern to meet all these goals, as showed in the following section.

6.2 Requirements

In this subchapter the specification and analysis of the requirements is shown. The functional requirements are prioritized according to the MoSCoW [85] scale:

- **Must Have:** it is a critical requirement for the success of the system.
- **Should Have:** it is a less critical requirement than the previous one, however, should be present in the final version of the system.
- **Could Have:** is a requirement whose implementation is not mandatory, but brings value to the system, increasing customer satisfaction. In case it is implemented, it will only be after the requirements with previously described priorities are completed.
- **Won't Have:** is a requirement that will not be included in this version from the project.

6.2.1 Functional requirements

Attending to the general goals of the application, every functional requirement with its prioritization is showed below, in table 2.

Functional Requirement	**Prioritization**
FR1 - Create and import a wallet	Must Have
FR2 - Transaction history	Must Have
FR3 - Check balance	Must Have
FR4 - Send payment	Must Have
FR5 - Receive payment	Must Have
FR6 - User onboarding	Should Have
FR7 - User authentication	Should Have
FR8 - User settings	Could Have
FR9 - Export wallet backup to pdf	Could Have
FR10 - Support for different languages	Could Have

Table 2 - Prototyped wallet function requirements prioritization

Next, is a summary of each functional requirement need outlining its primary goal within the wallet. A more detailed functional requirement specification, including acceptance criteria, is presented in appendix B.

FR1 - Create and import a wallet: The aim of this requirement is to allow users to create a wallet themselves and import an existing one to be able to send and receive money, check their transaction history, balance and more.

FR2 - Transaction history: The aim of this requirement is to allow users to easily check their transactions and filter them by date or description.

FR3 - Check balance: The aim of this requirement is to allow users to easily check their current balance.

FR4 - Send payment: The aim of this requirement is to allow users to easily send fast Bitcoin transactions without worrying about high fees or delayed confirmation of payment taking advantage of the Lightning Network.

FR5 - Receive payment: The aim of this requirement is to allow users to receive fast Bitcoin payments with low fees taking advantage of the Lightning Network.

FR6 - User onboarding: The aim of this requirement is to allow the user to be introduced to the application in a series of steps about the Lightning Network and its functionalities.

FR7 - User authentication: The aim of this requirement is to allow the user to have a PIN to access the wallet.

FR8 - User settings: The aim of this requirement is to allow users to easily change their wallet name, language, default displayed currency and delete wallet.

FR9 - Export wallet backup to pdf: The aim of this requirement is to allow users to have a backup of their wallet key to a pdf document, making it easier to recover in the future if needed.

FR10 - Support for different languages: The aim of this requirement is to enable the user to choose the display language of the application.

6.2.2 Non-functional requirements

Non-functional requirements are not directly linked to the wallet functionalities, but they are equally important, as they represent the characteristics that the system must support, which can sometimes impose restrictions on functional requirements.

The non-functional requirements specified for the application prototype are defined in the table below. These are also prioritized by the MoSCoW scale, the same used in the prioritization of functional requirements.

Non-Functional Requirement	**Prioritization**
NFR1 - Usability	Must Have

NFR2 - Extensibility	Must Have
NFR3 - Security	Could Have

Table 3 - Prototyped wallet non-function requirements prioritization

6.2.2.1 NFR1 - Usability

Usability concerns the specification of the interface presented to the user, as well as ease of use and iteration. As a result, the development of the wallet should take into account Nielson's heuristics [86], presented below:

1. **System status visibility**
    a. The design should always keep users informed about what is going on, through appropriate feedback within a reasonable amount of time.
2. **Correspondence between the system and the real world**
    a. The design should speak the users' language.
3. **Consistency and standards**
    a. Follow platform and industry conventions.
4. **Error prevention**
    a. Good error messages with designs carefully planned to prevent problems from occurring in the first place.
5. **Recognize instead of remembering**
    a. Minimize the user's cognitive load by making elements, actions, and options intuitive.
6. **Flexibility and ease of use**
    a. Shortcuts may speed up the interaction for the expert user such that the design can cater to both inexperienced and experienced users.
7. **Aesthetics and minimalist design**
    a. Interfaces should not contain information that is irrelevant or rarely needed.
8. **Help users recognize, diagnose, and recover from errors**
    a. Error messages should be expressed in plain language (no error codes), precisely indicate the problem, and constructively suggest a solution.

To assess the usability of the system, usability tests were carried out on the prototyped wallet, to validate the heuristics listed above. These tests are detailed in chapter 10.

6.2.2.2 NFR2 - Extensibility

Although the wallet is a prototype, it must be prepared to be able to accommodate possible changes in the future, either to add new features or to change existing ones. For this, the wallet architecture must be taken into account so that extensibility is possible. Repository pattern is mentioned in detail in subchapter 8.1, aiming to fulfill this requirement.

6.2.2.3 NFR3 - Security

Although the wallet uses Blockchain technology and the Lightning Network which are considered as secure, as explained in chapter 2 and 3, there are some additional security considerations that need to be addressed. The following security aspects are independent of Blockchain technology and must be addressed separately:

- A PIN to unlock the wallet will be used
- User PIN will be stored using encryption
- User data will be stored in an encrypted database

7 TECHNOLOGIES AND TOOLS

In this chapter, the tools and technologies chosen for the development of the project are presented, namely the programming language used, mobile system chosen, frameworks, integrated development environment (IDE) and management and development tools.

7.1 Technologies

In terms of technology, Android was selected as the operating system on which the program will run, while Android Studio was used to develop the entire mobile application. The language of choice was Kotlin and Retrofit 2 was used to make application programming interface (API) requests to third-party servers.

7.1.1 Android

Android [87] is an open-source mobile operating system that is primarily made for touchscreen mobile devices like smartphones and tablets. It is based on a modified version of the Linux kernel. Since 2011 and since 2013 for tablets, Android has been the most popular OS globally. The Google Play Store has over 3 million applications as of January 2021, and as of May 2021, it had the biggest installed base of any operating system with over 3 billion monthly active users.

Android was chosen since there was a possibility to choose between iOS and Android, with the intern having already some background in Android development, it was an easy choice.

7.1.2 Kotlin

Kotlin [88] is a cross-platform, statically typed[21] programming language with type inference[22] that can be used to create software in a wide range of application domains. It emphasizes on interoperability, safety, clarity, and tooling support.

On May 7, 2019, Kotlin replaced Java as Google's preferred language for Android application development [89]. Since the October 2017 release of Android Studio 3.0, Kotlin has been added as a substitute for the default Java compiler.

Kotlin has several advantages over Java [90]. The following are the most relevant:

[21] Process of verifying the type safety of a program.

[22] Automatic detection of the type of an expression in a formal language

- **Readability**
  - Kotlin has more readable and precise code than Java, which makes the program easier to understand. A Java developer can easily learn how to write Kotlin after a short learning curve.
- **Null Safety**
  - Null pointer exceptions, also referred to as "The billion-dollar mistake" by Tony Hoare [91], is among the most frequent mistakes that lead to applications crashing if Java is being used. Null safety is the default in Kotlin. It prohibits assigning variables with null values.
- **Interoperability**
  - Kotlin language is interoperable. As a result, Kotlin projects can use Java commands, Java libraries, and Java frameworks.
- **Immutability**
  - An object is said to be immutable if its state cannot be changed after it has been created. To make it simple for developers to understand which values can be changed, Kotlin variables are defined using the *val* or *var* keywords.

7.1.3 Retrofit 2

Retrofit [92] is a type-safe representational state transfer (REST) client for Android, Java and Kotlin developed by Square. The newest library version, Retrofit 2, provides a powerful framework for authenticating and interacting with APIs and sending network HTTP requests[23] with OkHttp. OkHttp is a third-party library that was also introduced by Square for sending and receiving HTTP-based network requests [93].

This library greatly simplifies downloading JSON[24] and XML[25] data from a Server. After making an HTTP request to the chosen server using Retrofit 2 REST framework, an HTTP response is then received from the server when ready. Retrofit 2 will then analyze the JSON or XML data received and convert it to a Data Class[26] making it easy to programmatically use the data received from the server. Retrofit 2 was used to make requests to the used third-party servers in the prototype.

[23] The aim of an HTTP request is to access a resource on the server.

[24] Data interchange format storing attribute-value pairs and arrays.

[25] File format for storing data using a set of defined rules.

[26] Classes specialized in holding data with similar format to JSON.

7.2 Development tools

At the implementation level, Android Studio was used to develop the whole mobile application.

7.2.1 Android Studio

Android Studio [94] is the official IDE for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development.

The following features are some of the many that exist in the current stable version:

- Gradle-based build support (build automation tool for multi-language software development)
- Android-specific refactoring and quick fixes
- Lint tools to catch performance, usability, version compatibility and other problems
- Android Virtual Device (Emulator) to run and debug applications in Android studio

7.3 Management tools

At the company level, Jira was used to help manage the project's tasks and GitLab to keep under control the various versions of software developed throughout the project. Confluence was used to manage project-related documentation. An example of such documentation is the UI drafts (appendix D).

7.3.1 GitLab

GitLab [95] is an open-source repository based on Git, commonly used in collaborative software development, it offers an online repository accompanied by a set of key functionalities that help in the management of the project, which includes version control and branching. GitLab is similar to the well-known GitHub but allows storing code on its own servers instead of third-party servers as is the case with GitHub.

Version control allows for developers to keep the project organized, but also in case of unexpected errors makes reversion to an older stable version easier while branching provides the freedom for the developers to engage in work, testing new functionalities without worrying about contamination of the main project.

7.3.2 Jira

Jira, developed by Atlassian [96], is a business process management tool used by agile teams to plan, track, and release software, supporting Scrum, Kanban, hybrid models, and

other workflows. With Jira, it is possible to create action plans to map all ongoing projects. The project board has a "drag and drop" interface that allows the user to manage each project in detail. In addition, the system has functionality to create user stories and items, plan sprints, and distribute tasks to the entire team. Users also have access to information from thousands of enterprise applications, from design and monitoring tools to source code and productivity applications.

As referenced, Jira was used to plan the project timeline following the already presented SCRUM methodology. Each feature was subdivided, when justified, into smaller sub-categories and given an estimate of completion in days using the Fibonacci sequence [97]. After the estimation was made, the features were then divided into weekly sprints according to time planned in each one, as seen in appendix C.

7.3.3 Confluence

Confluence [98], also developed Atlassian, by lets users organize and find the information needed. Group related pages together in a space dedicated to work, team, or cross-functional projects. Depending on permissions, a Confluence space can be accessed by just the user or the entire company. Another feature, page trees, create a hierarchical list of pages within a space, highlight topics on main pages, and help organize work.

Since there was no availability from WIT design team, Confluence was used to store the draft for the UI based on existing solutions in the market, as seen in appendix D.

8 ARCHITECTURE AND UI DESIGN

This chapter provides a high-level overview of the application prototype. The Architecture uses the Repository Pattern, which is explained first. Then, the decisions regarding the prototype UI design are given together with a group demonstration screens.

8.1 Repository pattern

The Repository pattern [99] was the chosen design pattern for the wallet architecture. This is the recommended app architecture by Google [100]. This pattern combines the advantages of Model View View-Model (MVVM) architecture with a much needed "Single Source of Truth" (SSOT). SSOT is a practice followed to centralized data access into a single class. SSOT was added since even following the MVVM pattern, some code duplication may still appear, especially when accessing data from a local or remote database.

MVVM is used in this pattern since it overcomes all drawbacks of Model View Presenter (MVP) and Model View Controller (MVC) design patterns. MVVM suggests separating the data presentation logic (Views or user interface) from the core business logic part of the application.

The components of MVVM are:

- **Model:** This layer is responsible for the abstraction of the data sources. ViewModel and Model work together to get and save the data.
- **View:** This layer serves to inform the ViewModel of the user's action. This layer simply observes the ViewModel without any application logic.
- **ViewModel:** It acts as a link between the Model and the View. It is in charge of transforming the Model's data. It provides data streams to the View. Callbacks are also utilized to update the View. It will request the data from the model.

The Repository design pattern facilitates de-coupling of the business logic, and the data access layers in the application with the former not having to have any knowledge on how data persistence would take place. The repository component will decide from where to get the data to provide to the ViewModel.

The main difference between the recommended Repository pattern and the used pattern, is that, due to the lack of time, the data access wasn't made using a database but using the own phone storage. Although the recommended pattern is used by developers, usually there are deviations from it, adding or altering layers in the recommended pattern.

