UNIVERSIDADE D
COIMBRA

Rodrigo Ribeiro Cardoso de Oliveira

# DESENVOLVIMENTO DE SOFTWARE PARA AUDITORIA DE TRANSAÇÕES BITCOIN

BITAUDIT

**Relatório de Estágio do Mestrado em Engenharia Informática, orientado pelo Professor Doutor Paulo Rupino da Cunha do Departamento de Engenharia Informática da FCTUC, Professor Doutor Hélder Sebastião e Professor Doutor Paulo Melo da Faculdade de Economia da Universidade de Coimbra, apresentado ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra**

junho de 2021

# FACULDADE DE CIÊNCIAS E TECNOLOGIA

# DESENVOLVIMENTO DE SOFTWARE PARA AUDITORIA DE TRANSAÇÕES BITCOIN

## BITAUDIT

| | |
|---|---|
| **Tipo de trabalho** | **Relatório de Estágio** |
| **Título** | Desenvolvimento de software para auditoria de transações Bitcoin |
| **Subtítulo** | BitAudit |
| **Autor/a** | Rodrigo Ribeiro Cardoso de Oliveira |
| **Orientador/a(s)** | Professor Doutor Paulo Rupino<br>Professor Doutor Hélder Sebastião<br>Professor Doutor Paulo Melo |
| **Identificação do Curso** | **2º Ciclo em Engenharia Informática** |
| **Área científica** | |
| **Especialidade/Ramo** | Engenharia de Software |
| **Ano** | **2020/2021** |

1 2 9 0

UNIVERSIDADE D COIMBRA

# Agradecimentos

Em primeiro lugar, o meu reconhecimento é dirigido aos meus orientadores, Professor Doutor Paulo Rupino da Cunha, Professor Doutor Hélder Sebastião e Professor Doutor Paulo Melo, pela disponibilidade, partilha de conhecimentos, esclarecimentos e críticas pertinentes, ao longo da realização do estágio e da elaboração do presente relatório.

Aos meus pais e irmã, pelo exemplo e por partilharem comigo os valores da pessoa que sou, bem como o seu apoio incondicional, tanto pessoal como académico, manifestado por cada palavra de motivação e de incentivo. E restantes familiares, que estiveram sempre disponíveis para me apoiar.

Por fim, mas não menos importantes, aos meus amigos, de Viseu e de Coimbra, pela presença assídua, pelos momentos de descontração e pela ajuda na concentração. Eles que tornaram todo este processo mais fácil.

Um grande obrigado a tod@s!

# Resumo

A Bitcoin é uma criptomoeda descentralizada, que permite efetuar pagamentos online, diretamente de uma parte para outra, sem passar por uma instituição financeira. É construída com base na *blockchain*, uma aplicação que recorre a mecanismos criptográficos para implementar livros-razão ou livros de registo de forma distribuída e segura.

O principal objetivo deste estágio consistiu no desenvolvimento ou adaptação de um *software open-source,* que possibilitasse estudar as transações registadas na *blockchain* da Bitcoin, de forma a permitir análises económicas.

A partir do aprofundamento de temas, como a análise forense realizada na rede Bitcoin e dos *softwares* de análise já existentes, definimos o processo e reunimos a documentação para o desenvolvimento do BitAudit, que incluiu a análise e validação de requisitos, definição da arquitetura do software, implementação e realização de testes. Foi utilizado o *BlockSci*, um *software open-source* para análise de *blockchain*, como base da nossa ferramenta.

Por fim, com a ferramenta desenvolvida, foram realizadas análises à rede Bitcoin, de modo a demonstrar parte do que poderia ser feito, por investigadores e autoridades.

O BitAudit contém diversas funcionalidades que possibilitam análises, como estudar um evento específico no tempo, a atividade de um endereço, transações e endereços associados, bem como a procura e obtenção de dados sobre endereços com características específicas. Com o crescente interesse na Bitcoin e, consequente aumento da atividade na sua rede, tanto em termos legais como ilegais, torna-se fundamental aperfeiçoar esta ferramenta, munindo-a de mais funcionalidades que permitam estudar todo o tipo de situações.

**Palavras-chave**: Bitcoin, Análises Económicas, Análises Forenses, BlockSci, Blockchain

# Abstract

Bitcoin is a decentralized cryptocurrency that allows online payments, directly from one party to another, without going through a financial institution. It is based on the blockchain, an application that uses cryptographic mechanisms to implement ledgers or logbooks in a distributed and secure way.

The main objective of this internship was to develop or adapt an open-source software, which would make it possible to study the transactions registered in the Bitcoin blockchain, to allow economic analysis.

Themes such as forensic analysis performed on the Bitcoin network and the existing blockchain analysis tools were studied so we could define the process and gather the documentation for the development of BitAudit, which included the analysis and validation of requirements, definition of the software architecture, implementation, and testing. BlockSci, an open source blockchain analysis software, was used as the basis of our tool.

Finally, with the developed tool, analyzes were carried out on the Bitcoin network, to demonstrate part of what could be done, by Investigators and Authorities.

BitAudit contains several functionalities that enable analysis, such as studying a certain specific event in time, the activity of an address, transactions, and associated addresses, searching and obtaining data about addresses with specific characteristics. With the growing interest in Bitcoin and, consequently, with the increase in activity on its network, both in legal and illegal terms, it is essential to improve this tool, providing it with more features that allow the study of all types of situations.

**Keywords**: Bitcoin, Economic Analyses, Forensic Analyses, BlockSci, Blockchain

# Índice

# Índice de Tabelas

# Índice de Figuras

# Lista de Abreviaturas e Siglas

| | |
|---|---|
| API | *Application Programming Interface* |
| BTC | *Bitcoins* |
| REST | *Representational State Transfer* |
| J2EE | *Java2 Platform Enterprise Edition* |
| UML | *Unified Modeling Language* |
| RAM | *Random Access Memory* |
| ID | Identificação |
| UED | *User Environment Design* |
| UI | *User Interface* |
| HTTP | *Hypertext Transfer Protocol* |
| PHP | *Hypertext Preprocessor* |
| HTML | *HyperText Markup Language* |
| CSS | *Cascading Style Sheets* |
| JSON | *JavaScript Object Notation* |
| UTF | *Unicode Transformation Format* |
| IDE | *Integrated Development Environment* |
| VPN | *Virtual Private Network* |
| IT | *Information technology* |
| CBOE | *Chicago Board Options Exchange* |

# 1. Introdução

## 1.1 Âmbito

Com o contínuo crescimento da Bitcoin, que à data de escrita deste relatório atinge valores históricos (1 BTC = 53.180€), torna-se importante aprofundar o seu conhecimento em vários domínios, nomeadamente financeiro e legal.

Desde que a Bitcoin ganhou popularidade, as criptomoedas tornaram-se a primeira escolha, não só de investidores e entusiastas de tecnologia, mas também de criminosos [1]. Detetar a atividade dos vários tipos de utilizadores é bastante exigente e desafiador, devido ao alto volume de transações e estrutura de dados Bitcoin [1]. Existem alguns métodos, propostos por diferentes autores [2][3][4][5], para detetar atividades ilegais ou apenas estudar a rede Bitcoin, em termos económicos e financeiros.

Perante este panorama, surge a necessidade de desenvolver uma ferramenta que vá ao encontro das necessidades de autoridades e investigadores, na qual estejam reunidas funcionalidades úteis para as análises que sejam necessárias realizar.

Face ao exposto, o presente estágio situa-se no âmbito de Sistemas de Informação e Engenharia de *Software.*

## 1.2 Definição de Objetivos

O principal objetivo deste estágio foi desenvolver ou adaptar um *software open-source,* que possibilitasse estudar as transações registadas na *blockchain* da Bitcoin, de forma a permitir análises económicas que foram realizadas com o apoio de professores da Faculdade de Economia.

Foram também definidos os seguintes objetivos específicos:

**1º Semestre:**

- Aprofundar os conhecimentos sobre a Bitcoin e a tecnologia *blockchain;*
- Situar o estado da arte sobre a análise forense;
- Analisar *softwares* de auditoria de Bitcoin existentes;
- Tomar decisões sobre o caminho a seguir para o desenvolvimento do *software*.

**2º Semestre:**

- Realizar explorações preliminares na rede Bitcoin;

- Definir os requisitos de auditoria para o *software*;
- Desenvolver/adaptar *software* de auditoria;
- Realizar análises económicas com o *software*;
- Escrever um artigo científico sobre o *software* desenvolvido.

## 1.3 Planeamento

No início do estágio foi elaborado um plano de tarefas para ser concretizado no decorrer do primeiro semestre, tendo em conta os objetivos apresentados (Figura 1).

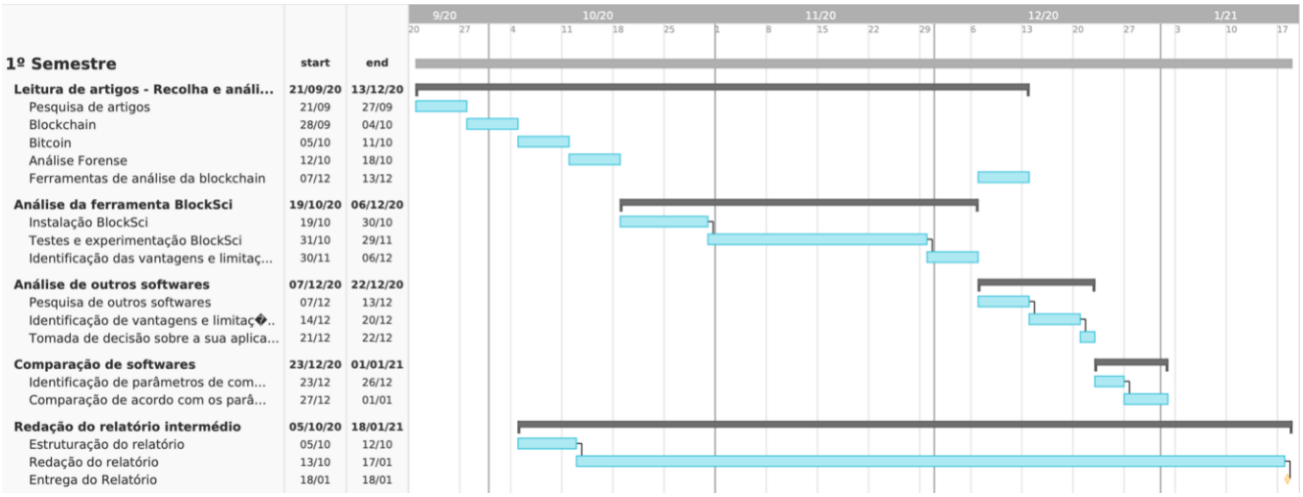


*Figura 1: Diagrama de Gantt estimado do 1º Semestre*

Devido a contratempos, nomeadamente a nível da instalação da ferramenta *BlockSci,* e dada a necessidade de aprofundar alguns dos conceitos abordados, o plano inicialmente definido sofreu algumas alterações. Como se pode observar no diagrama de Gantt correspondente ao concretizado (Figura 2), a primeira tarefa consistiu na leitura de artigos e aquisição de conhecimentos sobre *blockchain,* Bitcoin, análise forense e ferramentas de análise Bitcoin. De seguida, efetuou-se a análise da ferramenta *BlockSci*, já identificada no início do estágio como sendo bastante promissora. Após um estudo feito sobre outros *softwares*, foi efetuada uma comparação entre as ferramentas encontradas e extraídas conclusões. Paralelamente, foi estruturado e redigido o relatório intermédio sobre as atividades desenvolvidas no decorrer do 1º semestre.

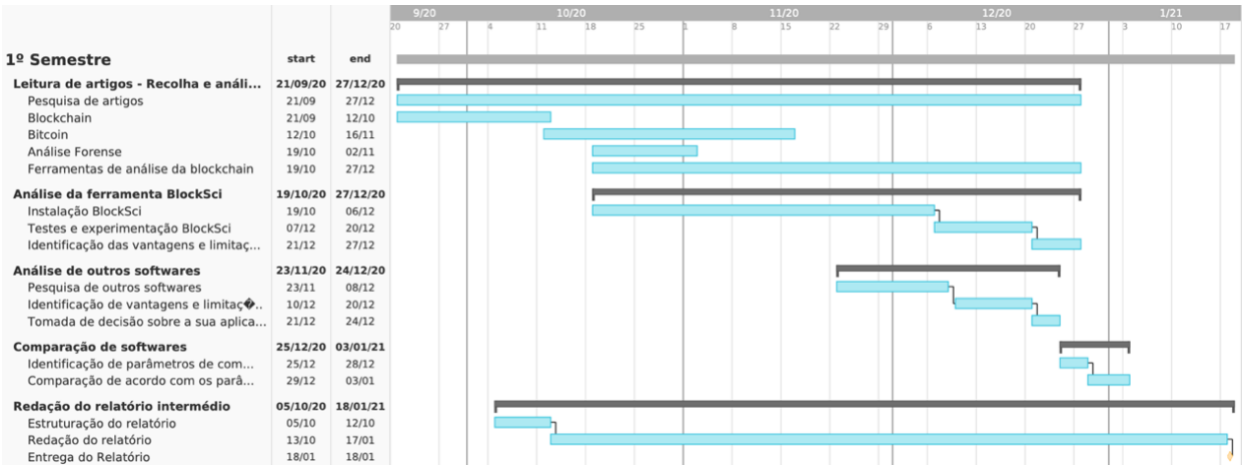


*Figura 2: Diagrama de Gantt concretizado do 1º Semestre*

Quanto ao plano de trabalhos para o 2º semestre e tendo em conta os objetivos do presente estágio, foram definidas as tarefas apresentadas na Figura 3.

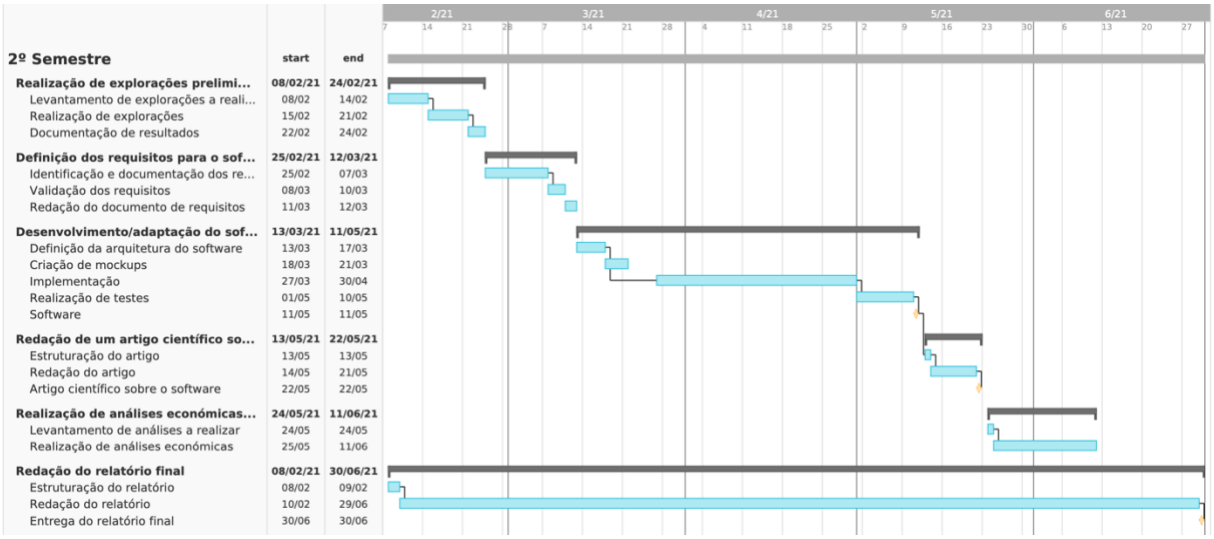


*Figura 3: Diagrama de Gantt estimado do 2º Semestre*

Como se pode observar na Figura 3, a metodologia escolhida para a implementação foi a *Waterfall*, que consiste num desenvolvimento de *software* sequencial, em que o progresso flui de forma contínua em direção à conclusão, através das fases de um projeto (análise de requisitos, arquitetura, implementação e testes) [6]. Esta escolha foi devida não só à falta de tempo disponível, mas também à definição prévia de toda a estrutura de desenvolvimento e de objetivos, não sendo provável a ocorrência de alterações significativas ao longo do processo.

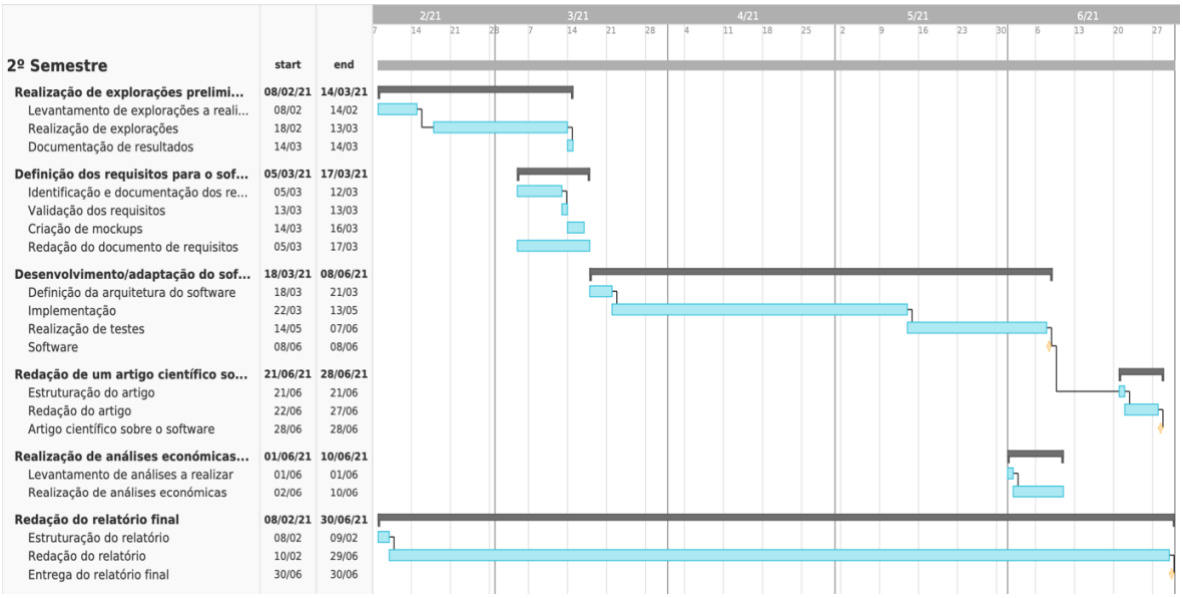


*Figura 4: Diagrama de Gantt concretizado do 2º Semestre*

Considerando que a realização de explorações na rede Bitcoin levou mais tempo que o estimado, devido à quantidade de dados existentes na *blockchain* (desde 2009 a 2014 apenas), o diagrama sofreu alterações (Figura 4). Podemos verificar que as explorações se sobrepuseram à definição dos requisitos do *software*, já que enquanto estas executavam, foi possível realizar outras tarefas em simultâneo. No que concerne à implementação da ferramenta e aos respetivos testes, houve também atrasos, nomeadamente a existência de erros no código e utilização de tecnologias sobre as quais tivemos que obter conhecimentos (PHP, *docker* e API em *Flask*). De seguida, foram realizadas análises com o *software* desenvolvido. Por fim, foi esboçada uma primeira versão do artigo relativo ao *software* implementado, que designamos de BitAudit. O presente relatório foi redigido à medida que as restantes atividades eram terminadas.

## 1.4 Estrutura do Documento

O relatório final é constituído por nove capítulos, o primeiro versa sobre a introdução do estágio e do relatório, o segundo apresenta o “Estado de arte” do nosso objeto de estudo, tendo início com a apresentação do percurso metodológico efetuado. De seguida, faz-se a concetualização e evolução da Bitcoin. A comparação e respetiva análise de *softwares* de auditoria, que tem por base parâmetros definidos, são referidos na última parte deste capítulo e serviram de ponto de partida para a decisão do percurso a seguir no 2º semestre.

O terceiro capítulo incide sobre a realização e documentação de explorações preliminares realizadas na rede Bitcoin, que permitiram conhecer de forma mais aprofundada o comportamento dos utilizadores e da rede envolvente.

Os capítulos quatro, cinco, seis e sete correspondem a apresentação do processo de desenvolvimento do *software*, nomeadamente a análise de requisitos, a definição da arquitetura, a implementação e a realização de testes.

Segue-se o capítulo oito, que contém a documentação das análises realizadas com o uso da ferramenta desenvolvida.

Por fim, apresentam-se as conclusões, limitações e sugestões de futuros trabalhos a desenvolver neste âmbito.

# 2. Estado de Arte

## 2.1 Metodologia de Pesquisa

A realização de uma investigação segue três fases essenciais: concetual, metodológica e empírica. Enquanto a fase concetual fornece à investigação as bases para a formulação de uma ideia e a delimitação do estudo, a fase metodológica corresponde ao conjunto de procedimentos utilizados para dar resposta aos objetivos definidos [7], que passamos a descrever.

Assim, dada a necessidade de obter informações iniciais sobre o tema, foram fornecidos pelos orientadores alguns artigos relativos à *blockchain*, às criptomoedas em geral, à Bitcoin e às atividades ilegais com recurso às criptomoedas.

Deu-se continuidade ao enquadramento teórico dos conceitos, através de uma pesquisa bibliográfica efetuada nas plataformas *Google Scholar* e *IEEExplore*, utilizando palavras-chave e a estrutura definida por Webster e Watson [8].

A primeira pesquisa foi efetuada no *IEEExplore* com as palavras "*Illegal, Cryptocurrencies*" (em que a vírgula significa "ou"). Foram encontrados e analisados 43 artigos, o que resultou na seleção de três [9][1][10] para integrarem o presente relatório, por permitirem, por um lado, o aprofundamento dos conhecimentos sobre a temática em estudo e, por outro, por fornecerem informações acerca do desenvolvimento de um *software* de análise de transações Bitcoin. Após a leitura dos *Abstract*, os restantes 40 foram descartados por não estarem diretamente relacionados com o que era pretendido para este relatório.

De seguida, foram utilizadas as palavras "*Forensics, Cryptocurrencies, Bitcoin"*, para obter artigos mais dirigidos à análise forense na Bitcoin e, dos quatro artigos encontrados, foi

filtrado um para um estudo mais aprofundado [11], por ser aquele que estava diretamente relacionado com o objeto de estudo.

Por fim, na plataforma *Google Scholar*, foram utilizadas as palavras "*Forensics, Cryptocurrencies, Bitcoin"*, com o mesmo propósito da pesquisa anterior, que retornou 18000 resultados. Na observação efetuada aos primeiros resultados, identificamos artigos já anteriormente encontrados e um novo [12] para ser analisado. Os restantes 33 artigos obtidos nas primeiras cinco páginas de pesquisa não eram relevantes para o caso de estudo.

Foram analisados os cinco artigos encontrados [1] [9][10][11][12], para além dos oito inicialmente explorados, de forma a aprofundar os conhecimentos relativos à análise forense da Bitcoin. Convém ainda realçar, que foram consultadas páginas na Internet alusivas à *blockchain* [13], Bitcoin *mining* [14][15], investimentos em Bitcoin [16], *softwares* de auditoria Bitcoin [17][18][19][20][21][22] e *mining pools* [15].

A pesquisa efetuada sobre os *softwares* de auditoria Bitcoin é descrita na secção 2.4.

## 2.2 Bitcoin

Bitcoin é uma criptomoeda descentralizada, uma versão puramente ponto a ponto de dinheiro eletrónico, que permite que pagamentos *online* sejam efetuados diretamente de uma parte para outra, sem passar por uma instituição financeira [23].

As *bitcoins* são divisíveis em "Satoshi", equivalente a 0,00000001 Bitcoin. Cada Bitcoin *holding* (conjunto de *bitcoins*) é identificado por um endereço análogo ao número de série de uma nota de banco, um *hash* [24].

O problema que se coloca com a generalidade de transações, com recurso a moedas digitais sem um intermediário, é que não se pode ter a certeza se um proprietário anterior não tenha gasto a mesma moeda duas vezes. A este problema dá-se o nome de *double-spending*. Quem recebe o dinheiro necessita de uma prova de que, no momento de cada transação, a maioria dos "nós" (um "nó" é um computador conectado à Internet, que possui uma cópia completa ou abreviada do *blockchain* da Bitcoin [25]) concordou que este foi recebido. Este problema foi resolvido com recurso a um *timestamp server*, que transmite o *hash* de um bloco de transações a toda a rede [26].

A Bitcoin é, na verdade, construída com base na *blockchain*, uma aplicação distribuída que recorre a mecanismos criptográficos para implementar livros-razão, ou livros de registo (designados *ledgers* na literatura anglo-saxónica), de forma distribuída e segura [13]. Podemos pensar na *blockchain* como um sistema operativo, como Microsoft Windows ou MacOS, e na Bitcoin como apenas uma das muitas aplicações que podem ser executadas nesse sistema operativo [27].

Os livros de registo, implementados com recurso a *blockchain*, são cadeias (*chain*) lineares de blocos de dados (*block*), ordenados por ordem cronológica. Cada bloco contém um ou mais registos imutáveis de transações [13].

A rede *blockchain* tem as seguintes propriedades-chave [27]:

- **Consenso**: Para uma transação ser válida, todos os participantes têm de concordar quanto à sua validade;
- **Proveniência**: Os participantes sabem de onde vem o ativo e como os seus proprietários mudaram ao longo do tempo;
- **Imutabilidade**: Nenhum participante da rede pode alterar uma transação após ter sido adicionada ao livro de registo. Se uma transação estiver em erro, uma nova deve ser utilizada para reverter esse erro. Ambas as transferências ficam visíveis;
- **Finalidade**: Um único livro de registo partilhado fornece um lugar para se determinar a propriedade de um ativo ou a conclusão de uma transação.

Um participante da rede armazena os endereços associados a cada pacote de Bitcoin que possui numa carteira. Semelhante a uma carteira de dinheiro convencional, o saldo de uma carteira de Bitcoin é a soma dos saldos de todos os endereços dentro da carteira. Embora os endereços Bitcoin individuais sejam projetados para serem anónimos, quando é usado mais do que um endereço para fazer uma compra, é possível inferir que os endereços pertencem à mesma carteira [24]. Normalmente, é utilizada uma chave privada de forma a aceder à carteira e autorizar transações, no entanto existem carteiras *multisig*, em que várias chaves privadas são necessárias para fazer as mesmas ações de uma carteira dita “normal” [28].

*Mining* é o processo de validação de novas transações, construindo e armazenando-as em blocos e, posteriormente, adicionados à *blockchain* [3]. *Bitcoins* são “*mined*” (criadas) ao adicionar registos de transações ao livro de registo público. Isto é feito com recurso à resolução de enigmas que aumentam deterministicamente de dificuldade e, uma vez resolvidos, podem ser facilmente verificados. Cada solução resulta num novo “bloco” e fornece a quem “minera” uma recompensa (atualmente 12,5 *bitcoins*), que serve de incentivo. Além de expandir a oferta de Bitcoin, cada bloco confirma uma coleção de transações recentes (transações desde o último bloco). Cada bloco também contém uma referência ao último bloco, formando assim uma "cadeia", dando origem ao termo *blockchain* [24].

Quando os blocos são gerados ao mesmo tempo ou muito próximos, o protocolo tem de lidar com um problema de consenso distribuído. A Bitcoin oferece uma solução única para o problema de consenso em sistemas distribuídos, uma vez que o poder de voto é diretamente proporcional ao poder de computação.

As transações são efetuadas ao assinar digitalmente, com a chave privada de quem efetua o pagamento, o *hash* da transação anterior e a chave pública do recetor da transferência. A assinatura na mensagem tem como função provar a autenticidade da transação a todos na rede. Após este processo, estas assinaturas são colocadas no final da cadeia (*public ledger*), permitindo que, quem recebe o dinheiro, possa verificar por onde este passou [23].

A solução implementada pelos criadores da Bitcoin para o problema do *double-spending* foi a criação de um *timestamp server* que faz com que seja possível provar que, na altura da transação, a maioria dos "nós" concorda que o dinheiro já foi recebido. Cada novo *timestamp* inclui o *timestamp* anterior no seu *hash*, formando uma cadeia, com cada *timestamp* adicional reforçando os anteriores [23].

Para ser possível implementar um *timestamp server*, é necessário um mecanismo de consenso, que neste caso, é um sistema de *proof-of-work*. Este sistema é realizado ao incrementar um *nonce* (número arbitrário que só pode ser usado uma vez) no bloco, até que seja encontrado um valor que forneça ao *hash* do bloco os *bits* zero necessários. Para modificar um bloco anterior, um atacante teria de voltar a fazer o *proof-of-work* do próprio bloco e de todos os blocos posteriores, o que é computacionalmente impraticável [23].

O modelo bancário tradicional atinge um nível de privacidade ao limitar o acesso das informações às partes envolvidas e a um intermediário de confiança (Figura 5). No caso da Bitcoin, a privacidade é conseguida através do "pseudoanonimato", ou seja, o *ledger* que contém todas as transações é público, mas as únicas informações disponíveis são os endereços de origem, o valor da transferência e os endereços recetores (Figura 6). Os proprietários dos endereços Bitcoin não são identificados explicitamente. Por outras palavras, qualquer pessoa pode ver que alguém está a enviar uma certa quantia a um destinatário, mas sem informações que ligam essa transação a alguém [23].

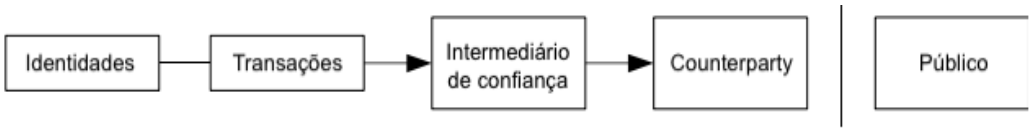


*Figura 5: Modelo de privacidade tradicional [23]*


*Figura 6: Novo modelo de privacidade [23]*

Como método de segurança adicional, um novo par de chaves deve ser usado para cada transação de modo a evitar a associação a um proprietário comum. Algumas transferências utilizam vários endereços como *inputs*, que necessariamente revelam que são do mesmo proprietário. Contudo, se o proprietário de uma chave for revelado, a ligação poderá divulgar outras transações que pertenceram ao mesmo dono [23].

Semelhante a qualquer investimento especulativo, comprar Bitcoin acarreta alguns riscos bem conhecidos, como por exemplo, o preço pode cair vertiginosamente e, apenas um ataque (*hacking*) à carteira (por exemplo, obter as credenciais de acesso) ou falha no seu armazenamento (por exemplo, carteiras de papel podem-se danificar ou carteiras de *hardware* podem ter algum tipo de avaria), pode fazer desaparecer todas as *bitcoins* ali depositadas. A Bitcoin tem registado aumentos muito consideráveis no preço, seguidos de algumas quedas. Investir em Bitcoin é semelhante a investir em ações, mas é muito mais volátil devido às suas oscilações diárias [16].

De uma forma resumida, o investimento em Bitcoin pode de alta recompensa e de alto risco, ou seja, tanto se consegue obter enormes lucros num curto espaço de tempo, como perder muito se o valor cair [16].

## 2.3 Análise Forense

O rápido crescimento da Bitcoin e o anonimato que fornece aos seus utilizadores criaram certos desafios relativos às regras impostas pela sociedade. Embora a Bitcoin tenha potenciais benefícios, como taxas mais baixas de transação, segurança e privacidade, as preocupações giram em torno do seu uso em comércio ilegal, financiamento terrorista e lavagem de dinheiro [24]. Com a criação de mecanismos de pagamento digital e anónimo, como a Bitcoin, houve um crescimento dos mercados "*darknet*" *online*, nos quais bens e serviços ilegais são comercializados [24].

Qualquer pessoa pode utilizar *bitcoins* com fins maliciosos e, por isso, existem muitas transações associadas a atividades ilegais. De acordo com uma pesquisa descrita por J. Han, J. Woo e J. W. K. Hong [10], cerca de US $ 35 mil milhões de atividades ilegais ocorrem anualmente nos EUA e na Europa, por exemplo, para compra e venda de armas, drogas e para lavagem de dinheiro. Posto isto, a deteção e a distinção entre transações ilegais e legítimas são essenciais para que a tecnologia Bitcoin se desenvolva num sistema financeiro de próxima geração [10].

O artigo "*Sex, Drugs, and Bitcoin: How Much Illegal Activity Is Financed through Cryptocurrencies?"* apresenta duas abordagens para estimar a população que pratica atividades ilegais, tais como, lavagem de dinheiro, tráfico de droga e o financiamento de

atividades terroristas. A população que pratica atividades ilegais é estimada com base em apreensões de *bitcoins* (combinadas com algumas outras fontes), que fornecem uma amostra de utilizadores conhecidos pelo facto de estarem envolvidos nesse tipo de atividades [24].

Face ao exposto e tendo conhecimento que a Bitcoin tem sido alvo de roubo e de outras atividades ilegais, um dos pontos-chave para a investigação forense é identificar endereços envolvidos em transferências ilícitas [11]. Assim, a primeira abordagem incide sobre as redes comerciais de utilizadores conhecidos por estarem envolvidos em atividades ilegais. É usada a *blockchain* Bitcoin para reconstruir a rede de transações entre os participantes do mercado. De seguida, é aplicado um tipo de análise de *cluster* de rede para identificar duas comunidades distintas nos dados, as comunidades legais e ilegais [24].

A segunda abordagem explora características, como por exemplo, o número e o tamanho das transações, que distinguem a utilização legal e ilegal de *bitcoins*, que podem ser usadas para a correspondência de padrões (definidos por um conjunto de propriedades de acordo com as pistas fornecidas). Esta correspondência tem como objetivo encontrar endereços que satisfaçam as características fornecidas [11], usadas em modelos de equações simultâneas que identificam a atividade ilegal, enquanto contabilizam a não aleatoriedade da amostra de utilizadores ilegais conhecidos [24].

Com estas abordagens, os autores do artigo chegaram à conclusão de que os utilizadores ilegais de Bitcoin tendem a realizar mais transações, em valores de menor porte, muitas vezes de forma repetida com uma determinada contraparte (entidade com a qual um determinado utilizador fez transações) e tendem a reter menos *bitcoins*. Esses recursos são consistentes com o uso da Bitcoin como sistema de pagamento, e não para investimento ou especulação. Os utilizadores ilegais também fazem maior uso de técnicas de transação que tentam esconder a sua atividade, como por exemplo, *CoinJoin* (método que combina múltiplos pagamentos de Bitcoin de vários *inputs* numa única transação para tornar mais difícil determinar qual a principal origem da transferência) e os seus picos de atividade (alturas em que existe um aumento do número de transações) após fecho de mercados de *darknet* e/ou apreensões policiais para realocação dos seus ativos. A quantidade de atividades na rede Bitcoin, associada ao comércio ilegal, diminui, por um lado, com o aumento de interesse por parte da população em geral (por exemplo, subida do valor de mercado e intensidade de pesquisa no Google) e, por outro, com o surgimento de criptomoedas alternativas, que fornecem mais privacidade que a Bitcoin e com menos mercados *darknet* ativos [24].

## 2.4 *Softwares* de Auditoria Bitcoin

*Softwares* de análise da Bitcoin fornecem uma ferramenta útil para quem deseja inspecionar a rede de transações em termos de, por exemplo, relações entre transações. Além disso, à medida que as criptomoedas prosperam e crescem como método de pagamento convencional, os *insights* sobre como as pessoas gastam as suas moedas virtuais tornam-se cada vez mais relevantes. Não apenas em termos de quais produtos ou serviços são comprados, por exemplo, num mercado negro online (exemplo, *Silk Road*), mas também o conhecimento de quanto tempo as pessoas mantêm a criptomoeda nas suas carteiras. Para as autoridades, a identificação desse tipo de atividades é importante para prevenir a lavagem de dinheiro, o financiamento do terrorismo, entre outros. Com recurso à análise das transações, os investigadores tentam combinar conexões e interações entre endereços. Algumas ferramentas já começaram a indexar conjuntos de transações com o objetivo de os reunir em grupos específicos [29], por exemplo, ao ter conhecimento de um endereço utilizado em compra e venda de produtos ilegais, são identificados endereços e transações associados à atividade do endereço inicial.

As investigações de *cybercrimes*, em geral, e de *ransomware* (forma de *malware* que encripta os arquivos da vítima e o atacante exige um resgate para restaurar o acesso aos dados, após o pagamento), confiam cada vez mais em ferramentas analíticas de *blockchain*, uma vez que muitos ataques geralmente usam criptomoeda para obter resgate. Os investigadores foram capazes de identificar pegadas digitais reveladores de informações relevantes acerca das identidades por trás desses crimes [29].

Nesta secção é feita uma comparação de *softwares* de forma a obter um melhor entendimento do que se faz em auditoria de Bitcoin e quais os melhores e mais completos para o fazer. Para facilitar esta comparação foi construída a Tabela 1, na qual estão reunidas as ferramentas selecionadas e os parâmetros utilizados.

Começou por se estudar o *BlockSci*, ferramenta já identificada previamente e definida como prioritária devido às suas características. De seguida, foi feita uma pesquisa no *IEEExplore* com as palavras-chave “Bitcoin”, “*analytics*” e “*forensics*”, que retornou dois resultados, dos quais, um deles era relativo a uma ferramenta de *mining*, e daí ter sido descartado. A partir do artigo selecionado [29] e da consulta do site https://www.1337pwn.com/best-open-source-blockchain-forensic-analysis-tools/ foram identificados os *softwares* abordados, que passamos de seguida a apresentar. Em ferramentas como *Matbea.net*, *Wallet Explorer*, *GraphSense*, *SoChain*, *SmartBit*, *Bitcoin Transaction Explorer* ou *Blockstream.info*, não foram encontradas mais informações para

serem descrita de uma forma completa, apesar de terem sido consideradas importantes para a comparação efetuada na Tabela 1.

### 2.4.1 Parâmetros de comparação

De acordo com a literatura consultada [29][30][5][4][3][31] (conjunto de artigos já analisados, o artigo encontrado na pesquisa descrita no paragrafo anterior e o *site* referido) e informações fornecidas pelos orientadores, foram identificados parâmetros para a comparação dos *softwares.* Os dois primeiros (análise visual e não-visual) são frequentemente utilizados para distinguir e comparar ferramentas nas secções "*Related work"*. Os parâmetros *Parser, Address clustering, Address tracking e Address tagging,* foram considerados como funcionalidades importantes num *software* de auditoria Bitcoin [29]. Os restantes, para além de terem sido sugeridos pelos orientadores, foram evidenciados na literatura consultada.

Apresentamos, de seguida, a comparação dos *softwares*, tendo como base os parâmetros:

- **Análise visual** – se tem um tipo de análise mais visual, com recurso a imagens 2D e 3D para facilitar a observação das análises executadas;
- **Análise não-visual** – se utiliza estatísticas descritivas e análises da rede para descrever propriedades da *blockchain;*
- ***Parser*** – se produz "*Core blockchain data"* (conjunto principal de dados para análise, gerado de forma a facilitar e tonar mais rápido cada estudo realizado)*,* que pode ser atualizado de forma incremental à medida que novos blocos chegam;
- ***Address clustering*** – se agrupa endereços que correspondem à mesma entidade ou estão ligados de alguma forma com algoritmos, em *clusters*;
- ***Address tracking*** – se faz o seguimento de como *bitcoins* de determinadas fontes (*inputs* de transações) são gastas ao longo do tempo por meio de transações e, eventualmente, são armazenadas em coletores (*outputs* de transações não gastas);
- ***Address tagging*** – se rotula endereços com identidades do mundo real;
- **Disponibilidade de código fonte**;
- **Linguagem de programação usada**;
- **Licença do código fonte** – onde podemos verificar o tipo de utilização que podemos dar ao código existente em *open-source*;
- **Atividade no projeto** – se continua com desenvolvimento ativo;
- **Apoio** – se há fóruns ou outros meios de apoio para quem utiliza e qual a data da última mensagem;

- **Instala e corre**;
- **Possibilidade de especificar as *queries* desejadas** – de forma manual, para fazer análises personalizadas ao que se pretende, por exemplo, se o utilizador quiser aceder aos *inputs* ou *outputs* de uma transação pelo *hash*, ser possível pesquisar;
- **Permite exportar resultados** – por exemplo, para o *Excel;*
- **Permite análise de períodos específicos ou datas especificas** – por exemplo, entre outubro e dezembro de 2020.

### 2.4.2 *BlockSci*

*BlockSci* é uma plataforma de *software* de *open-source* para análise de *blockchain*. É uma ferramenta versátil que dá suporte a diferentes criptomoedas e tarefas de análise. Incorpora uma base de dados analítica (em vez de transacional) na memória, tornando-a 15 a 600x mais rápida do que as ferramentas existentes [30]. Tem características muito interessantes, a destacar:

- Não está limitada à Bitcoin (inclui *Litecoin*, *Namecoin* e *Zcash*) [30].
- Inclui uma biblioteca de ferramentas analíticas e de visualização úteis, como a identificação de transações especiais (por exemplo, *CoinJoin*) e *links* de endereços entre si com base em heurísticas bem conhecidas (por exemplo, os *inputs* utilizados na mesma transação são controlados pela mesma entidade) [30].