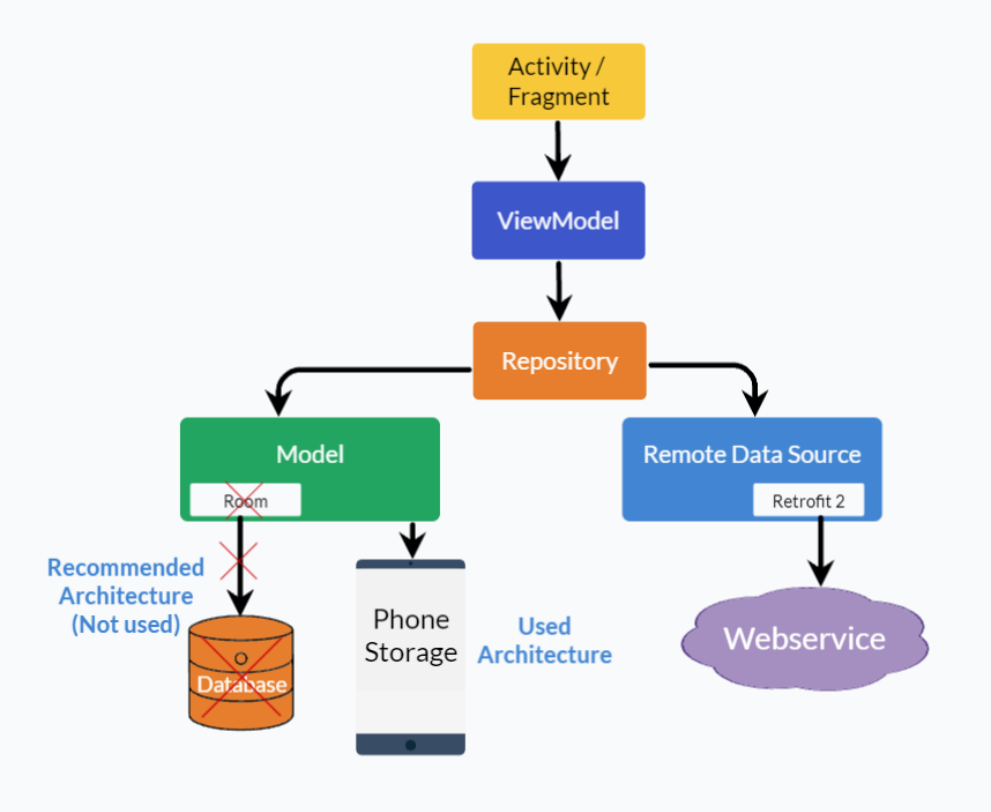


Figure 18 - Wallet prototype repository design pattern

Following this pattern, due to the result of modular code, well separated logic from the user interface and centralized access to the necessary data, adding new features or change existing ones becomes an easy process making this prototype easily extensible, fulfilling the extensibility non-functional requirement.

8.2 Wallet prototyped UI design

The application prototype followed M-Pesa design guidelines since this application was developed as a possible extension for M-Pesa.

In terms of the application design mockups, the WIT UI/UX design team wasn't available to provide the application design. Initially, a draft was developed using Confluence, taking into consideration existing wallet solutions (appendix D). Afterwards, every feature implemented was discussed in the weekly meeting, contemplating possible design options with Business Analyst Mylena Dias (WIT Software), and with Lead Engineer João Sousa (WIT Software).

Although the WIT design team wasn't available, having access to the design made for M-Pesa, design patterns used in M-Pesa were followed, such as: overall Android screen layout, buttons layouts, color schemes and letter fonts. These similarities are evident in the below wallet comparison between Figure 19,20,21 (M-Pesa) and Figure 22,23,24 (Prototype).

Figure 19 - M-Pesa pin screen

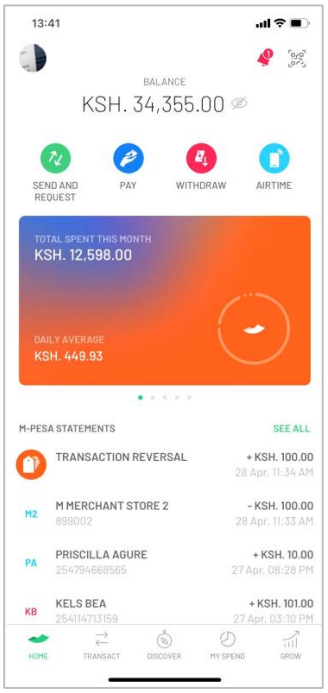


Figure 20 - M-Pesa home page screen

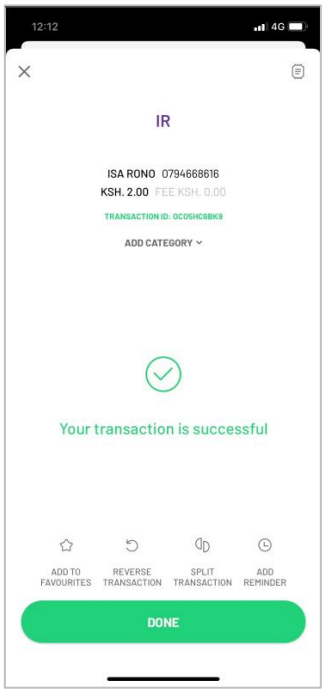


Figure 21 - M-Pesa payment confirm screen

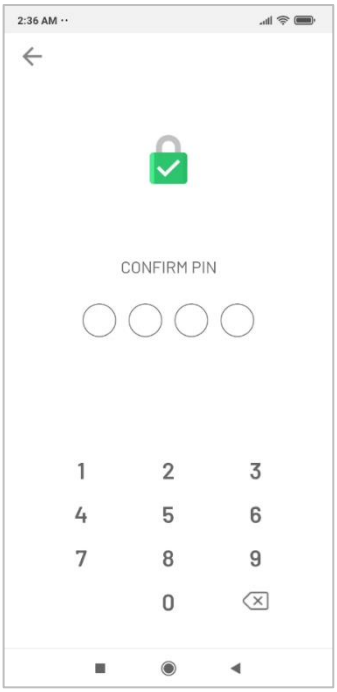


Figure 22 - Prototype pin screen

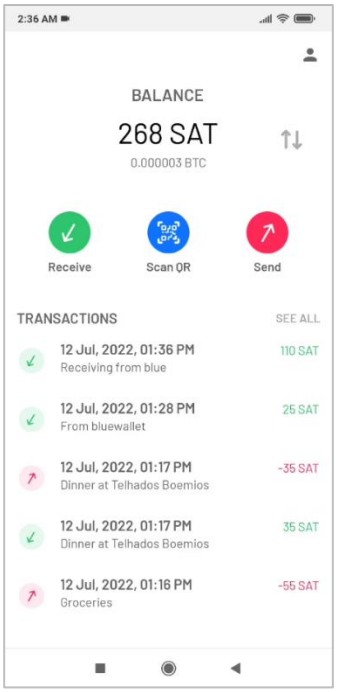


Figure 23 - Prototype home page screen

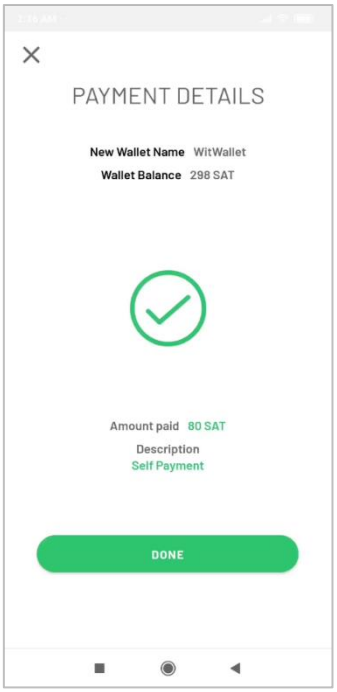


Figure 24 - Prototype payment confirm screen

9 IMPLEMENTATION

The wallet uses a public server provided by LNbits to interact with both Bitcoin Blockchain and the Lightning Network, delegating the internal details of this interaction to this server. A public server of Coingecko is also used to obtain the updated price of Bitcoin needed to make conversions from Bitcoin to Euro, a wallet feature. The wallet interacts with the LNbits and Coingecko server using HTTP requests following the server protocol, as explained in the Coingecko and Lightning Network use case below.

Other implementation details will also be mentioned including the use of encryption and libraries[27] to read inputs from the user and to generate QR codes.

9.1 Coingecko use case

The Coingecko server is used to convert the balance from Bitcoin to Euro. Because the Bitcoin price is highly volatile, converting the balance from Bitcoin to Euro requires the periodic checking of its value.

The interaction between the wallet and the Coingecko server, showed in Figure 25, is as follows:

1. User opens the wallet
2. An HTTP request is made to obtain the Bitcoin price
3. The wallet prototype backend receives the HTTP response from the Coingecko server with the Bitcoin price updated
4. The wallet prototype backend now has the updated Bitcoin prince allowing to switch between currencies at any moment

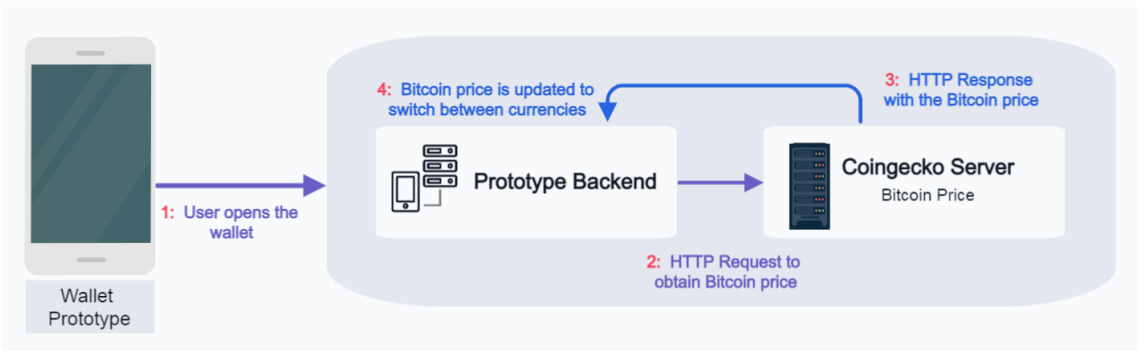


Figure 25 - Coingecko use case

[27] A collection of non-volatile resources used by computer programs.

This request is made at different times during the application use, including, for example, before creating a new invoice request, since it is possible to input the amount requested in the invoice in Euro, Satoshi[28] and Bitcoin.

9.2 Lightning Network use case

To better understand the flow of the architecture when making a request related to the Lightning or Bitcoin Network, using the LNbits server, an example of an invoice creation will be given.

The interaction between the wallet and the LNbits server, showed in Figure 26, is as follows:

1. User requests a creation of a new invoice with its amount and description
2. An HTTP request is made to the LNbits server to create the new invoice with the details received from the user
3. The LNbits server creates the invoice in the Lightning Network
4. Invoice is created and the LNbits server obtains the generated payment details
5. These invoice payment details are then sent back to the prototyped wallet backend through an HTTP response
6. After receiving the invoice details, the wallet communicates the necessary information to the user so he can share a QR code or a unique string allowing the invoice to be paid

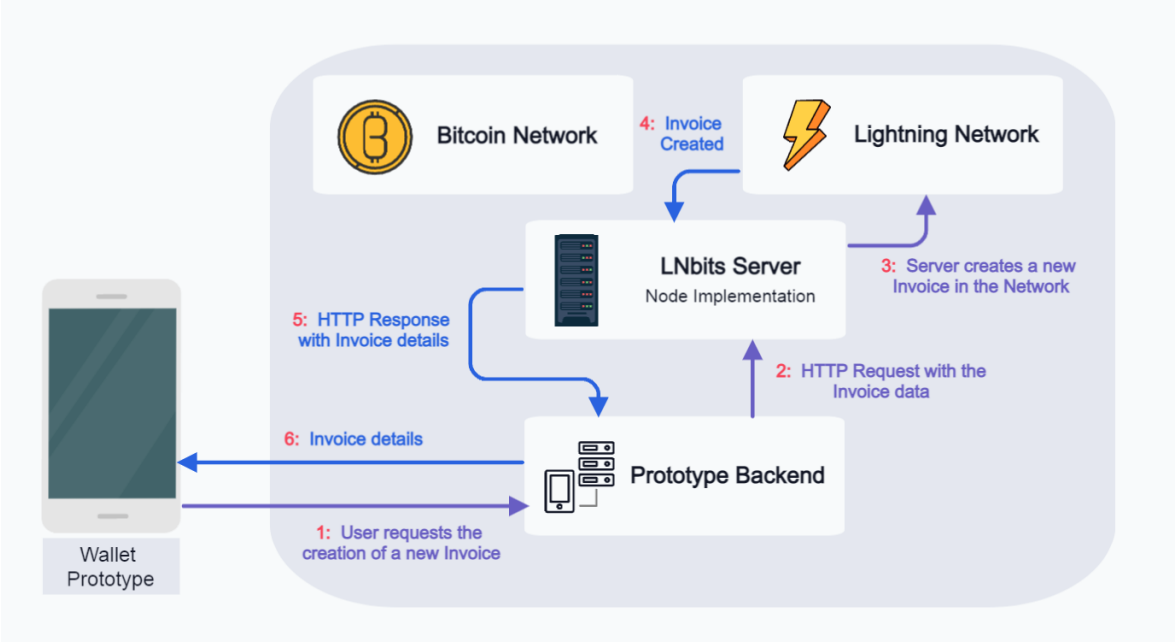


Figure 26 - Lightning Network use case

[28] Represents one hundred millionth of a single Bitcoin (0.00000001 BTC).

Although the presented use case is from an invoice creation, any other request related to the Lightning and Bitcoin Network have many similarities with this use case. The main change is the responsibility delegated to the LNbits server, which isn't the prototyped wallet concern.

9.3 PIN number pad

To create the necessary screens with a number keypad to write the wallet pin, a library was used to make it simpler to implement. Made by Github user Yoann Goular, the *numpadview* library [101] allows custom implementations such as changing the font of the numbers and icons on bottom right or left of the number pad.

The main reason of choosing this library is how easy it is programmatically to detect the number keypad pressed with a provided custom listener that detects every click on the keypad, including bottom left and right icon clicks. This library cuts on lengthy manual code that would have to be done to implement the same number keypad manually.

9.4 PIN encryption

Ideally, the wallet data would be stored in an encrypted database. But that is not a possibility in this project quantity of requirements prioritized before this one. Although the PIN is saved in the phone storage, encryption measures were taken, using symmetric encryption.

Symmetric encryption is based on the Advanced Encryption Standard (AES) [102]. The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information. It uses a secret key[29] to encrypt plain data as displayed in Figure 27, and uses 128-, 192-, and 256-bit keys to process 128-bit data locks. The same secret key is used to decipher the data with a decryption algorithm. Both decryption and encryption algorithms are showed in Figure 28.

[29] Normally a string of letters and numbers.

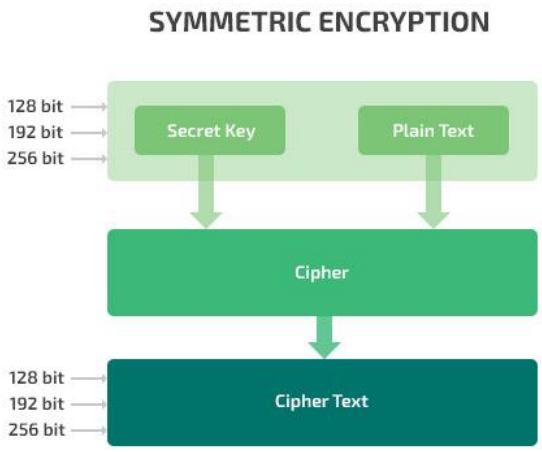


Figure 27 - Symmetric encryption example [103]

```
// Encryption algorithm
fun String.encryptCBC(): String {
    val iv = IvParameterSpec(SECRET_IV.toByteArray(charset( charsetName: "UTF-8")))
    val skeySpec = SecretKeySpec(SECRET_KEY.toByteArray(charset( charsetName: "UTF-8")),  algorithm: "AES")
    val cipher = Cipher.getInstance( transformation: "AES/CBC/PKCS5PADDING")
    cipher.init(Cipher.ENCRYPT_MODE, skeySpec, iv)
    val encrypted = cipher.doFinal(this.toByteArray())
    return Base64.encodeToString(encrypted, Base64.DEFAULT)
}

// Decryption algorithm
fun String.decryptCBC(): String {
    val iv = IvParameterSpec(SECRET_IV.toByteArray(charset( charsetName: "UTF-8")))
    val skeySpec = SecretKeySpec(SECRET_KEY.toByteArray(charset( charsetName: "UTF-8")),  algorithm: "AES")
    val cipher = Cipher.getInstance( transformation: "AES/CBC/PKCS5PADDING")
    cipher.init(Cipher.DECRYPT_MODE, skeySpec, iv)
    val original = cipher.doFinal(Base64.decode( str: this, Base64.DEFAULT))
    return String(original)
}
```

Figure 28 - Encryption and Decryption algorithm snippet

9.5 QR Codes

To be able to scan and decode QR codes with the camera, the *ZXing* barcode scanning library [104] was used. This library actively searches for QR codes when the camera is open, as showed in Figure 29, returning the decoded QR code when it finds one.

Additional settings can be used when customizing this library, some of them being:

- Choosing which camera to use
- Continuously scan or to scan only the first scannable QR code
- Enable or disable camera auto focus
- Enable or disable flash

Since it uses the phone camera, it requires user permission to use it, as displayed in Figure 30.

To create a new QR code, a custom function displayed in Figure 32 was used in which receives as arguments, the width and height of the QR code, along with the string of data to be encoded, returning a two-dimensional picture, as a rectangular matrix or grid of square pixels of the QR code. This picture is then presented in the prototype screen, as showed in Figure 31.

Figure 29 - Prototype QR code scan screen

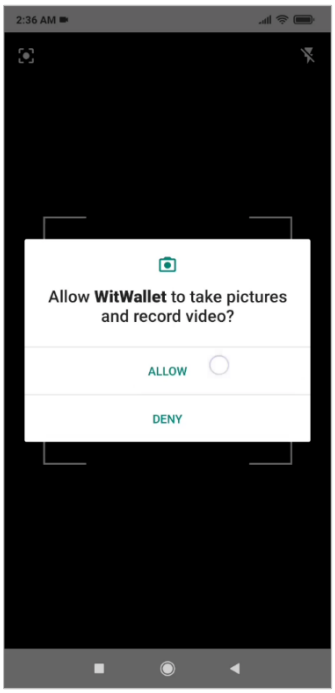


Figure 30 - Prototype camera permissions

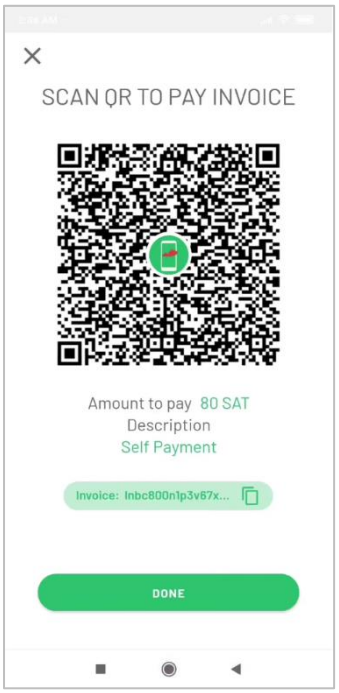


Figure 31 - Prototype payment details screen

```
fun generateQRbitmap(qrCode: String, qr_width: Int, qr_height: Int): Bitmap? {
    var bitmap: Bitmap? = null
    try {
        val result : BitMatrix? = MultiFormatWriter().encode(qrCode,
            BarcodeFormat.QR_CODE,qr_width,qr_height,mapOf(EncodeHintType.MARGIN to 0))
        if(result != null){
            bitmap = Bitmap.createBitmap(result.width, result.height, Bitmap.Config.RGB_565)
            for (x in 0 until result.width) {
                for (y in 0 until result.height) {
                    bitmap.setPixel(x, y, if (result.get(x, y)) Color.BLACK else Color.WHITE)
                }
            }
        }
    }catch (e: WriterException){
        Log.i(TAG_Frag_ReceivePayment, msg: "createQRcode: Exception: ${e.printStackTrace()}")
    }
    return bitmap
}
```

Figure 32 - QR code generation algorithm snippet

10 VALIDATIONS

This chapter presents the validations of the functional and non-functional requirements.

10.1 Functional requirements

Weekly meetings were held with Business Analyst Mylena Dias (WIT Software), and with Lead Engineer João Sousa (WIT Software) where the fulfillment of the requirements was confirmed through brief demonstrations.

All requirements with high priorities (Must Have and Should Have) have been implemented. Due to limited development time, none of lower priority (Could Have) were implemented. Table 4 gives an overview of the implemented functional requirements.

Functional Requirement	**Implemented**
FR1 - Create and import a wallet	Yes
FR2 - Transaction history	Yes
FR3 - Check balance	Yes
FR4 - Send payment	Yes
FR5 - Receive payment	Yes
FR6 - User onboarding	Yes
FR7 - User authentication	Yes
FR8 - User settings	No
FR9 - Export wallet backup to pdf	No
FR10 - Support for different languages	No

Table 4 - Prototyped implemented functional requirements

10.2 Non-functional requirements

This section demonstrates firstly how the usability non-functional requirements included in the project were validated. When it comes to the extensibility non-functional requirement, it has already been validated by following the Repository design pattern (Section 8.1). Some security measures were taken, but it was not possible to fulfil all security requirements. Table 5 gives an overview of the implemented non-functional requirements.

Functional Requirement	Implemented
NFR1 - Usability	Yes
NFR2 - Extensibility	Yes
NFR3 - Security	Partially

Table 5 - Prototype implemented non-functional requirements

10.2.1 Usability

Usability tests were carried out to evaluate the behavior of this non-functional requirement. The following table, Table 6, presents the heuristic identifier, represented in section 6.2.2.1, as well as a brief description of what was done to fulfill it.

Heuristic ID	Description
1	Every screen displays an explanatory title of the screen displayed (Figure 33, 34, 35).
2	All application icons are illustrative of the action aimed at match, as well as colors associated with receiving Bitcoin (green) and sending Bitcoin (red). A practical example is the home page in Figure 23.
3	Interfaces are consistent across the application as well as all the information that is displayed (Figure 33, 34, 35).
4	To prevent possible errors, the fields introduced by the user are validated, where feedback is given to the user (Figure 37). There are also hints on what to fill in this fields (Figure 35).
5	The interfaces were designed in such a way that a user does not need to memorize the actions they take, but rather recognize how to do them, through their simplicity (Figure 34, 35, 36).
6	Android “go back” functionality can always be used to go back to the previous screen in the prototype.
7	The application presents a design suitable for the intended purpose, taking the focus to the actions that really matter. This was possible by following M-Pesa design guidelines.
8	The feedback given to the user is explanatory, so that the user can recognize the error and, if possible, correct it (Figure 37).

Table 6 - Prototype usability tests

The following figures complement the descriptions given in Table 5.

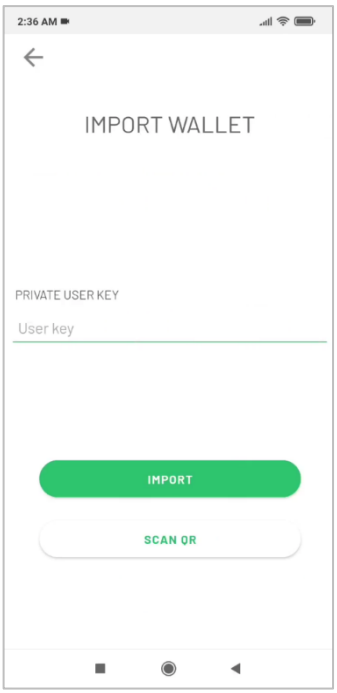


Figure 33 - Prototype import screen

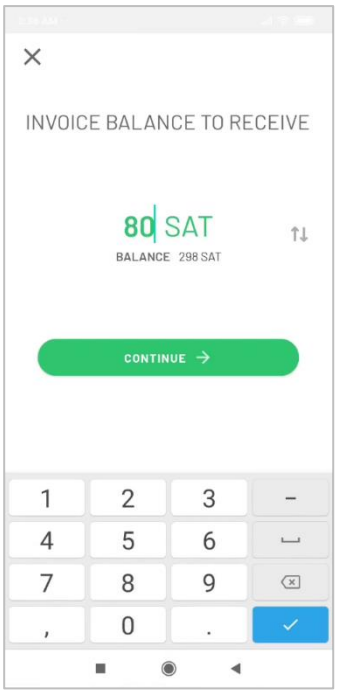


Figure 34 - Prototype scan invoice request

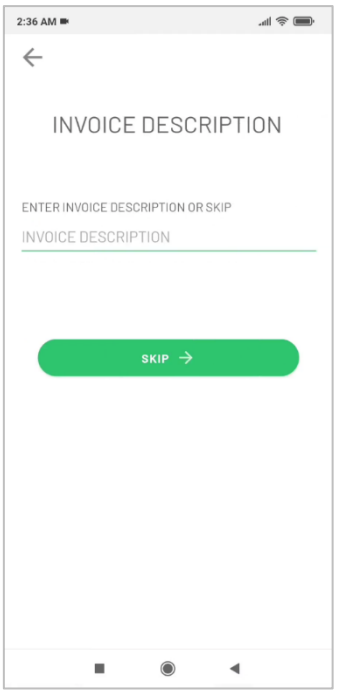


Figure 35 - Prototype invoice description

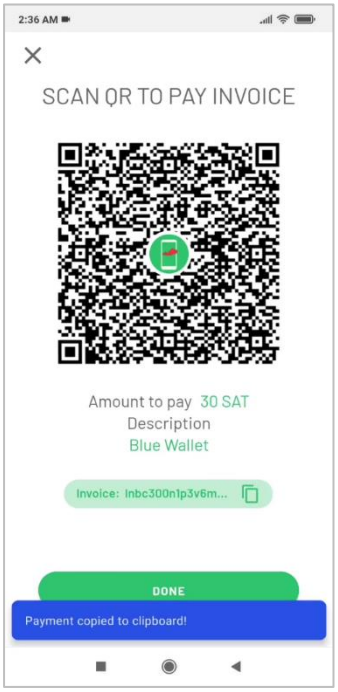


Figure 36 - Prototype payment copied feedback

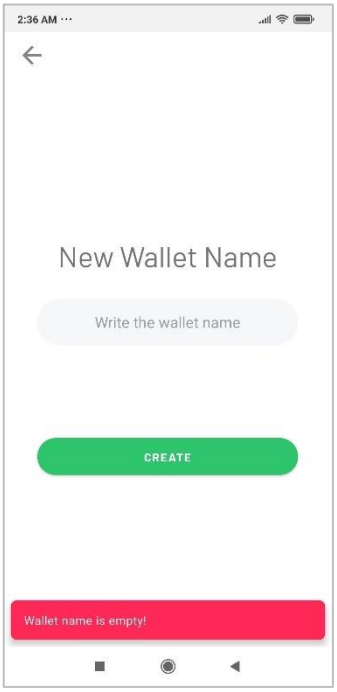


Figure 37 - Prototype create wallet empty name

10.2.2 Security

All requirements with high priorities (Must Have) have been implemented. Due to limited development time, the lower priority (Could Have) requirement was only partially implemented.

Using the Bitcoin and the Lightning Network, making and receiving payments is considered secure by many different reasons as mentioned before. Although, as stated in subsection 6.2.2.3, different aspects should be considered. The following table, Table 7, shows which aspects were followed.