O *BlockSci* possui dois componentes principais: o *parser* e a biblioteca de análise. Enquanto o *parser* gera os ficheiros de dados do *BlockSci*, a biblioteca de análise pode ser apontada para os dados gerados.

Foi utilizada uma máquina virtual com o sistema operativo *Ubuntu* 20.04 para a instalação e realização de experiências com a ferramenta. Na instalação, era necessário ter uma versão anterior do *Ubuntu* (18.04) e, por isso, foi utilizada uma imagem *docker* para ser possível a utilização do *software*. Depois de criada a imagem *docker* e seguir os passos de instalação do site https://citp.github.io/BlockSci/setup.html, foi feito o *parse* da *blockchain* para a utilização da biblioteca de análise disponível. Foi então gerado um ficheiro de configuração com as seguintes informações:

- Nome do ficheiro;
- Qual o tipo de moeda que se pretendia analisar, neste caso a Bitcoin;
- Nome da diretoria de destino, ou seja, para onde vão os ficheiros de dados do *BlockSci;*
- Nome da diretoria onde estão os dados da Bitcoin.

Após a criação deste ficheiro, foi feito o *parse* da *blockchain* com o comando: `blocksci_parser <nome_ficheiro> update`.

De seguida, foram feitos alguns testes para adaptação à ferramenta. Começou-se por fazer um bastante simples, que retornava o número de transações em cada bloco. Ao imprimir o comando `chain.blocks.select(lambda b: b.txes.size)`, o programa retornou o número de transações dos primeiros e dos últimos blocos analisados. O resultado foi "[1 1 1 ... 26 808 56]", que mostra o número de transações confirmadas em cada bloco, desde os primeiros identificados, em que apenas havia uma, até aos últimos onde havia mais, nos quais podemos verificar 26, 808 e 56 transações confirmadas. O segundo teste foi feito de forma a retornar todas as transações, em que a *transaction fee* fosse superior a 100 *satoshis*, com os comandos:

- `txes = chain.blocks.txes.where(lambda tx: tx.fee > 100)`
- `txes.to_list()`

Esta pesquisa resultou numa lista de grandes dimensões, o que significa que existem muitas transações com taxa superior a 100 *satoshis* com o seguinte formato:

- `[Tx(len(txins)=1,len(txouts)=2,size_bytes=235,block_height=2817,tx_index=2862), Tx(len(txins)=2, len(txouts)=2, size_bytes=414, block_height=2817, tx_index=2863), …]`

Como se pode verificar, para cada transação que fosse ao encontro do requisito acima descrito eram apresentadas informações como, o número de *inputs*, *outputs* e o tamanho em *bytes*. Por exemplo, na primeira transação, conseguimos verificar que foram utilizados dois endereços como entrada e dois como saída, enquanto na segunda, dois *inputs* para dois *outputs*. Caso existisse interesse, era possível aceder a cada elemento pelo seu índice na lista e assim aceder a outros valores de cada transação, como por exemplo, o seu *hash,* lista de *inputs,* lista de *outputs,* valor da *transaction fee* e o bloco que contém esta transação. O terceiro e último teste bem sucedido foi feito com o objetivo de encontrar a transação com o maior valor em BTC e algumas das suas propriedades. Foi conseguido da seguinte maneira:

- `txe_max = chain.blocks.txes.max(lambda tx: tx.fee)`
- `print(txe_max)`
- `print(txe_max.hash)`
- `print(txe_max.inputs.value)`
- `print(txe_max.outputs.value)`
- `print(txe_max.fee)`

Em que os resultados foram os seguintes:

- `Tx(len(txins)=11, len(txouts)=11, size_bytes=2011, block_height=254642, tx_index=22869692)`, a transação em questão;
- `4ed20e0768124bc67dc684d57941be1482ccdaa45dadb64be12afba8c8554537`, o *hash*;
- `[2000000000 2000000000 2000000000 2000000000 2000000000 2000000000 2000000000 2000000000 2000000000 2000000000 30144158000]`, os valores de *input* em BTC;
- `[2000000000 2000000000 2000000000 2000000000 2000000000 2000000000 2000000000 2000000000 2000000000 2000000000 10144158000]`, os valores de *output* em BTC;
- `20000000000`, o valor da *transaction fee.*

Com isto, conseguimos perceber o que se pode retirar de cada transação, de modo a ser utilizado noutras análises ou estudos e, neste caso, qual a transação com maior taxa.

Após esta adaptação ao *software*, foram feitos alguns testes para provar que a ferramenta funcionava da forma esperada, ao utilizar casos reais.

O primeiro caso estudado [5] foi o "*Payment to a killer?".* Em março de 2013, o fornecedor do *Silk Road* (mercado negro *online* e o primeiro mercado *darknet* moderno, mais conhecido como uma plataforma de venda de drogas ilegais), *FriendlyChemist*, supostamente, tentou chantagear o DPR ("*Dread Pirate Roberts*", alegado criador e operador do mercado negro "*Silk Road*") pelo sistema de mensagens privadas do *Silk Road*, afirmando ter provas de que possuía nomes e endereços de milhares de fornecedores. Exigiu US $ 500.000 pelo seu silêncio. O DPR pediu a outro utilizador, *redandwhite*, para "executar" o *FriendlyChemist*, fornecendo-lhe o seu nome completo e endereço. Em 31 de março de 2013, depois de concordar com os termos, o DPR enviou 1.670 BTC a *redandwhite* para matar o *FriendlyChemist*. O endereço do assassino é 1MwvS1idEevZ5gd428TjL3hB2kHaBH9WTL [5]. Com recurso a outra ferramenta "*Blockchain Explorer"* foi pesquisado esse endereço e retirado o *hash* da transação relativa a 1.670 BTC. Após encontrar a transação com esse *hash*, foram retirados os valores aos *outputs* dessa transferência, onde estava o valor pago ao assassino "[167000000000 13264000000]", com recurso ao *BlockSci*.

O segundo caso testado [5] foi o de um endereço conhecido por ter muitas *bitcoins* (1933phfhK3ZgFQNLGSDXvqCn32k2buXY8a) [5]. Com a ajuda do "*Blockchain Explorer"* foi pesquisado esse endereço e retirado o *hash* da primeira transação efetuada. Com o *BlockSci* foram retirados os *inputs* da transferência [3742109304115 211453580000 47320470692 48853000000 45606370000 113817100], valores enviados para a pessoa em questão, num valor total de 40954.56541907 BTC, que equivale a US $ 953.325.630,18.

Neste conjunto de testes também foi feita uma tentativa de *address clustering*, que não foi bem sucedida devido a não se conseguir converter uma *string* com um endereço num objeto *"Address"*, algo já reportado aos autores por terceiros nos "*Issues"* do *Github* da

ferramenta. O que seria necessário para ser colocado como parâmetro no método que iria fazer o referido *cluster*, *"cluster_with_address".*

### 2.4.3 *Blockchain Explorer*

O *Blockchain Explorer* [32][33] está entre as ferramentas de análise de *blockchain* mais conhecidas e utilizadas [17]. Esta ferramenta fornece uma série de recursos rápidos e fáceis de implementar para rastrear transações individuais, ao mesmo tempo que apresenta uma quantidade significativa de informações práticas, incluindo gráficos e estatísticas, sobre toda a rede Bitcoin. Além disso, o *Blockchain Explorer* permite, a quem analisa, associar as transações a um nome, ou *tag,* para posteriormente facilitar a identificação. É possível classificar uma série de transações numa rede de utilizadores e conduzir análises de tráfego sobre movimentação de dinheiro, ao utilizar heurísticas de agrupamento para dados fornecidos pelo *Blockchain Explorer* e informações publicamente disponíveis em fóruns de criptomoedas [17].

Este sistema cobre três componentes. O primeiro é um indexador que cria um índice de transações e endereços e, em seguida, adiciona uma *tag* de endereços suspeitos descobertos com recurso a informações obtidas por autoridades policiais. O segundo componente consiste nos recursos de análise: uma função de pesquisa, que fornece estatísticas e informações em relação a um endereço; um visualizador, que mostra as relações de endereços por transação; e um localizador, que descobre o caminho da transação em qualquer endereço, enquanto agrupa vários endereços que podem pertencer a uma carteira. A terceira componente é a interface da *web* para o utilizador do sistema [17]. Na Figura 7 verificamos alguns aspetos da página inicial do *Blockchain Explorer*, onde é possível pesquisar transações, endereços ou blocos de transações; apurar quais as transações ou blocos mais recentes e observar algumas informações, como o preço médio de cada Bitcoin e o volume de transações.

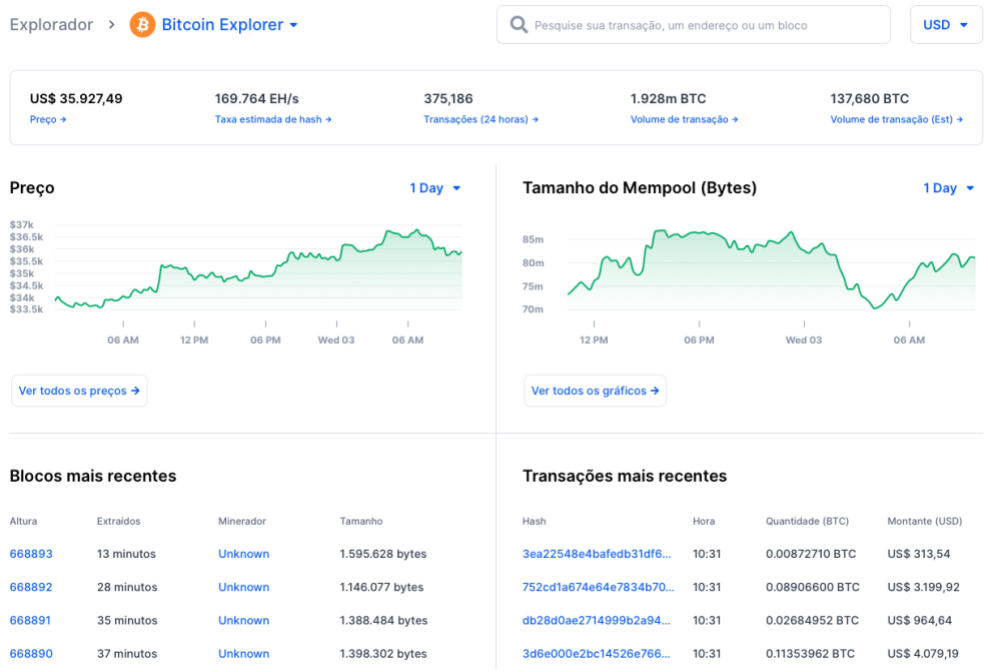


*Figura 7: Página inicial Blockchain Explorer [32]*

### 2.4.4 *Matbea.net*

*Matbea.net* foi projetado para receber informações sobre transações de criptomoedas, saldos de endereços e os seus proprietários [17]. Como se pode verificar na Figura 8, tanto é possível fazer uma pesquisa por transação, endereço, bloco ou *public key*, como ver quais os últimos blocos adicionados à *blockchain* (identificados pela "altura" (*height*), *timestamp*, tamanho em *bytes* e pelo *hash* do bloco). Um *software* algo parecido com o *Blockchain Explorer* [17].

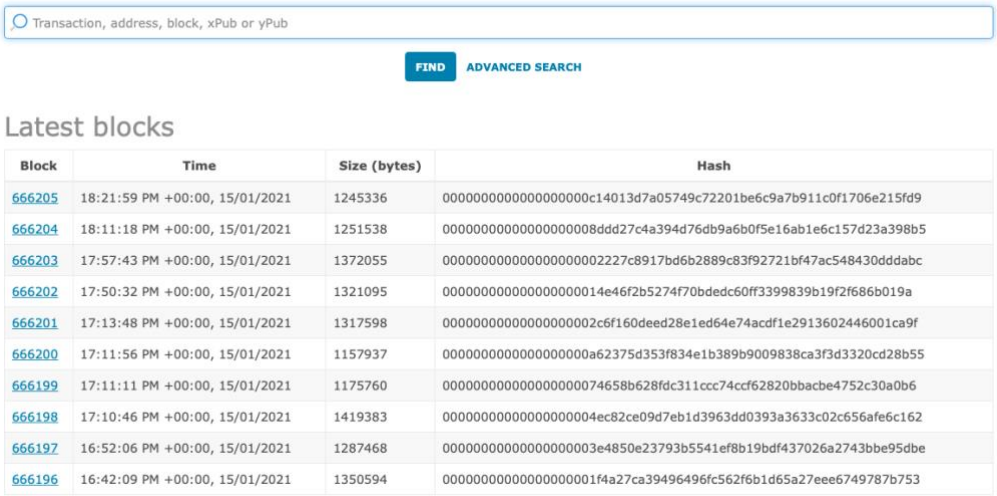


*Figura 8: Página inicial Matbea.net [18]*

### 2.4.5 *Wallet Explorer*

*Wallet Explorer* é uma ferramenta que monitoriza e exibe atividades dentro da rede Bitcoin. Este *software* utiliza uma heurística direta em que, no caso de dois endereços emergirem como entradas em qualquer transação específica são considerados como referentes ao mesmo indivíduo. É possível fazer diversas pesquisas, como por endereço, *hash* de transações e *wallet id.* E mostra ainda, quais são as carteiras mais conhecidas [17].

### 2.4.6 *OpReturnTool*

*OpReturnTool* é uma ferramenta utilizada para investigar *metadata* ("dados que fornecem informações sobre outros dados", ou seja, "dados sobre dados" [34]) relacionada à instrução OP_RETURN, que consiste num comando incluído no Bitcoin para inserir informações suplementares no *blockchain*. OP_RETURN é estabelecido para permitir até 80 *bytes* de dados e, quando uma transação, que compreende um campo OP_RETURN, é confirmada pela “mineração”, o conteúdo é inserido num bloco que persistirá dentro da *blockchain*.

*OpReturnTool* inspeciona as *blockchains* ao ler blocos consecutivos de blocos: cada bloco é analisado por um *thread* diferente. Cada “pedaço” analisado é definido no arquivo *input.txt* como um par de valores: o *hash* do primeiro e do último bloco do “pedaço”.

### 2.4.7 *BitConeView*

*BitConeView* é uma ferramenta para análise visual de fluxos de Bitcoin. É um *software* que permite fazer o seguimento gráfico de como *bitcoins* de determinadas fontes (entradas de transações) são gastas ao longo de um período de tempo por meio de transações e, eventualmente, como são armazenadas em vários coletores (saídas de transações não gastas). Pode ser usado para revelar padrões de fluxo Bitcoin de interesse em muitos domínios de aplicação, como acumulação, distribuição e mistura [4]. Contém análises como *Utxos Analysis*, em que analisa os *outputs* de cada bloco num determinado intervalo de tempo [4] e *Purity Analysis*, com o propósito de visualizar o quanto as *bitcoins* das transferências de origem foram “contaminadas” com outras *bitcoins*, ao longo do tempo [4].

O *BitConeView* analisa os dados extraídos do grafo de transações, designado de *tx-graph*. Para validar transações e blocos, os “nós” Bitcoin necessitam apenas de percorrer o *tx-graph*, em termos temporais, do presente para o passado. Em vez disso, o *BitConeView* avança de trás para a frente (no tempo) no grafo de transações e requer operações que não

são suportadas pelo *software* Bitcoin padrão. Nas experiências realizadas pelos criadores, são utilizados o *Insight-API* e *API REST,* disponibilizados pela *Chain.com*. Ambos permitem percorrer o grafo de transações para frente no tempo, mas fornecem diferentes compensações de eficiência e confiabilidade [4].

É uma aplicação baseada na *web*, a sua lógica compreende a exploração do *tx-graph*, cuja saída é o conjunto de dados que é mostrado ao utilizador. O servidor de aplicação é implementado em *Python* e utiliza estrutura *Flask*. Por fim, a renderização do *BitCone* é realizada pelo navegador da *web*, que também se encarrega da interação do utilizador. Este componente é baseado na biblioteca gráfica *D3.js* [4].

### 2.4.8 *GraphSense*

*GraphSense* é uma plataforma de análise de criptomoedas com ênfase em transparência algorítmica e escalabilidade. Fornece um painel para investigações *interativas* e controlo total de dados para a execução de tarefas analíticas avançadas. O *GraphSense* suporta as principais criptomoedas como Bitcoin, Bitcoin *Cash*, *Litecoin* e *Zcash*. Possui características como pesquisa de moeda, transações, inspeção e localização de caminhos. *GraphSense* usa *BlockSci* para analisar *blockchains* e filtrar *CoinJoins* [19]. Pode-se verificar um exemplo de utilização desta ferramenta na Figura 9.

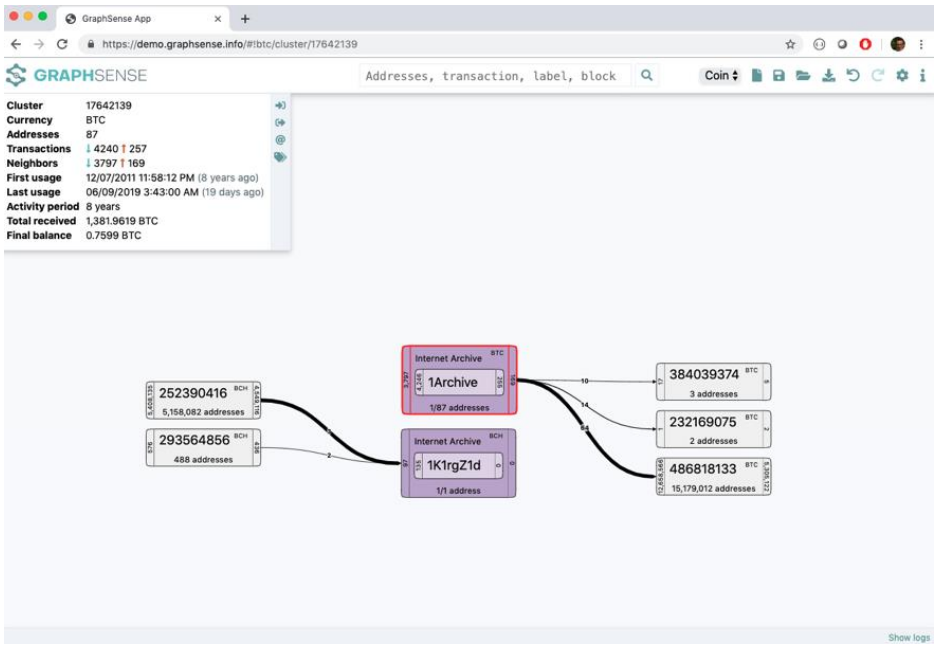


*Figura 9: Exemplo de utilização do GraphSense [19]*

### 2.4.9 *BitIodine*

O *BitIodine*, que analisa a *blockchain*, agrupa endereços, que provavelmente pertencem a um mesmo utilizador ou grupo de utilizadores. Classifica esses utilizadores e rotula-os, assim como, visualiza informações complexas extraídas da rede Bitcoin. O *BitIodine* rotula os utilizadores (semi-) automaticamente com informações sobre a sua identidade e ações, que são automaticamente retiradas de fontes de informação disponíveis ao público. O *BitIodine* também oferece suporte à investigação manual, encontrando caminhos entre endereços ou utilizadores [5].

Este *software* foi testado em vários casos do quotidiano. Por exemplo, foi descoberta uma conexão entre o fundador do *Silk Road* e um endereço que, provavelmente, pertence à carteira criptografada do *Silk Road*. Foi encontrada, também, a transação que, de acordo com o FBI, foi um pagamento do fundador do *Silk Road* a um assassino. Por fim, foi investigado o *ransomware CryptoLocker* e, começando por um endereço colocado por uma vítima, foi quantificado com precisão o número de resgates pagos e obtidas informações sobre as vítimas, com análise manual muito limitada [5].

### 2.4.10 *BitConduite*

*BitConduite* é uma ferramenta de análise visual para explorar a atividade financeira na rede Bitcoin. Este *software* oferece uma visão centrada na entidade das transações, tornando os dados acessíveis a especialistas não técnicos, por meio de um fluxo de trabalho guiado para classificação de entidades, de acordo com várias métricas de atividade. É também possível agrupar entidades por similaridade e explorar os dados de transações em diferentes escalas, desde grandes grupos de entidades até uma única entidade e as transações associadas [3].

O *BitConduite* consiste num *back-end* para preparação e gestão de dados, bem como para acesso a dados de alto desempenho, e um *front-end* com uma *interface* gráfica (*GUI*), que consiste em cinco *views* diferentes:

- ***Filter view***, painel que fornece uma visão geral sobre a distribuição temporal de transações, bem como histogramas para as medidas de atividade (número de transações, hora da primeira e da última transação, tempo ativo em dias, valor recebido em BTC, quantidade enviada em BTC, número de *inputs* e número de *outputs*);
- ***Tree view***, representa uma hierarquia de partições (também chamadas de classes e grupos) das entidades;

- ***Cluster view***, fornece funcionalidade para criar automaticamente grupos de entidades que partilham características semelhantes;
- ***Entity browser***, os resultados das etapas de filtragem e armazenamento em cluster no *BitConduite* são grupos de entidades, e o ***Entity browser*** é um navegador para esses grupos. Esta *view* ajuda a obter uma visão geral das características das entidades e a recuperar informações sobre entidades específicas;
- ***Transaction view***, exibe uma linha do tempo com transações para uma única entidade selecionada no ***Entity browser***. O que revela a distribuição temporal das transações e os valores.

São integradas numa aplicação *web* e atualizadas dinamicamente com cada alteração, e podem ser manipuladas independentemente para facilitar a exploração interativa dos dados [3].

### 2.4.11 *SoChain*

*SoChain* é um explorador de *blockchain* que exibe dados de um endereço, de transação e de bloco para Bitcoin e *altcoins* [17], como se pode ver na Figura 10. Oferece também uma *API* que permite criar aplicações em Bitcoin, *Litecoin*, *Dogecoin*, *Zcash* e *Dash*. É um *software* muito semelhante a outros, como por exemplo, o *Blockchain Explorer*.

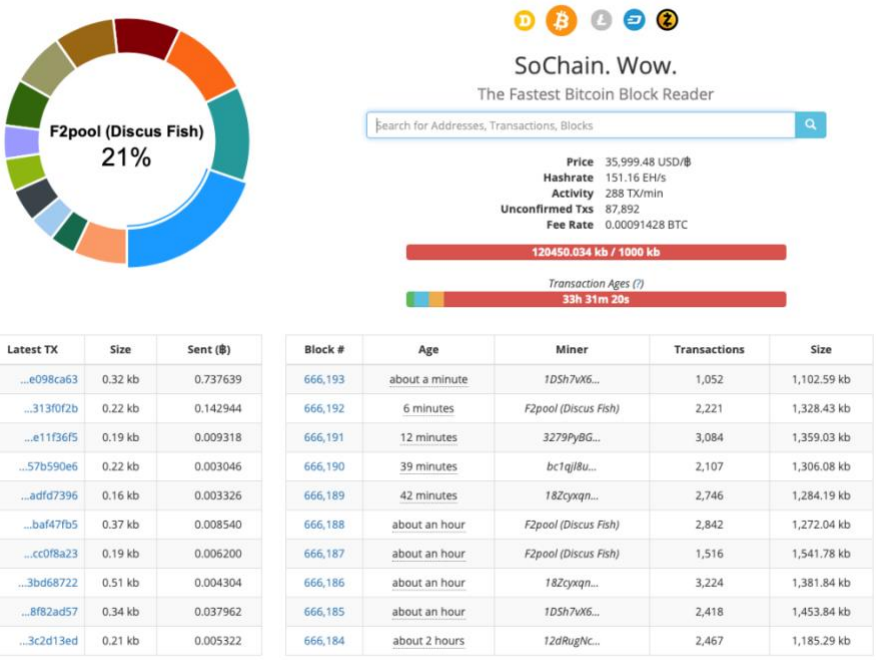


*Figura 10: Página inicial SoChain [20]*

### 2.4.12 *SmartBit*

*SmartBit* é outro *software* semelhante ao *Blockchain Explorer*, que inclui a capacidade de visualizar os blocos e transações mais recentes. A ferramenta fornece a contagem de blocos, de transações e *bitcoins* em circulação [17]. O *software* inclui recursos, como a visualização de blocos, de transações, de endereços, visualização de *scripts* e de *pools* de Bitcoin *mining* (grupos de pessoas que cooperam entre si para partilhar recompensas de Bitcoin *mining,* em troca de poder computacional para o fazer [15]).

### 2.4.13 *Bitcoin Transaction Explorer*

O *Bitcoin Transaction Explorer* é um explorador de blocos simples que o utilizador pode executar em cima de um "nó" completo. A ferramenta é executada como uma aplicação *web* que o utilizador pode correr em qualquer *J2EE Web* Container. O *software* inclui muitos recursos, como a visualização de blocos e de transações, um simulador de *mining* e visualização de *scripts* [17].

### 2.4.14 *BitExTract*

*BitExTract*, um sistema de análise visual interativo, é a primeira tentativa de explorar os padrões evolutivos de transação das *exchanges* de Bitcoin (*sites* que permitem aos utilizadores comprar e vender criptomoedas) de duas perspetivas, ou seja, *exchange versus exchange* e *exchange versus* cliente [35]. Em particular, *BitExTract* resume a evolução do mercado Bitcoin, observando as transações entre as compras e vendas da moeda ao longo do tempo por meio de uma exibição de sequência massiva. Um diagrama *node-link* descreve a rede comercial de *exchanges* e a sua distribuição de transação temporal. Além disso, o *BitExTract* incorpora várias barras paralelas numa linha de tempo para examinar e comparar os padrões de evolução das transações entre as diferentes *exchanges* [35].

É uma aplicação *web* com três partes principais, um módulo de armazenamento de dados, um de processamento de dados e um de visualização. O módulo de armazenamento de dados é baseado em *MongoDB* e reúne as informações necessárias, como transações de Bitcoin e notícias significativas sobre transferências de Bitcoin. O módulo *Python3* de processamento de dados agrupa as transações em duas categorias de transações de *exchange* para *exchange* e transações de *exchange* para cliente, e realiza agregação de tempo dos dados brutos na base de dados. Os dados processados são posteriormente utilizados para extrair estatísticas de alto nível e outros índices, como a participação de

mercado, o índice de posição da rede e a conectividade entre duas conversões de moeda. Estes dois módulos, baseados em *Python,* formam em conjunto o *back-end* e garantem ainda mais a eficiência do módulo de visualização. Este módulo permite que os utilizadores explorem os padrões evolutivos de transação de Bitcoin com três visualizações bem coordenadas. A exibição da sequência massiva resume a evolução do mercado de Bitcoin ao longo do tempo. A visualização da conexão descreve a rede comercial de *exchanges* e a sua distribuição de transação temporal. A visualização de comparação mostra as tendências de evolução de diferentes *exchanges*, levando em consideração as classificações da rede. Além disso, um painel adicional que lista todas os *sites* de compra e venda é fornecido para ajudar os utilizadores a obter uma visão geral e selecionar rapidamente uma determinada transação [35].

### 2.4.15 *Blockchair*

*Blockchair* é outro explorador e mecanismo de pesquisa universal de *blockchain*, útil para analistas e entusiastas de criptomoedas. Mostra diversas informações úteis, como gráficos e detalhes sobre transações e blocos de transações. Fornece uma *API* (paga) que contém acesso a dados contidos em 17 *blockchains*. A *API* oferece suporte a inúmeras consultas analíticas, como filtragem, classificação e agregação de dados de *blockchain* com tempo de atividade elevado [17].

### 2.4.16 *Blockstream.info*

*Blockstream.info* é outro explorador de blocos para Bitcoin e *Liquid*, que fornece informações de forma organizada. Permite explorar blocos, transações e endereços, mostrar *outputs* anteriores e detalhes de transações [17]. Tal como apresentado na Figura 11.

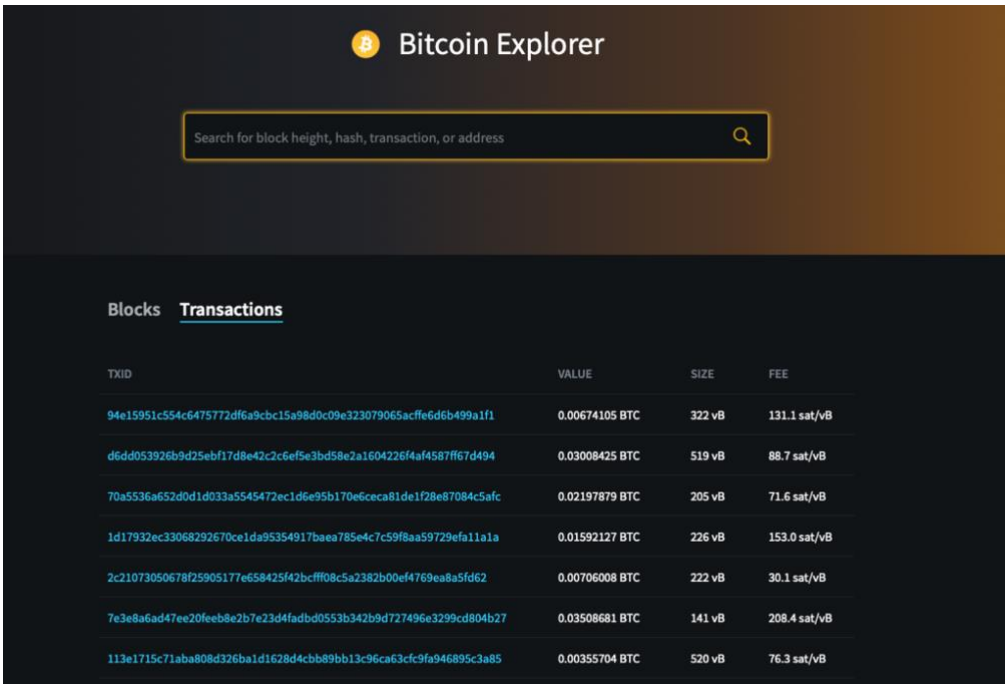


*Figura 11: Bitcoin Explorer [21]*

### 2.4.17 Outros *softwares*

A partir da pesquisa realizada e descrita no início desta subsecção, foram encontrados mais *softwares* que não foram utlizados para comparação, que apresentamos de seguida e pelas razões referidas:

- *BTCSpark,* desatualizado e com poucas informações. No *Github* da ferramenta direcionam para o *BlockSci* [22];
- *Bitcoin Block Explorer*, não funciona e sem informações;
- *Insight / Bitcore*, com poucas informações para proceder a uma comparação;
- *BlockPath*, muito semelhante a algumas ferramentas analisadas, para além disso, é significativamente mais lento a obter resultados, comparativamente aos outros;
- *BlockChainVis*, ferramenta em estado prematuro que se apresenta como sendo de análise visual, mas sem essa capacidade atualmente [36].

| | Análise visual | Análise não-visual | *Parser* | *Address clustering* | *Address tracking* | *Address tagging* | Disponibilidade de código fonte | Linguagem de programação usada | Licença do código fonte | Atividade no projeto | Apoio | Instala e corre | Possibilidade de especificar as *queries* desejadas | Permite exportar resultados | Permite análise de períodos específicos ou datas especificas |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BlockSci | Não | Sim | Sim | Sim | Não | Sim | Sim | C++, Python | Sim (GNU General Public License v3.0) | Até Novembro de 2020 | Sim | Sim | Sim | Não | Sim |
| Blockchain Explorer | Não | Sim | Sim | Sim | Sim | Sim | Sim | Python | Sim (MIT) | Sim | Sim | N/A (Versão web funcional) | Não | Não | Não |
| Matbea.net | Não | N/A | N/A | Não | Não | Sim | Não | JavaScript, PHP | N/A | N/A | N/A | Versão web funcional | Não | Não | Não |
| Wallet Explorer | Não | Não | N/A | Não | Não | Não | Não | N/A | N/A | N/A | N/A | Versão web funcional | Não | Sim | Não |
| OpReturnTool | Não | Não | N/A | Não | Não | Não | Sim | Java | Não apresenta m licença | Última atividade há 4 anos | Não | N/A | Sim | Não | Não |
| BitConeView | Sim | Sim | N/A | Não | Sim | Não | Não | Python | N/A | N/A | Não | Versão web funcional | Não | Não | Sim |
| BitIodine | Não | Sim | Sim | Sim | Sim | Sim | Sim | Rust | Sim (MIT) | Até junho de 2020 | Não | N/A | Sim | Sim | Sim |
| BitConduite | Sim | Não | N/A | Sim | Não | Sim | Não | N/A | N/A | N/A | Não | Não | Sim | Sim | Não |
| SoChain | Não | Não | N/A | Não | Não | Não | Não | N/A | N/A | N/A | N/A | Versão web funcional | Não | Não | Não |
| SmartBit | Não | Sim | N/A | N/A | N/A | Não | Não (Só algumas APIs) | N/A | N/A | N/A | N/A | Versão web funcional | Não | Sim | Não |
| Bitcoin Transaction Explorer | Não | Sim | Não | Não | Não | Não | Sim | Java | Sim (MIT) | Última atividade há 2 anos | Não | N/A | Sim | Não | Não |
| BitExtract | Sim | Não | Não | Não | Não | Não | Não | N/A | N/A | N/A | N/A | N/A | Não | Não | Sim |
| Blockchair | Sim | Sim | N/A | Sim | Não | Não | Não | N/A | N/A | Sim | Sim | Versão web funcional | Sim | Sim | Não |
| Blockstream.info | Não | Não | N/A | Não | Não | Não | Sim | JavaScript | Sim (MIT) | Sim | Sim | Versão web funcional | Não | Não | Não |
| GraphSense | Não | Sim | Sim | Sim | Sim | Sim | Sim | Python | Sim (MIT) | Sim | Sim | N/A | Sim | Não | Sim |

*Tabela 1: Comparação de softwares*

Devido à escassez de informações sobre certas ferramentas, não foi possível aprofundar algumas características e funcionalidades como era expectável. Com recurso à comparação apresentada na Tabela 1, conseguimos concluir que há três *softwares* que se destacam pela positiva, *BlockSci, Blockchain Explorer e GraphSense*, não só pela quantidade de informação disponível, mas também pelas funcionalidades que possuem. Outro aspeto que se pode retirar da tabela, é que a maior parte dos *softwares* analisados, ou não tem informações sobre se ainda se mantém (ou não) em atividade ou simplesmente não está ativo (os criadores deixaram de lançar novas atualizações ou correções do *software*). Para além disso, são poucas as ferramentas que permitem fazer *queries* desejadas, algo que limita quem quer aprofundar algum tipo de análise. Também são poucos aqueles que permitem exportar resultados, uma funcionalidade que facilitaria a documentação de análises feitas à rede Bitcoin. Para além disso, apenas quatro *softwares* permitem especificar períodos de tempo para análise, funcionalidade considerada muito importante para certos tipos de análise, como por exemplo, verificar por onde passou um certo endereço ao longo do último ano. Muitas ferramentas não disponibilizam o código fonte, algo que dificultou a comparação em alguns aspetos e eliminou a possibilidade de servirem de base para o desenvolvimento do nosso *software*.

Conclui-se, portanto, que o caminho a seguir seria utilizar o *BlockSci* como base e desenvolver uma ferramenta que reúna funcionalidades úteis para investigadores da área de economia e autoridades, que designamos de BitAudit.

# 3. Explorações Preliminares na Rede Bitcoin

De forma a obter um melhor entendimento de como funciona a rede Bitcoin, foram realizadas explorações preliminares, com recurso a ferramentas já existentes e disponíveis para utilização.

Estas explorações foram definidas após uma pesquisa bibliográfica efetuada na plataforma *IEEExplore*, utilizando palavras-chave, como: "*Bitcoin, exploration, network*", da qual foram obtidos seis artigos, dos quais foram filtrados dois [37][38]. Esta seleção prendeu-se, sobretudo, com os conteúdos apresentados e também pelo facto dos restantes quatro artigos já terem sido analisados na sequência de outras pesquisas efetuadas. Do artigo [38] foi retirado o [39] por fazer uma análise empírica de interesse à rede Bitcoin. Com as palavras-chave "*Bitcoin, Analysis, Network"*, houve um retorno de 281 artigos. Desta pesquisa foram retirados seis [40][41][42][43][44][45], por serem os que mais se relacionavam com o pretendido. Alguns dos excluídos eram duplicados, ou seja, já encontrados anteriormente.

Começou por se fazer uma análise mais generalizada, como por exemplo, a variação do preço da Bitcoin e o número de transações ao longo dos anos. O *software* selecionado para estas explorações iniciais foi o *Blockchain Explorer*, por ser o que fornecia um conjunto de recursos rápidos e fáceis de implementar para rastrear transações individuais, ao mesmo tempo que apresentava uma quantidade significativa de informações práticas, incluindo gráficos e estatísticas, sobre toda a rede Bitcoin.

Como se pode verificar na Figura 12, os valores de conversão de Bitcoin para as moedas convencionais, na imagem o Euro, continuam a bater recordes e a aumentar dia após dia. Isto mostra que o crescente interesse da população mundial e a contínua especulação em volta desta criptomoeda reforçam o estatuto da Bitcoin como a criptomoeda mais valiosa do mercado das moedas virtuais, baseadas na *blockchain*. Apesar disto, consegue-se entender também a volatilidade da moeda em questão, algo já referenciado na secção 2.2.

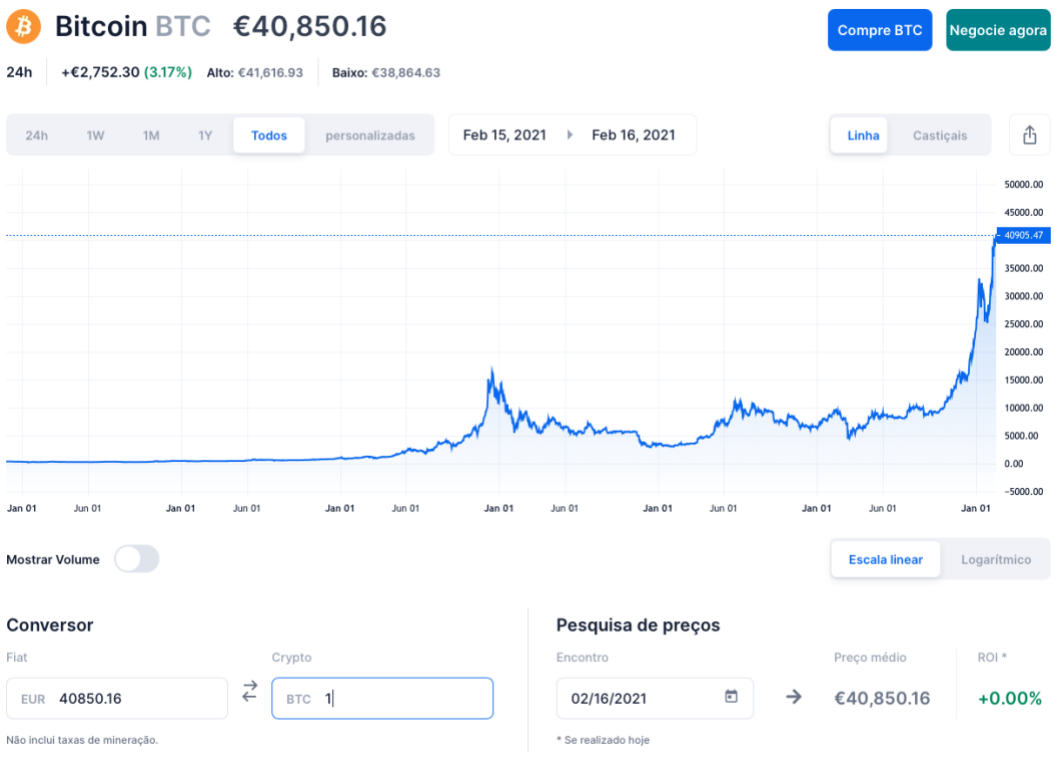


*Figura 12: Situação do valor da Bitcoin à data de escrita (16/02/2021) [32]*

Outro parâmetro considerado interessante nesta exploração é a receita dos *miners*, valor em Bitcoin recebido por cada verificação de um conjunto de transações, ou seja, um bloco. Na Figura 13, conseguimos observar algumas oscilações na recompensa atribuída. A principal razão para que isto aconteça é o Bitcoin *Halving,* em que, depois de cada 210.000 blocos minerados, ou de 4 em 4 anos, aproximadamente, a recompensa em bloco dada aos *miners* de Bitcoin, pelo processamento de transações, é cortada pela metade [46]. Este

fenómeno aconteceu em 2012, 2016 e 2020. Mas, apesar destas ocorrências, as receitas dos *miners* têm vindo a aumentar, mesmo após uma queda, devido à quantidade crescente de blocos confirmados.