Security Requirement	**Implemented**
A PIN to unlock the wallet	Yes
User PIN will be stored encrypted	Yes
User data will be stored in an encrypted database	No

Table 7 - Prototype followed security requirements

The prototyped wallet has the PIN functionality has showed in previous figures, for example, Figure 22. The PIN is also encrypted using symmetric encryption, explained in sub-chapter 9.2.3.

11 CONCLUSIONS AND FUTURE WORK

The internship's two primary objectives were accomplished with the prototyped application having more features than expected. This chapter presents the conclusions of the work carried out and the value that brought to WIT, followed by a brief presentation of the possible evolution of the solution created.

11.1 Personal growth and internship goals

The contact, for the first time, with concepts about Blockchain, Bitcoin and the Lightning Network constituted an initial challenge, having been successfully overcome. Overcoming this initial challenge was essential to achieve the first objective of the internship, making it possible to carry out a detailed analysis of the characteristics of the Lightning Network together with the identification of its advantages and limitations.

The entire technical process of making an Android application, while adhering to industry-recommended development standards that WIT Software uses, allowed me to grow my own technical skills and achieve the second main goal, which was, the creation of a prototype application that enables users to manage their Lightning Network wallet. The two main objectives of the project have been accomplished, considering the validation of the requirements as well.

This entire process was supported by well thought-out planning combined with a methodology that allowed all the work to be well structured and closely monitored by those interested in the product, allowing it to evolve over time, through constant feedback from WIT advisor, tutor, and business analyst.

The first interaction with some of the technologies used in the development of the project also contributed to my professional enrichment, being now more comfortable with the practices used in the company and in the job market.

From a personal perspective, this internship was an enriching experience that greatly contributed to my personal growth on a social level by establishing new relationships with the team and other WIT employees.

11.2 Internship value

The company improved its understanding of this new technology through the creation of this project, enabling WIT to evaluate the viability of potential investments in the future. The developed prototype can also be used as a possible extension for M-Pesa expanding its features with almost instant and feeless payments using Bitcoin. Additionally, the prototyped wallet can be made profitable by WIT if a private server is substituted for the

current public LNbits server. WIT will then receive the routing fees produced by the lightning server node.

11.3 Future work

Although the two main goals of the internship were achieved, there is always room for improvement. For instance, by implementing unfulfilled functional and non-functional requirements of the prototyped wallet. Therefore, the following potential future features could be included:

- **User settings**
    - Create a user private area where he can easily change their wallet name, language, default displayed currency and delete wallet.
- **Export wallet backup to pdf**
    - Allow the user to have a backup of their wallet key to a pdf document, making it easier to recover funds in the future if needed.
- **Wallet storage in an encrypted database**
    - Improve security by switching current storage to an encrypted database.
- **Private LNbits server**
    - Stop relying on a third-party service by creating a server based on LNbits for wallet management and extended functionalities. It also allows WIT Software to collect routing fees, monetizing the prototyped wallet.
- **New wallet functionalities**
    - As referenced in other wallets, adding new features will always benefit the wallet, those being the possibility for submarine swaps and BOLT #12 implementation, for example.

REFERENCES

[1] ISEC, "Bachelor in Informatics Engineering," [Online]. Available: https://www.isec.pt/PT/estudar/licenciaturas/EngInfor/. [Accessed 05 04 2022].

[2] ISEC, "ISEC," [Online]. Available: https://www.isec.pt/PT/default.aspx. [Accessed 05 04 2022].

[3] WIT Software, "WIT Software," [Online]. Available: https://www.wit-software.com/. [Accessed 05 04 2022].

[4] WIT Software, "About," [Online]. Available: https://www.wit-software.com/about/. [Accessed 05 04 2022].

[5] WIT Software, "Expertise," [Online]. Available: https://www.wit-software.com/expertise/. [Accessed 05 04 2022].

[6] W. Software, "RCS," [Online]. Available: https://www.wit-software.com/rcs/. [Accessed 05 04 2022].

[7] WIT Software, "Buzz," [Online]. Available: https://www.wit-software.com/buzz/. [Accessed 05 04 2022].

[8] WIT Sofware, "WCP," [Online]. Available: https://www.wit-software.com/wcp/. [Accessed 05 04 2022].

[9] ISEC, "Presentation," [Online]. Available: https://www.isec.pt/pt/instituto/#lnkApresentacao. [Accessed 05 04 2022].

[10] ISEC, "Facts and Numbers," [Online]. Available: https://www.isec.pt/pt/instituto/#lnkFactosNumeros. [Accessed 05 04 2022].

[11] Scrum, "Scrum," [Online]. Available: https://www.scrum.org/resources/what-is-scrum. [Accessed 10 04 2022].

[12] Investopedia, "What Is a Blockchain," [Online]. Available: https://www.investopedia.com/terms/b/blockchain.asp. [Accessed 10 04 2022].

[13] L. Bassotto, "Entendendo o que é Bitcoin e qual é a sua utilidade," Investificar, [Online]. Available: https://www.investificar.com.br/o-que-e-bitcoin-e-como-funciona/. [Accessed 10 04 2022].

[14] D. H. S. S. W. Bayer, "Sequences II: Methods in Communication, Security, and Computer Science," Springer, New York, NY, 1993.

[15] Wikipedia, "Blockchain," [Online]. Available: https://en.wikipedia.org/wiki/Blockchain#Types. [Accessed 10 05 2022].

[16] S. Roth, "An Introduction to Sidechains," CoinDesk, [Online]. Available: https://www.coindesk.com/learn/an-introduction-to-sidechains/. [Accessed 10 05 2022].

[17] A. Paszke, "Layer 2," Binance Academy, [Online]. Available: https://academy.binance.com/en/glossary/layer-2. [Accessed 10 05 2022].

[18] Coinbase, "What is "proof of work" or "proof of stake"?," Coinbase, [Online]. Available: https://www.radixdlt.com/post/what-is-proof-of-work. [Accessed 10 05 2022].

[19] R. Patel, "Byzantine Fault Tolerance (BFT) and its significance in Blockchain world," hcltech, [Online]. Available: https://www.hcltech.com/blogs/byzantine-fault-tolerance-bft-and-its-significance-blockchain-world. [Accessed 10 05 2022].

[20] River, "How Bitcoin Mining Pools Work," River, [Online]. Available: https://river.com/learn/how-bitcoin-mining-pools-work/. [Accessed 10 05 2022].

[21] J. Frankenfield, "51% Attack," Investopedia, [Online]. Available: https://www.investopedia.com/terms/1/51-attack.asp. [Accessed 10 05 2022].

[22] Coinbase, "What is "proof of work" or "proof of stake"?," Coinbase, [Online]. Available: https://www.coinbase.com/pt-PT/learn/crypto-basics/what-is-proof-of-work-or-proof-of-stake. [Accessed 10 05 2022].

[23] Identity Management Institute, "BLOCKCHAIN PROOF OF STAKE CAN PREVENT CYBERATTACK," Identity Management Institute, [Online]. Available: https://identitymanagementinstitute.org/blockchain-proof-of-stake-can-prevent-cyberattack/. [Accessed 10 05 2022].

[24] Cryptopedia, "What Are Public and Private Keys?," Cryptopedia, [Online]. Available: https://www.gemini.com/cryptopedia/public-private-keys-cryptography. [Accessed 10 05 2022].

[25] A. Baloian, "How To Generate Public and Private Keys for the Blockchain," Medium, [Online]. Available: https://baloian.medium.com/how-to-generate-public-and-private-keys-for-the-blockchain-db6d057432fb. [Accessed 10 05 2022].

[26] G. Iredale, "What Problems Does Blockchain Solve?," 101 Blockchains, [Online]. Available: https://101blockchains.com/problems-blockchain-solve/. [Accessed 15 05 2022].

[27] CCN, "Indian State Government Will Issue Birth Certificates on a Blockchain," CCN, [Online]. Available: https://www.ccn.com/indian-state-government-will-issue-birth-certificates-on-a-blockchain/. [Accessed 15 05 2022].

[28] R. J. page, "Inside the Jordan refugee camp that runs on blockchain," MIT Technology Review, [Online]. Available: https://www.technologyreview.com/2018/04/12/143410/inside-the-jordan-refugee-camp-that-runs-on-blockchain/. [Accessed 15 05 2022].

[29] G. Iredale, "Top Disadvantages Of Blockchain Technology," 101 Blockchains, [Online]. Available: https://101blockchains.com/disadvantages-of-blockchain/. [Accessed 15 05 2022].

[30] C. Baldwin, "Bitcoin worth $72 million stolen from Bitfinex exchange in Hong Kong," Reuters, [Online]. Available: https://www.reuters.com/article/us-bitfinex-hacked-hongkong-idUSKCN10E0KP. [Accessed 15 05 2022].

[31] W. S., "DeFi related hacks account for 76% of all major hacks in 2021," atlasVPN, [Online]. Available: https://atlasvpn.com/blog/defi-related-hacks-account-for-76-of-all-major-hacks-in-2021. [Accessed 15 05 2022].

[32] Interpol, "What is cryptojacking? How does it work?," Interpol, [Online]. Available: https://www.interpol.int/Crimes/Cybercrime/Cryptojacking. [Accessed 15 05 2022].

[33] A. TAYLOR, "Watch out for the ‘rug pull’ crypto scam that’s tricking investors out of millions," Fortune, [Online]. Available: https://fortune.com/2022/03/02/crypto-scam-rug-pull-what-is-it/. [Accessed 15 05 2022].

[34] Chainalysis, "The 2022 Crypto," Chainalysis, 2022.

[35] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," https://bitcoin.org/bitcoin.pdf, 2008.

[36] T. B. Lee, "When will the people who called Bitcoin a bubble admit they were wrong?," The Washington Post, [Online]. Available: https://www.washingtonpost.com/news/the-switch/wp/2013/11/05/when-will-the-people-who-called-bitcoin-a-bubble-admit-they-were-wrong/?variant=116ae929826d1fd3. [Accessed 16 07 2022].

[37] Vice, "When Governments Take Your Money, Bitcoin Looks Really Good," Vice, [Online]. Available: https://www.vice.com/en/article/ypp9v5/cyprus-spain-when-governments-take-your-money-bitcoin-looks-really-good. [Accessed 16 07 2022].

[38] R. B. Arjun Kharpal, "This ‘crypto winter’ is unlike any downturn in the history of digital currencies. Here’s why," CNBC, [Online]. Available: https://www.cnbc.com/2022/07/14/why-the-2022-crypto-winter-is-unlike-previous-bear-markets.html. [Accessed 16 07 2022].

[39] Numbrs, "Is bitcoin digital gold? A look at volatility.," Numbrs, [Online]. Available: https://www.numbrs.com/is-bitcoin-digital-gold-a-look-at-volatility/. [Accessed 16 07 2022].

[40] YCHARTS, "Bitcoin Average Confirmation Time," YCHARTS, [Online]. Available: https://ycharts.com/indicators/bitcoin_average_confirmation_time. [Accessed 17 05 2022].

[41] T. D. Joseph Poon, "The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments," https://lightning.network/lightning-network-paper.pdf, 2016.

[42] BitCube, "In-depth interpretation of Lightning Network (Part 1): Payment Channel," Medium, [Online]. Available: https://medium.com/@Bitcube-/in-depth-interpretation-of-lightning-network-part-1-payment-channel-a3871a6d9c9e. [Accessed 23 05 2022].

[43] P. Chan, "What is Lightning Network?," Genesis Block, [Online]. Available: https://genesisblockhk.com/what-is-lightning-network/. [Accessed 23 05 2022].

[44] bitfinex, "The Lightning Nodes – A beginner’s guide," bitfinex, [Online]. Available: https://blog.bitfinex.com/trading/the-lightning-nodes-a-beginners-guide/. [Accessed 25 05 2022].

[45] ION Lightning Network Wiki, "Node," ION Lightning Network Wiki, [Online]. Available: https://wiki.ion.radar.tech/tech/lightning/node. [Accessed 25 05 2022].

[46] btse, "Trampoline Nodes Explained: Lightning Innovation," btse, [Online]. Available: https://blog.btse.com/trampoline-nodes/. [Accessed 25 05 2022].

[47] B. Vu, "Exploring Lightning Network Routing," Lightning Labs Blog, [Online]. Available: https://blog.lightning.engineering/posts/2018/05/30/routing.html. [Accessed 25 05 2022].

[48] Arcane research, "The State of Lightning Volume 2," Arcane research, 2022.

[49] O. Dreaming, "Instant channels enable safe Lightning payments with unconfirmed funding," Medium, [Online]. Available: https://medium.com/@akumaigorodski/instant-channels-enable-safe-lightning-payments-with-unconfirmed-funding-8d640defa183. [Accessed 27 05 2022].

[50] Wikipedia, "Onion Routing," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Onion_routing#/media/File:Onion_diagram.svg. [Accessed 27 05 2022].

[51] ION Lightning Network Wiki, "Fees," ION Lightning Network Wiki, [Online]. Available: https://wiki.ion.radar.tech/tech/bitcoin/fees. [Accessed 27 05 2022].

[52] IBM, "What are smart contracts on blockchain?," IBM, [Online]. Available: https://www.ibm.com/topics/smart-contracts. [Accessed 03 06 2022].

[53] C. Ibeawuchi, "What are smart contracts in a blockchain?," Educative, [Online]. Available: https://www.educative.io/answers/what-are-smart-contracts-in-a-blockchain. [Accessed 03 06 2022].

[54] IBM, "What are smart contracts on blockchain?," IBM, [Online]. Available: https://www.ibm.com/topics/smart-contracts. [Accessed 03 06 2022].

[55] Lightning Network, "BOLT: Basis of Lightning Technology," Lightning Network, [Online]. Available: https://github.com/lightning/bolts. [Accessed 03 06 2022].

[56] Core Lightning, "Core Lightning," Elements Project, [Online]. Available: https://github.com/ElementsProject/lightning. [Accessed 03 06 2022].

[57] R. Russel, Interviewee, *#31- BOLT 12, Umbrel, Options and much more!*. [Interview]. 21 October 2021.

[58] N. Bambysheva, "Strike Announces Shopify Integration, Partnerships With NCR And Blackhawk Bringing Bitcoin Lightning Payments To Major Merchants," Forbes, [Online]. Available: https://www.forbes.com/sites/ninabambysheva/2022/04/07/strike-announces-shopify-integration-partnerships-with-ncr-and-blackhawk-bringing-bitcoin-lighting-payments-to-major-merchants/. [Accessed 13 07 2022].

[59] Bitcoin Visuals, "Lightning Network Capacity," Bitcoin Visuals, [Online]. Available: https://bitcoinvisuals.com/ln-capacity. [Accessed 13 07 2022].

[60] R. SHARMA, "Bitcoin's Lightning Network: 3 Possible Problems," Investopedia, [Online]. Available: https://www.investopedia.com/tech/bitcoin-lightning-network-problems/. [Accessed 04 06 2022].

[61] R. H. J. Y. Zhichun Lu, "General Congestion Attack on HTLC-Based Payment Channel Networks," Cryptology ePrint Archive, 2020.

[62] ION Lightning Network Wiki, "Lightning Wallets," ION Lightning Network Wiki, [Online]. Available: https://wiki.ion.radar.tech/tutorials/wallets. [Accessed 4 06 2022].

[63] Chirag, "Custodial vs. Non-Custodial Wallets: Understanding the Difference Points," Appinventiv, [Online]. Available: https://appinventiv.com/blog/custodial-vs-non-custodial-wallets/. [Accessed 05 06 2022].

[64] ION Lightning Network Wiki, "Submarine Swap," ION Lightning Network Wiki, [Online]. Available: https://wiki.ion.radar.tech/tech/research/submarine-swap. [Accessed 06 06 2022].

[65] Lightning Labs, "Lightning Network Daemon," Lightning Labs, [Online]. Available: https://github.com/lightningnetwork/lnd. [Accessed 07 06 2022].

[66] O. O. R. P. Andreas M. Antonopoulos, Mastering the Lightning Network, USA: O'Reilly Media, 2021.

[67] Lightning Labs, "Lightning Loop: A Non-Custodial Off/On Chain Bridge," Lightning Labs, [Online]. Available: https://github.com/lightninglabs/loop. [Accessed 10 06 2022].

[68] Tor project, "Tor," Tor project, [Online]. Available: https://www.torproject.org/. [Accessed 10 06 2022].

[69] Spiral, "Lightning Dev Kit," Spiral, [Online]. Available: https://github.com/lightningdevkit. [Accessed 10 06 2022].

[70] Namcios, "How Spiral, Jack Dorsey’s Rebranded Bitcoin Company, is accelerating adoption," Bitcoin maganize, [Online]. Available: https://bitcoinmagazine.com/business/inside-spiral-jack-dorseys-bitcoin-company. [Accessed 10 06 2022].

[71] LNbits, "LNbits, free and open-source lightning-network wallet/accounts system," LNbits, [Online]. Available: https://github.com/lnbits/lnbits-legend. [Accessed 10 06 2022].

[72] E. Próspero, "CashApp Shocks The World, Enables Payments Through The Lightning Network," Bitcoinist, [Online]. Available: https://bitcoinist.com/cashapp-enables-payments-lightning-network/. [Accessed 10 06 2022].

[73] S. DELORME, "Thoughts on CashApp's Lightning user flows," [Online]. Available: https://d.elor.me/2022/02/thoughts-on-cashapps-lightning-user-flows/. [Accessed 13 06 2022].

[74] G. Play, "Muun - Bitcoin and Lightning Wallet," Muun, [Online]. Available: https://play.google.com/store/apps/details?id=io.muun.apollo&hl=pt_PT&gl=US. [Accessed 13 06 2022].

[75] CoinCarp, "What is Chivo Wallet？," CoinCarp, [Online]. Available: https://www.coincarp.com/wallet/chivo-wallet/. [Accessed 13 06 2022].

[76] World Bank, "Personal remittances, received (% Of GDP) - El Salvador," World Bank, [Online]. Available: https://data.worldbank.org/indicator/BX.TRF.PWKR.DT.GD.ZS?locations=SV. [Accessed 15 06 2022].

[77] K. Majcher, "El Salvador's Chivo wallet keeps breaking, and users are seeking answers," The Block, [Online]. Available: https://www.theblock.co/post/131452/el-salvadors-chivo-wallet-keeps-breaking-and-users-are-seeking-answersEl Salvador's Chivo wallet keeps breaking, and users are seeking answers. [Accessed 13 06 2022].

[78] M. D. Salvo, "El Salvador Taps AlphaPoint to Fix State-Sponsored Bitcoin Wallet Chivo," Decrypt, [Online]. Available: https://decrypt.co/91775/el-salvador-alphapoint-fix-chivo-bitcoin-wallet. [Accessed 13 06 2022].

[79] Empreende con huevos, "Chivo Wallet introduction," [Online]. Available: https://emprendeconhuevos.com/chivo-wallet-apk/. [Accessed 13 06 2022].

[80] ACINQ, "Introducing Phoenix," ACINQ, [Online]. Available: https://medium.com/@ACINQ/introducing-phoenix-5c5cc76c7f9e. [Accessed 13 06 2022].

[81] F. Vold, "7 Popular Bitcoin Lightning Network Wallets for 2022," Crypto News, [Online]. Available: https://cryptonews.com/exclusives/7-popular-bitcoin-lightning-network-wallets-for-2022.htm. [Accessed 13 06 2022].

[82] Spark, "Minimalistic wallet GUI for Core Lightning," Spark, [Online]. Available: https://github.com/shesek/spark-wallet. [Accessed 13 06 2022].

[83] Google Play, "Spark Lightning Wallet," Google Play, [Online]. Available: https://play.google.com/store/apps/details?id=com.spark.wallet&hl=pt_PT&gl=US. [Accessed 13 06 2022].

[84] Wheatstones, "Solana's Future is Bright & Fast.," MEDIUM, [Online]. Available: https://medium.com/coinmonks/solanas-future-is-bright-fast-5cb9a782b6f6. [Accessed 15 06 2022].

[85] Airfocus, "MoSCoW Prioritization," Airfocus, [Online]. Available: https://airfocus.com/glossary/what-is-moscow-prioritization/. [Accessed 17 06 2022].

[86] J. Nielsen, "10 Usability Heuristics for User Interface Design," Nielsen Norman Goup, [Online]. Available: https://www.nngroup.com/articles/ten-usability-heuristics/. [Accessed 17 06 2022].

[87] Wikipedia, "Android," [Online]. Available: https://pt.wikipedia.org/wiki/Android. [Accessed 17 06 2022].

[88] WIKIPEDIA, "Kotlin," [Online]. Available: https://en.wikipedia.org/wiki/Kotlin_(programming_language). [Accessed 18 06 2022].

[89] H. J. Kamps, "https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/," Tech crunch, [Online]. Available: https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/. [Accessed 17 06 2022].

[90] ALIGNMINDS TECHNOLOGIES, "What Are The Advantages of Kotlin Over Java?," ALIGNMINDS TECHNOLOGIES, [Online]. Available: https://alignminds.com/advantages-kotlin-over-java/. [Accessed 17 06 2022].

[91] N. R. T. B. D. Mistake, "ALGOL W," in *QCon Software Development Conference*, San Francisco, 2009.

[92] Code Path, "Consuming APIs with Retrofit," Code Path, [Online]. Available: https://guides.codepath.com/android/consuming-apis-with-retrofit. [Accessed 17 06 2022].

[93] Square, INC, "OkHttp," OkHttp, [Online]. Available: https://square.github.io/okhttp/. [Accessed 17 06 2022].

[94] WIKIPEDIA, "Android Studio," WIKIPEDIA, [Online]. Available: https://en.wikipedia.org/wiki/Android_Studio. [Accessed 17 06 2022].

[95] Tech Targed, "What is GitLab," Tech Targed, [Online]. Available: https://www.techtarget.com/whatis/definition/GitLab. [Accessed 20 06 2022].

[96] Software Advice, "About Jira," Software Advice, [Online]. Available: https://www.softwareadvice.pt/software/4315/jira. [Accessed 20 06 2022].

[97] T. Ghose, "What is the Fibonacci sequence?," Live Science, [Online]. Available: https://www.livescience.com/37470-fibonacci-sequence.html. [Accessed 20 06 2022].

[98] GetApp, "About Confluence," GetApp, [Online]. Available: https://www.getapp.pt/software/90949/atlassian-confluence. [Accessed 20 06 2022].

[99] E. Lazuardy, "Repository Pattern in Android," Medium, [Online]. Available: https://medium.com/swlh/repository-pattern-in-android-c31d0268118c. [Accessed 25 06 2022].

[100] Google, "Recommended Application Architecture," Google, [Online]. Available: https://developer.android.com/jetpack/guide#recommended-app-arch. [Accessed 25 06 2022].

[101] Y. Goular, "Numeric keypad," [Online]. Available: https://github.com/yoanngoular/numpadview. [Accessed 25 05 2022].

[102] C. Bernstein, "Advanced Encryption Standard (AES)," Tech Target, [Online]. Available: https://www.techtarget.com/searchsecurity/definition/Advanced-Encryption-Standard. [Accessed 28 06 2022].

[103] T. Kriuchkov, "Top 7 Methods of Data Encryption in Android ApplicationsIt was originally published on https://www.apriorit.com/," Apriorit, [Online]. Available: https://www.apriorit.com/dev-blog/612-mobile-cybersecurity-encryption-in-android. [Accessed 28 06 2022].

[104] Zxing, "Xing barcode scanning library for Java, Android," [Online]. Available: https://github.com/zxing/zxing. [Accessed 29 06 2022].

[105] K. L., "The Blockchain Scalability Problem & the Race for Visa-Like Transaction Speed," towardsdatascience, [Online]. Available:

https://towardsdatascience.com/the-blockchain-scalability-problem-the-race-for-visa-like-transaction-speed-5cce48f9d44. [Accessed 15 05 2022].

[106] YCHARTS, "Bitcoin Average Transaction Fee," YCHARTS, [Online]. Available: https://ycharts.com/indicators/bitcoin_average_transaction_fee. [Accessed 17 05 2022].

[107] YCHARTS, "Bitcoin Average Transaction Fee," YCHARTS, [Online]. Available: https://ycharts.com/indicators/bitcoin_average_transaction_fee. [Accessed 17 05 2022].

Appendix A: INTERNSHIP PROPOSAL

isec Engenharia

Departamento de Engenharia Informática e de Sistemas

Proposta de Projeto/Estágio

Ano Letivo de 2021/2022
2º Semestre

Mobile Money – Blockchain Transactions

SUMÁRIO

Este estágio consiste na prototipagem de uma aplicação móvel de carteira pessoal de Mobile Money com base na rede Lightning

Ramo: **INDICAR O(S) RAMO(S) EM QUE SE ENQUADRA:**
☒ Desenvolvimento de Aplicações
☐ Redes e Administração de Sistemas
☐ Sistemas de Informação.

1. ÂMBITO

O crescimento das tecnologias Blockchain ao longo da última década tem vindo a revelar que as suas características de imutabilidade e verificabilidade distribuida são um factor desejável para as mais variadas áreas de desenvolvimento.

Uma dessas áreas é a área de FinTech. O nascimento das crypto-currencies, nomeadamente da BitCoin, veio abrir as portas a novas oportunidades de explorar o mundo financeiro recorrendo a informação completamente transparente e verificável, em que não existe um intermediário para mediar transacções e definir taxas.

A BitCoin foi a primeira crypto-currency e é a blockchain mais antiga, e que tem como seu foco principal a segurança e descentralização. A ausência de suporte nativo para smart contracts faz com que esta blockchain, por si só, não seja capaz de competir com o rico ecossistema de aplicações descentralizadas baseadas na Ethereum ou Solana. Além disso, a mainnet da BitCoin cresceu a um ponto em que o número de transacções que é capaz de fazer por segundo é bastante baixo e a criação de novos blocos na cadeia é cada vez mais lenta, tornando-a ainda menos atractiva para o desenvolviemnto de aplicações descentralizadas que se baseiem em blockchain.

Similar ao que ocorreu com outras blockchains, a BitCoin também viu o seu crescimento expandido com a criação de cadeias de nível 2, nomeadamente, a Lightning Network. O número de aplicações baseadas na Lightning Network tem vindo a aumentar e, com ele, o número de nós na rede também. A Lightning Network permite um throughput bastante mais elevado e reduz o custo associado às transacções na rede quase para 0.

De forma a explorar as potencialidades desta cadeia de nível 2, pretende-se desenvolver um protótipo de uma aplicação de Mobile Money.

O objectivo do estágio consiste na criação de um protótipo que permita a um utilizador executar actividades necessárias ao seu dia-a-dia, como enviar dinheiro para outros utilizadores, receber dinheiro de outros utilizadores, consultar o seu saldo actual e o ver o seu histórico de transacções.