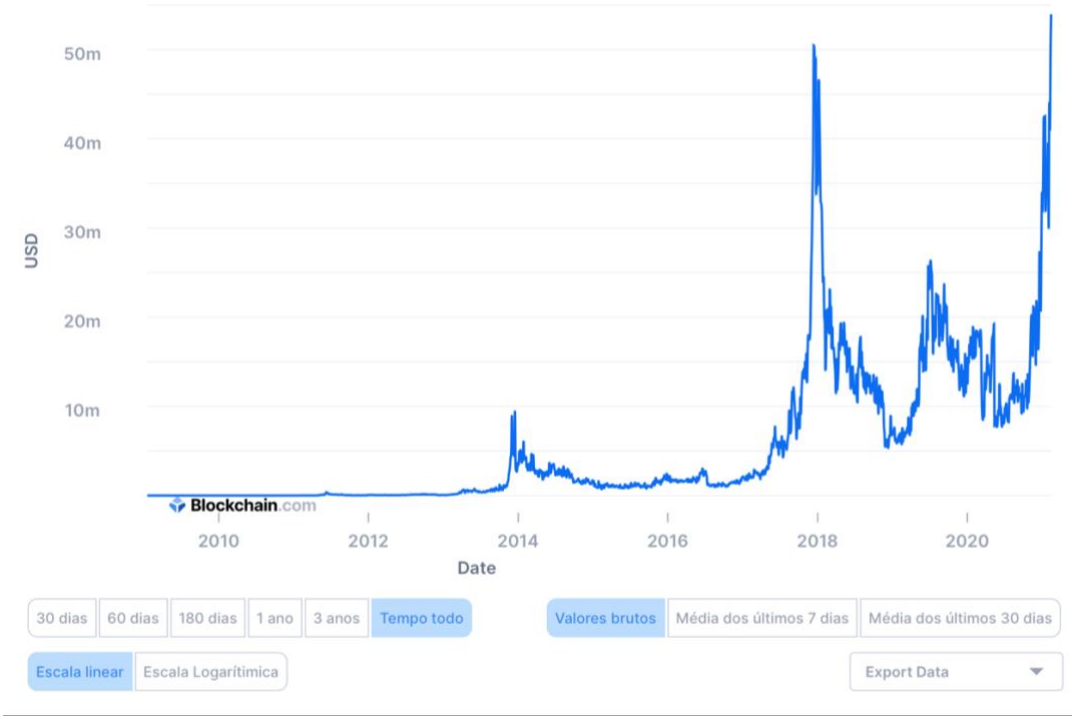


*Figura 13: Variação da receita dada aos miners de Bitcoin [32]*

Segundo I. Alqassem, I. Rahwan e D. Svetinovic, foram identificadas duas fases principais da vida do Bitcoin: a fase inicial e a fase de *trading* [42]. A fase inicial durou até o outono de 2010, durante a qual não havia nenhum valor no mundo real associado a um Bitcoin. De seguida, *MtGox*, uma exchange de *bitcoins*, foi fundada e a fase de *trading* de Bitcoin começou, através da qual esta moeda ganhou valor de mercado [42]. Por fim, a fase da especulação, identificada mas não analisada neste artigo [42], por se tratar da fase atual da Bitcoin. Como pode ser observado na Figura 14, com a Bitcoin a ganhar valor real de mercado, houve um grande aumento de transações. Isto também pode ser verificado com o início da fase de especulação, mas em números muito superiores, em que no dia com mais movimento em 2019, foram confirmadas quase 500.000 transações.


*Figura 14: Fases da Bitcoin [32]*

Ao analisar as variações de valor da Bitcoin (Figura 14) foram observadas algumas subidas e descidas interessantes de aprofundar. A primeira é uma descida localizada a partir de janeiro de 2018, após o conhecimento que a Coreia do Sul se preparava para proibir, completamente, transações com moedas digitais. A intenção tinha sido sugerida em dezembro (de 2017), quando o governo sul-coreano anunciou novas medidas para as transações regulares com as divisas digitais como, por exemplo, a utilização do nome verdadeiro para as transações. Dois meses antes, o país já tinha proibido o financiamento de empresas através do lançamento de novas criptomoedas [47]. A verdade é que, como se pode ver na Figura 15, existiu uma real descida de valor, ou seja, de valores a rondar 15.000€ por Bitcoin para valores a rondar os 6.000€.


*Figura 15: Descida de valor a janeiro de 2018 [32]*

A partir do início de 2019, verificou-se uma recuperação do valor anteriormente perdido, como se pode ver na Figura 16, até meados desse mesmo ano. A contribuir para esta subida esteve a postura mais otimista dos investidores, que estariam convictos de que as autoridades iriam aprovar legislação para que a negociação deste tipo de ativos fosse facilitada. O anúncio do *Facebook* sobre o lançamento de uma moeda própria virtual também ajudou a aumentar o otimismo no mercado [48]. Com esta aposta numa moeda digital, por parte de um "peso pesado tecnológico", é normal que o mercado das criptomoedas tenha maior interesse por parte dos investidores [49].

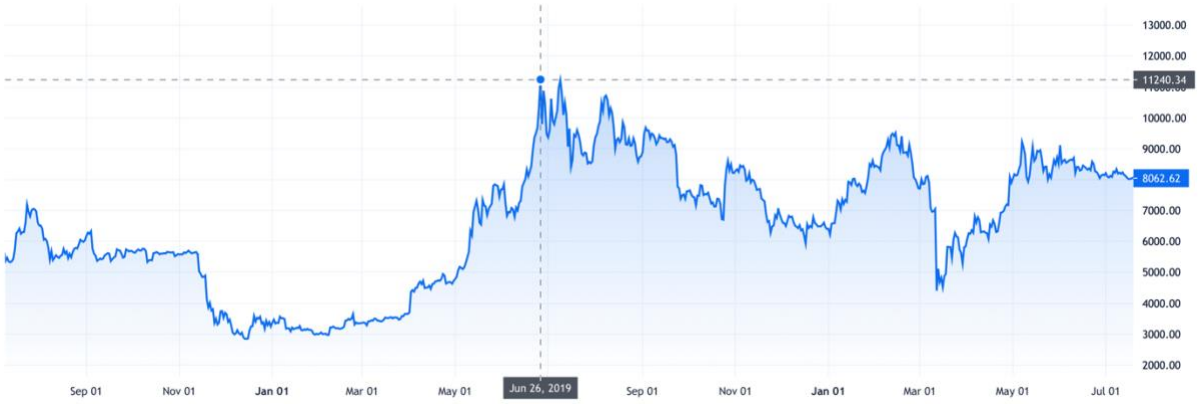


*Figura 16: Subida de valor na primeira metade de 2019 [32]*

Em março de 2020, a Bitcoin teve a maior queda diária da sua história (Figura 16). Do dia 11 para o dia 12 de março, esta criptomoeda registou uma desvalorização de 2.600€. Vários especialistas apontaram diversas razões para o sucedido, como por exemplo, queda das bolsas e a necessidade de liquidez. Uma empresa de investimentos chinesa, *Plus Teken,* foi acusada de ser um esquema em pirâmide e diz-se ter inserido 3 mil milhões de dólares em

criptomoedas [50], mas aquele que é considerado o maior motivo desta queda foi o início da pandemia na Europa e na América. A pandemia do COVID-19 e, consequentemente, a incerteza que gerou, fez com que ativos de risco, como a Bitcoin, pareçam ainda mais arriscados [50].

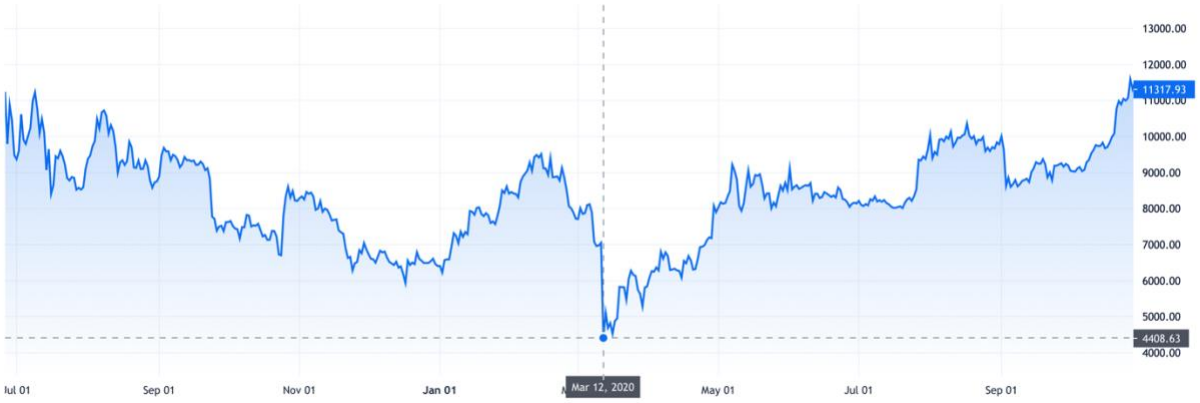


*Figura 17: Queda registada devido ao coronavírus [32]*

Por fim, verifica-se uma subida registada mais recentemente (1BTC = 42.665 €) (Figura 18), mas que ainda não pôde ser estudada de forma mais aprofundada. Contudo, considera-se que pode estar diretamente relacionada com o facto da *Tesla* ter feito um investimento de 1,5 mil milhões de dólares em *bitcoins*. A *Tesla* admitiu, inclusive, aceitar em breve criptomoedas como Bitcoin para a compra dos seus automóveis. Isto, aliado ao facto das constantes publicações no *Twitter* a favor das criptomoedas e da Bitcoin, podem ser ponderadas como as maiores razões para a valorização de mais de 20% em fevereiro do ano de 2021 [51].

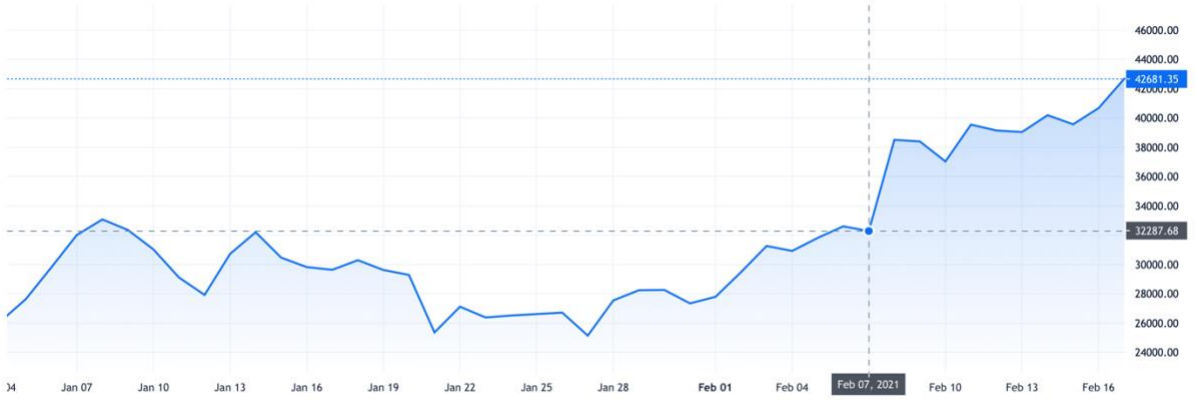


*Figura 18: Subida com o investimento da Tesla [32]*

Com recurso ao *Blockchain Explorer* conseguimos observar algumas características da rede Bitcoin. Começando pela recompensa dos *miners,* ao confirmarem blocos de transações*,* é possível concluir que apesar dos sucessivos cortes (Bitcoin *halving*), o valor total desta recompensa continua a aumentar devido à quantidade crescente de transações confirmadas. Para além disto, fomos capazes de identificar claramente as três fases de vida da Bitcoin, com recurso a um gráfico em que era possível observar as transações confirmadas ao longo dos anos. Por fim, verificamos algo já abordado na secção 2.2 relativo à Bitcoin como investimento, que é o facto de ser bastante volátil, ou seja, como certos acontecimentos ao longo dos últimos anos causaram quedas ou subidas no valor da Bitcoin.

Após este conjunto de análises, foi utilizado o *BlockSci* para a realização de explorações mais especificas, entre os anos 2009 e 2014 (inclusive, até maio), que não eram possíveis com o *Blockchain Explorer*.

Em primeiro lugar, foi feito um estudo relacionado com *Multisignatures* [30]. *Multisignature* (*multisig*) refere-se à solicitação de várias chaves para autorizar uma transação Bitcoin, em vez de uma única assinatura de uma chave. Tem várias aplicações [52]:

- Dividir a responsabilidade pela posse de *bitcoins* entre várias pessoas;
- Evitar um único ponto de falha, tornando substancialmente mais difícil o comprometimento da carteira;
- *Backup* M-de-N, em que a perda de uma única chave não leva à perda da carteira.

Considerando estas aplicações, foi feito um estudo da percentagem de utilização. Concluiu-se que, durante o período de tempo indicado, apenas 0,412% dos endereços utilizados são *multisig*. Sendo que estas aplicações podem ser consideradas bastante vantajosas para os seus utilizadores, era expectável uma maior percentagem de utilização.

De seguida, foram realizadas análises definidas após indicação dos professores orientadores. A primeira foi feita de forma a tirar conclusões da generalidade das transações ao longo destes cinco anos, em que se concluiu que a menor transação tem o valor de 0 *satoshis*, a maior tem de 55.000.000.000.000 *satoshis,* a média do valor de transações encontra-se nos 4.523.293.994,4 *satoshis* e a mediana nos 67.300.000 *satoshis.* A segunda está relacionada com Bitcoin *mining,* em que foi possível verificar qual a média da recompensa recebida por bloco em cada ano, o número de blocos confirmados por ano (que pode conter centenas ou milhares de transações confirmadas) e a percentagem de blocos confirmados correspondentes a esse ano (relativamente aos cinco anos estudados). Os resultados foram os seguintes (valores arredondados):

- Ano: 2009
    - Média de recompensa dos *miners*: 5.000.008.833,5 *satoshis*

    - Blocos confirmados: 3.2490
    - Percentagem de blocos confirmados: 11%
- Ano: 2010
    - Média de recompensa dos *miners*: 5.000.064.766,7 *satoshis*
    - Blocos confirmados: 67.920
    - Percentagem de blocos confirmados: 22,5%
- Ano: 2011
    - Média de recompensa dos *miners*: 5.005.176.359,1 *satoshis*
    - Blocos confirmados: 5.9627
    - Percentagem de blocos confirmados: 19,8%
- Ano: 2012
    - Média de recompensa dos *miners*: 4.803.235.858,1 *satoshis*
    - Blocos confirmados: 5.4526
    - Percentagem de blocos confirmados: 18,1%
- Ano: 2013
    - Média de recompensa dos *miners*: 2.524.079.775,2 *satoshis*
    - Blocos confirmados: 6.3433
    - Percentagem de blocos confirmados: 21%
- Ano: 2014
    - Média de recompensa dos *miners*: 2.508.002.997,9 *satoshis*
    - Blocos confirmados: 2.3004
    - Percentagem de blocos confirmados: 7,6%

Face ao exposto, foi possível concluir algo já observado anteriormente, o *Bitcoin halving,* que ocorreu no final de 2012, daí o valor de recompensa dos *miners* ter diminuído ligeiramente em relação aos dois anos anteriores e ter-se estabilizado nos 25 BTC nos anos seguintes.

Após esta análise, foi realizado um estudo baseado numa heurística chamada “heurística de propriedade de entrada comum”, ou seja, consiste numa suposição de que se uma transação tem várias entradas (ou *inputs*), todas elas pertencem à mesma entidade [3][5][23][37][41][42][53]. Para este estudo, foi utilizado um endereço conhecido por estar associado a *CryptoLocker* [54], que consiste numa ameaça de *malware* que ganhou notoriedade nos últimos anos. É um cavalo de Troia que infeta o computador de um individuo e procura ficheiros para encriptar. Após o computador ser infetado, os arquivos são "bloqueados" usando o que é conhecido como encriptação assimétrica. Este vírus exibe janelas de aviso, indicando que os dados serão destruídos se não houver o pagamento de um resgate para obter a chave privada [55]. Com esta análise foi possível reunir um conjunto de

mais de 100 endereços diferentes e 11 transações associadas ao endereço “18iEz617DoDp8CNQUyyrjCcC7XCGDf5SVb”. Os resultados foram os que se seguem:

- ['18iEz617DoDp8CNQUyyrjCcC7XCGDf5SVb',
  '1KP72fBmh3XBRfuJDMn53APaqM6iMRspCh',
  '18SaxWfBP2oLqdZygaNa6eQfHgtKkX3YNo',
  '1HSPr92v7ypRxsJFPFFx4ai7GhXSQma56d',
  '1BFguNQ3CWURg8TTqkiowF9dWbh4dgcqo4',
  '16jMEqCjJn91ckzbw7MpKmc2ivG5PguYTB',
  '17w9bMf2DEcDUbpfVFodb34LpJShHdCTRv',
  '12Z71WcLDC6pLQiyWATU2xx91455PeE3ki',
  '1PUHb4rzzVw82vGWrTx2esTqWV34qsiLoM',
  '1AU4Bdf13bCkc2fTkyAQc4zA25kuJQbu8H',
  '14ujLQwtnQjnxyp7KtyHKhj8dsgoBkTwcb',
  '1548XtfEVgbKjkzbB3KeZUcmf8vyWWZcae',
  '193tqzCaEGbUDw114QwZYoLJvEoB3Ao5K8',
  '12xkY3XezVZUsetYYxutJSUUSLQ2byyRBJ',
  '1DXFJ9L8i8YTQomQnF9s7mmFwsBQ2UtDwy',
  '18SghQmtGjDaUre3yyhsfBajeMrEDAkEtJ',
  '1LDKfUJGBEbso4gdbnJQ3HBm78H7hd1j6g',
  '13S4DGMrq2FNeMCR6w3TQwVGGDN641kMTK', ... ]

- Transações associadas, com o formato [n.º de *inputs*, n.º de *outputs*, valor da transação, valor da taxa, *hash* da transação]:
  - [9, 2, 1501045762, 0, 03e5fa6bae2c35f29e2a8a65ce0595a18c57b71cf6dd179336d9e9653cb91c51],
  - [8, 2, 1586410000, 0, 65608f4da1af1ff8e11b6d5ca48ad8166dedca3afb8c8ba310f970a8f6c4b058],
  - [10, 1, 2000000000, 0, 31e9c25c34cb9cce4c817df428d8b23af3d0d2cd0bf21925471fc2f9f3b56107],
  - [10, 1, 2000000000, 0, cc12b842dfefde64f79f97b62b7cacec995bd66b239773e0aad3728dfe6073ff],
  - [15, 1, 3000000000, 0, 6ed885f91701f4c6106d3b5a5ac60c2d42107b7b275f864fe105c58e606324bc],
  - [20, 1, 4000000000, 0, 7f166f4809fdeb646969d92441742be066d444f88de5d4be930e688079f02bec],
  - [20, 1, 4000000000, 0, dbada88d4787e9ab12b7cebfd9876f9533bd4a52f5142bf89e8f59be37764c63],

- [20, 1, 4000000000, 0, 93e0b6db29c2a8d0e684e8ae2a24660cd45855fac1420d36ea1e334460cb72b7],
- [66, 2, 12720780439, 0, 47f1dd4af19a5a4175ad2f85224fd654c785b5c9c65405ecda4f8bee3fa43155],
- [91, 2, 17708620569, 140000, 83796a7652fd7d1838a699e89f5f7045a67e2c8efb72a4bc057f40652d310327],
- [8, 2, 1156471000, 0, a5f2f4c0674d5a633afd7e628d42a50b6745e7830f6a0aa6e7647d7e5fb08d55]

Com estes dados, podemos verificar que foram transações que utilizaram muitos *inputs,* seguramente devido ao valor de transação elevado. Também podemos ficar com uma ideia de quais os endereços relacionados com este tipo de esquemas de *ransomware*, dado que foram encontrados mais de 100 endereços.

Uma questão levantada por especialistas desta área, foi "se *bitcoins* são geralmente usadas como investimento ou transferidas ativamente, por exemplo, para fazer compras" [3]. A maioria das entidades está envolvida num baixo número de transações [3] e, para confirmação, foi realizada uma análise que mostra a percentagem de endereços que apenas concretizaram uma transação no período definido anteriormente. Foram encontrados 29339219 que respeitam esta regra, que corresponde a 86,4% de todos os endereços usados em transações no período estudado.

Contrariamente, foram identificados os endereços mais utilizados no período de tempo referido:

- "1dice8EMZmqKvrGE4Qc9bUFf9PX3xaYDpcom", com 1.581.406 transações, que corresponde a 4,07% das transações da *blockchain*;
- "1dice97ECuByXAvqXpaYzSaQuPVvrtmz6com", com 1.065.818 transações, que corresponde a 2,74% das transações da *blockchain*;
- "1VayNert3x1KzbpzMGt2qdqrAThiRovi8com", com 781.891 transações, que corresponde a 2,01% das transações da *blockchain*.

Com esta análise conseguimos identificar os três endereços com mais transações, que correspondem, respetivamente, a quase 9% de todas as transações deste período. Para além disto, os primeiros dois endereços estarão relacionados com sites de apostas [54].

Por fim, foi realizada uma análise baseada na heurística designada "*shadow address guessing*". A heurística está relacionada com o "troco" das transações. O protocolo da Bitcoin "força" que o valor de entrada (*input*) das transações seja igual ao de saída (*output*). Isto significa que o valor não utilizado de uma transação deve ser devolvido. Para melhorar o

anonimato, é criado um *shadow address* automaticamente, que é utilizado para recolher o "troco" de volta ao utilizador que efetuou a transação. A heurística tenta prever qual dos endereços de saída realmente pertence ao mesmo utilizador que iniciou a transação da seguinte forma: se houver dois endereços de *output* (um beneficiário e um endereço de "troco", o que é verdadeiro para a grande maioria das transações), e um dos dois nunca apareceu antes no *blockchain*, enquanto o outro sim, então podemos assumir, com segurança, que o endereço que nunca apareceu antes é o *shadow address* gerado pelo cliente para coletar o "troco" de volta [5]. Para isto, foi utilizado novamente o endereço associado *CryptoLocker,* "18iEz617DoDp8CNQUyyrjCcC7XCGDf5SVb", como valor de entrada desta análise. Foram então reunidos, a partir da heurística exposta, todos os endereços que reuniam as condições descritas:

- '1HSN65qWF7EtBgrD85kjqeuYH7P5UaaSaZ';
- '1CWV5ZtnMcTyeTfopL3RTbutYD8yDeHTBY'.

Estes endereços podem ser úteis em várias situações, como por exemplo, seguir as *bitcoins* da carteira ou identificar mais endereços relacionados com este tipo de *ransomware*.

Com estas explorações, foi possível obter um melhor entendimento da rede Bitcoin como um todo e definir com mais clareza as funcionalidades num *software* de auditoria de transações Bitcoin. Em síntese, salientamos:

- O valor de conversão tem vindo a aumentar de forma significativa nos últimos anos, o que mostra um crescente interesse;
- Apesar do *Bitcoin halving*, as receitas dos *miners* não diminuíram, até pelo contrário, com recurso ao *Blockchain Explorer* [32] e ao *BlockSci* [56];
- A partir da visualização de gráficos no *Blockchain Explorer* [32] foi possivel observar que existem três fases principais da vida do Bitcoin: inicial, *trading* e a de especulação;
- Foi confirmada a volatilidade da Bitcoin, com várias subidas e descidas, cada uma associada a algum acontecimento em especial;
- Apesar das vantagens do *Multisignature*, até maio de 2014 (*blockchain* disponível na altura de estudo) teve muito pouca aderência;
- É possivel identificar endereços relacionados com esquemas de *ransomware*, ao conhecer pelo menos um endereço relacionado com esse tipo de atividades (ou outro tipo de atividades ilegais);
- Apesar da maior parte dos endereços serem utilizados apenas uma vez, os que mais transações efetuam possuem uma grande parte das transações totais da rede Bitcoin;

- A partir de uma heurística estudada [5], foi possivel, com um endereço, encontrar outros que possam ter sido utilizados para receber “troco” de transações. O que é muito útil para “seguir *bitcoins*” e ter uma ideia mais clara do que pode pertencer a uma certa entidade.

# 4. Análise de Requisitos

## 4.1 Introdução

A análise de requisitos é um processo centrado nas tarefas que determinam as necessidades ou condições de um projeto novo ou alterado, considerando as exigências, possivelmente conflituosas das várias partes interessadas, analisando, documentando e validando os requisitos do *software* ou sistema [57]. Devem ser documentados, acionáveis, mensuráveis, testáveis, rastreáveis, relacionados às necessidades ou oportunidades de negócios, identificados e definidos a um nível de detalhe suficiente para o *design* do sistema [58]. Esta análise é crítica para o sucesso ou o fracasso de um projeto [59].

Este capítulo contém os elementos necessários para *design* e desenvolvimento de *software*, requisitos utilizados como ponto de partida para a nossa ferramenta, ao descrever a sua finalidade, quem e como será utilizado.

## 4.2 *Problem Statement*

### 4.2.1 *Problem*

Como referido na secção 1.1 (Âmbito), surge a necessidade de desenvolver uma ferramenta que vá ao encontro das necessidades de autoridades e investigadores, em que estejam reunidas funcionalidades úteis para as análises à rede Bitcoin, que sejam necessárias realizar.

### 4.2.2 *Stakeholders*

De forma a definir melhor o nosso problema, foi necessário identificar as partes diretamente envolvidas e os respetivos interesses e necessidades.

Por um lado, temos os investigadores e economistas que, com a implementação desta ferramenta, vêm a sua tarefa de análise facilitada, uma vez que encontram as funcionalidades que necessitam para analisar as transações da rede Bitcoin reunidas num único lugar. Para

além da comodidade de ter uma ferramenta completa, também se torna mais eficiente a realização das mesmas porque, ao ter tudo agrupado e organizado, o analista irá aceder de forma eficaz a funcionalidades que possam ser úteis nas suas investigações na rede Bitcoin.

Por outro lado, temos as autoridades. Sendo que a deteção e a distinção entre transações ilegais e legítimas são aspetos importantes para que a Bitcoin seja uma criptomoeda viável para a população. Uma ferramenta que reúna um conjunto de funcionalidades para esse efeito pode ser particularmente útil.

### 4.2.3 Solução do Problema

Perante o panorama apresentado, surge a necessidade de construir uma ferramenta que reúna um conjunto de funcionalidades que permita efetuar diversas análises à *blockchain* da Bitcoin como, por exemplo, estudar a atividade de um certo endereço. Assim, pretende-se desenvolver/adaptar um *software* de auditoria de transações Bitcoin, que apresente as características necessárias e seja uma mais-valia nesta área.

### 4.2.4 Restrições

É importante referir que, no processo de desenvolvimento desta ferramenta, é necessário restringir a um campo de atuação, que no nosso caso se refere às Tecnologias de Informação, Engenharia de *Software* e Economia, mais especificamente sobre a rede Bitcoin.

A nossa proposta de *software* tem como base o *BlockSci* [30]. Dito isto, o trabalho a realizar está assente nas funcionalidades que esta ferramenta já possui e na tentativa de desenvolvimento de algo mais completo. Para além disso, o apoio dado por parte dos criadores do *BlockSci* está limitado aos "*issues*", que os utilizadores da ferramenta colocam no *Github,* onde se encontra o código base.

### 4.2.5 Riscos

Considerando que a nossa ferramenta será baseada noutra, é fundamental que esteja bem documentada e que funcione da maneira esperada. Caso isto não se verifique, pode comprometer o desenvolvimento deste *software* e causar atrasos ou mudanças de requisitos/funcionalidades.

## 4.3 *UML Use Case Diagram*

Um *goal-model* expressa as relações entre um sistema e o seu ambiente, ou seja, não apenas sobre o que o sistema deve fazer, mas também o porquê. Esclarece requisitos, ao especificar metas que levam a colocar questões como "porquê?" e "como?". Permite que grandes objetivos sejam decompostos. Lida com conflitos, quando o cumprimento de algo interfere na execução de outros pontos. Permite que a integridade dos requisitos seja medida e só podem ser considerados completos se cumprirem todas as metas [60]. Existem várias notações em uso para *goal-models* no desenvolvimento de *software*, entre os quais KAOS, i* e UML *Use Case Diagram* [60].

Foi selecionado o **UML *Use Case Diagram***, pois permite mostrar, de uma forma simples, as diferentes maneiras que o utilizador pode interagir com o sistema. Os casos de uso, uma vez especificados, podem ser denotados como representação textual e visual (ou seja, diagrama de caso de uso). “Um conceito chave da modelação dos casos de uso, é que nos ajuda a projetar um sistema na perspetiva do utilizador final” [61]. “É uma técnica eficaz para comunicar o comportamento do sistema nos termos do utilizador, especificando todo o comportamento do sistema de uma perspetiva externa” [61].

Os UML *Use Case Diagram*s são normalmente elaborados no estado inicial de desenvolvimento e aplicados à modelação de caso de uso para os seguintes fins [61]:

- Especificar o contexto de um sistema;
- Capturar os requisitos de um sistema;
- Validar uma arquitetura de sistemas;
- Impulsionar a implementação e gerir casos de teste.

### 4.3.1 Perspetiva do Sistema

Este sistema irá funcionar num domínio de investigação para fins académicos e criminal, ou seja, irá abranger as autoridades e investigadores com necessidades/interesses em aprofundar o conhecimento das transações efetuadas na rede Bitcoin e identificar situações ilegais.

### 4.3.2 *Use Case Diagram*

Nesta secção é apresentado o diagrama (Figura 19), que consiste em: **casos de uso** - ovais em formato horizontal que representam os diferentes usos que um utilizador pode ter; **atores** - bonecos que representam as pessoas que estão associadas ao sistema; **associações** - uma linha entre atores e casos de uso, para perceber quais atores estão envolvidos em quais casos de uso; e "**caixas de limite do sistema**" – onde se encontram todos os casos de uso pertencentes ao sistema.

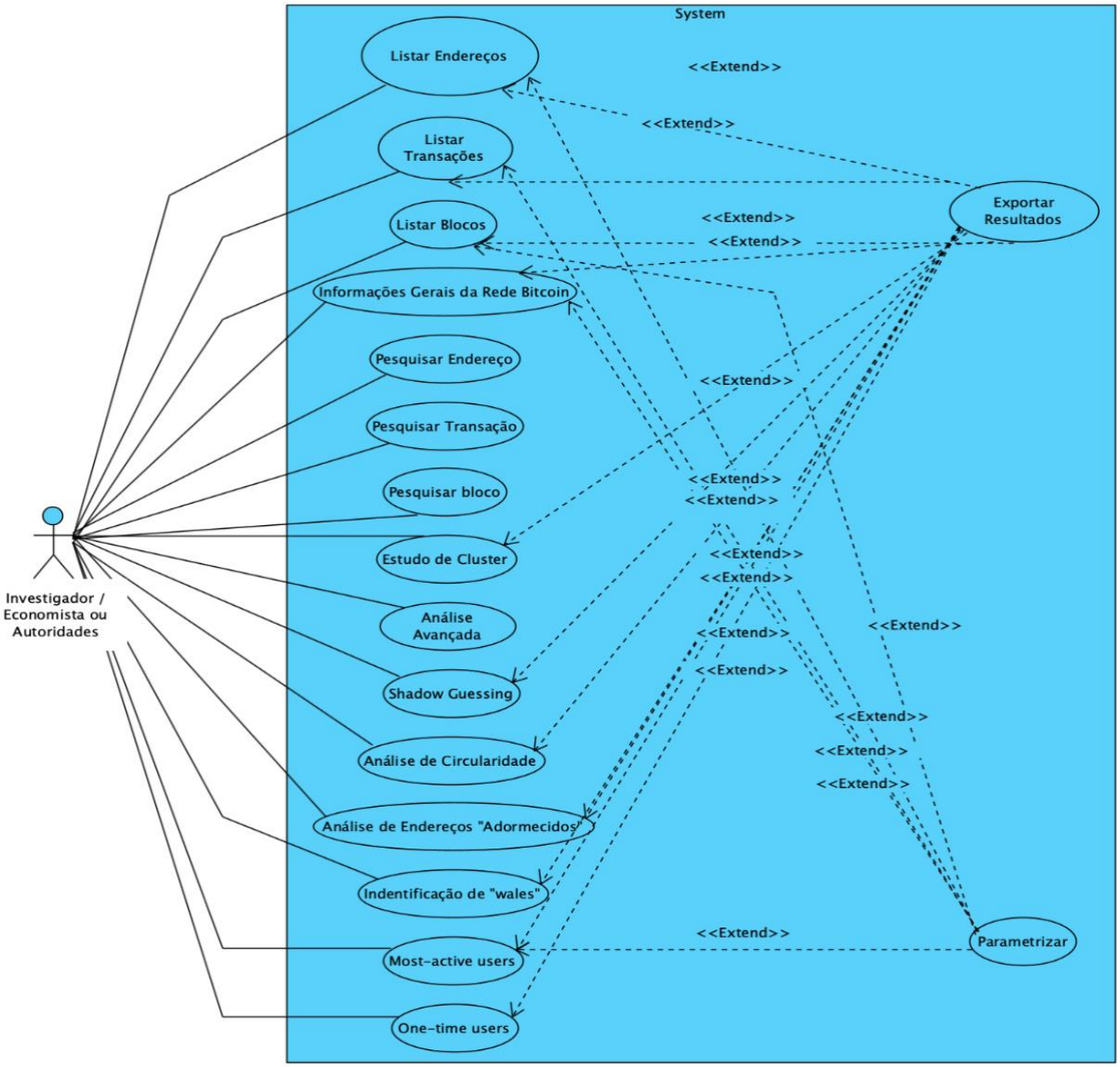


*Figura 19: Use Case Diagram*

## 4.4 Casos de Uso

### 4.4.1 Identificação dos atores

Os nossos atores consistem no **Investigador / Economista** e nas **Autoridades**. Podemos considerá-los como atores primários, sendo que todas as funcionalidades podem ser utilizadas por ambos.

Na próxima secção apresentamos uma análise de cada um dos casos de uso enunciados.

### 4.4.2 Análise dos Casos de Uso

Os casos de uso são fundamentalmente um formato de texto. Descrevem o comportamento do sistema, quando um utilizador interage com *software* [62]. Existem vários formatos para descrever casos de uso:

- ***Fully dressed***, composto por uma coluna de texto (não tabela), etapas numeradas, nenhuma instrução IF e uma convenção de numeração na secção de extensões, que envolve combinações de dígitos e letras [62];
- ***Casual***, texto livre;
- ***One-column table***, semelhante ao *fully dressed,* mas em formato de tabela;
- ***Two-column table****,* semelhante ao *one-column table,* mas divide-se em duas colunas, onde a primeira (da esquerda) corresponde às ações do ator primário e a segunda (da direita) corresponde às restantes;
- ***RUP style****,* baseado num conjunto de blocos de construção e elementos, descreve o que deve ser produzido, as habilidades exigidas e a explicação passo a passo que descreve como os objetivos de desenvolvimento específicos devem ser alcançados. Os principais blocos de construção, ou elementos, são os seguintes: funções (quem), produtos de trabalho (o quê) e tarefas (como) [63].

Foi utilizada uma adaptação do formato *fully dressed* para descrever os casos de uso. Este formato mostra muitos detalhes e é bem estruturado. É útil para obter uma compreensão aprofundada dos objetivos, tarefas e requisitos.

#### 4.4.2.1 Listar Endereços

**Nome do Caso de Uso:** Listar Endereços
**Âmbito:** Interação entre o utilizador e o sistema com o objetivo de obter uma lista de endereços
**Ator(es) Primário(s):** Investigador / Economista e Autoridades
**Pré-condições:**
1. O utilizador tem que ter instalado o *Docker*
2. O utilizador tem descarregado o *Bitcoin Core*
3. O utilizador tem pelo menos 60GB de RAM
**Pós-condições:** O sistema apresenta ao utilizador uma lista de endereços conforme requisitada pelo utilizador
**Cenário Principal de Sucesso:**
1. O utilizador abre o *software*
2. O utilizador escolhe a opção "*List*"
3. O utilizador escolhe a opção "*Addresses*"
4. O sistema apresenta os endereços
**Extensões (Caminhos alternativos):**
2a. O utilizador parametriza a listagem dos endereços
4a. O utilizador decide exportar os resultados
4b. O utilizador pretende ver detalhes de um dos endereços
4b1. O sistema mostra os detalhes do endereço selecionado

#### 4.4.2.2 Listar Transações de *bitcoins*

**Nome do Caso de Uso:** Listar Transações
**Âmbito:** Interação entre o utilizador e o sistema com o objetivo de obter uma lista de transações
**Ator(es) Primário(s):** Investigador / Economista e Autoridades
**Pré-condições:**
1. O utilizador tem que ter instalado o *Docker*
2. O utilizador tem descarregado o *Bitcoin Core*
3. O utilizador tem pelo menos 60GB de RAM
**Pós-condições:** O sistema apresenta ao utilizador uma lista de transações conforme requisitada pelo utilizador (da mais antiga para a mais recente)
**Cenário Principal de Sucesso:**
1. O utilizador abre o *software*
2. O utilizador escolhe a opção "*List*"
3. O utilizador escolhe a opção "*Transactions*"
4. O sistema apresenta as transações
**Extensões (Caminhos alternativos):**
2a. O utilizador parametriza a listagem das transações
4a. O utilizador decide exportar os resultados
4b. O utilizador pretende ver detalhes de uma das transações
4b1. O sistema mostra os detalhes da transação selecionada

#### 4.4.2.3 Listar Blocos da *blockchain*

**Nome do Caso de Uso:** Listar Blocos
**Âmbito:** Interação entre o utilizador e o sistema com o objetivo de obter uma lista de blocos de transações
**Ator(es) Primário(s):** Investigador / Economista e Autoridades
**Pré-condições:**
1. O utilizador tem que ter instalado o *Docker*
2. O utilizador tem descarregado o *Bitcoin Core*
3. O utilizador tem pelo menos 60GB de RAM
**Pós-condições:** O sistema apresenta ao utilizador uma lista de blocos de transações conforme requisitada pelo utilizador
**Cenário Principal de Sucesso:**
1. O utilizador abre o *software*
2. O utilizador escolhe a opção "*List*"
3. O utilizador escolhe a opção "*Blocks*"
4. O sistema apresenta os endereços
**Extensões (Caminhos alternativos):**
2a. O utilizador parametriza a listagem dos Blocos
4a. O utilizador decide exportar os resultados
4b. O utilizador pretende ver detalhes de um dos blocos
4b1. O sistema mostra os detalhes do bloco selecionado

#### 4.4.2.4 Obter informações gerais da rede Bitcoin

**Nome do Caso de Uso:** Obter informações gerais da rede Bitcoin
**Âmbito:** Interação entre o utilizador e o sistema com o objetivo de obter informações gerais da rede Bitcoin
**Ator(es) Primário(s):** Investigador / Economista e Autoridades
**Pré-condições:**
1. O utilizador tem que ter instalado o *Docker*
2. O utilizador tem descarregado o *Bitcoin Core*
3. O utilizador tem pelo menos 60GB de RAM
**Pós-condições:** O sistema apresenta ao utilizador um conjunto de informações sobre a rede Bitcoin
**Cenário Principal de Sucesso:**
1. O utilizador abre o *software*
2. O utilizador escolhe a opção "*General information*"
3. O sistema apresenta as informações requisitadas, como número de transações, volume de transações, médias relativas ao número e valores de transações, recompensas de *miners* e blocos minerados
**Extensões (Caminhos alternativos):**
2a. O utilizador parametriza quais informações quer receber
3a. O utilizador decide exportar os resultados

#### 4.4.2.5 Pesquisar Endereço

**Nome do Caso de Uso:** Pesquisar Endereço
**Âmbito:** Interação entre o utilizador e o sistema com o objetivo de obter informações sobre um determinado endereço
**Ator(es) Primário(s):** Investigador / Economista e Autoridades
**Pré-condições:**
1. O utilizador tem que ter instalado o *Docker*
2. O utilizador tem descarregado o *Bitcoin Core*
3. O utilizador tem pelo menos 60GB de RAM
**Pós-condições:** O sistema apresenta ao utilizador um conjunto de informações sobre o endereço fornecido pelo utilizador
**Cenário Principal de Sucesso:**
1. O utilizador abre o *software*
2. O utilizador escolhe a opção "*Search*"
3. O utilizador escolhe a opção "*Address*"
4. O utilizador escreve o endereço que quer pesquisar
5. O sistema apresenta as informações relativas ao endereço, como número de transações e transações associadas
**Extensões (Caminhos alternativos):**
5a. O utilizador decide exportar os resultados

#### 4.4.2.6 Pesquisar Transação de *bitcoins*

**Nome do Caso de Uso:** Pesquisar Transação
**Âmbito:** Interação entre o utilizador e o sistema com o objetivo de obter informações sobre uma determinada transação
**Ator(es) Primário(s):** Investigador / Economista e Autoridades
**Pré-condições:**
1. O utilizador tem que ter instalado o *Docker*
2. O utilizador tem descarregado o *Bitcoin Core*
3. O utilizador tem pelo menos 60GB de RAM
**Pós-condições:** O sistema apresenta ao utilizador um conjunto de informações sobre a transação fornecida pelo utilizador
**Cenário Principal de Sucesso:**
1. O utilizador abre o *software*
2. O utilizador escolhe a opção "*Search*"
3. O utilizador escolhe a opção "*Transaction*"
4. O utilizador escreve o *hash* da transação que quer pesquisar
5. O sistema apresenta as informações relativas à transação como data, valor de *input*, valor de *output*, valor de taxa de transação, bloco e endereços de *input* e de *output*
**Extensões (Caminhos alternativos):**
5a. O utilizador decide exportar os resultados