Sobre a Empresa:

A WIT desenvolve software para Operadores de Telecomunicações de vários continentes, tais como o Grupo Vodafone (Europa), Deutsche Telekom (Alemanha), Reliance Jio (Índia), KDDI, Softbank, NTT Docomo (Japão), Singtel (Singapura), Telstra (Australia), Unitel (Angola), Eir (Irlanda), Telecom Italia (Itália), Orange (França), Telefónica (Espanha), TeliaSonera (Suécia), Belgacom (Bélgica), Post Luxembourg (Luxemburgo), Bell (Canadá), Century Link (EUA) e Everything Everywhere (RU). O software desenvolvido pela WIT está presente em 46 países. A WIT tem escritórios em Portugal e Reino Unido e os seus centros de desenvolvimentos estão localizados em Coimbra, Porto, Leiria, Lisboa e Aveiro.

2. OBJECTIVOS

O presente projecto/estágio pretende atingir os seguintes objectivos genéricos:

- Explorar as características da Lightning Network e identificar as suas vantagens e limitações
- Criar um protótipo de Mobile Money baseado na Lightning Network

O objetivo do estágio é criar um protótipo que permita demonstrar:

1. Criar/registar uma wallet
2. Autenticação do utilizador
3. Consultar o saldo actual
4. Iniciar uma transacção
5. Receber um pagamento
6. Consultar histórico de transacções

3. PROGRAMA DE TRABALHOS

O estágio consistirá nas seguintes actividades e respectivas tarefas:

- ***T1*** – *Aquisição de Know-how* – Aquisição de conhecimentos sobre blockchain, BitCoin e Lightning Network
- ***T2*** – *Investigação do estado da arte* – Investigar soluções já existentes para a gestão de Mobile Money de forma descentralizada usando blockchain
- ***T3*** – *Análise de Requisitos e especificação funcional* – Proceder à análise dos requisitos para a funcionalidade e sua especificação.
- ***T4*** – *Especificação de UI* – Especificação da interface visual da funcionalidade com suporte da equipa de UI/UX da WIT.
- ***T5*** – *Implementação da fase 1* – Implementação incial do protótipo onde será possível o utilizador registar a sua wallet e ver o seu saldo corrente
- ***T6*** – *Implementação da fase 2* – Implementação das restantes funcionalidades permitindo ao utilizador realizar uma transacção, receber uma transacção e ver o seu histórico de transacções
- ***T7*** – *Relatório de estágio* – Elaboração do relatório de estágio com os resultados e objetivos atingidos.

4. CALENDARIZAÇÃO DAS TAREFAS

O plano de escalonamento dos trabalhos é apresentado em seguida:

	Meses												
Tarefas		N		N+1		N+2		N+3		N+4		N+5	
T1													
T2													
T3													
T4													
T5													
T6													
T7													
Metas	Início		M1	M2	M3	M4			M5			M6	M7

Início		Início dos trabalhos
M1	(Início + 4 Semanas)	Tarefa T1 terminada
M2	(Início + 6 Semanas)	Tarefa T2 terminada
M3	(Início + 8 Semanas)	Tarefa T3 terminada
M4	(Início + 10 Semanas)	Tarefa T4 terminada
M5	(Início + 16 Semanas)	Tarefa T5 terminada
M6	(Início + 22 Semanas)	Tarefa T6 terminada
M7	(Início + 24 Semanas)	Tarefa T7 terminada

5. LOCAL E HORÁRIO DE TRABALHO

Neste momento, a generalidades dos trabalhadores da empresa encontram-se em teletrabalho podendo o estágio decorrer nestes mesmos moldes. Deixamos, contudo, a indicação de que se a situação se alterar o local de trabalho poderá ser o escritório da WIT Software, em Taveiro ou nos nossos escritórios no Porto, Leiria, Aveiro, se assim for do interesse do aluno. O horário será a tempo completo. Durante o estágio o aluno terá ao seu dispor todos os equipamentos necessários para desempenhar as suas tarefas. O estágio é remunerado. Se o desempenho do aluno ao longo do mês for positivo, terá direito a receber uma bolsa mensal. Terá ainda acesso às formações da WIT Academy. No final do estágio, será feita uma avaliação do desempenho e dos conhecimentos adquiridos. Se o resultado for positivo o estagiário será convidado para fazer parte da equipa de desenvolvimento

6. TECNOLOGIAS ENVOLVIDAS

A linguagem a utilizar será definida durante o projecto. Sugere-se a utilização do Lightning Dev Kit (https://lightningdevkit.org/) para a integração com a Lightning Network.

7. METODOLOGIA

Serão usadas metodologias Agile.
A documentação de projeto deverá ser preferencialmente em Inglês.

8. ORIENTAÇÃO

O acompanhamento do estágio será feito não só pelo orientador, mas também por um tutor técnico que dará ao aluno todo o apoio necessário. O Orientador define os requisitos do estágio, define as prioridades e acompanha os resultados parciais do projeto. O Tutor dá todo o suporte técnico necessário, garante o cumprimento das tarefas e promove as meetings de acompanhamento do cumprimento dos objetivos.

Empresa: WIT Software
Nome: Jorge Sousa <jorge.sousa@wit-software.com>
Categoria: Head of Unit

DEIS-ISEC:

Nome <nome@isec.pt>
Categoria

Appendix B: REQUIREMENTS SPECIFICATION

Functional requirement: Create and Import a Wallet

Problem Statement:

There is no way currently to create or import an existing wallet in the system.

Feature Hypothesis:

The aim of this requirement is to allow users to create a wallet themselves and also import an existing one in order to be able to send/receive money, check transaction history, check balance and check account profile.

Benefit Hypothesis:

Better user experience creating and importing a wallet

Acceptance criteria:

- As user I should be able to click "Add a new Wallet" and be redirected to a screen with:
  1. - 1 field: Wallet name
  2. - 2 Buttons: One to create and one to import wallet

Create wallet button that can lead to three different scenarios:

Error Scenario 1#: Generic error. Such as: "Something went wrong, please try again later".

Error Scenario 2#: Empty wallet name.

Success Scenario: Progress bar informing the user the wallet is being created, leading after to the homepage.

- As user I should be able to click "Import Wallet" at add new wallet page and be redirected to a screen with:
  1. - 1 text field: Wallet string with necessary data to recover the wallet
  2. - 2 Buttons: One to Import Wallet with the data in the field and one to scan a QR code to automatically fill the field.

Import wallet button that can lead to three different scenarios:

Error Scenario 1#: Importing wallet error. Such as: "Invalid wallet information".

Error Scenario 2#: Empty import wallet field.

Success Scenario: Progress bar informing the user about the importing leading after to the homepage.

- As user after importing or creating a new wallet, I should be able to see the homepage with the following options:
    1. - Account details (user profile, change currency, change pin)
    2. - Balance
    3. - Send money
    4. - Receive money
    5. - Scan QR
    6. - Transaction History
    7. - Switch between currencies

Note: These features are only entry points to other screens that will be specified later on.

Functional requirement: Transaction History

Problem Statement:

Currently there is no way the user can check his transactions or filter them by date or description.

Feature Hypothesis:

The aim of this requirement is to allow users to easily check his last transactions or view his full transaction history and filter it by date or description.

Benefit Hypothesis:

1. Better user experience
2. Allows the user to easily check his transactions and filter them

Acceptance criteria:

- As user I should be able to see my last 5 transactions in the home screen, containing the following information:
    1. - Date of the transaction
    2. - Amount
    3. - Description
- As a user I should be able to see the value of each transaction in different currencies (BTC, SAT, EUR) by using the switch currency icon.
- As user I should be able to click on the button “See All” at the homepage and be redirected to a screen with the following information:
    1. - Scrollable list of transactions with the information described above
    2. - 1 field: Search by description
    3. - 1 date picker to filter by date
    4. - Switch icon that allows the user to switch between currencies

There are two differences when there are no transactions:

1. The homepage will display a message informing the user there are no transactions.
2. The "See All" button will display a message informing the user that there are no transactions.

Note: When loading transaction data, a progress bar will be displayed.

Functional requirement: Check Balance

Problem Statement:

Currently there is no way that the user can keep updated about the current balance.

Feature Hypothesis:

The aim of this requirement is to allow users to easily check his current balance.

Benefit Hypothesis:

Better user experience checking his updated balance easily and in real time.

Acceptance criteria:

- As user I should be able to see my balance in the home screen, being also able to switch between currencies (BTC, SAT, EUR) using the switch icon
- As user I should be able to request a balance update by pulling the home screen down (pull-to-refresh)
- As user the current wallet balance will be visible when performing a Send Transaction, so that the user knows how much balance he has available for sending

Functional requirement: Send Payment

Problem Statement:

People who own Bitcoin can't just use it as Fiat currency at the moment. There is no way to make micro or even regular Bitcoin payments with low fees and instantly.

Feature Hypothesis:

The aim of this requirement is to allow users to easily send Bitcoin without worrying about high fees or delayed confirmation of payment.

Benefit Hypothesis:

1. Better user experience
2. Saves users money in fees and time with instant payment

Acceptance criteria:

- As user I should be able to click on the button “Send” at the homepage and be redirected to a screen with the following information:
  1. - 3 text display fields: Amount to send, payment description and fee.
  2. - 1 field: Payment string
  3. - Switch icon that allows the user to switch between currencies
  4. - Scan QR code button
  5. - Pay button that can lead to three different scenarios:

  Error Scenario #1: Generic error. Such as: “Something went wrong. Please try again.”.

  Error Scenario #2: Insufficient funds error. Following message: “You have insufficient funds.”.

  Success Scenario: Payment made successfully with a pop-up confirming it, which contains a “Done” button that leads the user to the homepage.
- As user I should be able to pay an invoice by typing in the appropriate payment string or scanning the invoice QR code.
- As user after clicking in the scan QR button, I should be redirected to the camera to scan a QR code of a transaction to pay.

The QR scan that can lead to two different scenarios:

Error Scenario: Error scanning QR code. Such as: “Invalid QR code”.
Success Scenario: QR code scanned correctly, filling the send screen with data related to the payment transaction details.

- As user the current wallet balance will be visible when performing a Send Transaction, so that the user knows how much balance he has available for sending.

Functional requirement: Receive Payment

Problem Statement:

To receive Bitcoin, you have to pay high fees and wait for the time-consuming transaction to be confirmed. There is a barrier to receiving Bitcoin payments, especially micropayments with low fees and fast.

Feature Hypothesis:

The aim of this requirement is to allow users to receive Bitcoin payments fast with low fees.

Benefit Hypothesis:

1. Better user experience
2. Saves users money in fees and time receiving payments

Acceptance criteria:

- As user, I should be able to click on the button “Receive” at the homepage and be redirected to a screen with the following information:
  1. - 2 fields: Amount to be received and description
  2. - Switch icon that allows the user to switch between currencies
  3. - Create Invoice button that can lead to two different scenarios:

     Error Scenario: Generic error. Such as: “Something went wrong. Please try again.”.

     Success Scenario: Creates a new Invoice successfully, redirecting the user to a new screen with transaction details. This screen contains the QR code and payment string to pay the amount defined and allows the user to share it via different applications through a “Share” button on the screen. It also contains details about the invoice created such as the amount and description and a Button to return to the homescreen.
  4. - Scan QR code button can lead to the screen with details mentioned above.
- As user after clicking in the scan QR button, I should be redirected to the camera to scan a QR code of an Invoice to read.

The QR scan that can lead to two different scenarios:

Error Scenario: Error scanning QR code. Such as: “Invalid QR code”.
Success Scenario: QR code scanned correctly, and the user is redirected to the screen mentioned above with the fields fulfilled.

Functional requirement: User Onboarding

Problem Statement:

Currently there is no way to onboard the system to be familiar with the application. The application may be used by users with different levels of experience with Blockchain wallets and the Lightning Network, from users who never contacted with the technology to users who are experienced and may even bring their own wallet to the application.

Feature Hypothesis:

The aim of this requirement is to allow the user with less experience to be introduced to the application in a series of steps about the Lightning Network and how the application allows the user to send and receive funds.

Benefit Hypothesis:

Better user experience by giving a brief introduction to the user on how to use the application and its benefits.

Acceptance criteria:

- As user I need to be able to download the app and see the oboarding page with the following information's:
    1. - A commercial statement as title (such as: "The easiest way to send and receive Bitcoin!")
    2. - Two buttons: One to go to the next introduction screen and one to skip onboarding
    3. - Brand image / logo

Notes: The first screens of the application are extremely important for the end-user. They will define his first opinion about it: "Does this application do what I need? How do I use this application? ". These are all questions that the user has and should be immediately answered by the application as soon as the user launches it.

Functional requirement: User authentication

Problem Statement:

Currently there is no security feature to prevent any person to enter the wallet.

Feature Hypothesis:

The aim of this requirement is to allow the user to have a PIN to enter the wallet.

Benefit Hypothesis:

1. Better user experience
2. Improves the necessary security the user expects in a wallet

Acceptance criteria:

- As user I need to be able to create a new PIN when entering the wallet for the first time, after onboarding.
- As user I should be able to use a 4-digit PIN code to use to access the wallet.

Entering the 4-digit PIN can lead to three different scenarios:

Error Scenario #1: Wrong PIN error. Such as: "Wrong PIN, try again.".
Error Scenario #2: Wrong PIN 3 times error. Such as: "Wrong PIN. Login blocked for 6hours.".
Success Scenario: Message informing the user about the correct PIN inserted, redirecting the user to the homepage.

Notes: The PIN is always sent through the network encrypted. It also should never be persisted in its plain text form, neither on the mobile device nor on the server backend.