#### 4.4.2.7 Pesquisar Bloco da *blockchain*

**Nome do Caso de Uso:** Pesquisar Bloco
**Âmbito:** Interação entre o utilizador e o sistema com o objetivo de obter informações sobre um determinado bloco de transações
**Ator(es) Primário(s):** Investigador / Economista e Autoridades
**Pré-condições:**
1. O utilizador tem que ter instalado o *Docker*
2. O utilizador tem descarregado o *Bitcoin Core*
3. O utilizador tem pelo menos 60GB de RAM
**Pós-condições:** O sistema apresenta ao utilizador um conjunto de informações sobre o bloco requisitado pelo utilizador
**Cenário Principal de Sucesso:**
1. O utilizador abre o *software*
2. O utilizador escolhe a opção "*Search*"
3. O utilizador escolhe a opção "*Block*"
4. O utilizador escreve a "altura" do bloco que quer pesquisar
5. O sistema apresenta as informações relativas ao bloco como número de transações, data de confirmação do bloco, *nonce*, volume de transações, recompensa do *miner* e transações
**Extensões (Caminhos alternativos):**
5a. O utilizador decide exportar os resultados

#### 4.4.2.8 Obter uma entidade a partir de um endereço

**Nome do Caso de Uso:** Obter uma Entidade a partir de um Endereço
**Âmbito:** Interação entre o utilizador e o sistema com o objetivo de obter informações sobre uma entidade ao juntar endereços, transações (e outras informações) associadas a um determinado endereço
**Ator(es) Primário(s):** Investigador / Economista e Autoridades
**Pré-condições:**
1. O utilizador tem que ter instalado o *Docker*
2. O utilizador tem descarregado o *Bitcoin Core*
3. O utilizador tem pelo menos 60GB de RAM
**Pós-condições:** O sistema apresenta ao utilizador um conjunto de informações sobre a entidade relativa ao endereço fornecido
**Cenário Principal de Sucesso:**
1. O utilizador abre o *software*
2. O utilizador escolhe a opção "*Address -> Entity*"
3. O utilizador escreve o endereço que quer obter informações sobre a entidade correspondente.
4. O sistema apresenta as informações relativas à entidade como número de endereços, número de transações como *inputs* e total, média de transações por endereço, média de valor de transação, volume de transações e respetivos endereços e transações
**Extensões (Caminhos alternativos):**
4a. O utilizador decide exportar os resultados
4a1. O utilizador identifica a entidade com uma *tag*
4b. O utilizador pretende ver detalhes de um dos endereços
4b1. O sistema mostra os detalhes do endereço selecionado

#### 4.4.2.9 Estudo de Cluster

**Nome do Caso de Uso:** Estudo de Cluster
**Âmbito:** Interação entre o utilizador e o sistema com o objetivo de obter um conjunto de endereços relacionados com o endereço fornecido.
**Ator(es) Primário(s):** Investigador / Economista e Autoridades
**Pré-condições:**
1. O utilizador tem que ter instalado o *Docker*
2. O utilizador tem descarregado o *Bitcoin Core*
3. O utilizador tem pelo menos 60GB de RAM
**Pós-condições:** O sistema apresenta ao utilizador um conjunto de endereços relacionados com o endereço fornecido
**Cenário Principal de Sucesso:**
1. O utilizador abre o *software*
2. O utilizador escolhe a opção "*Cluster with Address*"
3. O utilizador escreve o endereço que pretende fazer o estudo de *cluster*
4. O sistema apresenta os endereços e transações provenientes do estudo e informações como número de endereços, número de transações, saldo do *cluster*, média de transações por endereço, média de valor de transação e volume de transações
**Extensões (Caminhos alternativos):**
4a. O utilizador decide exportar os resultados
4b. O utilizador pretende ver detalhes de um dos endereços
4b1. O sistema mostra os detalhes do endereço selecionado
4c. O utilizador pretende ver detalhes de uma das transações
4c1. O sistema mostra os detalhes da transação selecionada

#### 4.4.2.10 Análise Avançada

**Nome do Caso de Uso:** Análise Avançada
**Âmbito:** Interação entre o utilizador e o sistema com o objetivo de fazer uma análise mais avançada
**Ator(es) Primário(s):** Investigador / Economista e Autoridades
**Pré-condições:**
1. O utilizador tem que ter instalado o *Docker*
2. O utilizador tem descarregado o *Bitcoin Core*
3. O utilizador tem pelo menos 60GB de RAM
4. O utilizador tem conhecimento do *Python Module Reference* do *BlockSci*
**Pós-condições:** O sistema apresenta os resultados obtidos pela pesquisa personalizada
**Cenário Principal de Sucesso:**
1. O utilizador abre o *software*
2. O utilizador escolhe a opção "*Advanced analysis*"
3. O utilizador escreve código em *Python* da análise que pretende fazer
4. O sistema apresenta os resultados da análise feita
**Extensões (Caminhos alternativos):**
4a. O utilizador decide exportar os resultados

#### 4.4.2.11 *Shadow Guessing*

**Nome do Caso de Uso:** *Shadow Guessing*
**Âmbito:** Interação entre o utilizador e o sistema com o objetivo de obter um conjunto de endereços utilizados como "troco" em transações que um endereço fornecido é utilizado como *input*.
**Ator(es) Primário(s):** Investigador / Economista e Autoridades
**Pré-condições:**
1. O utilizador tem que ter instalado o *Docker*
2. O utilizador tem descarregado o *Bitcoin Core*
3. O utilizador tem pelo menos 60GB de RAM
**Pós-condições:** O sistema apresenta ao utilizador um conjunto de endereços possivelmente utilizados para receber "troco" de transações
**Cenário Principal de Sucesso:**
1. O utilizador abre o *software*
2. O utilizador escolhe a opção "*Shadow Guessing*"
3. O utilizador escreve o endereço
4. O sistema apresenta os endereços provenientes do estudo e informações, como número de endereços, número de transações, média de transações por endereço, média de valor de transação e volume de transações
**Extensões (Caminhos alternativos):**
4a. O utilizador decide exportar os resultados
4b. O utilizador pretende ver detalhes de um dos endereços
4b1. O sistema mostra os detalhes do endereço selecionado

#### 4.4.2.12 Análise de Circularidade

**Nome do Caso de Uso:** Análise de Circularidade
**Âmbito:** Interação entre o utilizador e o sistema com o objetivo de analisar se é feita circulação de *bitcoins* por várias carteiras antes de regressar à original
**Ator(es) Primário(s):** Investigador / Economista e Autoridades
**Pré-condições:**
1. O utilizador tem que ter instalado o *Docker*
2. O utilizador tem descarregado o *Bitcoin Core*
3. O utilizador tem pelo menos 60GB de RAM
**Pós-condições:** O sistema apresenta ao utilizador um conjunto de endereços pelos quais a Bitcoin passou antes de regressar, respetivas datas e respetivos montantes
**Cenário Principal de Sucesso:**
1. O utilizador abre o *software*
2. O utilizador escolhe a opção "*Circularity Analysis*"
3. O utilizador escreve o endereço
4. O sistema apresenta as informações provenientes do estudo
**Extensões (Caminhos alternativos):**
4a. O utilizador decide exportar os resultados
4b. O utilizador pretende ver detalhes de um dos endereços
4b1. O sistema mostra os detalhes do endereço selecionado

#### 4.4.2.13 Análise da *Blockchain* em períodos específicos

**Nome do Caso de Uso:** Análise da *Blockchain* em períodos específicos
**Âmbito:** Interação entre o utilizador e o sistema com o objetivo de analisar períodos de elevada atividade, que possam ser associados a um determinado evento identificável no tempo
**Ator(es) Primário(s):** Investigador / Economista e Autoridades
**Pré-condições:**
1. O utilizador tem que ter instalado o *Docker*
2. O utilizador tem descarregado o *Bitcoin Core*
3. O utilizador tem pelo menos 60GB de RAM
**Pós-condições:** O sistema apresenta ao utilizador um conjunto de informações relativas ao período definido inicialmente (como o n.º de endereços e montantes transacionados)
**Cenário Principal de Sucesso:**
1. O utilizador abre o *software*
2. O utilizador escolhe a opção "*Blockchain Analysis*"
3. O utilizador define as datas do período a ser analisado
4. O sistema apresenta as informações provenientes do estudo, como valores das transações menor e maior, valor médio de transação, número de transações, volume de transações, número de endereços como *input* e *output*, valores das taxas maior e menor, valor médio da taxa, recompensa total dos *miners*, valor médio da recompensa dos *miners* e número de blocos confirmados
**Extensões (Caminhos alternativos):**
4a. O utilizador decide exportar os resultados

#### 4.4.2.14 Análise de Endereços "Adormecidos"

**Nome do Caso de Uso:** Análise de Endereços "Adormecidos"
**Âmbito:** Interação entre o utilizador e o sistema com o objetivo de analisar qual a percentagem de endereços e *bitcoins* inativos durante um determinado período de tempo
**Ator(es) Primário(s):** Investigador / Economista e Autoridades
**Pré-condições:**
1. O utilizador tem que ter instalado o *Docker*
2. O utilizador tem descarregado o *Bitcoin Core*
3. O utilizador tem pelo menos 60GB de RAM
**Pós-condições:** O sistema apresenta ao utilizador a percentagem de endereços inativos ao longo do tempo (e quantos desses endereços e *bitcoins* foram criados nos primeiros tempos da *blockchain*)
**Cenário Principal de Sucesso:**
1. O utilizador abre o *software*
2. O utilizador escolhe a opção "*Dorment Addresses"*
3. O utilizador define o período de inatividade que quer pesquisar
4. O sistema apresenta as informações provenientes do estudo, como número de endereços, número de transações, número médio de transações por endereço, valor médio de transação, volume de transações e endereços respetivos
**Extensões (Caminhos alternativos):**
4a. O utilizador decide exportar os resultados
4b. O utilizador pretende ver detalhes de um dos endereços

4b1. O sistema mostra os detalhes do endereço selecionado

#### 4.4.2.15 Identificação de carteiras (ou endereços) “whales”

**Nome do Caso de Uso:** Identificação de carteiras (ou endereços) “whales”
**Âmbito:** Interação entre o utilizador e o sistema com o objetivo de identificar endereços com mais de 1 milhão de *bitcoins*
**Ator(es) Primário(s):** Investigador / Economista e Autoridades
**Pré-condições:**
1. O utilizador tem que ter instalado o *Docker*
2. O utilizador tem descarregado o *Bitcoin Core*
3. O utilizador tem pelo menos 60GB de RAM
**Pós-condições:** O sistema apresenta ao utilizador a atividade desses endereços ao longo do tempo
**Cenário Principal de Sucesso:**
1. O utilizador abre o *software*
2. O utilizador escolhe a opção “*Whales*”
3. O sistema apresenta as informações provenientes do estudo como número de endereços, número de transações, número médio de transações por endereço, valor médio de transação, volume de transações e endereços respetivos
**Extensões (Caminhos alternativos):**
3a. O utilizador decide exportar os resultados
3b. O utilizador pretende ver detalhes de um dos endereços
3b1. O sistema mostra os detalhes do endereço selecionado

#### 4.4.2.16 *Most-active users*

**Nome do Caso de Uso:** *Most-active Users*
**Âmbito:** Interação entre o utilizador e o sistema com o objetivo de obter informações e dados associados aos endereços ou entidades mais ativas na rede Bitcoin
**Ator(es) Primário(s):** Investigador / Economista e Autoridades
**Pré-condições:**
1. O utilizador tem que ter instalado o *Docker*
2. O utilizador tem descarregado o *Bitcoin Core*
3. O utilizador tem pelo menos 60GB de RAM
**Pós-condições:** O sistema apresenta ao utilizador uma lista dos 10 endereços mais ativos ao longo do período de tempo definido
**Cenário Principal de Sucesso:**
1. O utilizador abre o *software*
2. O utilizador escolhe a opção “*Most-active users*”
3. O utilizador define o período temporal
4. O sistema apresenta os endereços mais ativos no período temporal definido
**Extensões (Caminhos alternativos):**
4a. O utilizador decide exportar os resultados
4b. O utilizador pretende ver detalhes de um dos endereços
4b1. O sistema mostra os detalhes do endereço selecionado

#### 4.4.2.17 *One-time users*

**Nome do Caso de Uso:** *Most-active Users*
**Âmbito:** Interação entre o utilizador e o sistema com o objetivo de obter informações e dados associados aos endereços que apenas têm uma transação na rede Bitcoin
**Ator(es) Primário(s):** Investigador / Economista e Autoridades
**Pré-condições:**
1. O utilizador tem que ter instalado o *Docker*
2. O utilizador tem descarregado o *Bitcoin Core*
3. O utilizador tem pelo menos 60GB de RAM
**Pós-condições:** O sistema apresenta ao utilizador uma lista dos 10 endereços utilizados apenas uma vez
**Cenário Principal de Sucesso:**
1. O utilizador abre o *software*
2. O utilizador escolhe a opção "*One-time users*"
3. O sistema apresenta as informações provenientes do estudo como número de endereços, número de transações, número médio de transações por endereço, valor médio de transação, volume de transações e endereços respetivos
**Extensões (Caminhos alternativos):**
3a. O utilizador decide exportar os resultados
3b. O utilizador pretende ver detalhes de um dos endereços
3b1. O sistema mostra os detalhes do endereço selecionado

## 4.5 Requisitos

Esta secção apresenta todos os requisitos funcionais, obtidos com auxílio dos casos de uso descritos, e não funcionais identificados para o desenvolvimento desta ferramenta. Os requisitos funcionais são listados primeiro e de seguida os não funcionais. Cada requisito tem um identificador (ID), uma descrição e uma prioridade *MoSCoW* (*Must Have, Should Have, Could Have* e *Won't Have*).

### 4.5.1 Requisitos Funcionais

| ID | Descrição | Prioridade |
|---|---|---|
| RF01 | O utilizador deve ser capaz de listar endereços, com ou sem parametrização (em termos de número de transações e/ou saldo). | *Must Have* |
| RF02 | O utilizador deve ser capaz de listar transações, com ou sem parametrização (em termos de valor de taxa, valor e/ou período temporal). | *Must Have* |

| | | |
|---|---|---|
| RF03 | O utilizador deve ser capaz de listar blocos de transações, com ou sem parametrização (em termos de número de transações, recompensa do *miner* e/ou período temporal). | *Must Have* |
| RF04 | O utilizador deve ser capaz de obter informações gerais sobre a rede Bitcoin, como número de transações, volume de transações, médias relativas ao número e valores de transações, recompensas de *miners* e blocos minerados. | *Could Have* |
| RF05 | O utilizador deve ser capaz de pesquisar um endereço à sua escolha e visualizar os detalhes do mesmo, como número e respetivas transações associadas. | *Must Have* |
| RF06 | O utilizador deve ser capaz de pesquisar uma transação (com o seu *hash*) à sua escolha e visualizar os detalhes da mesma, como data, valor de *input*, valor de *output*, valor de taxa de transação, bloco e endereços de *input* e de *output.* | *Must Have* |
| RF07 | O utilizador deve ser capaz de pesquisar um bloco de transações (com a sua *height*) à sua escolha e visualizar os detalhes do mesmo, como número de transações, data de confirmação do bloco, *nonce*, volume de transações, recompensa do *miner* e transações. | *Must Have* |
| RF08 | O utilizador deve ser capaz de obter várias informações sobre a possível entidade a que pertence um certo endereço (outros endereços que possam pertencer à mesma entidade e transações), ao fornecer um endereço à sua escolha. Estas informações incluem número de endereços, número de transações como *inputs* e total, média de transações por endereço, média de valor de transação, volume de transações e respetivos endereços e transações. | *Must Have* |
| RF9 | O utilizador deve ser capaz de fazer um estudo de *cluster* ao fornecer um certo endereço. Permite reunir informações como endereços associados, transações associadas, saldo do *cluster*, média de transações por endereço, média de valor de transação e volume de transações. | *Must Have* |
| RF10 | O utilizador deve ser capaz de escrever uma análise personalizada em *Python*. | *Could Have* |
| RF11 | O utilizador deve ser capaz de fazer um estudo de *"Shadow Guessing"* ao fornecer um certo endereço. Permite reunir | *Must Have* |

| | | |
|---|---|---|
| | endereços usados para receber “troco” das transações em que o endereço fornecido foi utilizado como *input* e informações, como número de transações, média de transações por endereço, média de valor de transação e volume de transações. | |
| RF12 | O utilizador deve ser capaz de exportar para um ficheiro os resultados das análises feitas ao fornecer o nome que quer que o *output* tenha | *Must Have* |
| RF13 | O utilizador deve ser capaz de analisar se é feita circulação de *Bitcoins* por várias carteiras antes de regressar à original ao fornecer um endereço. Isto permite reunir uma lista de endereços pelos quais a Bitcoin passou antes de regressar, respetivas datas e respetivos montantes. | *Must Have* |
| RF14 | O utilizador deve ser capaz de analisar períodos de elevada atividade que possam ser associados a um determinado evento identificável no tempo (ex: uma determinada notícia). Isto permite reunir informações, como número de transações, volume de transações, médias relativas ao número e valores de transações, recompensas de *miners* e blocos minerados. | *Must Have* |
| RF15 | O utilizador deve ser capaz de analisar endereços “adormecidos” e visualizar informações, como número de endereços, número de transações, número médio de transações por endereço, valor médio de transação, volume de transações e endereços respetivos. | *Must Have* |
| RF16 | O utilizador deve ser capaz de identificar carteiras (ou endereços) “*whales*”, ou seja, endereços com mais de 1 milhão de *bitcoins*. Isto permite verificar o número de endereços, número de transações, número médio de transações por endereço, valor médio de transação, volume de transações e endereços respetivos. | *Must Have* |
| RF17 | O utilizador deve ser capaz de identificar os endereços mais ativos num certo período temporal. Isto permite reunir informações e dados associados aos endereços ou entidades mais ativas na rede Bitcoin no período definido. | *Could Have* |
| RF18 | O utilizador deve ser capaz de identificar os endereços que sejam “*one-time users*” (com apenas uma transação) num certo período temporal. Isto permite reunir informações e dados | *Could Have* |

| | | |
|---|---|---|
| | associados aos endereços com apenas uma transação na rede Bitcoin no período definido, como número de endereços, número de transações, número médio de transações por endereço, valor médio de transação, volume de transações e endereços respetivos. | |

*Tabela 2: Requisitos funcionais*

### 4.5.2 Requisitos Não-Funcionais

Esta seção descreve os seguintes requisitos não funcionais do sistema: **Disponibilidade**, **Confiabilidade**, **Desempenho** e **Usabilidade**. Segurança, *Manageability*, *Maintainability* e Custo não foram considerados por não se aplicarem ao nosso *software*.

#### 4.5.2.1 Disponibilidade

É a capacidade do sistema estar disponível para serviço quando solicitado por utilizadores finais ou outros sistemas. Está diretamente relacionado ao tempo de inatividade.

| ID | Descrição | Prioridade |
|---|---|---|
| DIS01 | O sistema deve estar sempre disponível a partir do momento que o utilizador inicia o programa. Ao identificar que algo de errado ocorre, o programa volta a iniciar-se. | *Must Have* |

*Tabela 3: Requisitos de disponibilidade*

#### 4.5.2.2 Confiabilidade

É a capacidade de um sistema executar suas funções sob condições estabelecidas por um período de tempo específico. É expresso como uma probabilidade (Frequentemente referido como uma propriedade relativa expressa pelo tempo médio entre falhas (MTBF)).

| ID | Descrição | Prioridade |
|---|---|---|
| CONF01 | O sistema deve correr sem falhas que terminem o programa. Ao ocorrer uma falha, o sistema deve recuperar sem que isso interfira com a utilização normal da ferramenta. | *Must Have* |

*Tabela 4: Requisitos de confiabilidade*

#### 4.5.2.3 Desempenho

Desempenho: diz respeito à velocidade de operação de um sistema. Medido por duas métricas:

- **Tempo de resposta**: a rapidez com que o sistema reage a uma entrada do utilizador ou de outro sistema;
- **Rendimento**: quanto trabalho (ou seja, operações) o sistema pode realizar num determinado período de tempo.

| ID | Descrição | Prioridade |
|---|---|---|
| DES01 | O sistema deve ser capaz de iniciar em menos de 5 segundos, em condições ideias de ligação à Internet. | *Must Have* |
| DES02 | O sistema deve ser capaz de mostrar os resultados de análises menos complexas, como pesquisas e estudos de *cluster* em menos de 30 segundos. | *Must Have* |

*Tabela 5: Requisitos de desempenho*

#### 4.5.2.4 Usabilidade

É a facilidade de uso e aprendizagem de um *software*. É composto por vários atributos: **Aprendizagem**, **Eficiência**, ***Memorability***, **Erros** e **Satisfação** [64].

| ID | Descrição | Prioridade |
|---|---|---|
| US01 | O sistema deve utilizar um diálogo simples e natural, para fácil compreensão do utilizador, familiarizado com termos de economia e relativos à *blockchain*. | *Must Have* |
| US02 | A linguagem utilizada deve ser simples e de conhecimento geral do utilizador, familiarizado com termos de economia e relativos à *blockchain*. | *Must Have* |

| | | |
|---|---|---|
| US03 | O sistema deve ser feito de forma a facilitar a realização de tarefas básicas pela primeira vez por parte do utilizador. | *Must Have* |
| US04 | O utilizador, após ter conhecimento do sistema, deve ser capaz de executar as tarefas de forma rápida e eficaz. | *Must Have* |
| US05 | O utilizador, após ter conhecimento do sistema, ao ausentar-se por algum tempo deve ser capaz restabelecer a proficiência. | *Must Have* |
| US06 | Quando o utilizador comete um erro, o sistema deve fornecer *feedback* sobre o mesmo através de mensagens de erro esclarecedoras. | *Must Have* |
| US07 | Quando o utilizador comete um erro, o sistema deve recuperar facilmente, sem afetar o uso da ferramenta. | *Must Have* |

*Tabela 6: Requisitos de usabilidade*

## 4.6 Modelos de Interface / UED /UI

Foi criado o *User Environment Design* (UED) e escolhidas três áreas de foco: **Página Inicial**, ***Search*** e ***List***. Posto isto, foi colocado em prática o desenvolvimento de *mockups*, que permite ter uma ideia de como será o sistema. De seguida, foram verificadas as 10 heurísticas [65] (*checklist* de usabilidade), ou seja, 10 princípios gerais para o *design* da interface do utilizador, onde se procurou perceber se estas se aplicavam ou se não fazia sentido a sua aplicação. São chamadas de "heurísticas" porque têm mais a natureza de regras práticas do que diretrizes de usabilidade específicas [65].

### 4.6.1 *User Environment Design (UED)*

***User Environment Design*** tem o propósito de mostrar como se navega no sistema. Mostra cada parte do *software*, como apoia o trabalho do utilizador, exatamente qual função está disponível naquela parte e como o utilizador entra e sai de cada componente do sistema. O *design* contextual usa o diagrama ***User Environment Design*** (UED), que exibe as áreas de foco, ou seja, áreas que são visíveis para o utilizador ou que lhe são relevantes (Figura 20). As áreas de foco podem ser definidas como funções num sistema que apoia um determinado tipo ou parte do trabalho. O **UED** também apresenta como as áreas de foco se relacionam entre si [66].

Como se pode verificar na Figura 20, a nossa *Main Focus Area* será a **Página Inicial**, a partir da qual será possível aceder a qualquer outra *Focus Area* (o contrário também se verifica).

As restantes *Focus Areas* são as que se seguem:

- ***List***, na qual o utilizador pode decidir se quer listar endereços, transações ou blocos com a possibilidade de parametrização consoante a opção escolhida;
- ***Search***, onde o utilizador pode pesquisar endereços, transações (com o *hash*) e blocos (com a *height*) e, posteriormente, ver os detalhes da pesquisa efetuada;
- ***Cluster with Address***, permite ao utilizador fazer um estudo de *cluster* que possibilita ver endereços e transações associados ao endereço fornecido. Posteriormente, é possível ver os detalhes dos endereços e transações resultantes;
- ***Address -> Entity,*** permite ao utilizador reunir endereços que poderão estar associados à carteira a que pertence o endereço fornecido. Posteriormente, é possível ver os detalhes dos endereços resultantes;
- ***Shadow Guessing***, permite ao utilizador reunir endereços que poderão ter sido utilizados para receber "troco" de transações em que endereço fornecido foi utilizado como *input*. Posteriormente, é possível ver os detalhes dos endereços resultantes;
- ***Circularity Analysis***, permite ao utilizador analisar se é feita circulação de *bitcoins* por várias carteiras antes de regressar à original. Posteriormente, é possível ver os detalhes dos endereços resultantes;
- ***Blockchain Analysis***, onde o utilizador pode analisar períodos de elevada atividade, que possam ser associados a um determinado evento identificável no tempo;
- ***Dorment Addresses*****,** permite ao utilizador analisar qual a percentagem de endereços e *bitcoins* inativos durante um determinado período de tempo;
- ***Whales***, permite ao utilizador analisar a atividade de endereços, que tenham mais de 1 milhão de *bitcoins*, ao longo do tempo;
- ***Most-active Users***, permite ao utilizador obter informações e dados associados aos endereços ou entidades mais ativas na rede Bitcoin. Posteriormente, é possivel ver os detalhes dos endereços resultantes;
- ***One-time Users***, permite ao utilizador obter informações e dados associados aos endereços que efetuaram apenas uma transação na rede Bitcoin. Posteriormente, é possivel ver os detalhes dos endereços resultantes;
- ***General Information***, onde o utilizador pode obter informações gerais da rede Bitcoin, de acordo com as datas escolhidas.

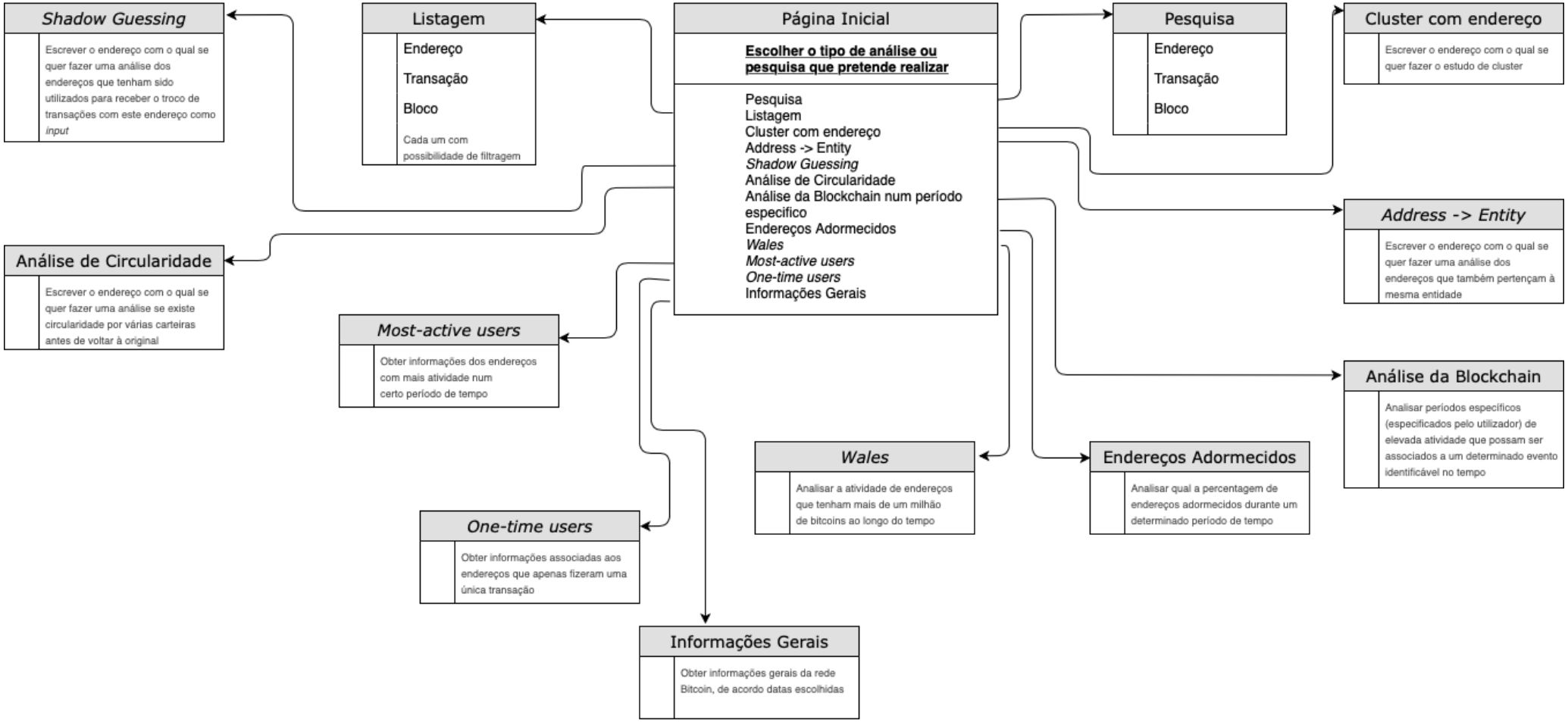


*Figura 20: User Environment Design (UED)*

### 4.6.2 Protótipos / *Mockups*

Como referido anteriormente, de forma a ter uma ideia de como será o *software,* foram desenvolvidos *mockups* para as três áreas de foco: **Página Inicial, *Search*** e ***List***.

Nas Figura 21 e 22 podemos observar uma estimativa de como será a **Página Inicial**, composta pelo conjunto de análises que o utilizador poderá selecionar. Para além disso, é possível, através do símbolo identificado com três traços horizontais, abrir um menu em que se pode escolher "*Home*" (para o caso de não estar), ver informações sobre o programa e uma componente de ajuda para alguma dúvida que o utilizador possa ter. Este menu está presente em toda a ferramenta.

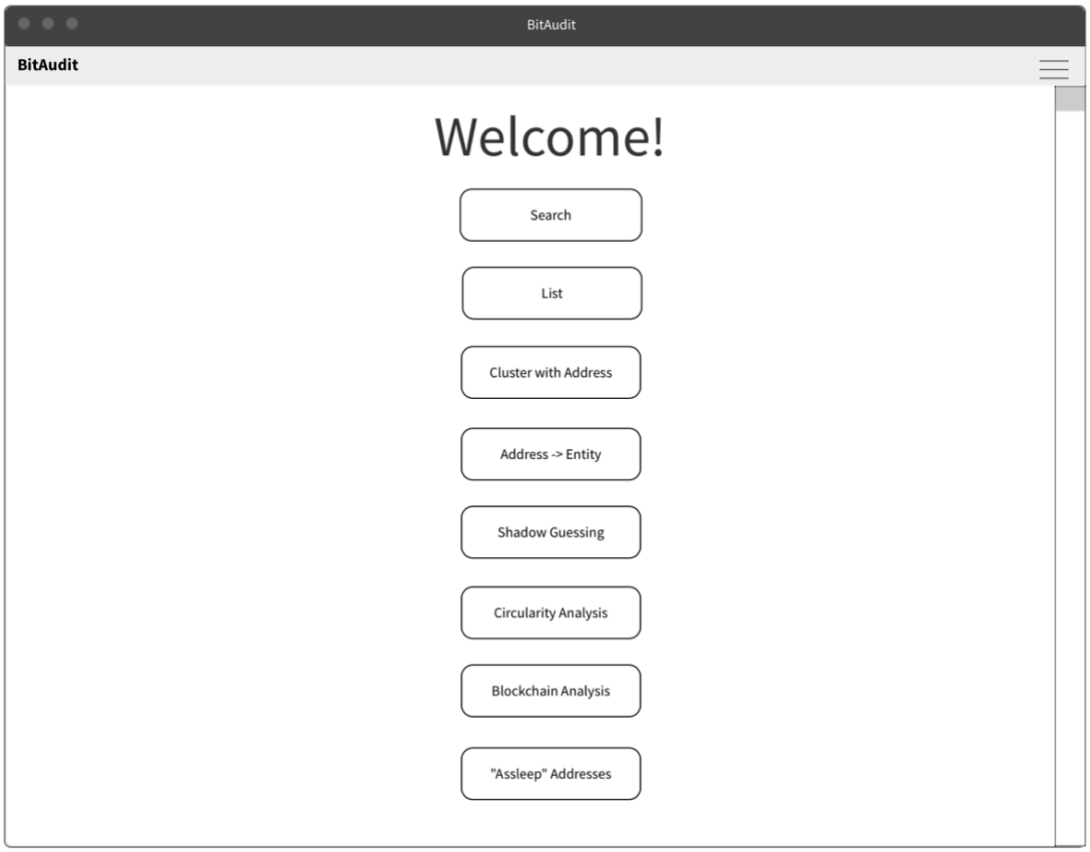


*Figura 21: Página Inicial (1)*

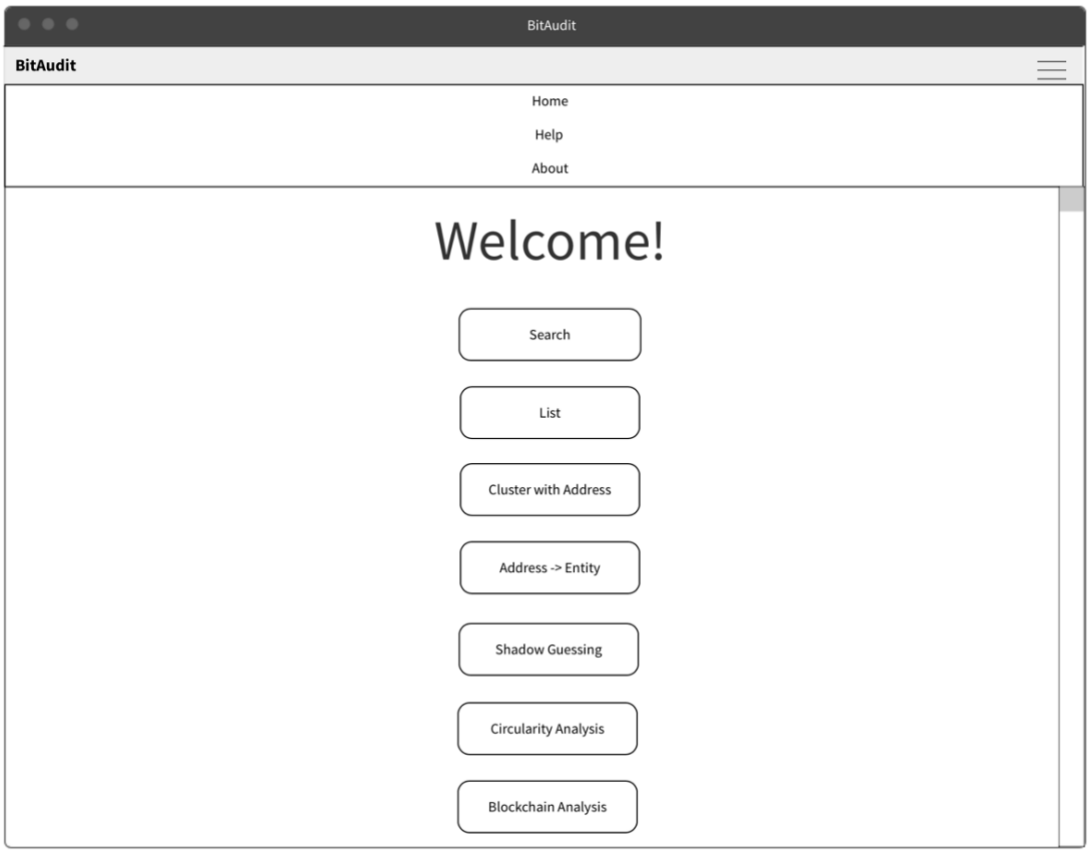


*Figura 22: Página Inicial (2)*

Na Figura 23, temos a página relativa à pesquisa de endereços, transações e blocos (***Search***). Através do *dropdown,* é possível escolher o tipo de pesquisa a fazer e, consoante essa escolha, o utilizador escreve na caixa de texto, o endereço, *hash* (transação) ou *height* (bloco). Também conseguimos verificar como é o resultado da pesquisa de endereço, onde se encontram detalhes, como número de transações, *balance* (saldo) e as transações, que também se podem ver ao clicar no símbolo ao lado direito do *hash* da transação.

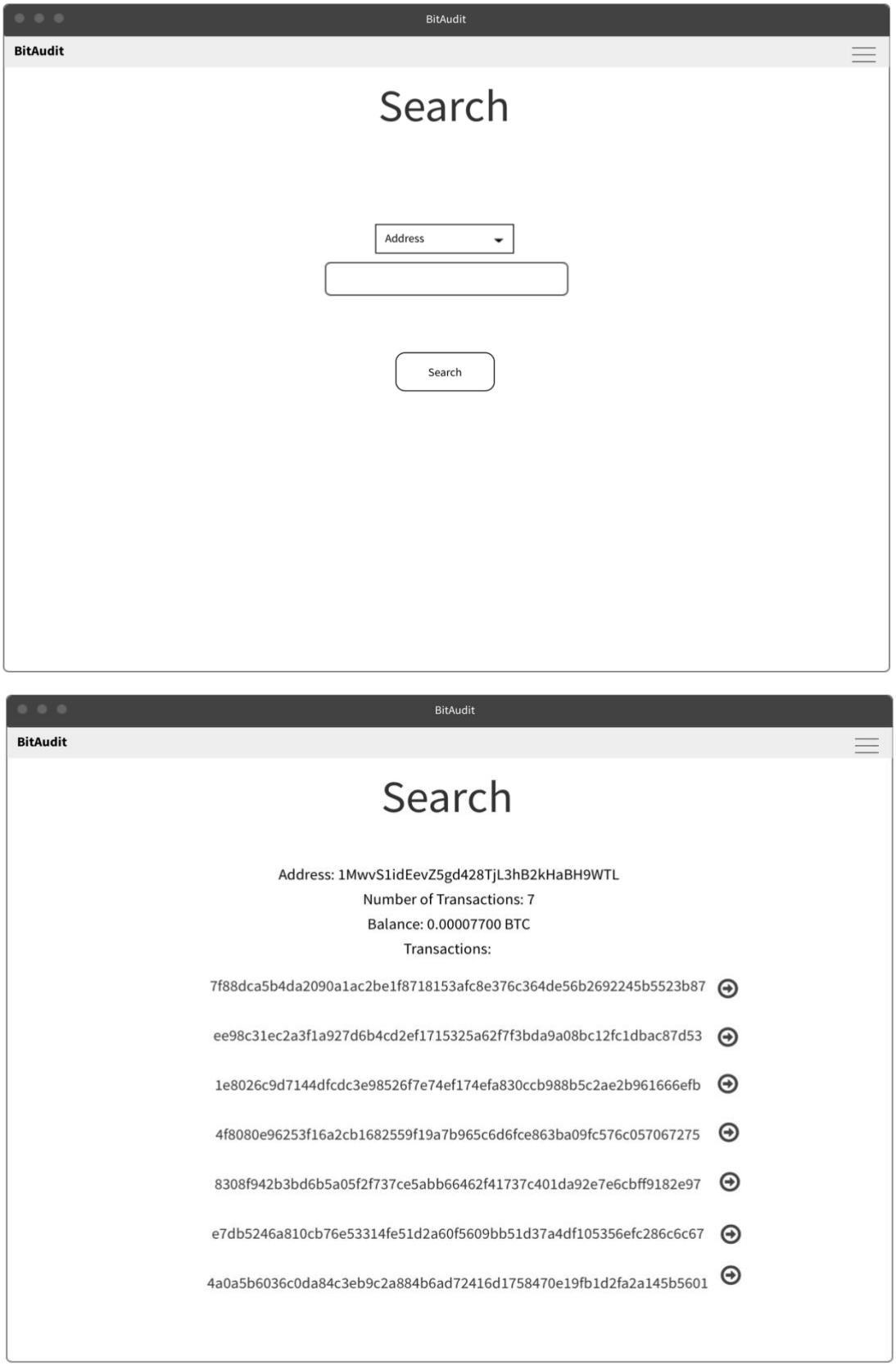


*Figura 23: Search (de endereço)*

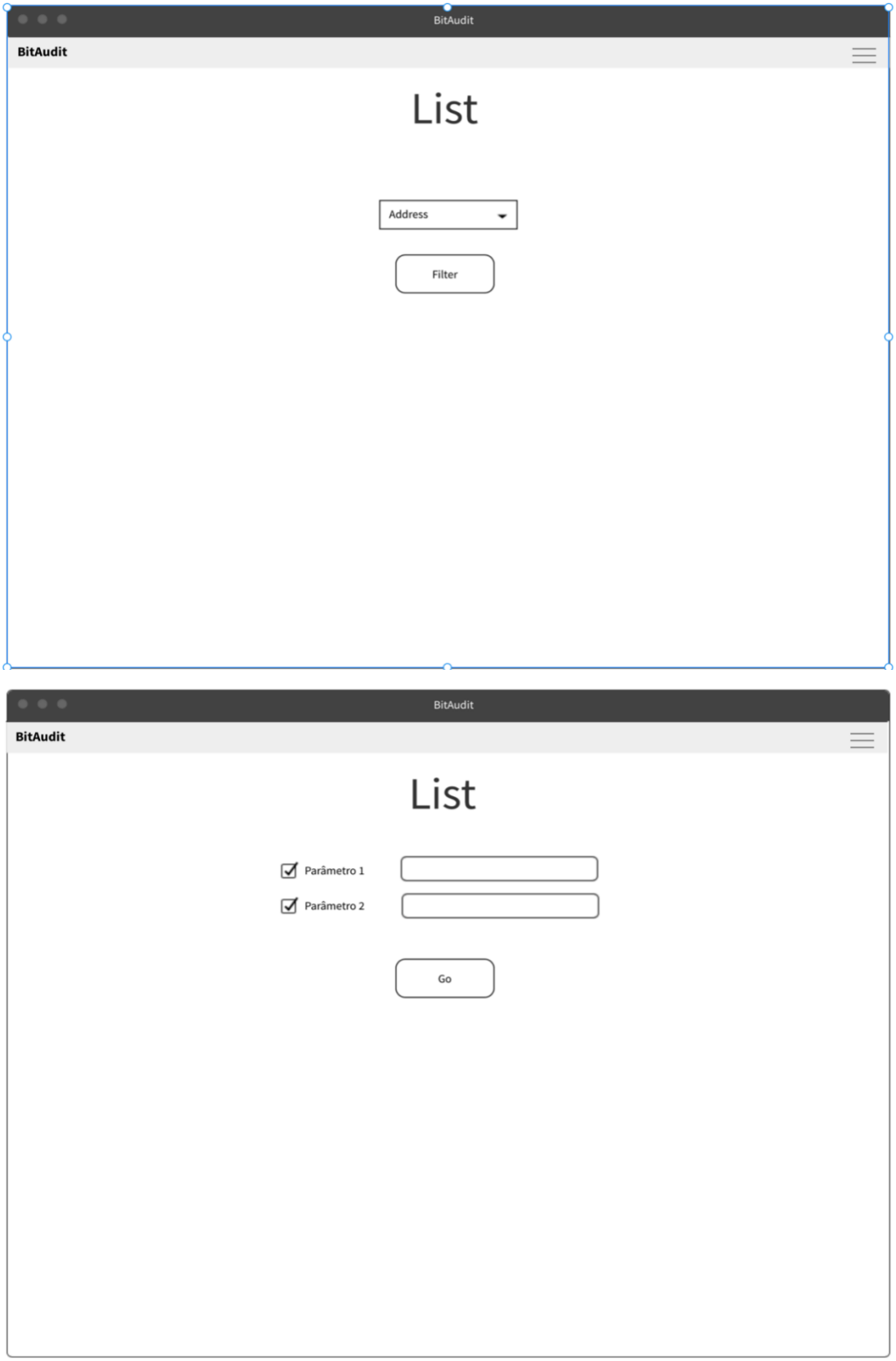


*Figura 24: Listagem de endereços, transações ou blocos*

Por fim, na Figura 24 temos a página relativa ao ***List***, onde é possível fazer a listagem de endereços, transações ou blocos, escolhido com o *dropdown*. Após esta escolha, aparecem os parâmetros de filtragem que vão ser diferentes consoante a seleção anterior. Todos estes parâmetros podem ou não ser utilizados para a filtragem, ao fazer "*check*" do lado esquerdo de cada parâmetro, sendo que, os que não forem selecionados não são considerados.