Functional requirement: User Settings

Problem Statement:

Currently there is no user settings or ways to change user or wallet data.

Feature Hypothesis:

The aim of this feature is to allow users to easily check change their wallet name, language, set default currency balance displays and delete wallet and more.

Benefit Hypothesis:

1. Better user experience
2. Allows the user to easily change their personal wallet settings

Acceptance criteria:

- As user I should be able to set my default currencies in the home screen.
- As user I should be able to change my wallet name
- As user I should be able to delete my wallet permanently

Functional requirement: Export wallet backup to pdf

Problem Statement:

Currently there is no way to save the user recovery key in a pdf document.

Feature Hypothesis:

The aim of this requirement is to allow users to have a backup of their wallet key to a pdf document, making it easier to recover in the future if needed.

Benefit Hypothesis:

1. Better user experience
2. Allows the user to have a pdf with all necessary information to recover the wallet

Acceptance criteria:

- As user I should be able to have the necessary data to recover the wallet in a pdf, downloading it by clicking the “download recovery pdf” in the user settings

Functional requirement: Support for different languages

Problem Statement:

Currently the application has only one language, English.

Feature Hypothesis:

The aim of this requirement is to allow the user to use the application with his own language.

Benefit Hypothesis:

1. Better user experience
2. Diversifies the application, having the support for multiple languages, covering a bigger number of users

Acceptance criteria:

- As user I should be able to go the user settings and change the application language.

Functional requirement: Wallet storage in an encrypted database

Problem Statement:

Currently the application stores its own data in the application phone.

Feature Hypothesis:

The aim of this requirement is to improve user data security, using an encrypted database.

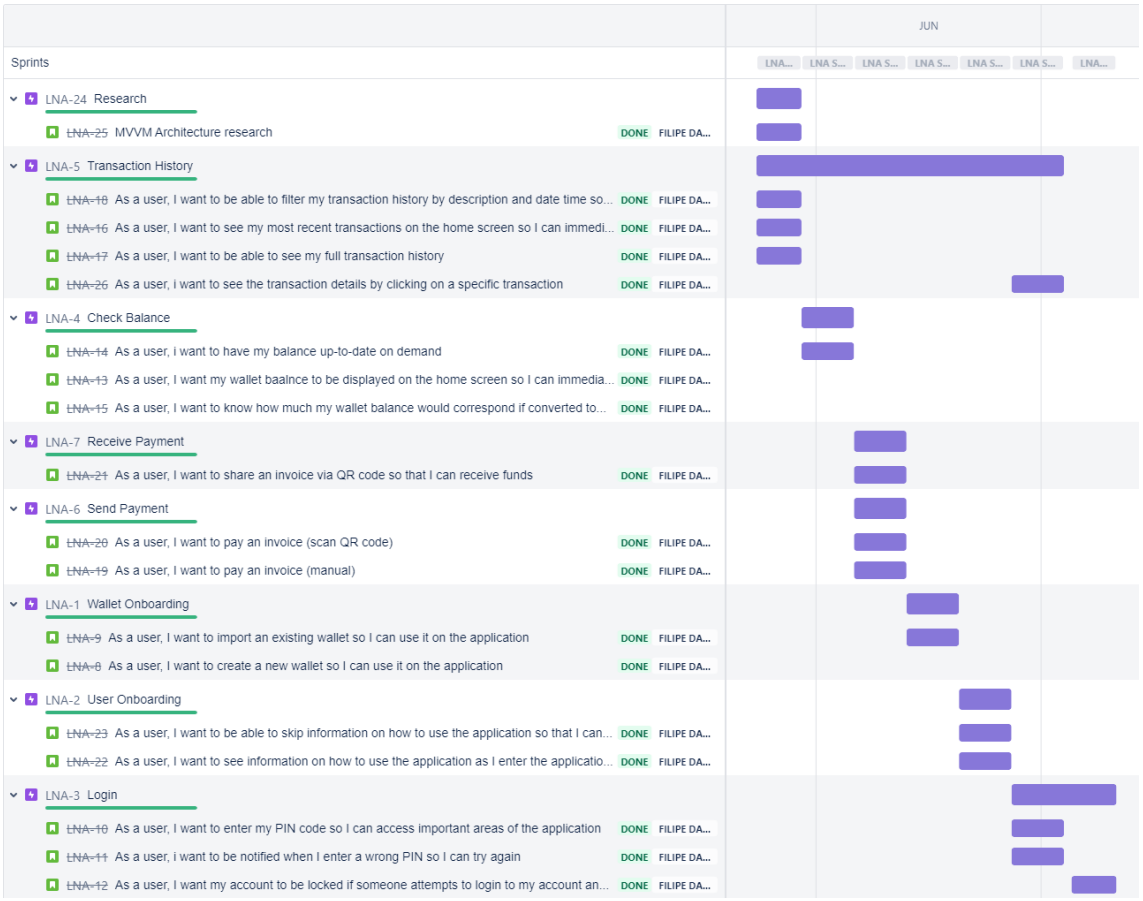
Benefit Hypothesis:

1. Better user experience
2. Improves the necessary security the user expects in a wallet

Acceptance criteria:

- As user I should know that my data secured

Appendix C: JIRA TASKS SCHEDULE



Appendix D: CONFLUENCE UI PROTOTYPE DRAFT

[Draft] User Interface - Proposal

- Onboarding
  - User Introduction
  - Create/Recover Wallet
    - Create a New Wallet
    - Recover Wallet
    - Import Wallet
    - User Authentication
    - Notes
- Home Page
  - Display Balance
  - Statements
  - Actions
  - Notes
- Receive Payment
  - Statements
  - Actions
  - Notes
- Send Payment
  - Statement
  - Actions
  - Notes
- [NOTES, FEATURES TO ADD] QR Code Screen

This page defines the guidelines for the User Interface of the mobile application that will enable end-users to perform transactions via Lightning Network.

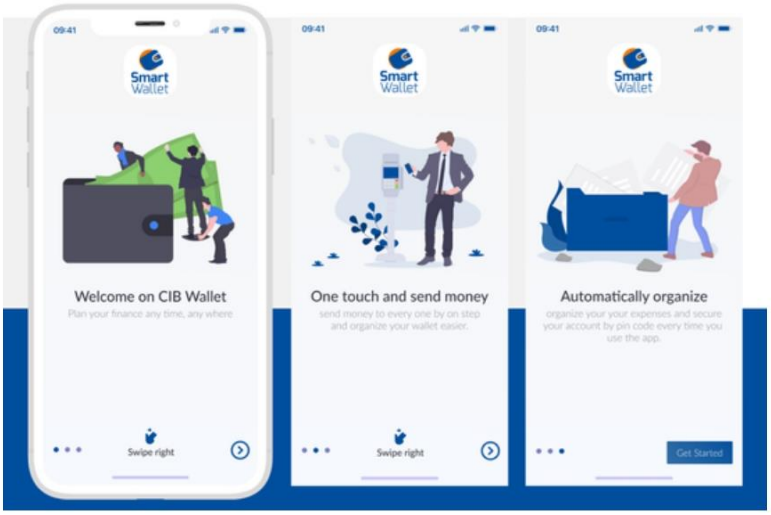
As a way to expedite the availability of the User interface for development and align the user Experience to an already existing project, we will be using the M-Pesa mobile application for Kenya, BlueWallet and Muun Wallet as a baseline for the prototype.

Namely, we will also be using the section for Pochi la Biashara as a guide to the development some of the features of the prototype.

Onboarding

User Introduction

The user will be presented with a brief tutorial with concise and clear introduction on how to use the wallet when they first open the application.

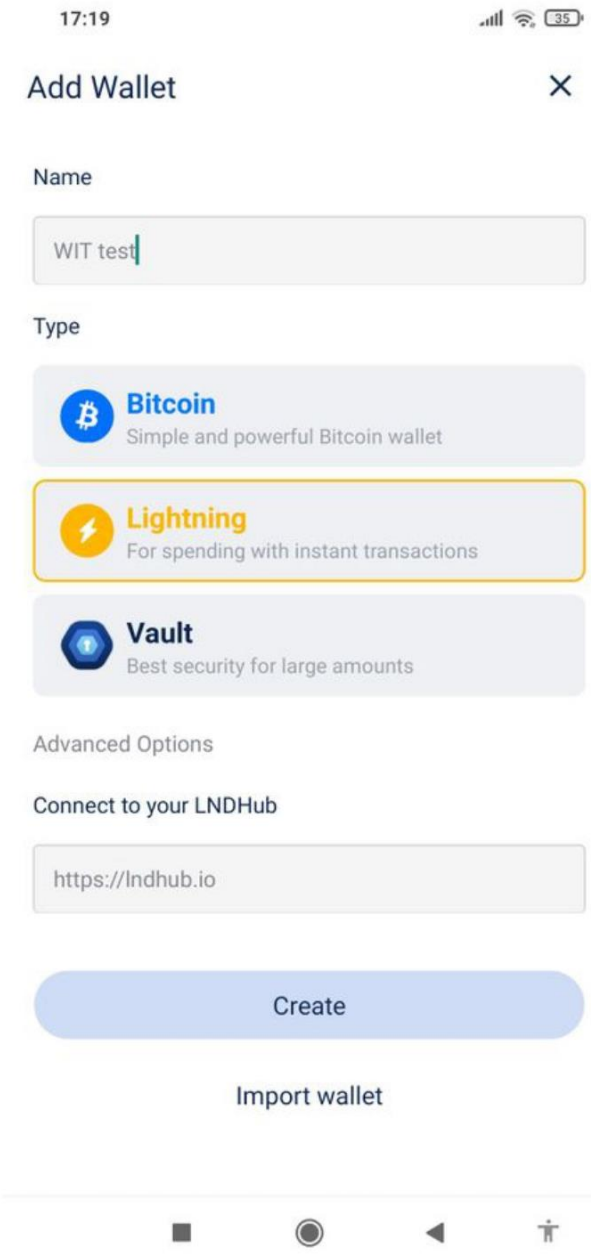


Create/Recover Wallet

To create or restore an account, the user is presented with a screen containing two buttons for each choice.

Create a New Wallet

To make a new wallet, the user only needs to give it a name.



Recover Wallet

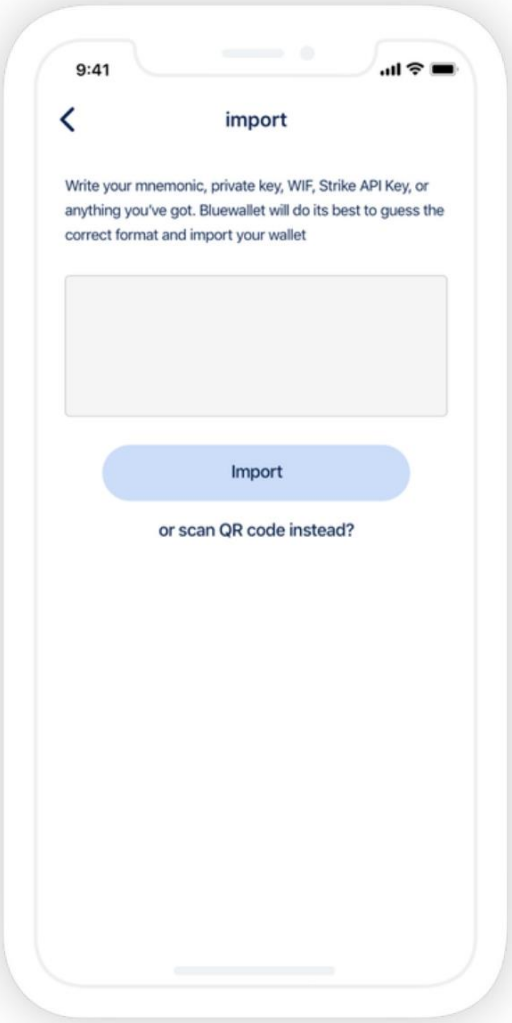
After clicking the create button, the user will be redirect to a screen informing the Wallet was created along with a QR code and link that allows the user to restore it.

When the user clicks on the wallet recovery string, a toast message for 2 seconds will be shown “Wallet backup copied to clipboard!”