### 4.6.3 Aplicação da *Checklist* de Usabilidade

Como referenciado, foram identificadas as três áreas de foco para aplicar a *checklist* de usabilidade: **Página Inicial**, ***Search*** e ***List***, tal como apresentado nas Figuras 21, 22, 23 e 24. A partir da análise detalhada destas figuras foi possível identificar se as heurísticas de usabilidade são verificadas.

**1. Diálogo simples e natural**

Utilizamos uma interface simples, de fácil compreensão e com uma linguagem adequada para qualquer pessoa que use o *software*. Optamos por colocar o mínimo possível de informação por página, ou seja, apenas o necessário, de forma a evitar sobrecarregar a interface, sendo utilizado o nome "BitAudit*"* como botão para voltar à página inicial, permitindo assim, auxiliar esta simplicidade e organizar melhor as funcionalidades.

**2. Usar a linguagem do utilizador**

A linguagem é simples e de conhecimento geral do utilizador, não sendo usada qualquer linguagem técnica, mas sim termos que são naturalmente entendidos por qualquer pessoa. Quando utilizada linguagem mais técnica existe sempre algum tipo de explicação.

**3. Minimizar a carga de memória do utilizador**

O utilizador apenas necessita de recordar quais as opções mais importantes. As unidades encontram-se corretamente explicitadas, como se pode observar, por exemplo, na Figura 23, que apresenta o valor do saldo do endereço em *bitcoins*. As ações efetuadas são consistentes, visto que o botão *Home* (tal como o "BitAudit") redireciona-nos sempre para a página inicial, o botão *About* leva-nos às informações do *software* e o botão *Help* leva-nos para a página de ajuda e a questões mais frequentes já respondidas.

**4. Consistência**

O tipo de letra é o mesmo em todo o *software*. O também acontece com a palete de cores e o tipo de *design* usado. Como abordado no ponto 3, todas as ações são consistentes entre si. O mesmo pode ser dito da informação, uma vez que todas as informações estão dispostas sempre da mesma forma.

**5. Feedback**

Apesar de não estar representado nos *mockups,* o utilizador terá sempre *feedback* para que seja possível saber se os seus *inputs* e ações estão a ser executados de forma correta e se o sistema está a realizar a tarefa solicitada.

**6. Saídas claramente identificadas**

Como se pode observar nas Figuras 21, 22, 23 e 24, o utilizador pode navegar por estes menus e sempre voltar atrás, caso o pretenda fazer, de maneira fácil e intuitiva.

**7. Atalhos**

Na Figura 22 conseguimos identificar três atalhos para as opções gerais do sistema, *Home, About* e *Help*. O utilizador pode também usar o atalho de retroceder do seu *browser*, sempre que queira ir para a opção anterior.

**8. Boas mensagens de erro**

O sistema apresenta mensagens de erro claras e explícitas, de forma a permitir uma compreensão imediata do utilizador de qual será o erro ocorrido e, se possível, ajudá-lo a sair de uma situação não desejada (como, por exemplo, uma falha inesperada do programa ou do carregamento de informação).

**9. Prevenir erros**

De forma a evitar erros, a plataforma limita os *inputs* que o utilizador pode fornecer, o que poderia originar erros caso isso não existisse. A redução de *inputs* necessários nas funcionalidades disponíveis contribui para a prevenção de erros originados por parte dos utilizadores.

**10. Ajuda e documentação**

O sistema apresenta um menu de ajuda, que pode ser acedido através do botão *Help* (representado na Figura 22), no qual o utilizador pode encontrar uma explicação das funcionalidades da aplicação, suporte para possíveis erros que podem ocorrer aquando do seu uso e também uma página para mais alguns esclarecimentos no botão *About*.

# 5. Arquitetura do *Software*

Após a identificação e análise de requisitos, neste capítulo será abordada a arquitetura do *software* a desenvolver. Esta manifesta as primeiras tomadas de decisão de *design* do sistema, cujas ligações iniciais são fulcrais para o seu restante desenvolvimento, para a sua implementação e vida útil de manutenção [67]. É, então, o primeiro ponto em que as decisões de *design* pelas quais o sistema em construção se rege poderão ser analisadas [67].

A qualidade dos diagramas de arquitetura de *software* têm impacto nos seguintes aspetos: na comunicação, tanto dentro como fora da equipa de desenvolvimento; na integração de uma nova equipa; na identificação de risco; e na modelação de ameaças [68]. Uma boa arquitetura facilita o entendimento de todos sobre o *software* em construção, tornando a equipa mais eficiente [68]. Os diagramas que se podem realizar neste âmbito são uma ótima forma de demonstrar o planeamento da construção de um sistema [68]. Para tal, foi utilizado o modelo C4 e a ferramenta draw.io.

O modelo C4 foi criado para auxiliar equipas de desenvolvimento a descrever e comunicar a arquitetura do sistema durante as sessões iniciais de *design,* bem como a documentar de forma de retrospetiva uma base de código pré existente. É, pois, uma maneira de criar mapas do código, em vários níveis de detalhe [68]. Este modelo considera as estruturas estáticas de um sistema de *software* em termos de *containers, como* aplicações, armazenamentos de dados e *microservices*, bem como em termos de componentes e código. E as “pessoas” (ou utilizadores) servem-se dos sistemas de software construídos. O modelo C4 divide-se em 4 níveis de detalhe [68]. No presente trabalho, a análise foi feita até ao segundo nível, que consiste num diagrama de *containers*, que tem como objetivo ampliar o âmbito do sistema e expor os blocos de construção técnicos de alto nível [68].

## 5.1 Estrutura Geral

Esta secção contém uma explicação geral acerca da forma como se estruturou o *software*. Sem uma descrição detalhada de cada componente, pretende-se obter um entendimento do sistema como um todo e de como é possível atingir os requisitos não-funcionais descritos na secção 4.5.2.

Na Figura 25 é possível entender quais são os principais componentes do sistema. O utilizador acede ao *software* através de http//:localhost:5000 no *browser*, numa comunicação à *web application* feita através de HTTP. Este passo irá permitir o acesso às funcionalidades pretendidas em páginas desenvolvidas em PHP, HTML e CSS. Por sua vez, este vai comunicar com a API, que utiliza o *BlockSci* como base para percorrer a *blockchain* e realizar todas as análises pretendidas.

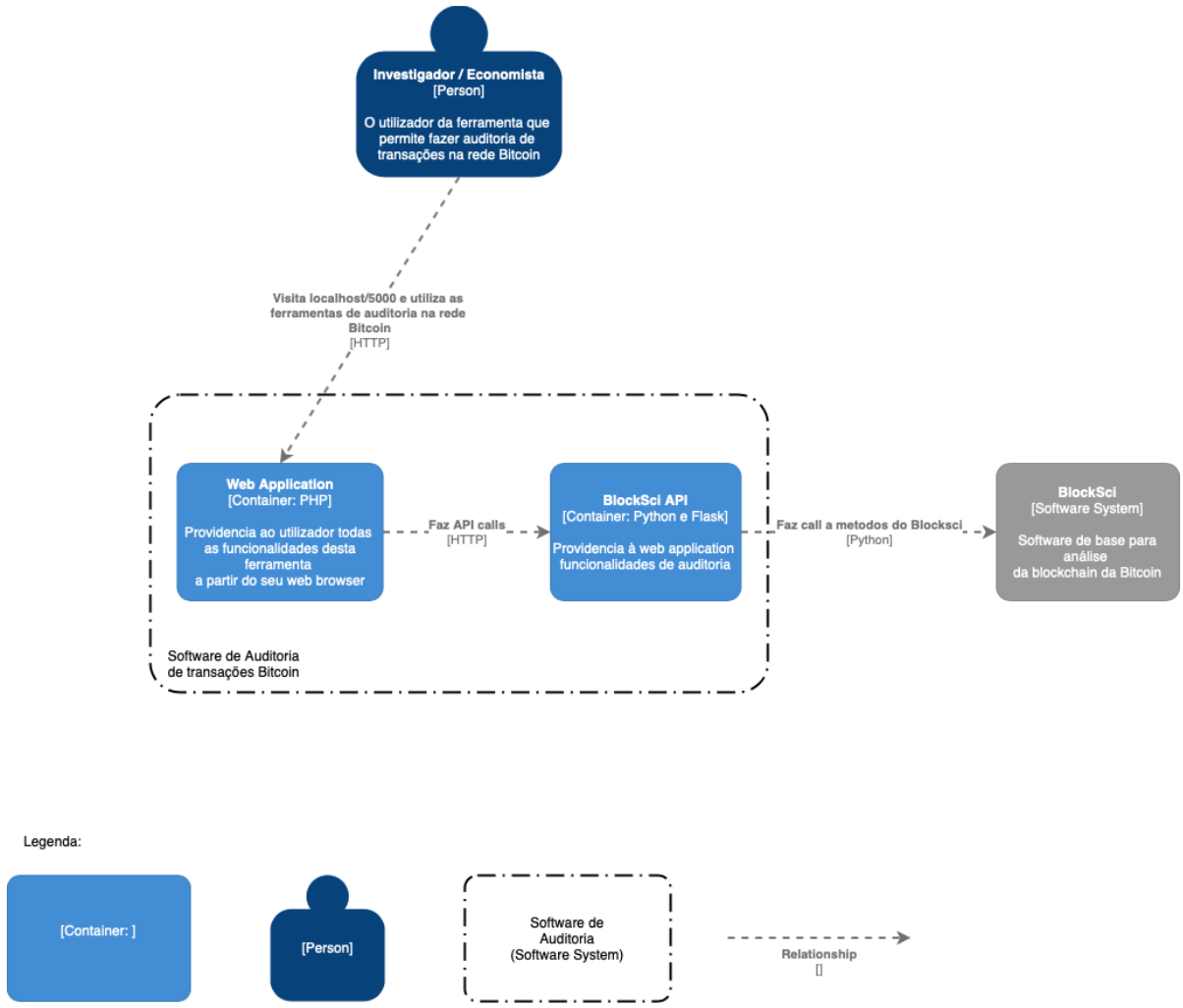


*Figura 25: Arquitetura do sistema (modelo C4)*

## 5.2 *Docker*

*Docker* é uma ferramenta construída para facilitar a criação, implementação e execução de aplicação com recurso a *containers* [69]. Os *containers* permitem ao programador juntar numa aplicação todas as parcelas que necessita, tais como bibliotecas e outras dependências, e a implemente como um pacote [69]. Com este procedimento e com o auxílio do *container*, quem desenvolve, pode certificar-se que o programa será executado em qualquer outra máquina *Linux*, independentemente de quaisquer configurações personalizadas, que podem ser diferentes da máquina usada para escrever e testar o código [69].

Ao contrário de uma máquina virtual, que cria um sistema operativo virtual completo, o *Docker* permite que os programas usem o mesmo *kernel Linux* do sistema em que estão a ser executados, exigindo apenas que as aplicações sejam enviadas com tarefas que ainda não estão em execução no computador *host* [69]. Esta função possibilita um aumento significativo de desempenho e a redução do tamanho do programa [69].

Neste caso, são utilizados dois *containers*, um para a componente com a qual o utilizador interage, ou seja, um servidor *Apache* e outro com uma API desenvolvida em *Flask* (*Python*), que está em contacto direto, no mesmo *container,* com o programa utilizado como base deste sistema, o *BlockSci*.

## 5.3 API em *Flask*

Flask é uma *microframework web* escrita em *Python*. É classificado como uma *microframework*, já que não requer ferramentas ou bibliotecas particulares [70][71]. Este não possui camada de abstração de base de dados, validação de formulário ou quaisquer outros componentes nos quais bibliotecas de terceiros pré-existentes fornecem funções comuns [71]. No entanto, o Flask oferece suporte a extensões que podem adicionar recursos do programa como se fossem implementados no próprio *Flask* [71]. Existem extensões para mapeadores relacionais de objetos, para validação de formulários, tratamento de *uploads* e várias tecnologias de autenticação aberta, bem como ferramentas comuns relacionadas à estrutura [71][72].

Para um projeto desta dimensão, o *Flask* possuía o que era pretendido, dado não existir a necessidade de estabelecer ligação com bases de dados e ser fundamental uma rápida curva de aprendizagem [73]. Além do exposto, é uma *microframework* leve, simples e flexível, que tem a possibilidade de utilizar um grande número de extensões complementares [73].

Optou-se pelo desenvolvimento de uma API separada da *web application*, existindo assim flexibilidade para ser utilizada em projetos futuros de uma forma mais simples, permitindo aos utilizadores personalizar o serviço, ao implementar novas funções e/ou fazer análises dos dados, de acordo com suas necessidades [74] e desenvolver diferentes tipos de *interface*, em *browser* e/ou *mobile*. Para além disso, as APIs são bastante úteis para integrar ou incorporar os serviços noutras aplicações e programas. Esta integração ou sincronização não depende do sistema operativo ou da linguagem de programação [74].

## 5.4 *Web Application*

O *Apache HTTP Server* é um *software* de servidor da *Web* gratuito e de *open-source*, suportado em várias plataformas. É desenvolvido e mantido por uma comunidade aberta de *developers* sob o domínio da *Apache Software Foundation* [75].

A grande maioria das instâncias do *Apache HTTP Server* corre em sistemas operativos *Linux* [76], no entanto, as versões atuais correm também em *Microsoft Windows, OpenVMS* e numa ampla variedade de sistemas do tipo *Unix* [77][78].

A escolha deste servidor prendeu-se com a sua facilidade de adequação às necessidades do nosso projeto, bem como a sua facilidade de integração com o *Docker.* Por outro lado, possui uma vasta informação na Internet, que facilita o esclarecimento de eventuais questões que vão surgindo ao longo do projeto. Paralelamente, foi utilizada a linguagem PHP para o desenvolvimento da *interface web*, por apresentar uma curva de aprendizagem relativamente rápida [79], adequada às limitações em termos temporais.

O PHP (um acrónimo para PHP - *Hypertext Preprocessor*) é uma linguagem de *script open source* de uso geral, largamente utilizada, sendo especialmente apropriada para o desenvolvimento *web,* com a possibilidade de utilização dentro do HTML [80].

HTML (*Hyper Text Markup Language)* é a linguagem de marcação padrão para a criação de páginas da *web*. Tem a capacidade de descrever a estrutura de uma página da *web,* consistindo numa série de elementos que informam o *browser* sobre como exibir o conteúdo. Os elementos HTML identificam os componentes exibidos ao utilizador, como "isto é um título", "isto é um parágrafo" ou "isto é um link” [81].

# 6. Implementação

O objetivo deste capítulo é explicar todo o processo relativo ao desenvolvimento do *software*, de acordo com o conjunto de requisitos reunidos no capítulo 4 e a arquitetura definida no capítulo 5.

Em primeiro lugar será abordado o ambiente em que o *software* foi desenvolvido, tanto a nível de *hardware* como de *software*. De seguida, serão abordadas quais as configurações realizadas para o desenvolvimento. Por fim, serão referenciadas todas as funcionalidades implementadas, não implementadas e alteradas, relativamente ao inicialmente estipulado nos requisitos.

O *software* desenvolvido foi colocado num repositório do *Github* e fornecidas todas as credenciais e instruções de utilização (ver anexo) aos professores orientadores para trabalhos futuros.

## 6.1 Ambiente de Desenvolvimento

Na implementação de *software*, um ambiente é um sistema de computador ou conjunto de sistemas em que um programa ou componente de *software* é implementado. Em casos simples, como desenvolver e executar imediatamente um programa na mesma máquina, pode haver um único ambiente. Porém, para uso industrial, o ambiente de desenvolvimento - no qual as alterações são feitas originalmente, - e o ambiente de produção - o que os utilizadores finais usam -, são separados, em geral, com vários estados intermediários [82].

Especificamente, para o ambiente de desenvolvimento, é fundamental tomar as melhores decisões possíveis, dado ser o local onde se reúnem todas as ferramentas necessárias para a execução do produto e simulação de utilização real, após lançamento ao público. Escolhas acertadas nesta fase facilitam o trabalho de quem testa e desenvolve o sistema.

Tal como referido na secção 2.4.2 (*BlockSci*), foi utilizada uma máquina virtual com as seguintes configurações necessárias para que o *software* fosse executado de forma mais rápida:

- *Ubuntu* 20.04
- 64 GB de RAM
- 1TB de armazenamento

Sendo que o *BlockSci* foi a base para a nossa ferramenta, foram estas as especificações mantidas para o seu desenvolvimento. Também algo já referido na secção 2.4.2 (*BlockSci*), surgiu a necessidade de utilizar uma versão mais antiga do *Ubuntu* (18.04). A opção que surgiu foi a de utilizar *Docker*, algo que facilitará também a utilização por parte utilizador final. Foram então configurados dois *containers* separados que comunicam por chamadas HTTP - protocolo da camada de aplicação desenvolvido para comunicação entre navegadores e servidores web [83] - com respostas em JSON - formato que utiliza texto legível por humanos para armazenar e transmitir objetos de dados, que consistem em pares de atributo-valor e valores serializáveis [84].

Por fim, foi utilizado o *Nano*, um editor de texto do terminal para sistemas operativos *Unix* e *Linux*. Este inclui todas as funcionalidades básicas de um editor de texto, como destaque de sintaxe, vários *buffers*, pesquisa e substituição com suporte a expressões regulares, verificação ortográfica e codificação UTF-8 [85]. A tomada desta decisão, bem como a da não utilização de qualquer *IDE* (ambiente de desenvolvimento integrado), deveu-se ao facto de o desenvolvimento ter sido feito a partir da máquina virtual e do terminal de um *Macbook,* com ligação à máquina virtual a partir de um *VPN.* Deste modo, tornou-se possivel a utilização do mesmo *software* para desenvolver em qualquer uma das situações.

## 6.2 Configurações

Foi utilizado *docker-compose* para que fosse possivel, com um único comando, criar e iniciar todos os serviços da configuração realizada. *Compose* é uma ferramenta para definir e executar aplicações *Docker* de vários *containers*. Com esta ferramenta, é utilizado um ficheiro *YAML* para configurar os serviços da aplicação (Figura 26).

O nosso ficheiro em específico foi dividido em dois serviços distintos: a API, que utiliza o BlockSci como base, e o servidor *web*.

```
version: '3.7'

services:
    blocksci:
        build: "/home/admin/BITCOIN/build/blocksci_dock_0.7.0"
        container_name: blocksci
        restart: always
        volumes:
            - /home/admin/BITCOIN:/data/BITCOIN
        ports:
            - "5000:80"

    web:
        image: "php:7.2-apache"
        container_name: web_container
        restart: always
        volumes:
            - ./web:/var/www/html
        ports:
            - "8888:80"
        depends_on:
            - blocksci
```

*Figura 26: docker-compose-yml*

O **serviço da API** começa por definir onde está o *Dockerfile* a partir do qual será construída a imagem e corrido o *container*. Este ficheiro contempla tudo o que é necessário para utilizar o *BlockSci* e correr uma REST API, como se pode verificar na Figura 27.

```
FROM ubuntu:18.04
LABEL maintainer="Paulo Melo <pmelo@fe.uc.pt>"

RUN apt-get update && apt-get install -y software-properties-common python3-software-properties
RUN add-apt-repository ppa:ubuntu-toolchain-r/test -y && apt-get update
RUN apt install -y cmake libtool autoconf libboost-filesystem-dev libboost-iostreams-dev libboost-serialization-dev libboost-thread-dev libboost-test-dev
\libssl-dev libjsoncpp-dev libcurl4-openssl-dev libjsoncpp-dev libjsonrpccpp-dev libsnappy-dev zlib1g-dev libbz2-dev liblz4-dev libzstd-dev libjemalloc-dev libsparsehash-dev python3-dev python3-pip git
RUN update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-7 60 --slave /usr/bin/g++ g++ /usr/bin/g++-7

WORKDIR /root
RUN git clone https://github.com/bitcoin-core/secp256k1
WORKDIR /root/secp256k1
RUN sh ./autogen.sh
RUN ./configure --enable-module-recovery
RUN make -j7 install

WORKDIR /root
RUN git clone https://github.com/citp/BlockSci.git
WORKDIR /root/BlockSci
RUN git submodule init
RUN git submodule update --recursive
RUN git fetch --all
RUN git checkout
RUN ln -s external libs
RUN cp -r libs/range-v3/include/meta /usr/local/include
RUN cp -r libs/range-v3/include/range /usr/local/include
RUN mkdir -p /root/BlockSci/release

WORKDIR /root/BlockSci/external/rocksdb
RUN make -j7 static_lib
RUN make -j7 shared_lib
RUN make install

WORKDIR /root/BlockSci/release
RUN CC=gcc-7 CXX=g++-7 cmake -j7 -DCMAKE_BUILD_TYPE=Release ..
RUN make install

WORKDIR /root/BlockSci/
RUN CC=gcc-7 CXX=g++-7 pip3 install -e blockscipy

RUN pip3 install --upgrade pip
RUN pip3 install --upgrade multiprocess psutil jupyter pycrypto matplotlib pandas dateparser

RUN mkdir /root/BlockSci/external/bitcoin-api-cpp/release
WORKDIR /root/BlockSci/external/bitcoin-api-cpp/release
RUN cmake -DCMAKE_BUILD_TYPE=Release -j7 ..
RUN make install
RUN mkdir /data
WORKDIR /data
COPY . .
RUN apt-get update && apt -y install python3-pip && pip install flask && pip install flask-restful && pip install pandas
CMD ["python3", "main.py"]
```

*Figura 27: Dockerfile*

De seguida, foi definida uma pasta de ficheiros que pudesse ser acedida dentro do *container,* com a palavra-chave "*volumes*". Isto permite, entre outras coisas, modificar o código sem reconstruir a imagem. Por fim, é ligado o *container* ao porto 80 e a máquina *host* ao porto 5000. Este serviço utiliza o porto padrão para o servidor da *web Flask*, 5000.

O *container* que inclui o servidor *web*, contrariamente ao primeiro, utiliza uma imagem pública retirada do *site*: https://hub.docker.com: `php:7.2-apache`. Também é definida uma pasta acessível dentro do *container* que, neste caso, vai conter os ficheiros relativos às páginas *web*. É ligado o *container* ao porto 80 e a máquina *host* ao porto 8888, onde se vai estabelecer a conexão a partir de um *browser* fora do *container*. Por fim, contém a informação de que este ~~se~~ vai apenas iniciar após o serviço da API com a palavra-chave "*depends_on*".

De forma a correr os dois *containers* em simultâneo, apenas é necessário correr o comando `docker-compose up` (com a opção de adicionar a *flag* `–build`, se a intenção for criar as imagens antes de iniciar os *containers*) na diretoria onde se encontra o ficheiro *docker-compose.yml*. Ao correr este comando, vão ser iniciados o servidor *web*, onde o utilizador poderá interagir com a ferramenta através do *browser* no endereço *localhost* e no porto 8888, e a REST API, no qual vão ser efetuadas todas as operações de análise e transmitidas ao servidor *web*, no endereço 0.0.0.0 e no porto 5000.

## 6.3 Funcionalidades

As funcionalidades que compõem o *software* foram desenvolvidas da seguinte forma:

- **Pesquisa** de **endereços**, **transações** (com o *hash*) ou **blocos** (com a *height*). Isto permite ao utilizador ver os detalhes de cada um destes elementos. Relativamente aos endereços, o programa vai mostrar o número de transações (como *input* e *output*), o saldo do endereço e uma lista de transações às quais se pode aceder para obter mais detalhes sobre cada uma, como se pode observar na Figura 28. No caso das transações, é possível verificar a data e hora em que foi feita a transferência, bem como o valor total de *input* (em *bitcoins*), o valor total de *output* (em *bitcoins*), a taxa (em *bitcoins*) e a *height* do bloco em que se insere. Neste, é possível verificar os seus pormenores e uma lista dos endereços de *input* e *output*, dos quais também se pode ver os detalhes, tal como se pode verificar na Figura 29. Por fim, temos os blocos, nos quais conseguimos observar o número e uma lista de transações que os compõem (das quais também se pode aceder de forma mais pormenorizada), o *nonce*, a data e a hora de confirmação do bloco, o volume de transações (em *bitcoins*) e a recompensa do *miner* após a confirmação (em *bitcoins*), conforme a Figura 30.

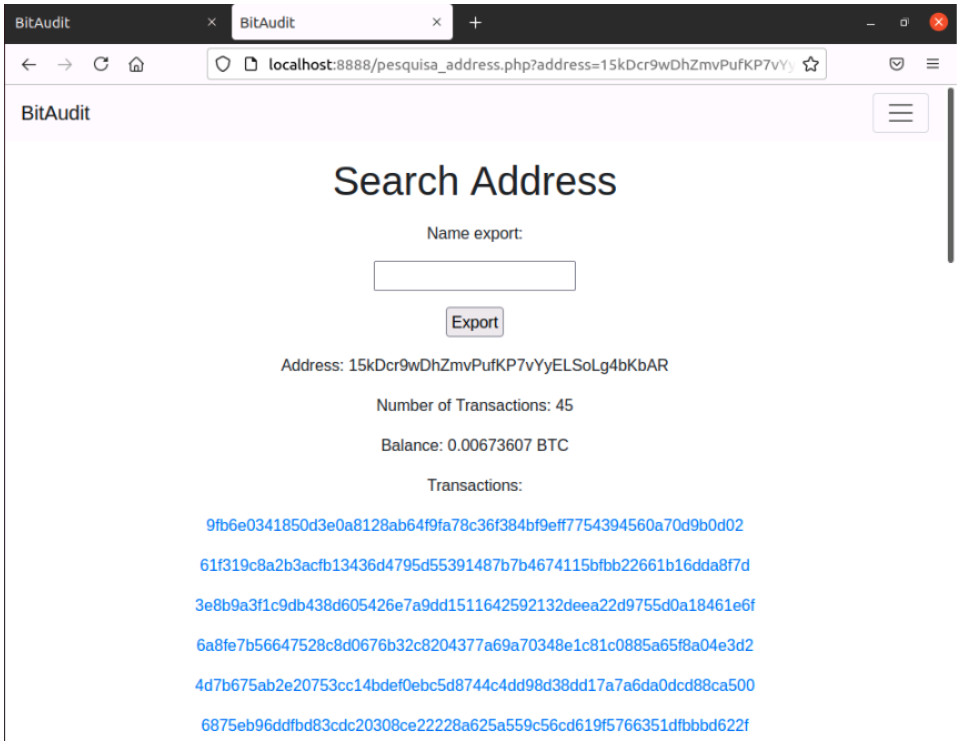


*Figura 28: Exemplo da pesquisa de um endereço*

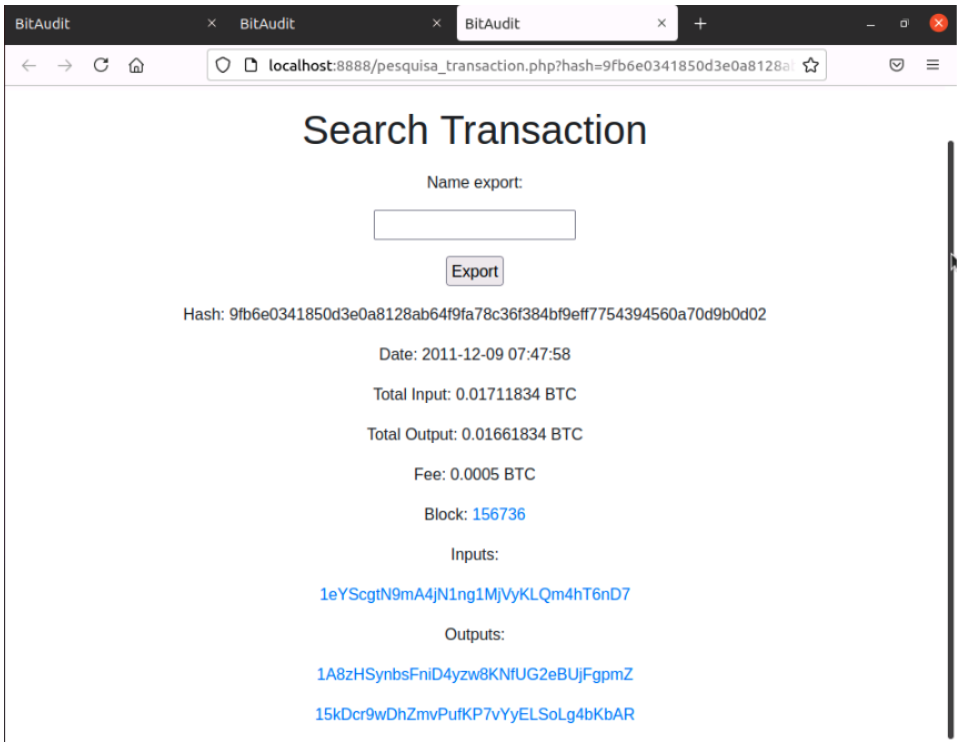


*Figura 29: Exemplo da pesquisa de uma transação*

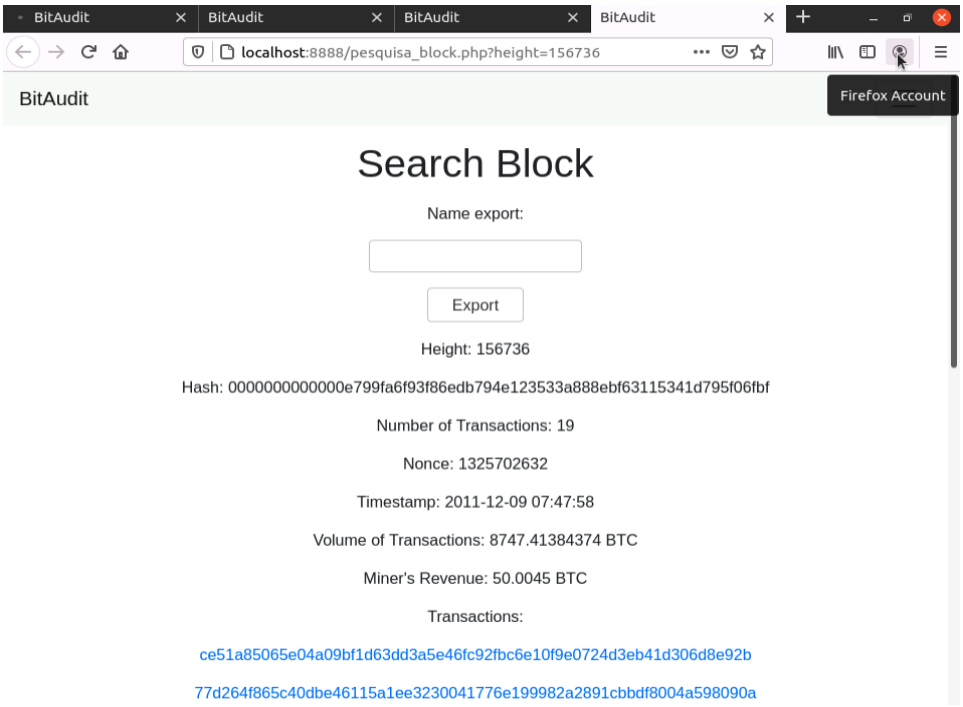


*Figura 30: Exemplo da pesquisa de um bloco*

Em termos de código presente na *REST API* é simples, acede-se diretamente a funcionalidades já presentes no *BlockSci*, para o caso do endereço da transação, escolhendo o que se quer retornar ao servidor *web* (em JSON) para mostrar ao utilizador, como ilustra a Figura 31.

```
def search(string, type):
        global tmp
        if(type == "address"):
                try:
                        address = chain.address_from_string(string)
                        tmp = [string, str(address.input_txes_count() + address.output_txes_count()), str(address.balance(-1)/(10**8)), address.txes.to_list()]
                        return [string, str(address.input_txes_count() + address.output_txes_count()), str(address.balance(-1)/(10**8)), address.txes.to_list()]
                except:
                        return []

        elif(type == "transaction"):
                try:
                        txe = chain.tx_with_hash(string)
                        tmp = [string, str(txe.block_time), str(txe.input_value/(10**8)), str(txe.output_value/(10**8)), str(txe.fee/(10**8)), str(txe.block.height), txe.inputs.to_list(), txe.outputs.to_list()]
                        return [string, str(txe.block_time), str(txe.input_value/(10**8)), str(txe.output_value/(10**8)), str(txe.fee/(10**8)), str(txe.block.height), txe.inputs.to_list(), txe.outputs.to_list()]
                except:
                        return []

        else:
                lista_aux = blocks.to_list()
                for i in range(len(lista_aux)):
                        if(str(lista_aux[i].height) == string):
                                block = lista_aux[i]

                                tmp = [string, str(block.hash), str(block.tx_count), str(block.nonce), str(block.time), str(block.output_value/(10**8)), str(block.revenue/(10**8)), block.txes.to_list()]
                                return [string, str(block.hash), str(block.tx_count), str(block.nonce), str(block.time), str(block.output_value/(10**8)), str(block.revenue/(10**8)), block.txes.to_list()]

                return []
```

*Figura 31: Excerto de código relativo à pesquisa*

- **Listagem** de **endereços**, **transações** ou **blocos** com possibilidade de parametrização, dependendo da opção escolhida. No que diz respeito aos endereços, é possível parametrizar a listagem tanto em termos de número de transações associadas, como de saldo atual, como se verifica na Figura 32. No caso das transações, pode-se selecionar a taxa (em *bitcoins*), o valor (em *bitcoins*) e a data de início e/ou de fim (Figura 33). Por

último, relativamente aos blocos, a filtragem pode ser feita com o número de transações dentro do bloco, quantidade movida (em *bitcoins*) e a data de início e/ou de fim (Figura 34).

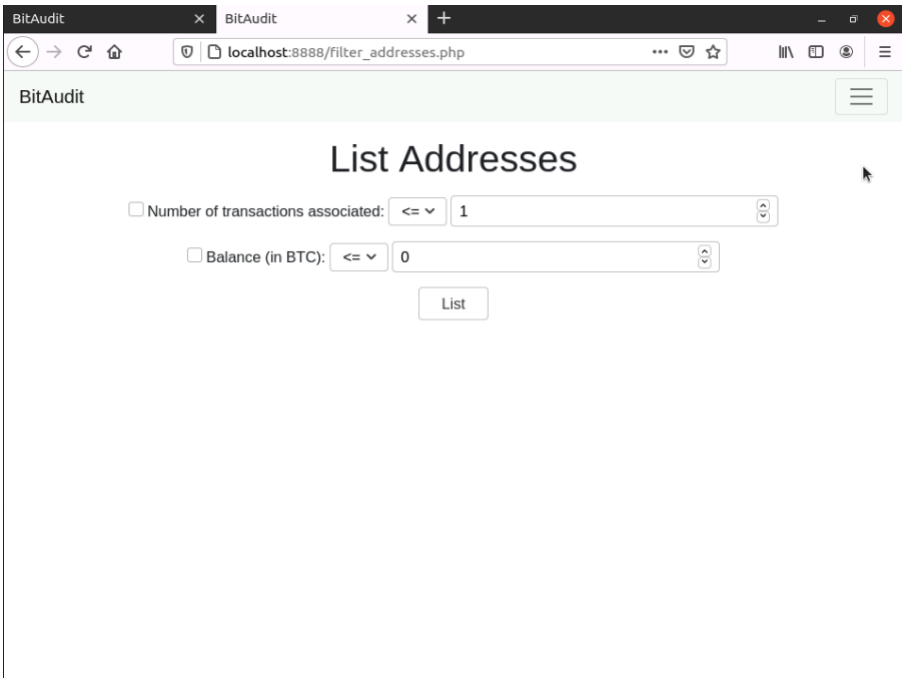


*Figura 32: Parametrização da listagem de endereços*

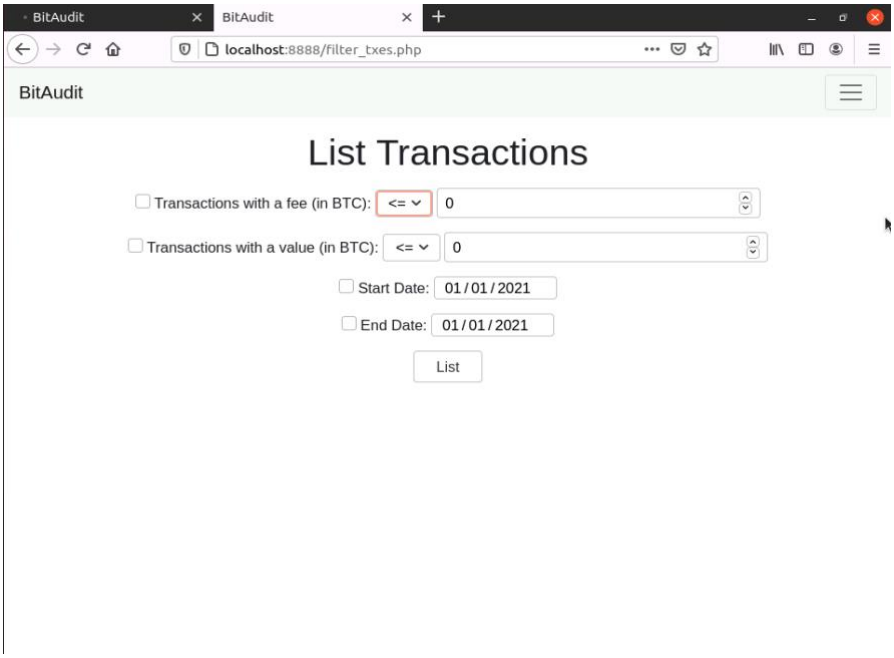


*Figura 33: Parametrização da listagem de transações*

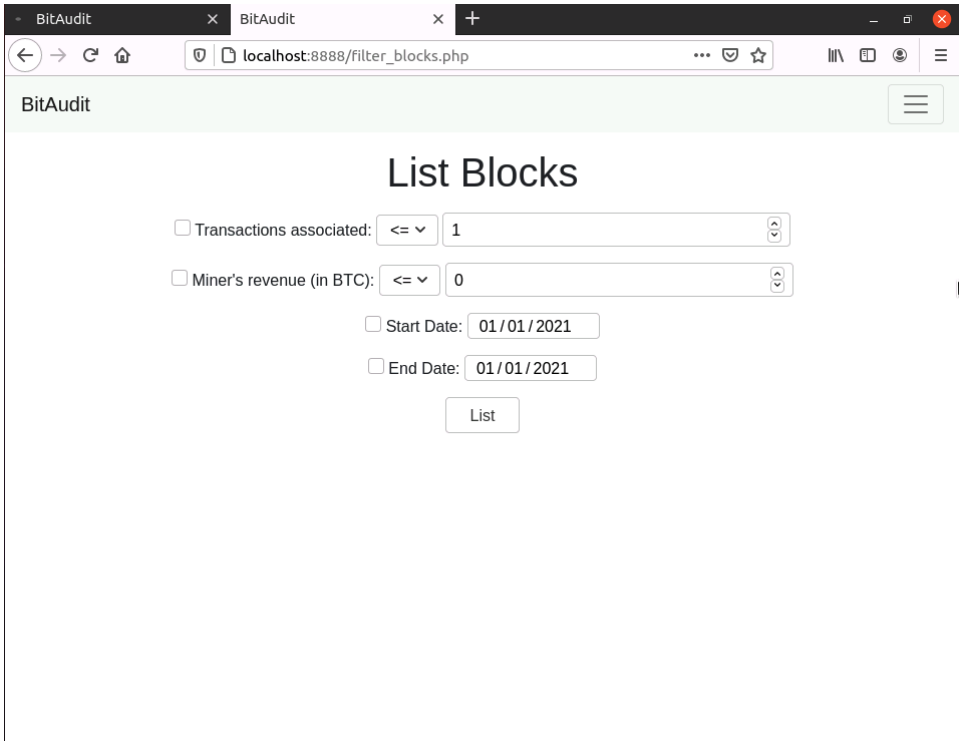


*Figura 34: Parametrização da listagem de blocos*

No que diz respeito ao código, é em tudo bastante semelhante, tanto para endereços, transações ou blocos. A partir de uma função disponível na ferramenta de base, `where()` (Figura 35), que permite fazer a filtragem diretamente de todos os parâmetros (um ou mais de cada vez) à exceção da data, que é feita como está descrito na Figura 36.

```
elif(option == "transaction"):
        if(bool1 == "True" and bool2 == "True"):
                if(option_1 == ">=" and option_2 == ">="):
                        lista_txes = blocks.txes.where(lambda tx: tx.fee >= int(param1*(10**8)) and tx.input_value >= int(param2*(10**8)))

                elif(option_1 == "<=" and option_2 == ">="):
                        lista_txes = blocks.txes.where(lambda tx: tx.fee <= int(param1*(10**8)) and tx.input_value >= int(param2*(10**8)))

                elif(option_1 == "==" and option_2 == ">="):
                        lista_txes = blocks.txes.where(lambda tx: tx.fee == int(param1*(10**8)) and tx.input_value >= int(param2*(10**8)))

                elif(option_1 == ">=" and option_2 == "=="):
                        lista_txes = blocks.txes.where(lambda tx: tx.fee >= int(param1*(10**8)) and tx.input_value == int(param2*(10**8)))

                elif(option_1 == "<=" and option_2 == "=="):
                        lista_txes = blocks.txes.where(lambda tx: tx.fee <= int(param1*(10**8)) and tx.input_value == int(param2*(10**8)))

                elif(option_1 == "==" and option_2 == "=="):
                        lista_txes = blocks.txes.where(lambda tx: tx.fee == int(param1*(10**8)) and tx.input_value == int(param2*(10**8)))

                elif(option_1 == ">=" and option_2 == "<="):
                        lista_txes = blocks.txes.where(lambda tx: tx.fee >= int(param1*(10**8)) and tx.input_value <= int(param2*(10**8)))

                elif(option_1 == "<=" and option_2 == "<="):
                        lista_txes = blocks.txes.where(lambda tx: tx.fee <= int(param1*(10**8)) and tx.input_value <= int(param2*(10**8)))

                elif(option_1 == "==" and option_2 == "<="):
                        lista_txes = blocks.txes.where(lambda tx: tx.fee == int(param1*(10**8)) and tx.input_value <= int(param2*(10**8)))
```

*Figura 35: Excerto de código que mostra como é feita a filtragem*

```
def compara_datas(data_1, data_2, data_3):
        if((data_1 - data_2).days >= 0 and (data_3 - data_1).days >= 0):
                return True

        else:
                return False
```

*Figura 36: Excerto de código que mostra como é feita a filtragem por data (posterior ao código mostrado na Figura 35)*

- **Estudo de *cluster***, que permite ver endereços e transações associados a um determinado endereço (fornecido como *input* pelo utilizador). Posteriormente, é possível ver os detalhes dos endereços e transações resultantes (Figura 37). Ao utilizarmos uma função de base do *BlockSci*, a explicação dada pelos seus criadores é a seguinte [56]: "O módulo de *cluster* do *BlockSci* fornece aos utilizadores a capacidade de aplicar técnicas de armazenamento em *cluster* baseadas em heurísticas realizadas a uma instância da *blockchain*. O *ClusterManager* é a principal entrada neste módulo. Usando-o, é possível abrir um *clustering* já produzido ou criar um novo com base numa heurística de mudança de endereço à escolha. O módulo de agrupamento do *BlockSci* é um esforço simples para explorar técnicas de agrupamento baseadas em heurísticas. Os utilizadores não devem presumir que os resultados serão corretos na prática. Fornecer um mecanismo de agrupamento mais preciso é um projeto de pesquisa em andamento. Um *clustering* consiste numa lista de objetos *blocksci.cluster.Cluster*, cada um com uma lista de endereços que foram marcados como membros do *cluster*. Os utilizadores podem encontrar com eficiência em qual *cluster* um endereço está, bem como descobrir quais endereços um *cluster* contém".

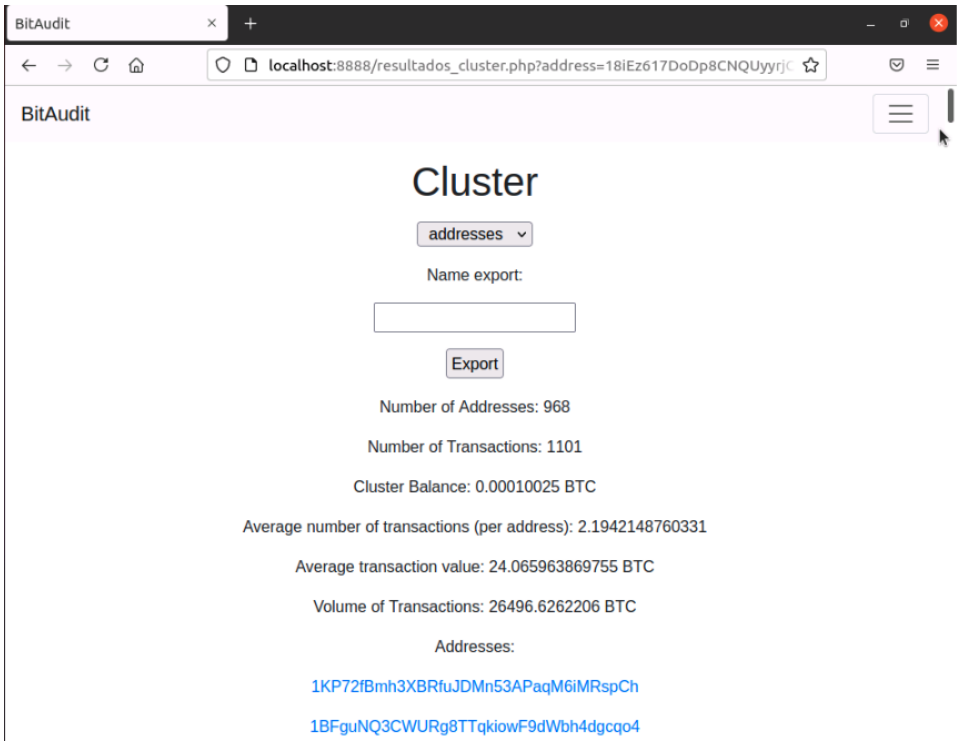


*Figura 37: Exemplo de estudo de cluster ao endereço 1EGZQ7hayB35tH6mt3g6YXWxRD8VCKsnWu*

***"Address -> Entity"***, que reúne endereços que podem estar associados à carteira à qual pertence o endereço (fornecido como *input* pelo utilizador). Neste caso, foi baseado na seguinte heurística: "As transações com vários *inputs* ocorrem quando um utilizador deseja realizar uma transferência e o valor do pagamento excede o valor de cada uma das *bitcoins* disponíveis na sua carteira. Para evitar a realização de várias transações para completar o pagamento, tendo que pagar várias taxas de transação, os utilizadores escolhem um conjunto de endereços da carteira (desse utilizador) de modo que seu valor total corresponda ao pagamento e que seja realizado em transações com vários *inputs*. Isto significa que sempre (ou quase sempre) que uma transação tem vários endereços de entrada, possamos assumir com segurança que pertencem à mesma carteira e, portanto, ao mesmo utilizador." [5]. É possível ver os detalhes dos endereços e transações resultantes. Relativamente ao código, segue a heurística descrita anteriormente e reúne todas as transações realizadas por esse conjunto de endereços (Figura 38).

```
    same_owner.append(address)

    for bl in data:
        for i in range(bl.txes.to_list()):
            inputs = bl.txes.to_list()[i].inputs.to_list()

            for j in range(len(inputs)):
                if((type(inputs[j].address) is not blocksci.NonStandardAddress) and (type(inputs[j].address) is not blocksci.OpReturn) and (type(inputs[j].address) is not blocksci.Multisi
                    if(inputs[j].address.address_string in same_owner):
                        aux += inputs
                        final_txes.append(bl.txes.to_list()[i])
                        valor_txes += bl.txes.to_list()[i].input_value
                        break

    for i in range(len(aux)):
        same_owner.append(aux[i].address.address_string)

    # duplicates
    seen = set()
    final = [x for x in same_owner if x not in seen and not seen.add(x)]
```

*Figura 38: Excerto de código de como é utilizada a heurística descrita para a funcionalidade "Address->Entity"*

- **"*Shadow Guessing*"**, que reúne endereços que possam ter sido usados para receber "troco" de transações em que o endereço, fornecido como *input* pelo utilizador, foi utilizado como entrada. É baseado na heurística: "Se houver dois endereços de saída, e um dos dois nunca apareceu antes no *blockchain*, enquanto o outro apareceu, então podemos assumir com segurança que o que nunca apareceu antes é o *shadow address* gerado pelo cliente para receber as alterações de volta." [5]. É possível ver os detalhes dos endereços resultantes. Como na funcionalidade anterior, em termos de código, segue o que foi descrito na heurística, como se pode verificar na Figura 39.

```
for bl in data:
        for i in range(len(bl.txes.to_list())):
                outputs = bl.txes.to_list()[i].outputs.to_list()
                inputs = bl.txes.to_list()[i].inputs.to_list()
                inputs_strings = transform_list_to_string(inputs)
                outputs_strings = transform_list_to_string(outputs)

                if((address in inputs_strings) and (len(outputs) == 2)):
                        if((outputs_strings[0] not in addr_record) and (outputs_strings[1] in addr_record)):
                                shadow_address.append(outputs_strings[0])
                        elif((outputs_strings[1] not in addr_record) and (outputs_strings[0] in addr_record)):
                                shadow_address.append(outputs_strings[1])
                        else:
                                pass

                addr_record += inputs_strings + outputs_strings
```

*Figura 39: Excerto de código de como é utilizada a heurística descrita para a funcionalidade "Shadow Guessing"*

- **Análise da *Blockchain***, que estuda períodos de atividade específicos (fornecidos pelo utilizador) que possam ser associados a um determinado evento identificável no tempo. Mostra ao utilizador dados, como o valor da menor e maior transação, número de transações total, número de blocos e valor total transacionado (Figura 40). Relativamente ao código, é feito ao percorrer os blocos e transações daquele período temporal e ir retirando os valores pertinentes para o cálculo dos dados apresentados ao utilizador.

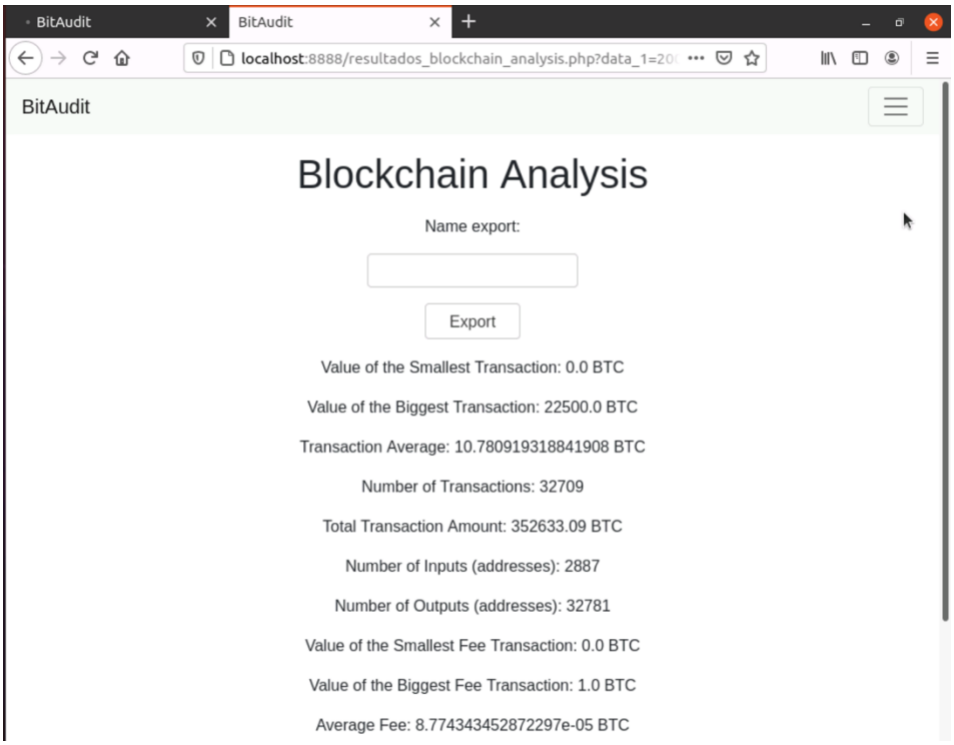


*Figura 40: Exemplo de análise do período de 01/01/2009 a 01/01/2010*

- **Endereços inativos** (ou “adormecidos”), em que são analisados alguns aspetos, como volume de transações e número médio de transações por endereço, de endereços inativos durante um determinado período de tempo escolhido pelo utilizador. É possivel ver os detalhes dos endereços resultantes. Relativamente ao código, o programa vai percorrer as transações *blockchain,* em cada uma é verificado se os endereços participantes estão ou não inativos, na duração que o utilizador indica. Optou-se por percorrer as transações e não os endereços, já que, ao chegar ao ponto em que se os endereços estão a efetuar ou a receber transações, significa que não estão inativos. Isto permite que o tempo de análise seja menor (Figura 41).

```
        for bl in blocks_1:
                if(today_date.year - block.time.year < years):
                        break

                for tx in bl.txes.to_list():
                        lista = tx.inputs.to_list() + tx.outputs.to_list()

                        for i in range(len(lista)):
                                if((type(lista[i].address) is not blocksci.NonStandardAddress) and (type(lista[i].address) is not blocksci.OpReturn)
and (type(lista[i].address) is not blocksci.MultisigAddress)):
                                        if(lista[i].address.address_string not in aux):
                                                if(len(lista[i].address.input_txes.to_list()) + len(lista[i].address.output_txes.to_list()) == 0):
                                                        final.append(lista[i].address.address_string)

                                                else:
                                                        if(len(lista[i].address.input_txes.to_list()) != 0 and len(lista[i].address.output_txes.to_list()) != 0):
                                                                if((today_date - lista[i].address.input_txes.to_list()[-1].block_time).days >= (years*365) and
(today_date - lista[i].address.output_txes.to_list()[-1].block_time).days >= (years*365)):
                                                                        final.append(lista[i].address.address_string)

                                                        elif(len(lista[i].address.input_txes.to_list()) != 0 and len(lista[i].address.output_txes.to_list()) == 0):
                                                                if((today_date - lista[i].address.input_txes.to_list()[-1].block_time).days >= (years*365)):
                                                                        final.append(lista[i].address.address_string)

                                                        elif(len(lista[i].address.input_txes.to_list()) == 0 and len(lista[i].address.output_txes.to_list()) != 0):
                                                                if((today_date - lista[i].address.output_txes.to_list()[-1].block_time).days >= (years*365)):
                                                                        final.append(lista[i].address.address_string)

                                                aux.append(lista[i].address.address_string)
```

*Figura 41: Excerto de código relativo à procura de endereços inativos*

- **“Whales”**, nos quais são identificados endereços cujo saldo é superior a 1.000 BTC, ~~e~~ apresentados alguns dados que possam ser relevantes, tais como volume de transações e valor médio por transação. Em geral é semelhante à listagem de endereços, com a ressalva de ser mais detalhado.

- **Endereços mais ativos**, em que são reunidas informações e dados associados aos dez endereços mais ativos na rede Bitcoin, num período à escolha do utilizador. É possível ver os detalhes dos endereços resultantes, o número de transações que cada um efetuou e quanto é que isto representa em termos percentuais em relação aos restantes endereços no mesmo período temporal (Figura 42).

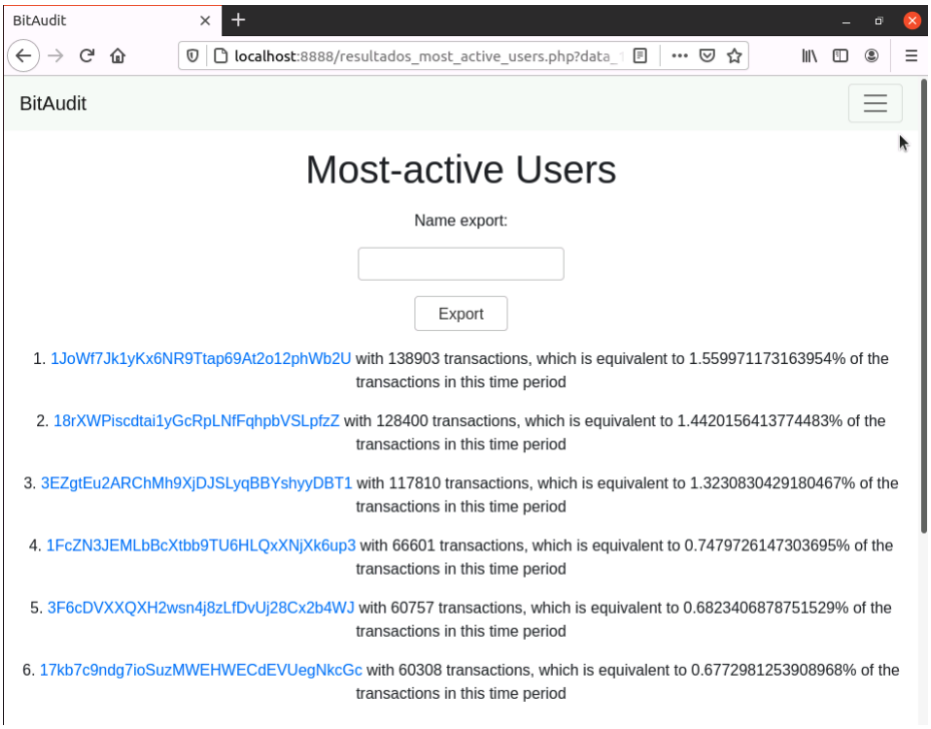


*Figura 42: Exemplo de análise dos endereços mais ativos no período de 01/03/2020 a 01/04/2020*

No que diz respeito à implementação, são percorridas as transações daquele período temporal e identificados os endereços que são utilizados e colocados num dicionário com o respetivo número de transações. Após a obtenção de todos os dados necessários, são colocados e ordenados (*insertion sort* retirado e adaptado [86]) para mostrar os mais ativos por ordem decrescente (Figura 43).

```
        high_score = [["1" , 0] , ["2" , 0], ["3" , 0], ["4" , 0], ["5" , 0], ["6" , 0] , ["7" , 0], ["8" , 0], ["9" , 0], ["10" , 0]]

        for bl in data:
                if(compara_datas(bl.time, data_2, data_3)):
                        Flag = True

                        for i in range(len(bl.txes.to_list())):
                                #print("Here 1")
                                inputs = bl.txes.to_list()[i].inputs.to_list() + bl.txes.to_list()[i].outputs.to_list()

                                for j in range(len(inputs)):
                                        if((type(inputs[j].address) is not blocksci.NonStandardAddress) and (type(inputs[j].address) is not blocksci.OpReturn)
and (type(inputs[j].address) is not blocksci.MultisigAddress)):
                                                #print("Here 2")
                                                if(inputs[j].address.address_string in dic_addr_tx.keys()):
                                                        #print("Here 2")
                                                        dic_addr_tx[inputs[j].address.address_string] += 1
                                                else:
                                                        #print("Here 3")
                                                        dic_addr_tx.setdefault(inputs[j].address.address_string, 1)

                                total += 1

                else:
                        if(Flag):
                                break

                        else:
                                continue


        for name in dic_addr_tx.keys():
                if(dic_addr_tx[name] >= high_score[9][1]):
                        high_score.append([name, dic_addr_tx[name]])
                        aux = high_score
                        high_score = insertionSort(aux)
                        high_score.pop()
```

*Figura 43: Excerto de código relativo à procura dos endereços mais ativos num período específico*

Estudo de **endereços utilizados apenas uma vez**, que reúne informações e detalhes associados aos endereços que apenas foram utilizados uma vez na rede Bitcoin, seja para receber ou enviar *bitcoins*, num determinado período de tempo definido pelo utilizador. É possivel ver os detalhes dos endereços resultantes (Figura 44).

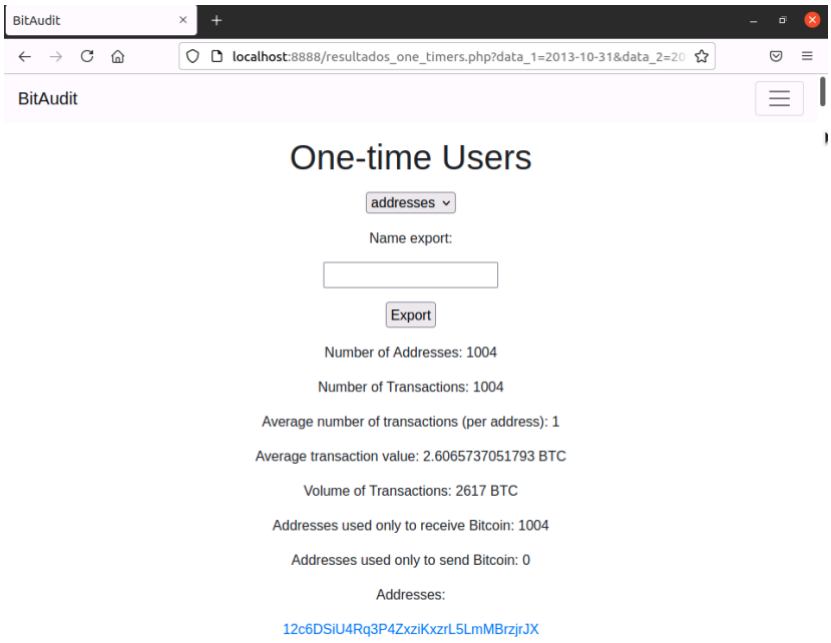


*Figura 44: Exemplo de "one-time users" no dia 31 de outubro de 2013*

Quanto ao código, este vai percorrer as transações do período temporal, no qual vão ser identificados os endereços que realizaram uma única transação entre as datas indicadas pelo utilizador (Figura 45).

```
        for bl in data:
                if(compara_datas(bl.time, data_2, data_3)):
                        Flag = True

                        for i in range(len(bl.txes.to_list())):
                                inputs = bl.txes.to_list()[i].inputs.to_list() + data[i].outputs.to_list()

                                for j in range(len(inputs)):
                                        if((type(inputs[j].address) is not blocksci.NonStandardAddress) and (type(inputs[j].address) is not blocksci.OpReturn)
and (type(inputs[j].address) is not blocksci.MultisigAddress)):
                                                if(inputs[j].address.address_string not in aux):
                                                        if(inputs[j].address.input_txes_count() + inputs[j].address.output_txes_count() == 1):
                                                                one_timers += 1
                                                                valor_tx += data[i].input_value
                                                                total_tx += 1
                                                                aux.append(inputs[j].address.address_string)
                                                                addresses.append(inputs[j].address.address_string)

                                                                if(inputs[j].address.input_txes_count() == 1):
                                                                        one_timers_env += 1

                                                                else:
                                                                        one_timers_rec += 1

                                                        else:
                                                                aux.append(inputs[j].address.address_string)

                else:
                        if(Flag):
                                break

                        else:
                                continue
```

*Figura 45: Excerto de código relativo à procura dos endereços com apenas uma transação*

- Estudo da **atividade de endereço(s)**, em que o *software* vai retornar os detalhes relativos à atividade do endereço ou ficheiro com endereços, fornecidos pelo utilizador. Mostra certos dados, como mês com mais transações, número e volume de transações (Figura 46). Para o fazer, apenas foram analisadas as transações do endereço ou do ficheiro e calculados os valores necessários para mostrar ao utilizador.

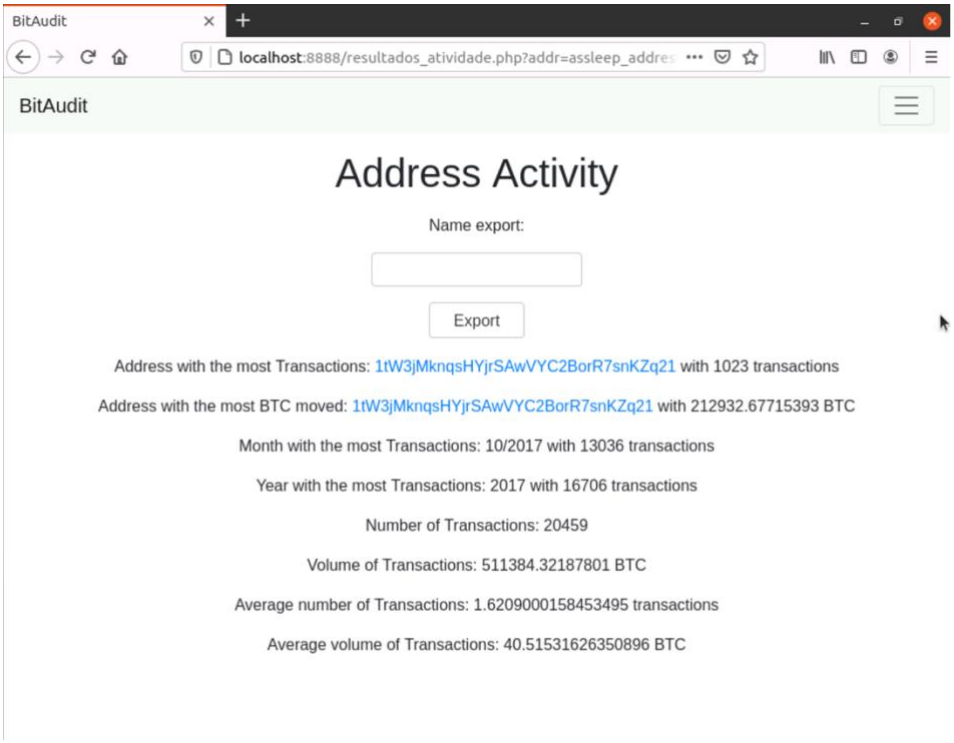


*Figura 46: Exemplo de análise de atividade de um conjunto de endereços*

- **"Wash Trade"**, identifica situações em que uma entidade (ou utilizador) envia *bitcoins* de um endereço (fornecido como *input* pelo utilizador) para outro que também controla [24]. Isto é pertinente pois utilizadores "legais" têm poucos motivos para realizar tais ações para ocultar os seus movimentos (e incorrer nos custos associados). Em contraste, os utilizadores envolvidos em atividades ilegais provavelmente utilizam este tipo de técnicas de ocultação. Como tal, o uso de serviços como este poderá ser indicador que um utilizador está envolvido em atividades ilegais [24]. O programa mostra a entidade em que o endereço se insere e se foram identificadas situações de *wash trading.* Relativamente ao código, o programa reúne os endereços que pertencem à mesma entidade ao utilizar ***"Address -> Entity"***. De seguida, verifica se existe algum caso de *wash trading*, ao seguir o que foi descrito anteriormente, que consiste em perceber se alguns dos endereços pertencente a uma entidade enviou *bitcoins* a outro igualmente pertencente (Figura 47).

```
resultados = address_entity(addr)

for addr in resultados[0]:
        # output_txes
        aux = chain.address_from_string(addr)
        #print(addr)
        for tx in aux.input_txes.to_list():
                for out in tx.outputs.to_list():
                        if((type(out.address) is not blocksci.NonStandardAddress) and (type(out.address) is not blocksci.OpReturn) and (type(out.address) is not blocksci.MultisigAddress)):
                                if (out.address.address_string in resultados[0]):
                                        wash_trade.append([str(tx.hash), addr, out.address.address_string])
```

*Figura 47: Exemplo da identificação da entidade do endereço e posterior procura de casos de wash trading*

- **Exportar**, todas as funcionalidades descritas anteriormente (exceto a de listagem de blocos) têm a possibilidade de exportar os resultados obtidos, tanto a nível de endereços, transações, estatísticas ou tudo o que a ferramenta retornou. Estas opções variam entre todas as funcionalidades (Figuras 48, 49 e 50).


*Figura 48: Exemplo em que apenas se pode exportar os resultados que aparecem ao utilizador*


*Figura 49: Exemplo em que se pode exportar tudo ou apenas as estatísticas ou endereços*


*Figura 50: Exemplo em que se pode exportar tudo ou apenas as estatísticas, endereços ou transações*

## 6.4 Funcionalidades Não Implementadas

De seguida são apresentadas as funcionalidades não implementadas e as respetivas justificações:

- "Utilizador avançado", devido a limitações de tempo e não ter sido definida como *must have* nos requisitos;
- "Informações gerais", que apenas foi alterado o nome da funcionalidade para "*Blockchain Analysis*", tendo sido considerado mais adequado ao que realmente era feito;
- "Análise de circularidade*"*, foi considerada uma funcionalidade de alguma complexidade tanto em termos de código como de eficiência e tempo de execução. Foi substituída por "*Wash Trade"* que, como descrito anteriormente, identifica situações em que um utilizador envia *bitcoins* de um endereço para outro que também controla, consideravelmente semelhante à análise de circularidade que se pretendia desenvolver.

# 7. Testes

O teste pode mostrar apenas a presença de falhas, não a sua ausência, sendo esta uma limitação teórica fundamental. É quase impossível encontrar todas as falhas num programa, por outras palavras, quem realiza os testes costuma chamar um teste de sucesso (ou efetivo) se encontrar um erro [87].

Trata-se, assim, de um método utilizado para verificar se o produto de *software* atende aos requisitos expectáveis e para garantir que esteja livre de defeitos. Envolve a execução de

componentes do sistema com o uso de ferramentas manuais ou automatizadas para avaliar uma ou mais propriedades de interesse. O objetivo do teste é identificar erros, lacunas ou requisitos ausentes em contraste com os requisitos reais [88].

Existem benefícios na realização dos testes, dos quais se destacam:

- ***Cost-Effective***: é uma das vantagens mais importantes. Testar qualquer projeto de IT atempadamente ajuda a economizar dinheiro a longo prazo. Quando os *bugs* são detetados nos testes, é mais fácil corrigir [88];
- **Segurança**: é o benefício mais vulnerável e sensível, que ajuda a remover riscos e problemas precocemente. Os clientes procuram produtos confiáveis [88];
- **Qualidade do produto**: é um requisito essencial de qualquer produto de *software*. Os testes garantem que um produto de qualidade seja entregue aos clientes [88];
- **Satisfação do cliente**: o principal objetivo de qualquer produto é a satisfação do cliente [88].

Segundo o planeamento e a metodologia de desenvolvimento escolhida (*Waterfall*), os testes teriam início após a fase de implementação. No entanto, foram realizados testes unitários após a finalização de cada fase, de modo a verificar que todas as funcionalidades estavam bem implementadas e de acordo com os requisitos previamente definidos. As pequenas falhas encontradas foram prontamente corrigidas. Para além disso, e ainda durante o desenvolvimento, foram realizados os testes de integração. Após a implementação foram postos em prática os testes de sistema.

Nesta fase, foram ainda realizados testes não funcionais de modo a validar os requisitos não funcionais.

Todos os resultados esperados para os testes foram retirados de *softwares* como *Blockchain Explorer* e *Wallet Explorer*, que são dos mais utilizados pela comunidade.

## 7.1 Testes Funcionais

Os testes funcionais verificam o que o sistema realmente faz, ou seja, se cada funcionalidade do *software* se comporta conforme especificado no documento de requisitos. Baseia-se em testar todas as funcionalidades, ao fornecer entrada apropriada para verificar se a saída real corresponde à saída esperada ou não [89].

### 7.1.1 Testes Unitários

Os testes unitários são realizados para verificar se os módulos individuais do código-fonte estão a funcionar corretamente, ou seja, testar cada unidade do *software* separadamente pelo *developer* no ambiente de desenvolvimento [89]. Também podem ser chamados de *Module* ou *Component Tests* [89].

Após a implementação de cada funcionalidade, foram realizados testes de modo a verificar o seu correto funcionamento. Para estes testes foram definidos dados de entrada e o que era pretendido que o programa retornasse. Quando os resultados não correspondiam ao desejado, eram efetuadas modificações ao código de modo a corrigir a situação. Para agilizar os processos, os testes, bem como o desenvolvimento da ferramenta, foram realizados com parte da *blockchain* (de 2009 a 2014).

### 7.1.2 Testes de Integração

Os testes de integração consistem no processo de testar a conectividade ou transferência de dados entre alguns módulos na unidade. É também conhecido como *I&T Testing* ou *String Testing* e é subdividido em abordagem *Top-Down*, *Bottom-Up* e mista (combinação de abordagens ascendente e descendente) [89]. É tipicamente feito em *Black-Box Testing,* que é um método no qual os testes são efetuados sem a necessidade de ter conhecimento da estrutura do código, detalhes de implementação e caminhos internos [90]. Este tipo de teste concentra-se principalmente nos valores de entrada e de saída e é inteiramente baseado em requisitos e especificações de *software* [90].

No nosso caso, após a implementação de certas funcionalidades e devidamente testadas individualmente, foram realizados testes em conjunto. Tal como para os testes unitários, foi definido um conjunto de valores de entrada e o que era pretendido que o programa retornasse. Quando não acontecia o previsto, eram efetuadas alterações de modo a corrigir a situação.

Este tipo de testes ocorreu especificamente da interação nas situações que se seguem:

- Listagem de endereços com saldo superior a 1000 *bitcoins* para ser utilizada no cálculo de certos valores pertinentes para serem mostrados ao utilizador. Isto é feito na funcionalidade “*Whales”*;
- Obtenção de endereços pertencentes à mesma entidade do endereço fornecido pelo utilizador para que seja possivel verificar se houve alguma situação de *wash trading*;
- Exportação dos resultados de todas as funcionalidades (exceto listagem de blocos);

- Após a exportação de endereços, o ficheiro resultante pode ser utilizado para ser estudada a atividade desses mesmos endereços, na funcionalidade "*Address Activity*".

Para agilizar estes processos, estes testes, tal como os anteriores, foram realizados com *blockchain* parcial.

### 7.1.3 Testes de Sistema

Os testes de sistema são também incluídos em *Black-Box Testing* e servem para garantir que o *software* funciona em todos os sistemas de utilização final [89]. Para simulação, quem desenvolve "veste a pele" de utilizador final.

No capítulo seguinte são documentadas as análises realizadas com a utilização do *software*. Como este tipo de testes pretende simular o modo de operação de um utilizador final, as análises serviram também o propósito dos testes de sistema.

A realização destes testes permitiu a validação do correto funcionamento do sistema como um todo. Quando foram encontradas falhas nos casos de estudo definidos, esse teste foi marcado como falhado. A tabela seguinte mostra exatamente isso: o ID de teste, a funcionalidade, os dados de entrada, os resultados esperados e obtidos.

Ao contrário dos testes anteriormente realizados, para estes testes e para as análises documentadas no capítulo seguinte, a versão da *blockchain* utilizada é mais atualizada, ou seja, até 2020. Face ao exposto, existem algumas funcionalidades que percorrem toda a *blockchain* para realizar a análise correspondente e demoram vários dias até ficarem completas, por isso, o reunir endereços de uma entidade, encontrar endereços de troco ou identificar situações de *wash trading,* não foi possivel documentar com este tipo de testes, devido a restrições temporais.

| ID | Funcionalidade | Dados de Entrada | Resultados Esperados | Resultados Obtidos |
|---|---|---|---|---|
| S01 | Pesquisa | Endereço (15kDcr9wDhZmvPufKP7vYyELSoLg4bKbAR) | Dados relativos ao endereço (N.º de transações = 45, saldo = 0,00673607) | N.º de transações = 45, saldo = 0,00673607 (teste bem sucedido) |
| S02 | Pesquisa | Endereço inválido (15kDcr9wDhZmvPufKP7vYyELS) | Mensagem de erro | Mensagem de erro (teste bem sucedido) |
| S03 | Pesquisa | *Hash* (0eaad8bd8812697f7b59d0a0a5ed89702279b7d33d949fef759c738e5fb13419) | Dados relativos à transação (Data e hora de confirmação = 2011- | Data e hora de confirmação = 2011-12-22 06:37, |

| | | | | |
|---|---|---|---|---|
| | | | 12-22 06:37, Total *input* = 0,55673607 BTC, Total *output* = 0,55673607 BTC, taxa = 0,0 BTC e bloco = 158.555) | Total *input* = 0,55673607 BTC, Total *output* = 0,55673607 BTC, taxa = 0,0 BTC e bloco = 158.555 (teste bem sucedido) |
| S04 | Pesquisa | *Hash* inválido (0eaad8bd8812697f7b59d0a0a5ed8970227) | Mensagem de erro | Mensagem de erro (teste bem sucedido) |
| S05 | Pesquisa | *Height* (100) | Dados relativos ao bloco (N.º de transações = 1, Nonce = 1573057331, recompensa = 50 BTC, hora de confirmação = 2009-01-11 08:00) | N.º de transações = 1, Nonce = 1573057331, recompensa = 50 BTC, data e hora de confirmação = 2009-01-11 08:00 (teste bem sucedido) |
| S06 | Pesquisa | *Height* inválida (100000) | Mensagem de erro | Mensagem de erro (teste bem sucedido) |
| L01 | Listagem (endereços) | Saldo >= 100 BTC<br>Nº transações >= 0 | Listagem de endereços | Listados 1.420.388 endereços (teste bem sucedido) |
| L02 | Listagem (endereços) | Saldo >= 10000000 BTC | Mensagem de erro | Mensagem de erro (teste bem sucedido) |
| L03 | Listagem (transações) | Valor >= 100 BTC<br>Até junho de 2013 | Listagem de transações | Mais de 1.000.000 de transações (teste bem sucedido) |
| L04 | Listagem (transações) | Taxa <= 0 BTC<br>A partir de janeiro de 2017 | Nenhuma transação | Nenhuma transação (teste bem sucedido) |
| L05 | Listagem (transações) | Data inicial mais recente que a final (sem outros parâmetros) | Mensagem de erro | Mensagem de erro (teste bem sucedido) |
| L06 | Listagem (blocos) | Recompensa do *miner* <= 25 BTC<br>Até janeiro de 2014 | Listagem de blocos | 240.228 blocos (teste bem sucedido) |

| | | | | |
|---|---|---|---|---|
| L06 | Listagem (blocos) | Recompensa do *miner* >= 50 BTC Ano de 2019 | Nenhum bloco | Nenhum bloco (teste bem sucedido) |
| L08 | Listagem (blocos) | Data inicial mais recente que a final (sem outros parâmetros) | Mensagem de erro | Mensagem de erro (teste bem sucedido) |
| C01 | *Cluster* | Endereço (18iEz617DoDp8CNQUyyrjCcC7XCGDf5SVb) | Endereços, transações e outras informações sobre o estudo de *cluster* realizado | N.º de endereços = 968, N.º de transações = 6.828, Saldo do *cluster* = 948,3 BTC, Média de transações por endereço = 5,5, Valor médio de transação = 20,4 BTC e Volume de transações = 139.276 BTC (e respetivos endereços e transações) (teste bem sucedido) |
| C02 | *Cluster* | Endereço inexistente (15kDcr9wDhZmvPufKP7vYyELS) | Mensagem de erro | Mensagem de erro (teste bem sucedido) |
| AE01 | *Address -> Entity* | - | - | - |
| AE02 | *Address -> Entity* | Endereço inexistente (15kDcr9wDhZmvPufKP7vYyELS) | Mensagem de erro | Mensagem de erro (teste bem sucedido) |
| SG01 | *Shadow Guessing* | - | -- | - |
| SG02 | *Shadow Guessing* | Endereço inexistente (15kDcr9wDhZmvPufKP7vYyELS) | Mensagem de erro | Mensagem de erro (teste bem sucedido) |
| BA01 | Análise da *Blockchain* | Dia 1 de outubro de 2013 | Dados da *blockchain* relativos ao período temporal escolhido | Valor da menor transação = 0 BTC, valor da maior transação = 22.500 BTC, Média do valor de transação = 10,8 BTC, N.º de transações = 32.709, Volume de transações = |

| | | | | |
|---|---|---|---|---|
| | | | | 352.633 BTC, N.º de *inputs* = 2.887, N.º de *outputs* = 32781, Valor da menor taxa = 0 BTC, valor da maior taxa = 1 BTC, valor médio da taxa = 8,7 * 10^-5 BTC |
| BA02 | Análise da *Blockchain* | Datas incompatíveis | Mensagem de erro | Mensagem de erro (teste bem sucedido) |
| DA01 | Endereços inativos | 11 anos inativos | Listagem de endereços e dados associados | Listados 49259 endereços, N.º de transações = 10.261, Valor médio de transação = 306,6 BTC e Volume de transações = 3.146.323 BTC (teste bem sucedido) |
| DA02 | Endereços inativos | 15 anos inativos (impossível) | Mensagem de erro | Mensagem de erro (teste bem sucedido) |
| MU01 | Endereços mais ativos | Dia 1 de outubro de 2013 | Endereços mais ativos relativos ao período temporal escolhido | Figura 67 (teste bem sucedido) |
| MU02 | Endereços mais ativos | Datas incompatíveis | Mensagem de erro | Mensagem de erro (teste bem sucedido) |
| OU01 | Endereços utilizados apenas uma vez | Dia 1 de outubro de 2013 | Endereços utilizados apenas uma vez (e essa utilização ocorreu neste período temporal) | 1.004 endereços encontrados nesse período |
| OU02 | Endereços utilizados apenas uma vez | Datas incompatíveis | Mensagem de erro | Mensagem de erro (teste bem sucedido) |
| AA01 | Análise de Atividade (um ou mais endereços) | Conjunto de endereços | Dados da atividade dos endereços fornecidos como *input* | Endereço com maior volume de transações e N.º de transações é o |

| | | | | |
|---|---|---|---|---|
| | | | | mesmo, outubro de 2017 foi mês com maior atividade e 2017 o ano, N.º de transações = 20.459, volume de transações = 511.384 BTC, N.º médio de transações por endereço = 1,6 e o volume médio de transações por endereço = 40,5 BTC (teste bem sucedido) |
| AA02 | Análise de Atividade (um ou mais endereços) | Endereço inexistente (15kDcr9wDhZmvPufKP7vYyELS) | Mensagem de erro | Mensagem de erro (teste bem sucedido) |
| WT01 | *Wash Trade* | - | - | - |
| WT02 | *Wash Trade* | Endereço inexistente (15kDcr9wDhZmvPufKP7vYyELS) | Mensagem de erro | Mensagem de erro (teste bem sucedido) |

*Tabela 7: Resultados dos testes de sistema realizados*

## 7.2 Testes Não-Funcionais

Este tipo de testes avalia se o sistema está de acordo com os requisitos não-funcionais definidos. O teste não-funcional pode estar relacionado com vários aspetos do *software*, como *performance*, carga, *stress*, escalabilidade, segurança e compatibilidade [89]. O foco principal é garantir a melhor experiência para o utilizador em todos estes aspetos. No nosso caso, foram apenas realizados testes de *performance* para verificar se vão ao encontro dos requisitos estipulados.