Your wallet has been created
Please save this wallet backup. It allows you to restore the wallet in case of loss.
blue
lndhub://110f1aa619d915b24827:
bd54edcc3d4f8cb21bbb@https://lndhub.io
OK, I have saved it

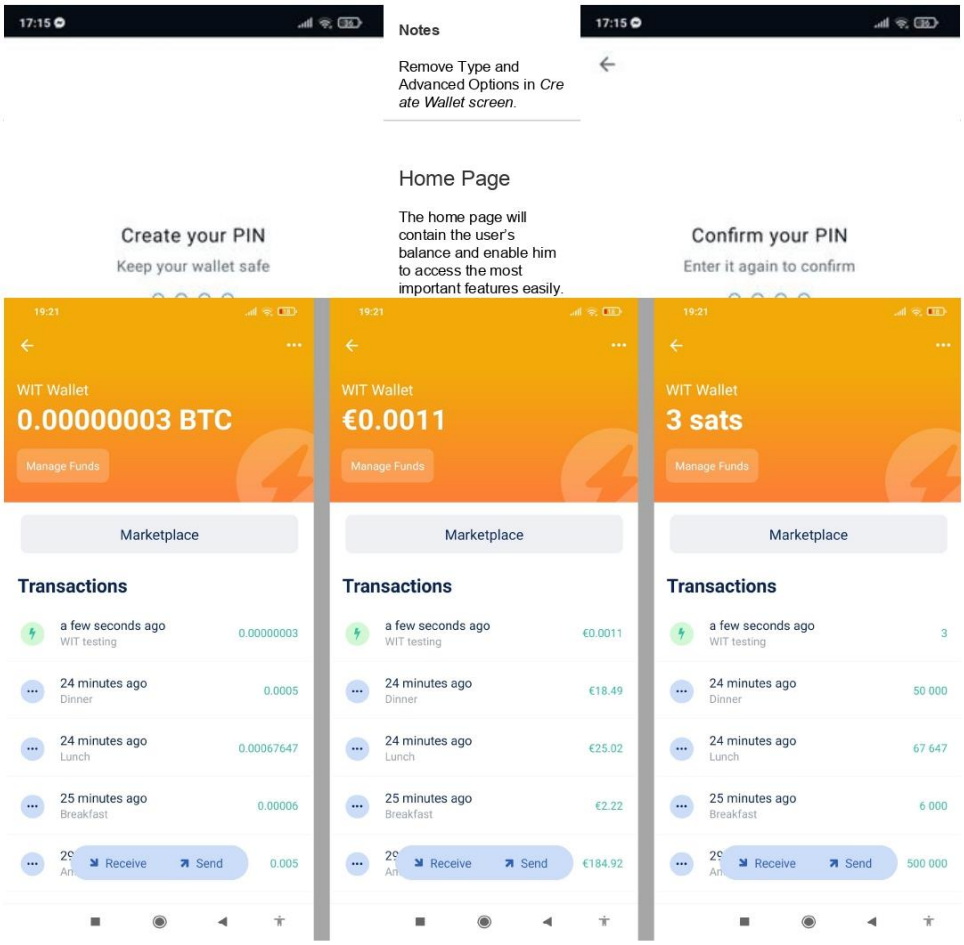
Import Wallet

The user will be able to import a wallet recurring to a mnemonic, private key, or a QR (**TBD**).



User Authentication

After saving the required data to store the wallet, the user has to define a pin for the application.



Display Balance

The balance is displayed on the top of the home page along with the wallet name and will be reloaded when the user is entering the application.

While a new balance is being retrieved, a progress animation is displayed on the top.

The user is able to trigger a refresh of the balance by pushing the screen down (push-to-refresh).

Statements

Displays the transactions in a scrollable list.

Allows to switch between currencies (EUR,SAT,BTC) by clicking in the balance (change in settings only maybe?).

By clicking on a transaction, the user is redirected to another screen, where he is able to see transaction information.

Actions

There are three main actions in the main screen, being:

- Receive Payments: after clicking the *Receive* button, the user will be redirected to a screen where he can make an invoice based on an amount or scan a QR invoice to receive payments.
- Send Payments: by clicking this option with the *send* button , the user will be redirected to a new screen to insert the destination address or QR code of the invoice to be paid.
- Account: A button/icon on the top right of the screen for the user to enter his private settings area where he is able to manage his Lightning Wallet settings

Notes

Remove top left back icon and possible top right icon too (It will be used for settings maybe).

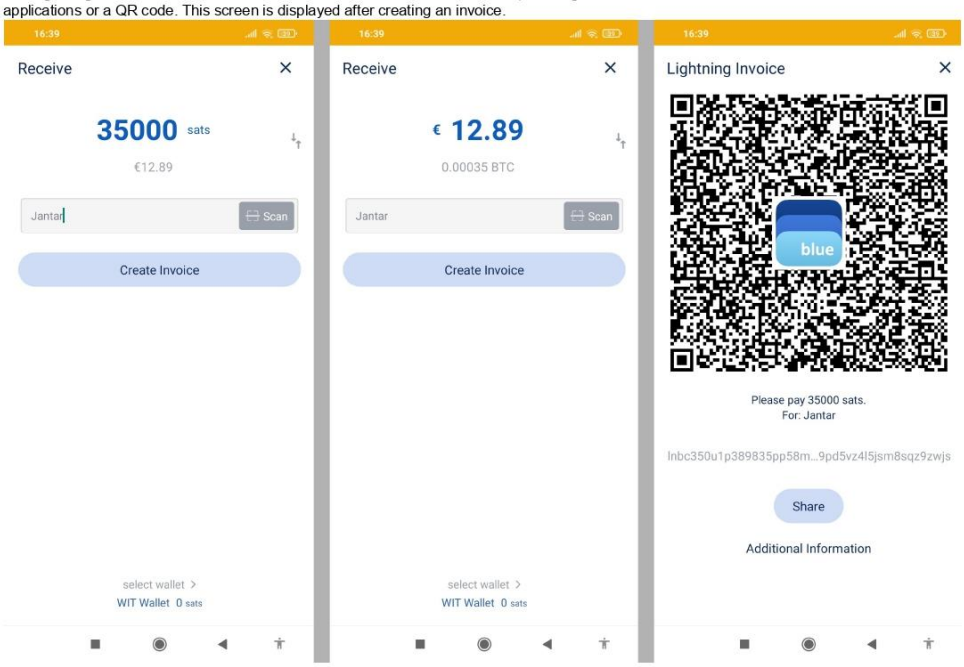
Remove *marketplace* and *manage funds* buttons.

Alter the "three dots" icon on the top-right for another one and use it for the account settings, keeping only 2 options on the quick actions (Receive,Send).

Receive Payment

The *Receive* screen allows the user to indicate the amount he wants to receive, together with the message associated with that invoice. It also allows the scan directly an invoice through a QR code.

The *Lightning Invoice* details information about the invoice such as the amount, message. It also enables the user to share it via different applications or a QR code. This screen is displayed after creating an invoice.



Statements

The "switch" icon on the right side of the invoice balance allows the user to switch between currencies.

The “X” icon on the top right allows the user to go back to the main screen.

When the user clicks on the invoice string, a toast message for 2 seconds will be shown “Address copied to clipboard!”

Actions

Create Invoice button will redirected the user to a screen detailing the invoice and allowing the user to share it with ease, allowing to copy the *invo ice string* or even share the QR code.

The *Scan* option allows you to get bitcoin by scanning a QR code.

The *Share* button in the details screen allows the user to share the invoice link through different platforms.

Notes

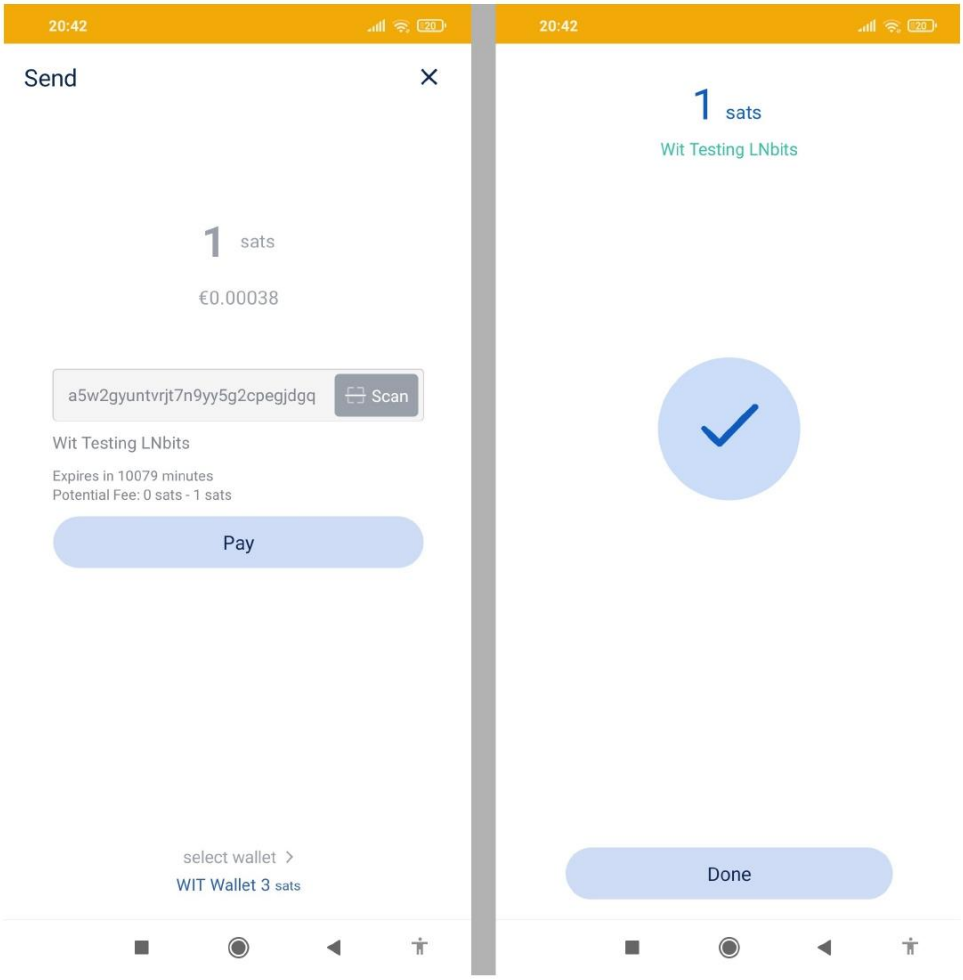
The “select wallet” button won’t exist.

Send Payment

The send payment screen allows the user to pay an invoice by typing in the appropriate payment string or scanning the invoice QR code.

By clicking “Scan”, the user is redirected to the camera to scan a QR code of a transaction to pay. If successful, the fields with information about the transaction will be automaticaly filled from the QR code.

The transaction details can also be filled out manually.



Statement

The “X” icon on the top right allows the user to go back to the main screen.

Transaction details are filled automatically after scanning or filling the invoice field.

Actions

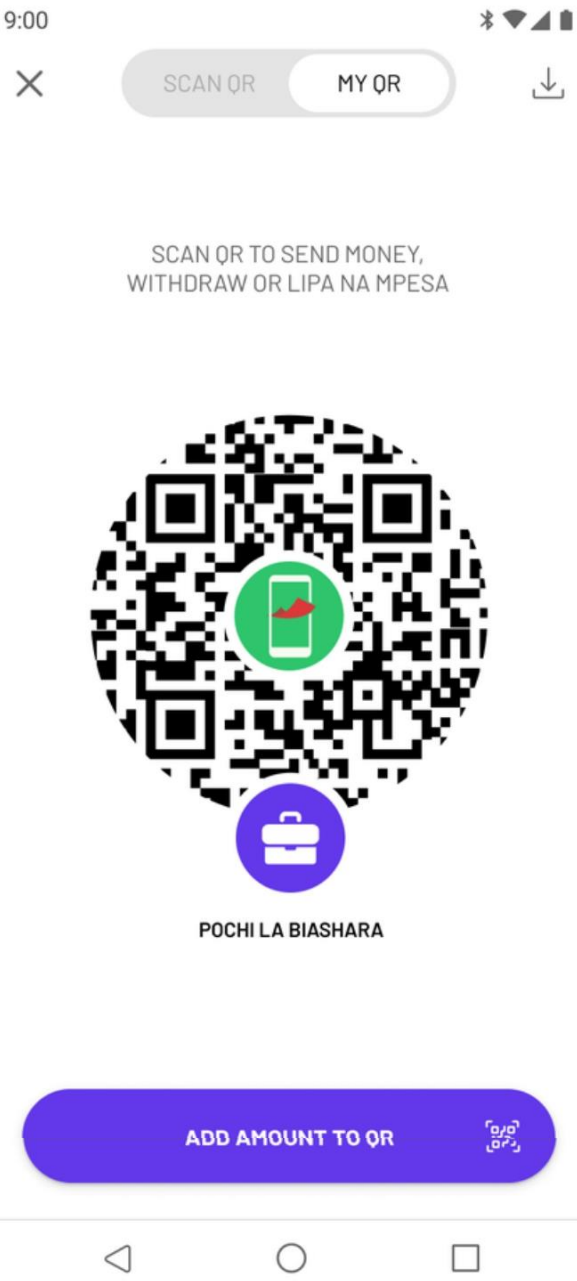
The *pay* button takes the user to a new page that confirms that the payment was completed. If there is an error, a message will be displayed instead, alerting the user.

The *Scan* option allows you to get bitcoin by scanning a QR code.

Notes

The “select wallet” button won’t exist.

[NOTES, FEATURES TO ADD] QR Code Screen



The screen opens with the user’s QR code open

By clicking the top-left cross, the user returns to the previous screen (should be the home screen), dismissing this one.

Remove “Pochi la Biashara” text under the QR code and don’t use the icon either (eventually, might be interesting to put a “Lightning Network” icon). Replace it with the beginning of the user’s public address, elapsized, and allow user to copy to clipboard by clicking a “copy” icon at the end of the text. When he clicks it, show a toast message for 2 seconds saying “Address copied to clipboard!”

Remove the center icon of the QR code

The circular QR code frame is optional (nice to have)

Replace the text “Scan QR to Send Money, Withdraw or Lipa na M-Pesa” with “Scan QR to send money with Lightning!”

By clicking the download button on the top-right, the user downloads his QR code. He can then share it with another user though a different application if he so desires.

By clicking the button “Add amount to QR” the user is prompted to add an amount to facilitate the next transaction. The amount is then encoded on the QR code.

By clicking “Scan QR” on the top switch, the user is redirected to the camera and scans a QR code of another user. If successful, the user is redirected to the flow to send money with lightning (another option in the home screen), with the fields already pre-filled from the QR code. If all fields are already available, he then only needs to complete the transaction by entering his wallet’s PIN code/password.

He also has an option to load a QR code from a file, for situations where the QR code was sent through a different channel other than scanning with the camera.

.