### 7.2.1 Testes de *Performance*

Os testes de *performance* constituem um método utilizado para testar a velocidade, tempo de resposta, estabilidade, confiabilidade, escalabilidade e uso de recursos de um *software* sob determinado volume de trabalho. O principal objetivo deste tipo de teste é identificar e eliminar os *bottlenecks* de desempenho no sistema [91]. Apenas foram testadas

análises mais simples para este efeito, sendo que um dos requisitos era que as análises mais simples demorariam até 30 segundos (DES02) (como pesquisa e estudo de *cluster*, identificados na tabela com um *) e que o sistema iniciaria em menos de cinco segundos. Também estavam planeados testes a análises mais complexas, mas, por limitações temporais, foram realizados outros tipos de testes. A Tabela 8 apresenta os testes e os respetivos resultados. Contém o ID do teste, os dados de entrada e tempo despendido para a apresentação de resultados. Aqui podemos verificar que o requisito DES02, relativo ao desempenho, se verifica, enquanto o DES01 (do início do sistema) é ultrapassado por mais de um segundo, que não parece algo preocupante. Nos restantes, é possivel observar que apenas a listagem de endereços demora mais tempo.

| ID | Funcionalidade | Dados de Entrada | Tempo |
|---|---|---|---|
| TP1 | Pesquisa * | Endereço (15kDcr9wDhZmvPufKP7vYyELSoLg4 bKbAR) | 00 min 00,68 seg |
| TP2 | Pesquisa * | *Hash* (0eaad8bd8812697f7b59d0a0a5ed8 9702279b7d33d949fef759c738e5fb13419) | 00 min 00,52 seg |
| TP3 | Pesquisa * | *Height* (15855) | 00 min 02,15 seg |
| TP4 | Listagem (transações) | Valor da transação >= 100 BTC | 03 min 49,59 seg |
| TP5 | Listagem (blocos) | Recompensa do *miner* = 50 BTC | 01 min 58,05 seg |
| TP5 | Listagem (endereços) | Saldo >= 100 BTC<br>Nº de Transações >= 0 | 2.161 min 03,00 seg |
| TP7 | Estudo de *Cluster* * | Endereço (15kDcr9wDhZmvPufKP7vYyELSoLg4 bKbAR) | 00 min 00,90 seg |
| TP8 | *One-timers* | De 01/01/2010 a 01/06/2010 | 00 min 47,62 seg |
| TP9 | Análise da *blockchain* | De 01/01/2010 a 01/06/2010 | 00 min 07,20 seg |
| TP10 | Análise da *blockchain* | De 01/01/2019 a 02/01/2019 | 00 min 07,23 seg |
| TP11 | Análise da *blockchain* | De 01/01/2017 a 01/02/2017 | 01 min 10,16 seg |
| TP12 | Atividade endereço (s) * | Endereço (15kDcr9wDhZmvPufKP7vYyELSoLg4 bKbAR) | 00 min 01,62 seg |
| TP13 | Atividade endereço (s) * | 960 endereços | 00 min 04,30 seg |
| TP14 | Inicio do sistema | - | 00 min 06,21 seg |

*Tabela 8: Resultados dos testes de performance realizados*

# 8. Análises com a Ferramenta

Neste capítulo são documentadas as análises efetuadas com o propósito de estudar casos de interesse com o *software* desenvolvido no decorrer do presente estágio. Estas foram

realizadas numa *blockchain* mais atualizada (comparativamente ao desenvolvimento e testes iniciais), até 2020, inclusive.

O primeiro caso analisado é referente ao início da negociação de contratos futuros da Bitcoin na CBOE (*Chicago Board Options Exchange*, um dos maiores mercados de opções do mundo com contratos focados em ações individuais, índices e taxas de juros [92]) a 10 de dezembro de 2017. “Um contrato de futuros é um acordo legal para comprar ou vender um determinado produto ou ativo a um preço predeterminado num momento específico no futuro. O comprador de um contrato futuro está a assumir a obrigação de comprar o ativo subjacente quando o contrato futuro expirar. O vendedor do contrato de futuros está a assumir a obrigação de fornecer o ativo subjacente na data de vencimento.” [93]. Este anúncio despoletou elevadas expectativas de uma aceitação e utilização generalizada desta criptomoeda [94]. Começou por se fazer uma análise à *blockchain* em três períodos diferentes para proceder a comparações, nos cinco dias anteriores (Figura 51), no dia do início das negociações de futuros (10 de dezembro de 2017) (Figura 52) e nos 5 dias seguintes (Figura 53). Após estas análises, foi feito o estudo dos endereços mais ativos nestes mesmos períodos (Figuras 54, 55 e 56).

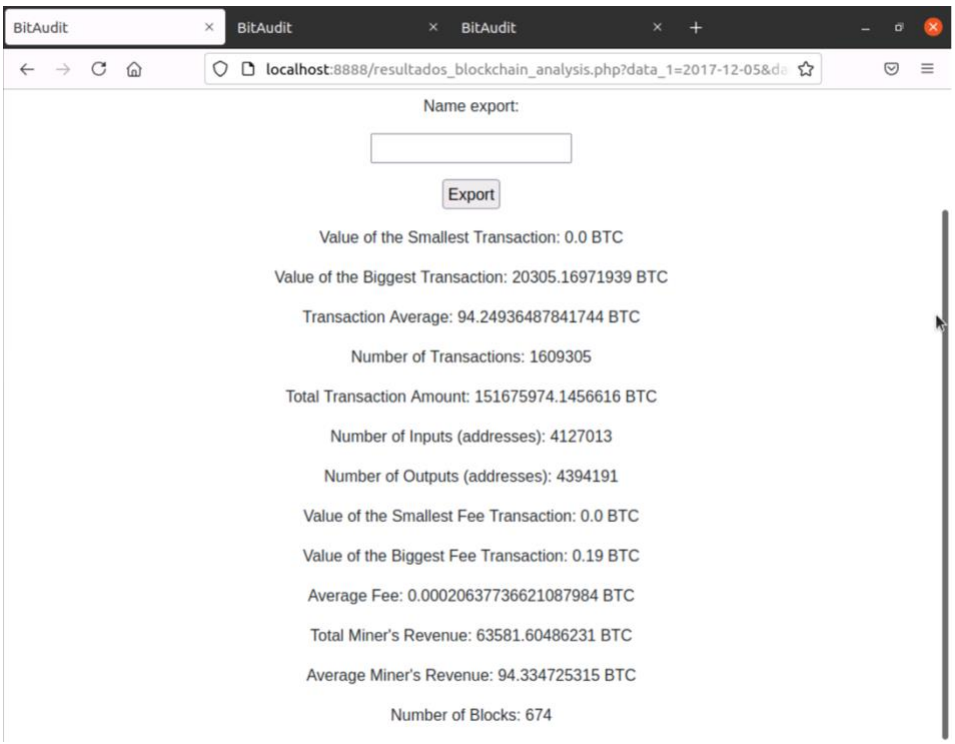


*Figura 51: Análise da blockchain nos cinco dias anteriores a 10 de dezembro de 2017*

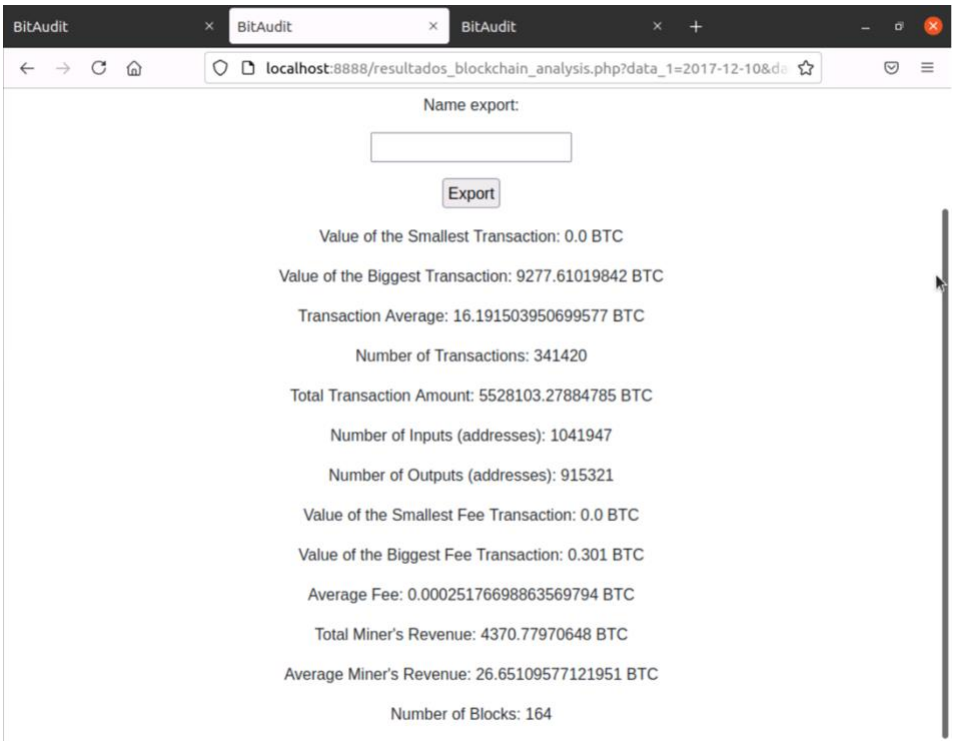


*Figura 52: Análise da blockchain no dia 10 de dezembro de 2017*

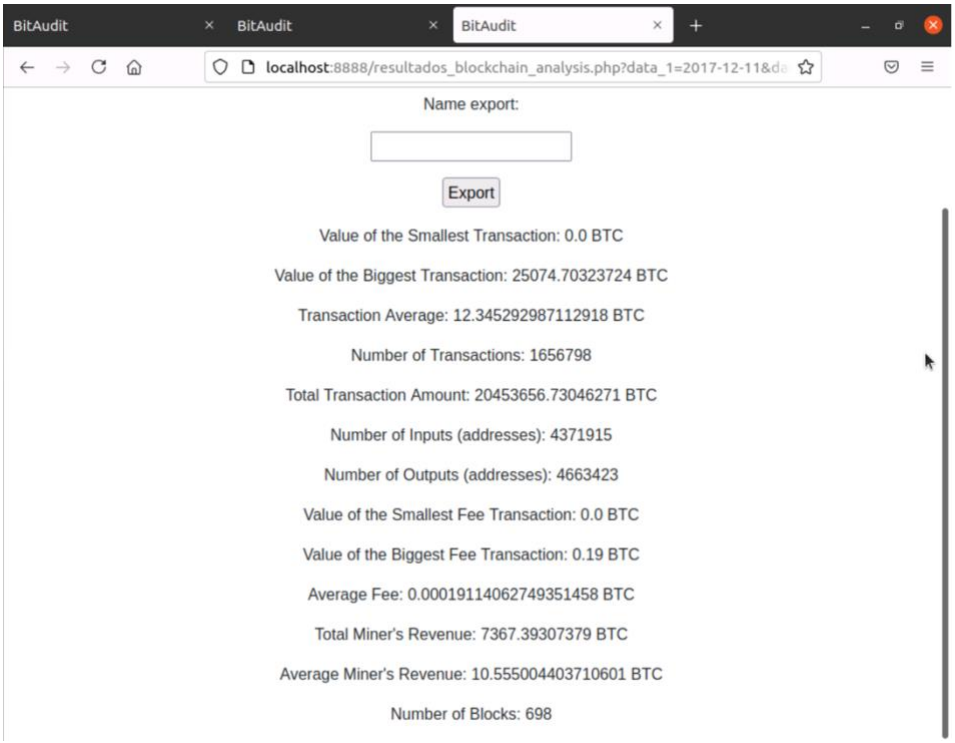


*Figura 53: Análise da blockchain nos cinco dias após 10 de dezembro de 2017*

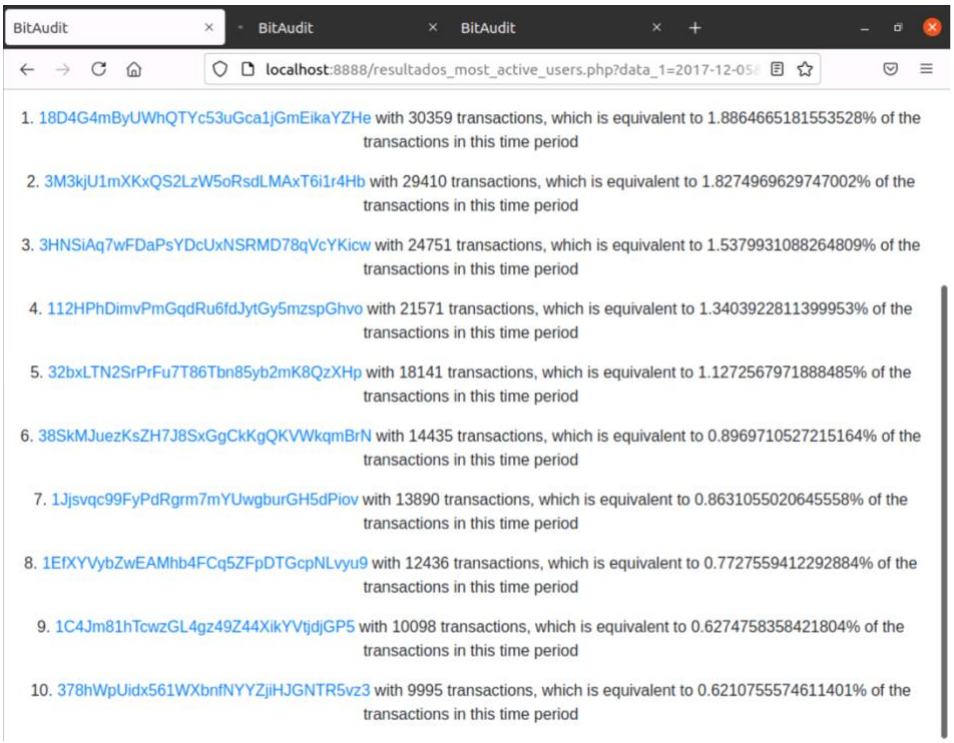


*Figura 54: Endereços mais ativos nos cinco dias anteriores a 10 de dezembro de 2017*

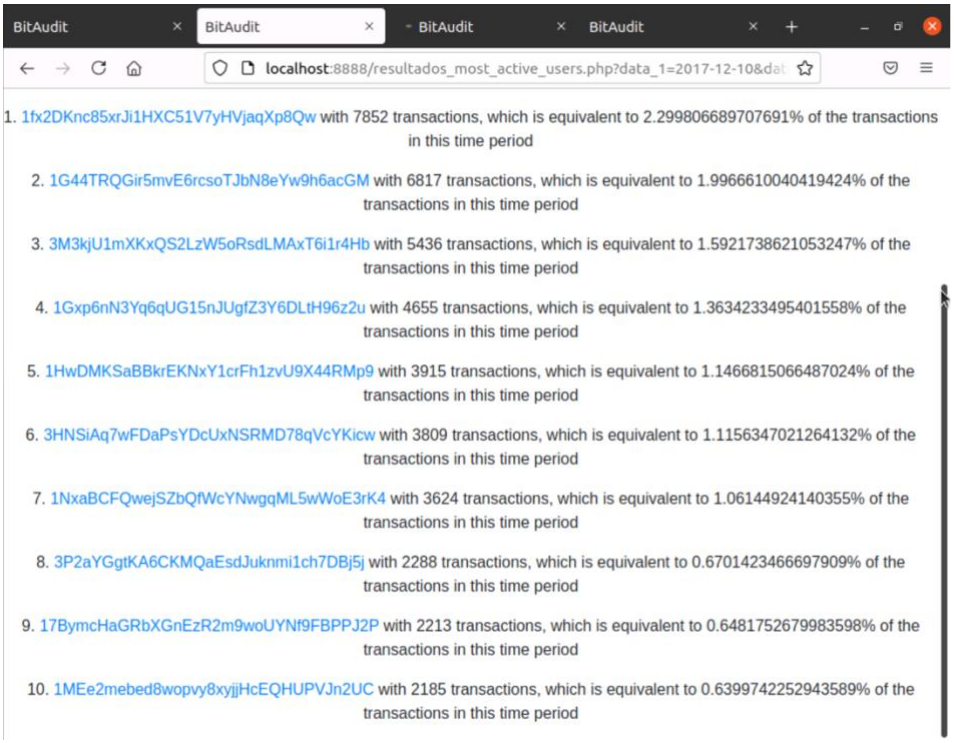


*Figura 55: Endereços mais ativos no dia 10 de dezembro de 2017*

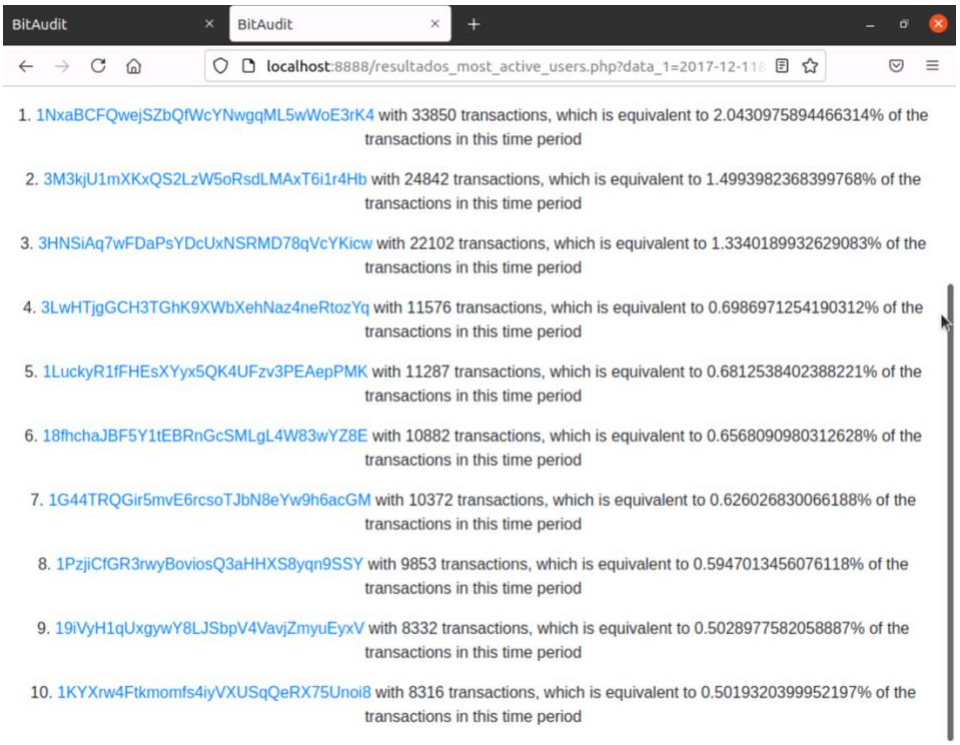


*Figura 56: Endereços mais ativos nos cinco dias após 10 de dezembro de 2017*

Face ao exposto, é possível perceber que houve um aumento do número de transações nos cinco dias após o dia 10 de dezembro, relativamente aos anteriores. Apesar disso, houve uma diminuição na média de valor transacionado. Isto pode estar diretamente relacionado com o aumento de preço (nas *exchanges*) de quase 40% durante esse período [32]. Relativamente aos endereços mais ativos, consegue-se verificar que, apenas um endereço se repete ao longo dos três períodos. Isto mostra uma variação nos endereços mais ativos ao longo deste tempo.

O segundo refere-se à maior queda, em termos percentuais, registada na rede Bitcoin até à data de escrita deste relatório [50], já abordada no capítulo três (Explorações Preliminares na Rede Bitcoin). Ocorreu no início de março e, num período de 24 horas, foi observada uma queda de 37% [50]. Primeiro, foi realizada uma análise à *blockchain* em três períodos diferentes para proceder a comparações, nos cinco dias anteriores à queda (Figura 57), no dia da queda (12 de março de 2020) (Figura 58) e nos 5 dias seguintes (Figura 59). À semelhança da análise anterior, de seguida, foi feito o estudo dos endereços mais ativos nestes mesmos períodos (Figuras 60, 62 e 63).

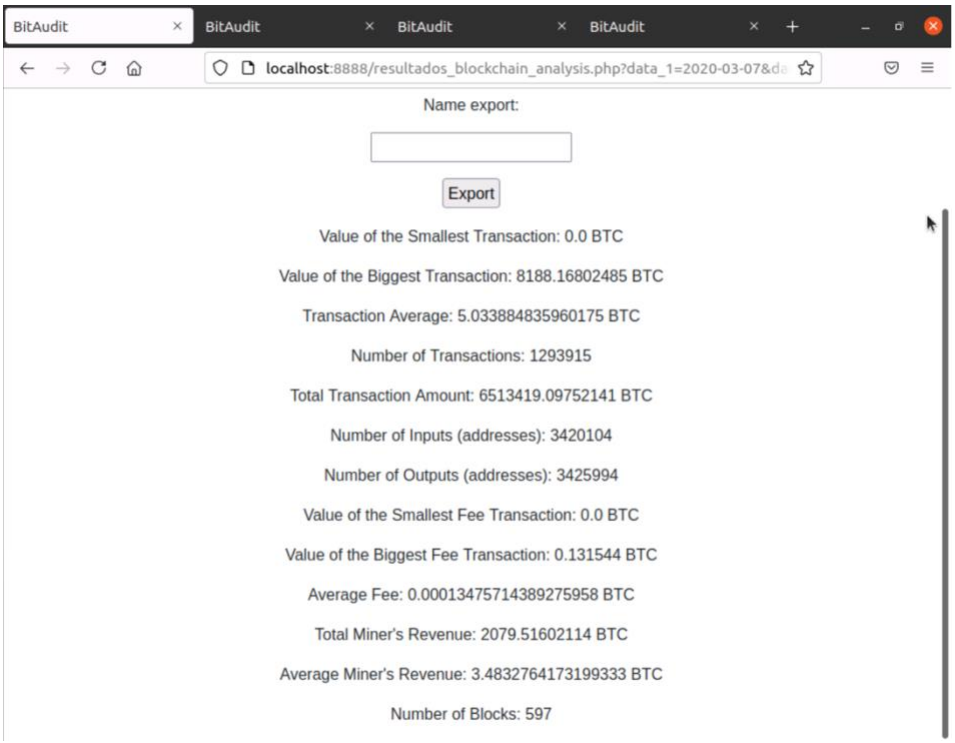


*Figura 57: Análise da blockchain nos cinco dias anteriores a 11 de março de 2020*

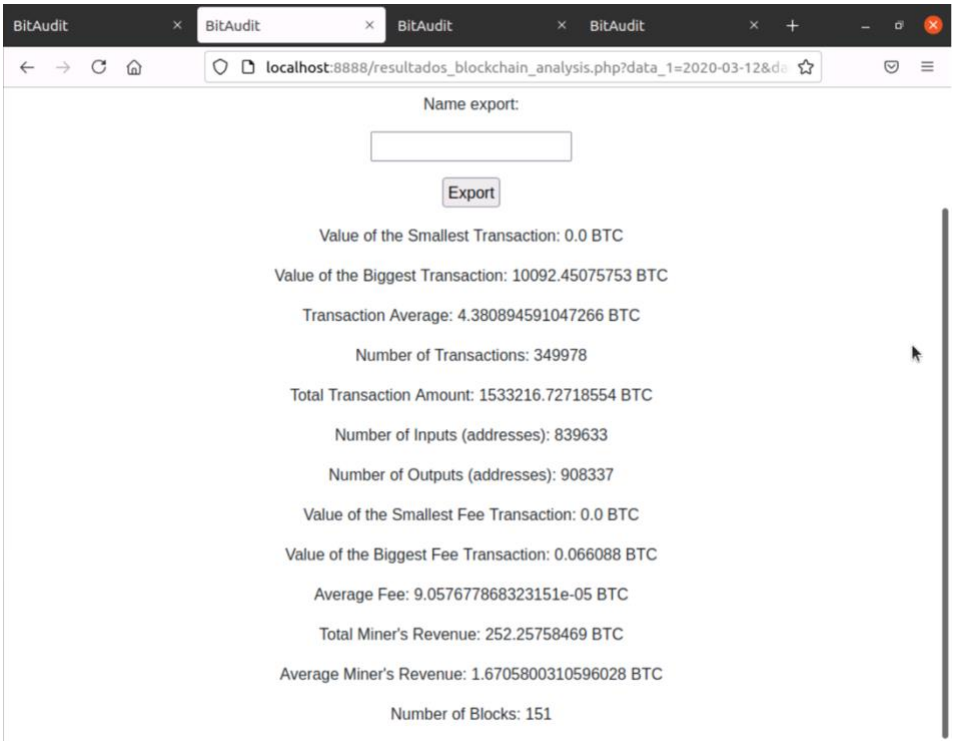


*Figura 58: Análise da blockchain no dia 11 de março de 2020*

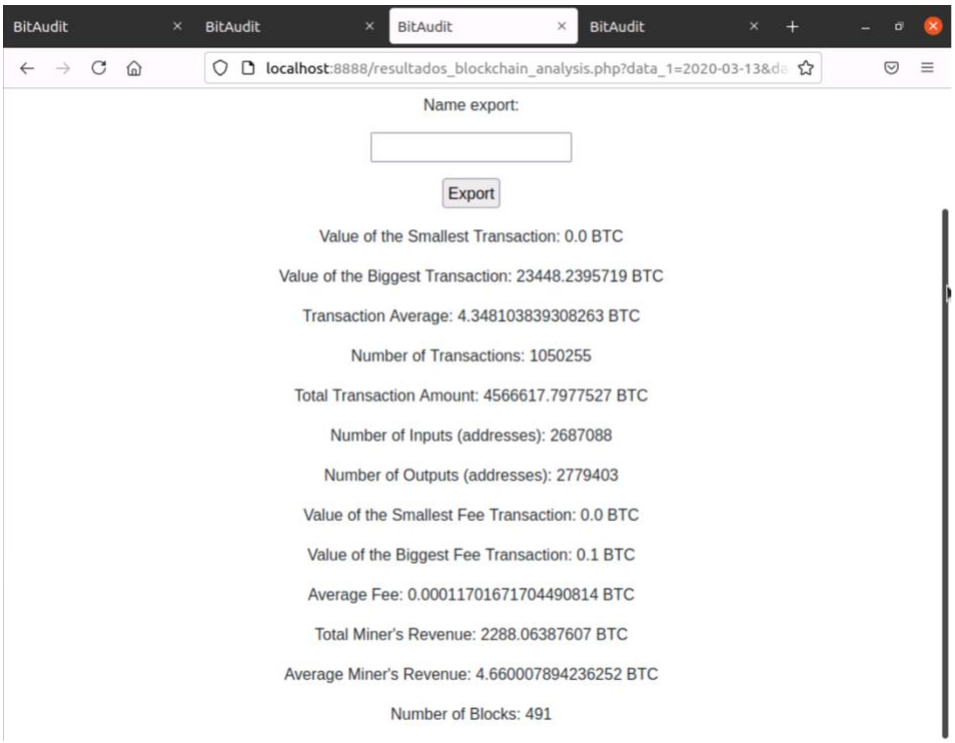


*Figura 59: Análise da blockchain nos cinco dias após 11 de março de 2020*

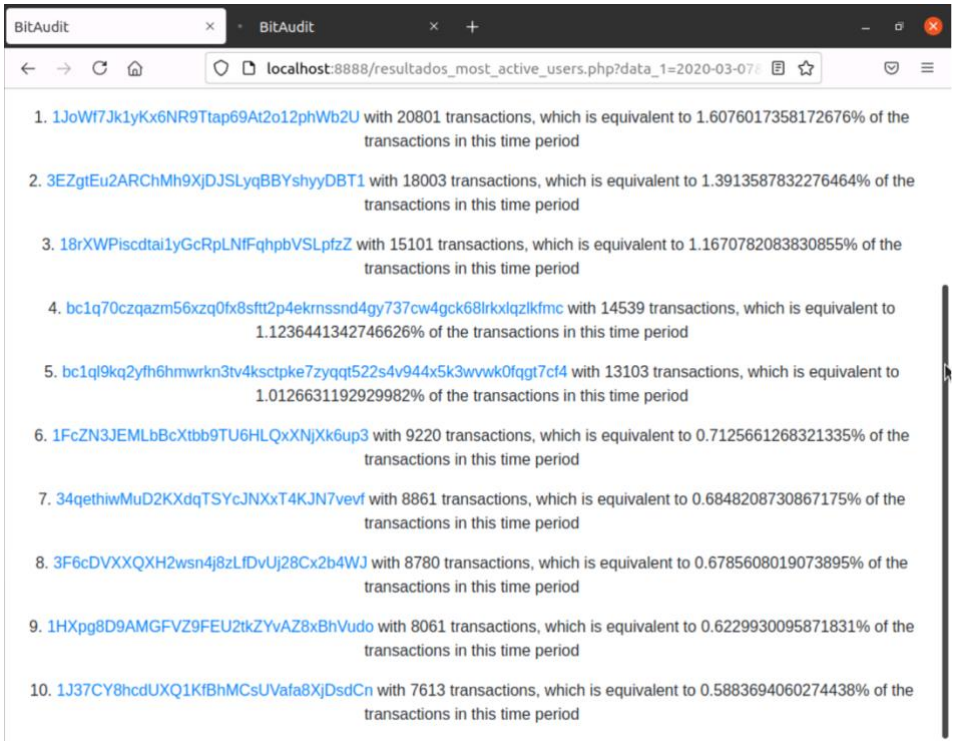


*Figura 60: Endereços mais ativos nos cinco dias anteriores a 11 de março de 2020*

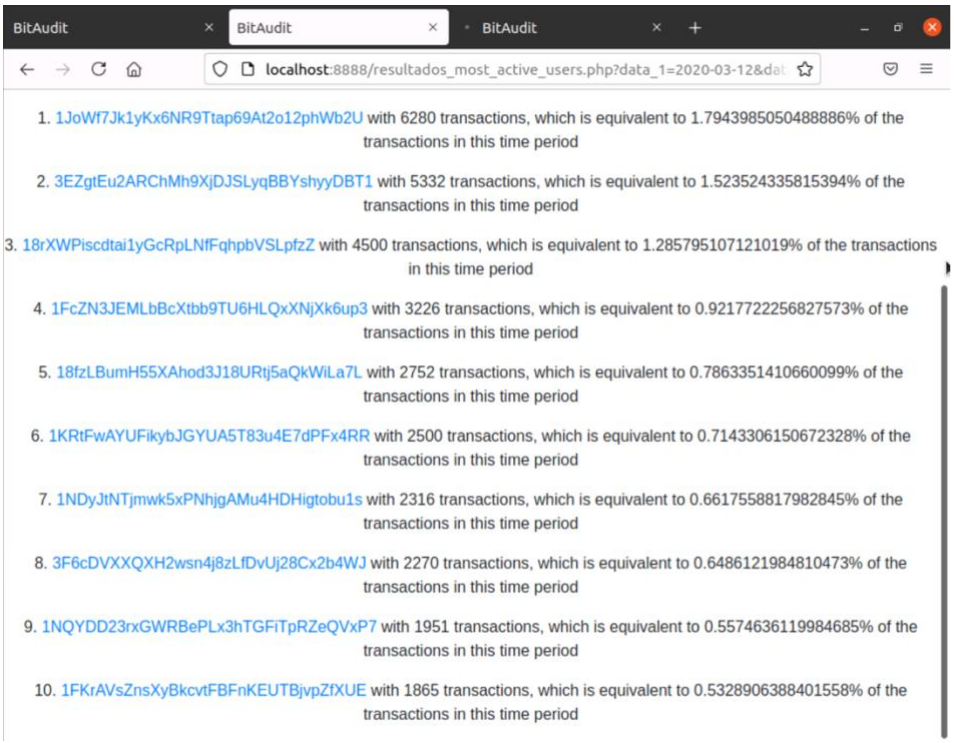


*Figura 61: Endereços mais ativos no dia 11 de março de 2020*

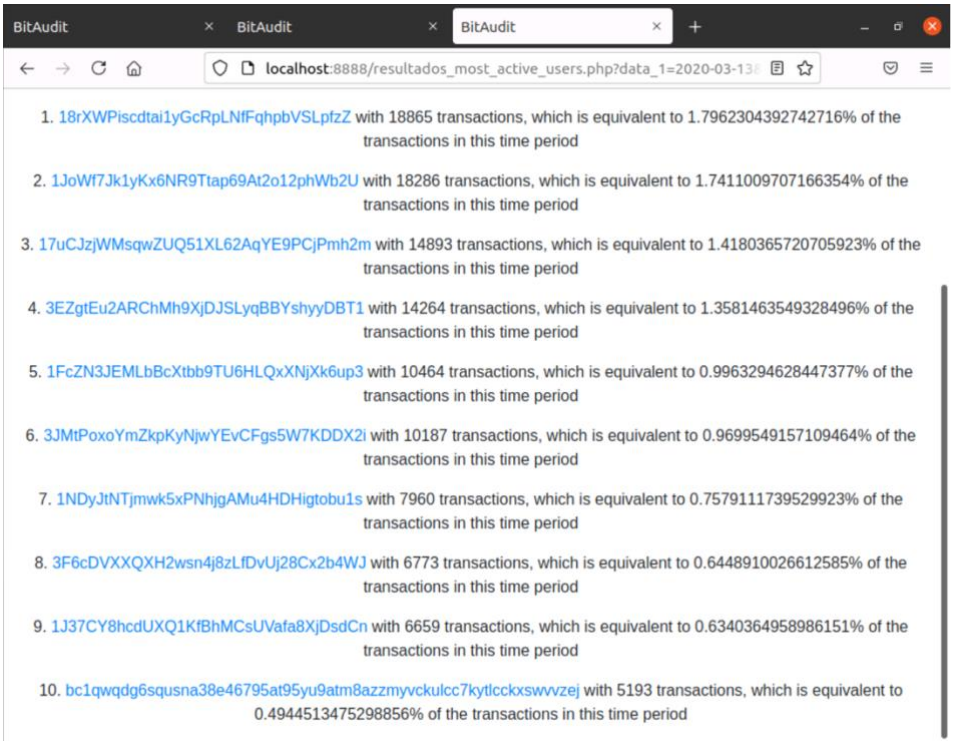


*Figura 62: Endereços mais ativos nos cinco dias após 11 de março de 2020*

A partir dos resultados obtidos, é possivel perceber que houve uma descida em termos de valor médio transferido, número e volume (em BTC) de transações. Tendo sido verificada uma das maiores quedas de valor, em termos percentuais, da história da Bitcoin, poderia ter ocorrido um aumento dos valores anteriormente mencionados, mas tal não aconteceu. O que pode significar que, na altura, os seus utilizadores começaram a perder "confiança" na moeda ou que apenas tentaram evitar vender de forma impulsiva, com o objetivo de recuperar algum investimento. A partir do estudo feito aos endereços mais ativos, ao contrário da análise anterior, consegue-se identificar alguns que se repetem ao longo de todo o período temporal especificado. A partir destes endereços poderia realizar-se outras análises como "*Address->Entity*", "*Wash Trade*" ou apenas exportar e analisar a sua atividade.

O terceiro caso analisado é relativo ao fecho do mercado negro *Silk Road*, mais conhecido por vender produtos ilegais com o uso de *bitcoins*, a 1 de outubro de 2013. Este tipo de eventos leva a alterações na forma como certos utilizadores se comportam, seja para movimentar o mais rapidamente moedas com ligação ao mercado negro, seja para diminuir a atividade e não serem detetados [24], o que pode ser interessante investigar. Inicialmente, foi realizada uma análise à *blockchain* em três períodos diferentes para proceder a comparações, nos cinco dias anteriores (Figura 62), no dia do acontecimento (1 de outubro de 2013) (Figura 63) e nos cinco dias seguintes (Figura 64).

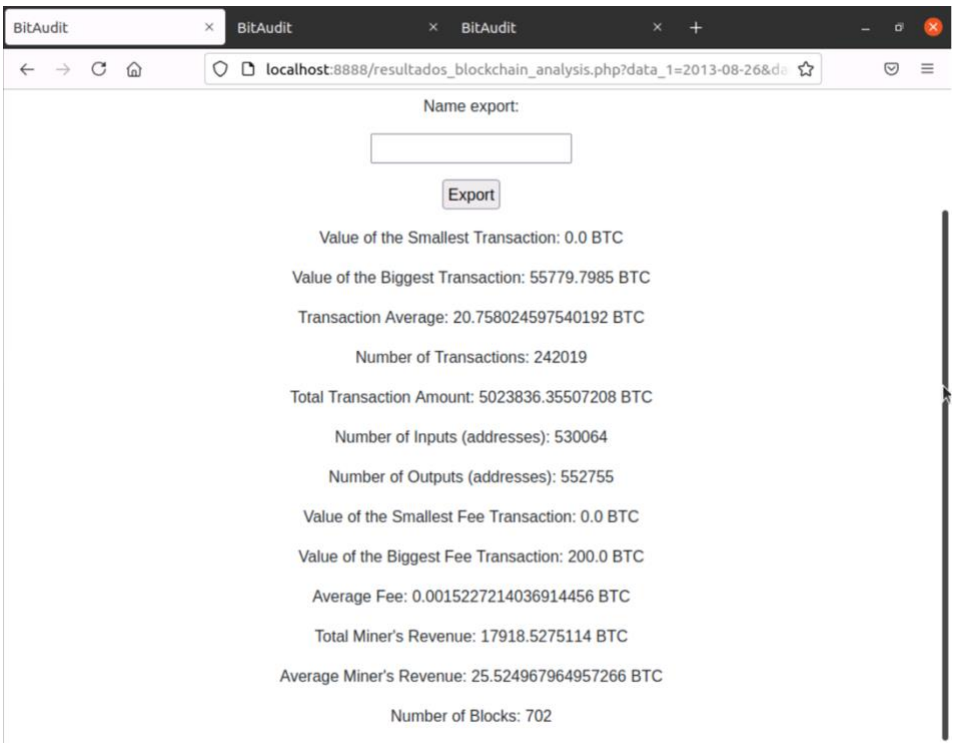


*Figura 63: Análise da blockchain nos cinco dias anteriores a 1 de outubro de 2013*

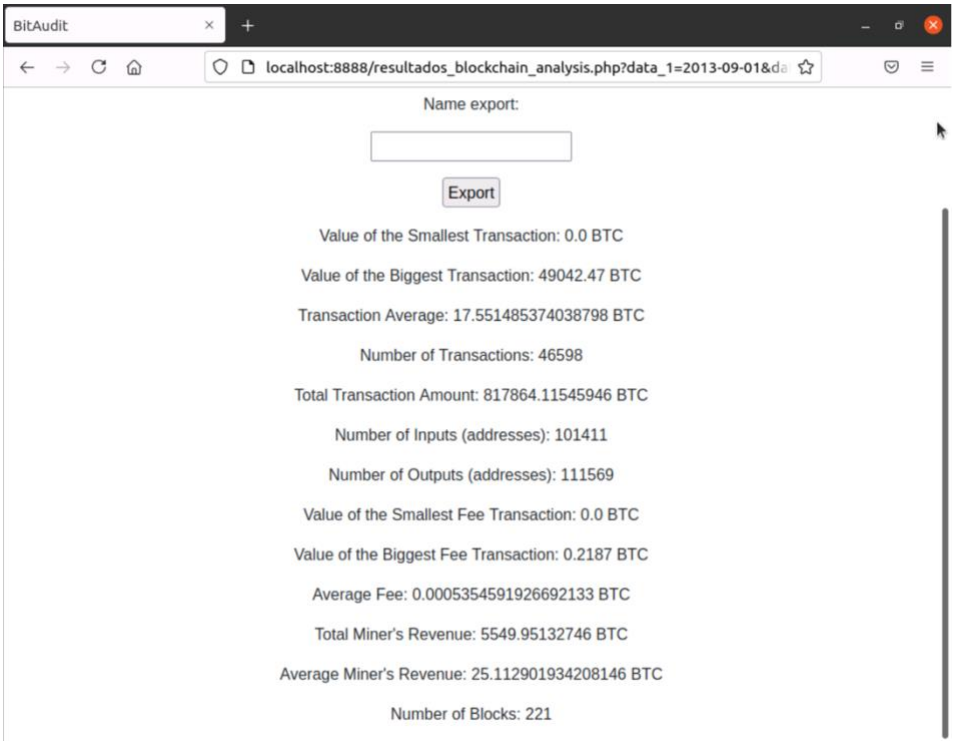


*Figura 64: Análise da blockchain no dia 1 de outubro de 2013*

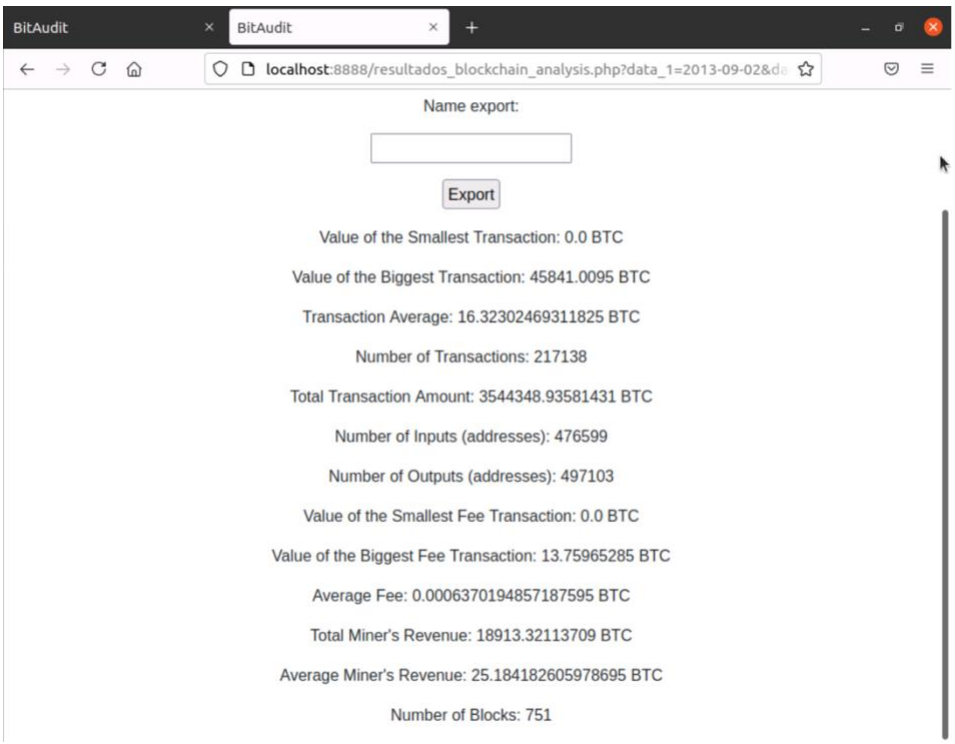


*Figura 65: Análise da blockchain nos cinco dias após 1 de outubro de 2013*

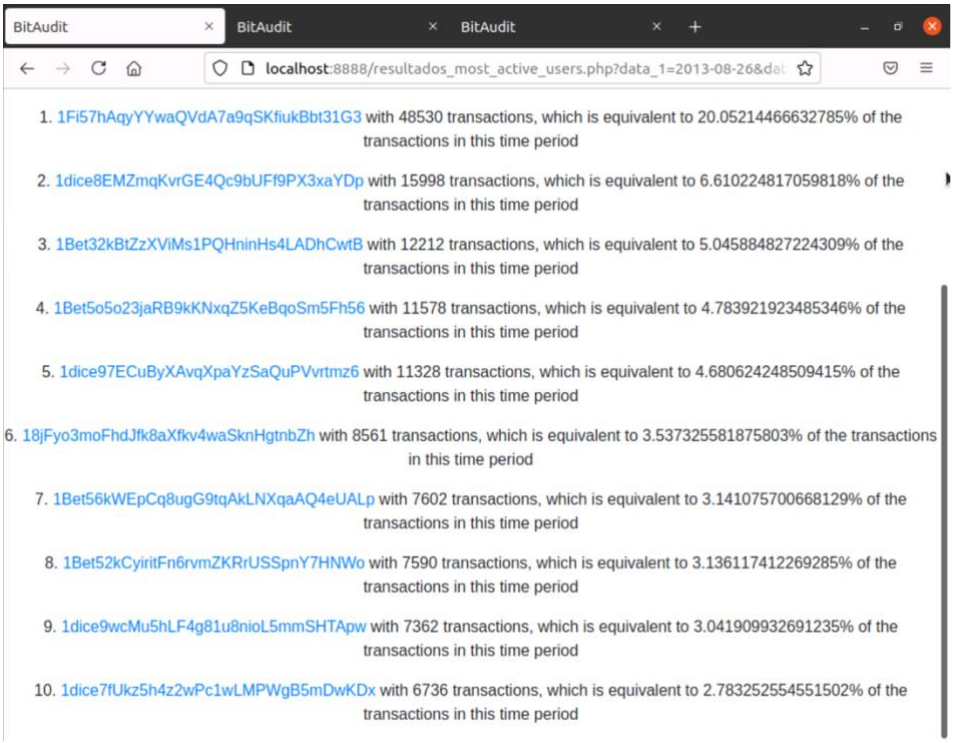


*Figura 66: Endereços mais ativos nos cinco dias anteriores a 1 de outubro de 2013*

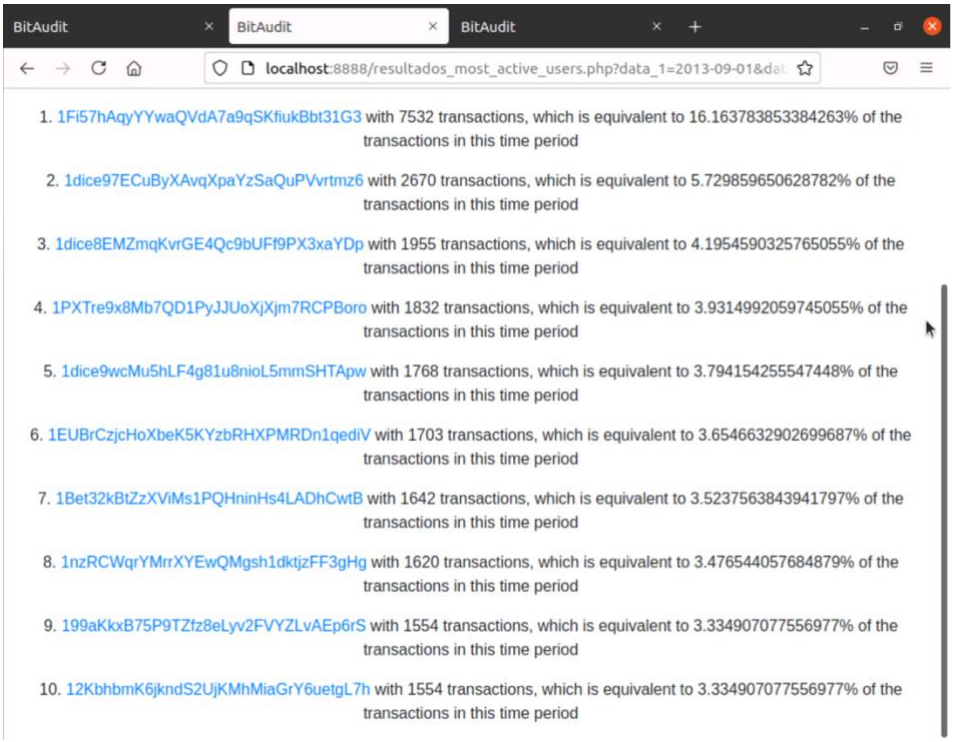


*Figura 67: Endereços mais ativos no dia 1 de outubro de 2013*

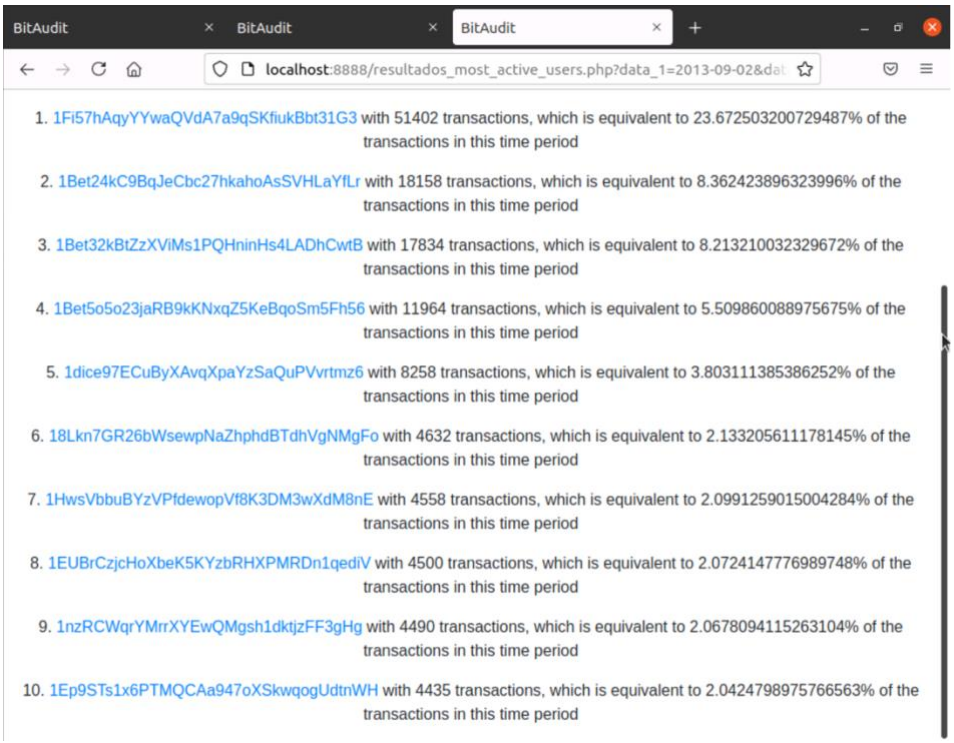


*Figura 68: Endereços mais ativos nos cinco após 1 de outubro de 2013*

Feita a análise da *blockchain* nos períodos descritos, é possivel observar que este evento levou a uma diminuição, tanto a nível do volume como do número de transações. De destacar o endereço mais ativo que reúne aproximadamente 20% das transações de todo o período temporal definido. De notar ainda, que existem mais endereços que se repetem nos três resultados.

Por último, o quarto caso, é referente a um endereço conhecido por estar associado a *CryptoLocker*, “18iEz617DoDp8CNQUyyrjCcC7XCGDf5SVb” [54], já estudado no capítulo três (Explorações Preliminares na Rede Bitcoin). Primeiro, foi utilizada a funcionalidade que permite fazer um estudo de *cluster*, de modo a reunir um conjunto de endereços que estivessem associados ao primeiro e que pudessem ser estudados (Figura 69). Após a exportação, os endereços foram utilizados para estudar a respetiva atividade do conjunto obtido (Figura 70).

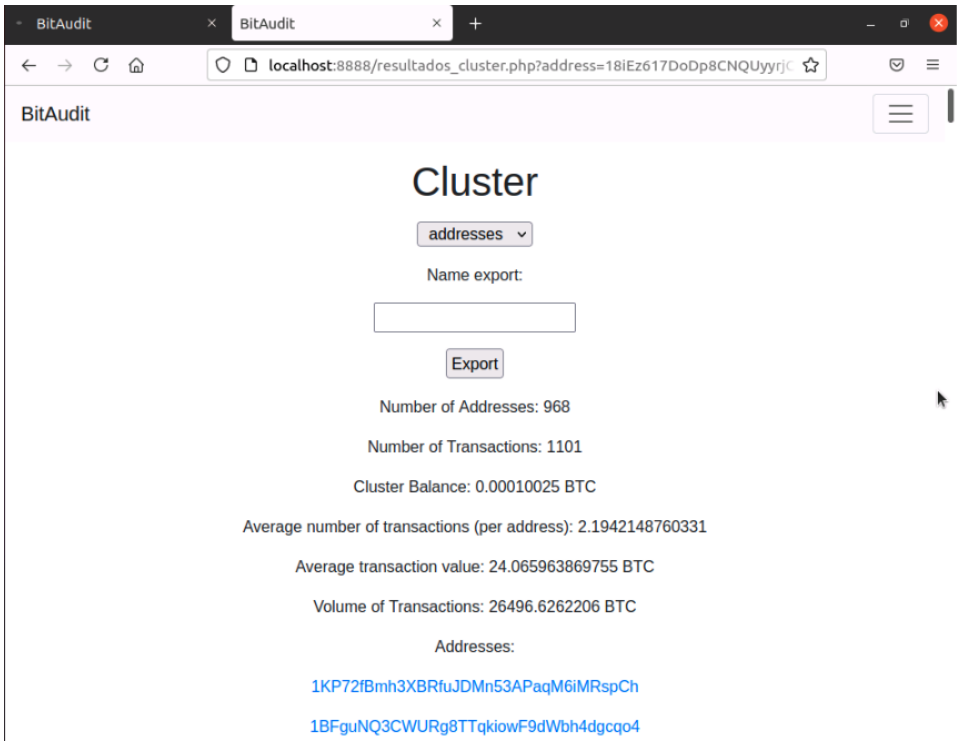


*Figura 69: Cluster do endereço "18iEz617DoDp8CNQUyyrjCcC7XCGDf5SVb"*

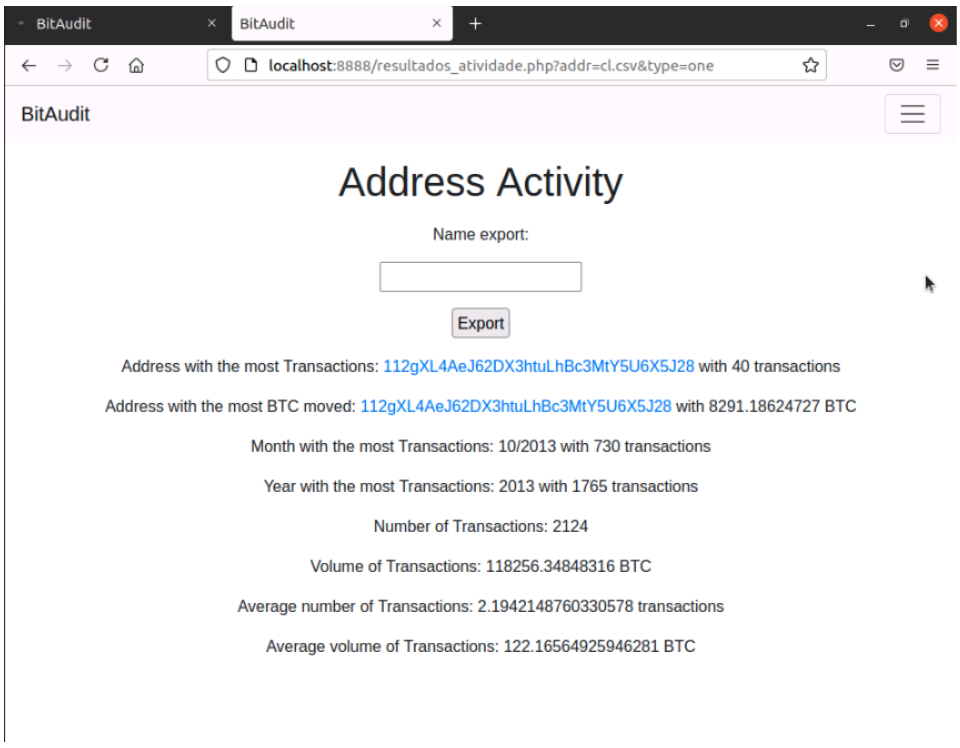


*Figura 70: Análise da atividade dos 968 endereços provenientes do estudo de cluster*

Apesar de só ter sido possivel concretizar estas duas análises, identificamos parte do que se consegue fazer com a ferramenta, de forma a estudar um endereço em específico.

Neste capítulo apresentamos algumas das análises que o *software* permite realizar. Assim, salientamos que o mesmo fornece ferramentas para investigadores e /ou autoridades, ou seja, por um lado, permite analisar especificamente endereços identificados como sendo utilizados para situações ilegais, com funcionalidades como "*Wash Trade", "*Análise de *Cluster"*, "Análise de Atividade" ou "*Address->Entity*". Por outro lado, possibilita a análise a situações ou eventos específicos, de forma a reunir dados que possam posteriormente ser analisados e tratados noutras ferramentas, sendo possivel exportar os resultados obtidos.

Foram ainda delineadas análises complementares, por exemplo, como descobrir situações de *wash trade* ou reunir os endereços restantes de uma entidade, mas devido a limitações de tempo, já que estas funcionalidades requerem dias para serem concluídas, não foi possível concretizar.

# 9. Conclusão

O principal objetivo deste estágio foi desenvolver ou adaptar um *software open-source,* que possibilitasse analisar as transações registadas na *blockchain* da Bitcoin, de forma a permitir realizar análises económicas, que foi cumprido.

Numa primeira etapa, foi efetuado o estado de arte, na qual foram reunidas informações e adquiridas competências essenciais para desenvolver/adaptar uma ferramenta de auditoria de Bitcoin. Durante esta fase, foram aprofundados os conhecimentos sobre a Bitcoin, uma criptomoeda descentralizada, construída com base na *blockchain*, sobre a análise forense na rede desta criptomoeda e analisados os *softwares* de auditoria Bitcoin existentes. Foi possível compreender a importância de prever e estimar a população de utilizadores ilegais, que apesar de existirem alternativas no mercado que certificam maior segurança e anonimato [14], a Bitcoin fornece "pseudo anonimato", que dificulta a associação de um endereço a uma "entidade". Também verificámos que foram já realizados estudos sobre como identificar operações ilegais com o uso de Bitcoin, tais como investigar redes comerciais de utilizadores conhecidos por estarem envolvidos em atividades ilegais, explorar certas características que distinguem utilizadores de *bitcoins* legais e ilegais e um método de sobreamostragem para detetar dados de transações ilegais de *bitcoins*. Relativamente aos *softwares* de análise, foi possível fazer uma comparação entre as ferramentas encontradas, que permitiu definir um rumo para este projeto. Optámos por utilizar o *BlockSci* [30], como *software* base para o desenvolvimento da nossa ferramenta, e definir as suas funcionalidades a partir de outros *softwares* [3][5][19][33] e de indicações fornecidas pelos orientadores.

Posteriormente à tomada de decisões, foram realizadas explorações preliminares com o intuito de obter um melhor entendimento da rede Bitcoin, auxiliadas por ferramentas já existentes como o *BlockSci* e o *Blockchain Explorer*.

O passo seguinte foi definir o processo e reunir a documentação necessária ao desenvolvimento do BitAudit, que incluiu a análise e validação de requisitos, definição da arquitetura do *software*, implementação e realização de testes.

Por fim, com a ferramenta desenvolvida, foram realizadas análises à rede Bitcoin, de modo a demonstrar parte do que poderia ser feito, por investigadores e autoridades, nomeadamente, como se poderia estudar certos aspetos (como atividade e endereços associados) de um endereço, identificado previamente. Contudo, salientamos que muito ficou por mostrar devido ao tempo requerido para a realização de algumas análises.

## 9.1 Limitações e Trabalho Futuro

O percurso foi longo, com alguns obstáculos que se tornaram desafios. Apesar de muito ter sido feito, existem sempre aspetos que podem ser aperfeiçoados ou adicionados.

Relativamente ao *software* desenvolvido, poderiam ter sido implementadas mais funcionalidades como, por exemplo, de deteção de atividade ilegal, com auxílio de inteligência artificial. Certas funcionalidades poderiam incluir mais parametrização, algo que se notou ser necessário nos testes, por terem tempos de execução tão longos.

Assim, ficam algumas sugestões para futuros estudos/investigações sobre esta temática, tomando como ponto de partida o nosso BitAudit.

# 10. Referências


[1] S. Samanta, B. K. Mohanta, S. P. Pati, and D. Jena, “A framework to build user profile on cryptocurrency data for detection of money laundering activities,” *Proc. - 2019 Int. Conf. Inf. Technol. ICIT 2019*, pp. 425–429, 2019, doi: 10.1109/ICIT48102.2019.00081.

[2] J. Lorenz, M. I. Silva, D. Aparício, J. T. Ascensão, and P. Bizarro, “Machine learning methods to detect money laundering in the Bitcoin blockchain in the presence of label scarcity,” May 2020, [Online]. Available: http://arxiv.org/abs/2005.14635.

[3] C. Kinkeldey, J. D. Fekete, T. Blascheck, and P. Isenberg, “BitConduite: Visualizing and analyzing entity activity on the bitcoin network,” *arXiv*, pp. 1–12, 2019.

[4] G. Di Battista, V. Di Donato, M. Patrignani, M. Pizzonia, V. Roselli, and R. Tamassia, “Bitconeview: Visualization of flows in the bitcoin transaction graph,” *2015 IEEE Symp. Vis. Cyber Secur. VizSec 2015*, 2015, doi: 10.1109/VIZSEC.2015.7312773.

[5] M. Spagnuolo, F. Maggi, and S. Zanero, “Bitiodine: Extracting intelligence from the bitcoin network,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8437, pp. 457–468, 2014, doi: 10.1007/978-3-662-45472-5_29.

[6] “Product development: The Waterfall methodology (model) in software development,” *https://learn.marsdd.com/article/product-development-the-waterfall-methodology-model-in-software-development/* (accessed June 6, 2021) .

[7] M.-F. Fortin, *O processo de investigação: Da concepção à realização*, 5th ed. Loures: Lusociência, 2009.

[8] J. (Queenís S. of B. Q. U. Webster and R. (Terry C. of B. T. U. of G. T. Watson, “ANALYZING THE PAST TO PREPARE FOR THE FUTURE: WRITING A LITERATURE REVIEW,” *Free Radic. Biol. Med.*, 2002.

[9] M. Rustem, K. Sergey, K. Anastasia, G. Muhamat, G. Venera, and K. Aleksey, “Problems of criminal responsibility for illegal circulation of cryptocurrency,” *Proc. - Int. Conf. Dev. eSystems Eng. DeSE*, vol. October-20, pp. 996–999, 2019, doi: 10.1109/DeSE.2019.00185.

[10] J. Han, J. Woo, and J. W. K. Hong, “Oversampling techniques for detecting bitcoin illegal transactions,” *APNOMS 2020 - 2020 21st Asia-Pacific Netw. Oper. Manag. Symp. Towar. Serv. Netw. Intell. Humanit.*, pp. 330–333, 2020, doi: 10.23919/APNOMS50412.2020.9236780.

[11] Y. Wu, A. Luo, and D. Xu, “Forensic analysis of bitcoin transactions,” *2019 IEEE Int. Conf. Intell. Secur. Informatics, ISI 2019*, pp. 167–169, 2019, doi: 10.1109/ISI.2019.8823498.

[12] M. Weber *et al.*, “Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics,” *arXiv*, no. 10, 2019.

[13] F. de C. Boavida, “Mas, afinal, o que é a blockchain?,” *https://www.sinalaberto.pt/mas-afinal-o-que-e-a-blockchain/*, 2020 (accessed Dec. 15, 2020).

[14] “O que é Bitcoin? Entenda de vez esse conceito!,” *https://www.comocomprarcriptomoedas.com/o-que-e-bitcoin/*, 2018 (accessed Jan. 17, 2021) .

[15] J. Tuwiner, “Bitcoin Mining Pools,” *https://www.buybitcoinworldwide.com/mining/pools/*, 2020 (accessed Jan. 10, 2021) .

[16] C. McCormack, “Is Bitcoin a Good Investment?,” *https://www.benzinga.com/money/is-bitcoin-a-good-investment/*, 2020 (accessed Dec. 22, 2021).

[17] S. Hoi, “Best Open-Source Blockchain Forensic Analysis Tools,” *https://www.1337pwn.com/best-open-source-blockchain-forensic-analysis-tools/*, 2020 (accessed Dec. 26, 2020) .

[18] “Matbea.net,” *https://matbea.net* (accessed Dec. 26, 2020) .

[19] M. Rella, M. Romiti, R. Stütz, and B. Haslhofer, “GraphSense.: Cryptoasset Analytics Platform,” *https://graphsense.info*, 2021 (accessed Jan. 9, 2021).

[20] “SoChain,” *https://sochain.com*, 2021 (accessed Dec. 26, 2020) .

[21] “blockstream.info,” *https://blockstream.info*, 2020 (accessed Dec. 26, 2020) .

[22] J. Rubin and H. Cooper, “Bitcoin Spark Framework (BTCSpark),” *https://github.com/JeremyRubin/BTCSpark*, 2017 (accessed Dec. 26, 2020) .

[23] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” *Artif. Life*, vol. 23, no. 4, pp. 552–557, 2017, doi: 10.1162/ARTL_a_00247.

[24] S. Foley, J. R. Karlsen, and T. J. Putniņš, “Sex, Drugs, and Bitcoin: How Much Illegal Activity Is Financed through Cryptocurrencies?,” *Rev. Financ. Stud.*, vol. 32, no. 5, pp. 1798–1853, May 2019, doi: 10.1093/rfs/hhz015.

[25] “Bitcoin Node,” *https://www.pcmag.com/encyclopedia/term/bitcoin-node* (accessed Apr. 28,

2021) .
[26] Mark, “Bitcoin Explained,” *https://www.mycryptopedia.com/bitcoin-explained/*, 2018 (accessed June 6, 2021) .
[27] M. Gupta, *Blockchain for dummies*, Limited. Hoboken: John Wiley & Sons, Inc., 2017.
[28] “O que é uma Carteira Multisig?,” *https://academy.binance.com/pt/articles/what-is-a-multisig-wallet*, 2021 (accessed June 21, 2021) .
[29] A. Balaskas and V. N. L. Franqueira, “Analytical Tools for Blockchain: Review, Taxonomy and Open Challenges,” *2018 Int. Conf. Cyber Secur. Prot. Digit. Serv. Cyber Secur. 2018*, pp. 1–8, 2018, doi: 10.1109/CyberSecPODS.2018.8560672.
[30] H. Kalodner *et al.*, “BlockSci: Design and applications of a blockchain analysis platform,” *Proc. 29th USENIX Secur. Symp.*, pp. 2721–2738, 2020.
[31] M. Bartoletti, A. Bracciali, S. Lande, and L. Pompianu, “A general framework for blockchain analytics,” *arXiv*, 2017.
[32] “Blockchain Explorer,” *https://www.blockchain.com/explorer* (accessed Feb. 10, 2021) .
[33] H. Kuzuno and C. Karam, “Blockchain explorer: An analytical process and investigation environment for bitcoin,” *eCrime Res. Summit, eCrime*, pp. 9–16, 2017, doi: 10.1109/ECRIME.2017.7945049.
[34] “Metadata,” *https://en.wikipedia.org/wiki/Metadata*, 2020 (accessed Dec. 26, 2020) .
[35] X. Yue *et al.*, “BitExTract: Interactive Visualization for Extracting Bitcoin Exchange Intelligence,” *IEEE Trans. Vis. Comput. Graph.*, vol. 25, no. 1, pp. 162–171, 2019, doi: 10.1109/TVCG.2018.2864814.
[36] Z. Zhong *et al.*, “SilkViser: A visual explorer of blockchain-based cryptocurrency transaction data,” *arXiv*, no. August, 2020, doi: 10.1109/vast50239.2020.00014.
[37] S. Phetsouvanh, F. Oggier, and A. Datta, “EGRET: Extortion graph exploration techniques in the bitcoin network,” *IEEE Int. Conf. Data Min. Work. ICDMW*, vol. 2018-Novem, pp. 244–251, 2019, doi: 10.1109/ICDMW.2018.00043.
[38] F. Oggier, S. Phetsouvanh, and A. Datta, “BiVA: Bitcoin network visualization and analysis,” *IEEE Int. Conf. Data Min. Work. ICDMW*, vol. 2018-Novem, pp. 1469–1474, 2019, doi: 10.1109/ICDMW.2018.00210.
[39] D. Kondor, M. Pósfai, I. Csabai, and G. Vattay, “Do the rich get richer? An empirical analysis of the Bitcoin transaction network,” *PLoS One*, vol. 9, no. 2, 2014, doi: 10.1371/journal.pone.0086197.
[40] J. Xiaomeng, Z. Fan, L. Shenwen, Y. Jinglin, and H. Ketai, “Data Analysis of Bitcoin Blockchain Network Nodes,” *Proc. 15th IEEE Conf. Ind. Electron. Appl. ICIEA 2020*, pp. 1891–1895, 2020, doi: 10.1109/ICIEA48937.2020.9248092.
[41] H. Xi, Z. Fan, L. Shenwen, M. Hongliang, and H. Ketai, “A Review on Data Analysis of Bitcoin Transaction Entity,” *Proc. 15th IEEE Conf. Ind. Electron. Appl. ICIEA 2020*, pp. 159–164, 2020, doi: 10.1109/ICIEA48937.2020.9248197.
[42] I. Alqassem, I. Rahwan, and D. Svetinovic, “The Anti-Social System Properties: Bitcoin Network Data Analysis,” *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 50, no. 1, pp. 21–31, 2020, doi: 10.1109/TSMC.2018.2883678.
[43] A. Kumar *et al.*, “Empirical Analysis of Bitcoin network (2016-2020),” *2020 IEEE/CIC Int. Conf. Commun. China, ICCC Work. 2020*, no. October, pp. 96–101, 2020, doi: 10.1109/ICCCWorkshops49972.2020.9209945.
[44] A. Biryukov and S. Tikhomirov, “Transaction Clustering Using Network Traffic Analysis for Bitcoin and Derived Blockchains,” *INFOCOM 2019 - IEEE Conf. Comput. Commun. Work. INFOCOM WKSHPS 2019*, pp. 204–209, 2019, doi: 10.1109/INFCOMW.2019.8845213.
[45] L. Nan and D. Tao, “Bitcoin mixing detection using deep autoencoder,” *Proc. - 2018 IEEE 3rd Int. Conf. Data Sci. Cyberspace, DSC 2018*, pp. 280–287, 2018, doi: 10.1109/DSC.2018.00047.
[46] L. Conway, “Bitcoin Halving,” *https://www.investopedia.com/bitcoin-halving-4843769*, 2021 (accessed Mar. 3, 2021) .
[47] K. Pequenino, “Bitcoin e outras moedas em queda com Coreia do Sul a querer travar transacções,” *https://www.publico.pt/2018/01/11/tecnologia/noticia/bitcoin-tropeca-com-coreia-do-sul-a-ponderar-proibir-transaccoes-com-criptomoedas-1798950*, 2018 (accessed Feb. 17, 2021) .
[48] S. Antunes, “Bitcoin triplica de valor desde abril,” *https://www.jornaldenegocios.pt/trading/detalhe/bitcoin-triplica-de-valor-desde-abril*, 2019 (accessed Feb 17, 2021) .
[49] P. Pinto, “Bitcoin já ultrapassou os 9 mil dólares! Moeda digital está imparável,”

*https://pplware.sapo.pt/informacao/bitcoin-ja-ultrapassou-os-9-mil-dolares/*, 2019 (accessed Feb. 17, 2021) .
[50] R. Gregorio, “Bitcoin tem maior queda diária da história e derrete 50% em uma semana; e agora?,” *https://valorinveste.globo.com/mercados/cripto/noticia/2020/03/14/bitcoin-tem-maior-queda-diaria-da-historia-e-derrete-50percent-em-uma-semana-e-agora.ghtml*, 2020 (accessed Feb. 17, 2021) .
[51] J. Tomé, “Investimento da Tesla em bitcoin pode ser investigado. Vítor Constâncio critica Musk e ‘zeros e uns,’” *https://www.dinheirovivo.pt/empresas/tecnologia/investimento-da-tesla-em-bitcoin-pode-ser-investigado-vitor-constancio-critica-musk-e-bitcoin-13337891.html*, 2021 (accessed Feb. 19, 2021) .
[52] “Multisignature,” *https://en.bitcoin.it/wiki/Multisignature*, 2019 (accessed Mar. 2, 2021) .
[53] F. Reid and M. Harrigan, “An analysis of anonymity in the Bitcoin system,” *Proc. - 2011 IEEE Int. Conf. Privacy, Secur. Risk Trust IEEE Int. Conf. Soc. Comput. PASSAT/SocialCom 2011*, no. July, pp. 1318–1326, 2011, doi: 10.1109/PASSAT/SocialCom.2011.79.
[54] “Wallet Explorer,” *https://www.walletexplorer.com* (accessed Mar. 7, 2021) .
[55] “Cryptolocker Virus Definition,” *https://usa.kaspersky.com/resource-center/definitions/cryptolocker* (accessed Mar. 7, 2021).
[56] “BlockSci Version 0.7 Documentation,” *https://citp.github.io/BlockSci/index.html* (accessed June 6, 2021) .
[57] G. Kotonya and I. Sommerville, *Requirements Engineering: Processes and Techniques*. 1998.
[58] “Requirements analysis,” *https://en.wikipedia.org/wiki/Requirements_analysis#cite_note-2* (accessed May 21, 2021) .
[59] A. Abran, J. W. Moore, P. Bourque, and R. Dupuis, *Guide to the Software Engineering Body of Knowledge*, 3rd ed. 2014.
[60] “Goal modeling,” *https://en.wikipedia.org/wiki/Goal_modeling* (accessed May 1, 2021).
[61] “What is Use Case Diagram?,” *https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/* (accessed Feb. 11, 2021).
[62] A. Cockburn, *Writing Effective Use Cases*. Addison-Wesle, 2000.
[63] Https://en.wikipedia.org/wiki/Rational_Unified_Process, “Rational Unified Process.” (accessed May 1, 2021).
[64] “Usability,” *https://en.wikipedia.org/wiki/Usability* (accessed June 21, 2021) .
[65] J. Nielsen, “Ten Usability Heuristics,” [Online]. Available: http://intra.iam.hva.nl/content/1112/verdieping1/research_for_design/intro-en-materiaal/RfD-Heuristic-Evaluation.pdf.
[66] H. Beyer and K. Holtzblatt, “Contextual design,” *Interactions*, vol. 6, no. 1, pp. 32–42, 1999, doi: 10.1145/291224.291229.
[67] “Why Is Software Architecture Important?,” *https://people.ece.ubc.ca/matei/EECE417/BASS/ch02lev1sec4.html* (accessed May 20, 2021) .
[68] S. Brown and M. Costa, “O modelo C4 de documentação para Arquitetura de Software,” *https://www.infoq.com/br/articles/C4-architecture-model/*, 2018 (accessed May 20, 2021) .
[69] “What is Docker?,” *https://opensource.com/resources/what-docker* (accessed June 6, 2021) .
[70] Armin Ronacher, “Foreword,” *https://web.archive.org/web/20171117015927/http://flask.pocoo.org/docs/0.10/foreword*, 2013 (accessed June 6, 2021) .
[71] “Flask (web framework),” *https://en.wikipedia.org/wiki/Flask_(web_framework)* (accessed June 6, 2021) .
[72] “Flask Extensions,” *https://web.archive.org/web/20180517082208/http://flask.pocoo.org/extensions/* (accessed June 6, 2021) .
[73] “Flask vs Django: How to Understand Whether You Need a Hammer or a Toolbox,” *https://steelkiwi.medium.com/flask-vs-django-how-to-understand-whether-you-need-a-hammer-or-a-toolbox-39b8b3a2e4a5*, 2019 (accessed June 6, 2021) .
[74] “Advantages of using an API,” *https://www.meteosim.com/en/advantages-of-using-api/*, 2019 (accessed June 6, 2021) .
[75] “Apache HTTP Server,” *https://en.wikipedia.org/wiki/Apache_HTTP_Server* (accessed June 6, 2021) .
[76] “OS/Linux Distributions using Apache,” *https://secure1.securityspace.com/s_survey/data/man.201808/apacheos.html*, 2018 (accessed June 6, 2021) .
[77] “Platform Specific Notes - Apache HTTP Server Version 2.4,”

*https://httpd.apache.org/docs/2.4/platform/index.html.en* (accessed June 6, 2021) .
[78] "Secure Web Server," *https://vmssoftware.com/products/secure-web-server/* (accessed June 6, 2021) .
[79] O. Romanyuk, "PHP vs JavaScript: How to Choose the Best Language for Your Project," *https://www.freecodecamp.org/news/php-vs-javascript-which-technology-will-suit-your-business-better/* (accessed June 6, 2021) .
[80] "O que é o PHP?," *https://www.php.net/manual/pt_BR/intro-whatis.php* (accessed June 6, 2021) .
[81] "HTML Introduction," *https://www.w3schools.com/html/html_intro.asp* (accessed June 6, 2021) .
[82] "Deployment environment," *https://en.wikipedia.org/wiki/Deployment_environment* (accessed June 6, 2021) .
[83] "HTTP," *https://developer.mozilla.org/pt-BR/docs/Web/HTTP*, 2021 (accessed June 6, 2021) .
[84] "JSON," *https://en.wikipedia.org/wiki/JSON* (accessed June 6, 2021) .
[85] "How to Use Nano, the Linux Command Line Text Editor," *https://linuxize.com/post/how-to-use-nano-text-editor/*, 2019 (accessed June 6, 2021) .
[86] "Insertion Sort," *https://www.geeksforgeeks.org/insertion-sort/*, 2021 (accessed June 6, 2021) .
[87] P. Ammann and J. Offutt, *Introduction to Software Testing*, 2nd ed. Cambridge: Cambridge University Press, 2016.
[88] "What is Software Testing? Definition, Basics & Types," *https://www.guru99.com/software-testing-introduction-importance.html* (accessed June 6, 2021) .
[89] Rajkumar, "What Is Software Testing," *https://www.softwaretestingmaterial.com/software-testing/* (accessed June 6, 2021) .
[90] "What is Black-Box Testing?," *https://www.guru99.com/black-box-testing.html* (accessed June 6, 2021) .
[91] "Performance Testing Tutorial," *https://www.guru99.com/performance-testing.html* (accessed June 6, 2021) .
[92] B. Bonani, "O que é a CBOE?," *https://bugg.com.br/o-que-e-a-cboe/*, 2018 (accessed June 18, 2021) .
[93] H. Garcia, "O que é um contrato de futuros?," *https://www.rankia.pt/bolsa/o-que-e-um-contrato-de-futuros/*, 2019 (accessed June 18, 2021) .
[94] E. Caetano, "Bitcoin chega à alta finança. Até onde irá a febre das moedas digitais?," *https://observador.pt/especiais/bitcoin-chega-a-alta-financa-ate-onde-ira-a-febre-das-moedas-digitais/*, 2017 (accessed June 18, 2021) .

# Anexo

Utilização:

- cd /home/admin/compose (no terminal)
- docker-compose up (adicionar --build antes do “up” caso seja a primeira vez)
- http://localhost:8888 (para iniciar a ferramenta no *browser*)