UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Luís Felipe Brasil Corrêa

MONETIZOR: API DE PAGAMENTOS AUTOMATIZADOS UTILIZANDO *BLOCKCHAIN* PARA *E-SPORTS*

Florianópolis

2019

Luís Felipe Brasil Corrêa

MONETIZOR: API DE PAGAMENTOS AUTOMATIZADOS UTILIZANDO *BLOCKCHAIN* PARA *E-SPORTS*

Tese submetida ao Curso de Graduação em Sistemas de Informação para a obtenção do Grau de Bacharel em Sistemas De Informação.
Orientador: Prof. José Eduardo de Lucca

Florianópolis

2019

Catalogação na fonte elaborada pela biblioteca da
Universidade Federal de Santa Catarina

A ficha catalográfica é confeccionada pela Biblioteca Central.

Tamanho: 7cm x 12 cm

Fonte: Times New Roman 9,5

Maiores informações em:
http://www.bu.ufsc.br/design/Catalogacao.html

Luís Felipe Brasil Corrêa

MONETIZOR: API DE PAGAMENTOS AUTOMATIZADOS UTILIZANDO *BLOCKCHAIN* PARA *E-SPORTS*

Esta Tese foi julgada aprovada para a obtenção do Título de "Bacharel em Sistemas De Informação", e aprovada em sua forma final pelo Curso de Graduação em Sistemas de Informação.

Florianópolis, 01 de Junho 2019.

Banca Examinadora:

Prof. José Eduardo de Lucca
Presidente

Prof. José Eduardo de Lucca
Orientador

Prof. Renato Fileto

Prof. Elder Rizzon Santos

Prof. Fernando Augusto da Silva Cruz

Dedico este trabalho aos meus pais, namorada, familiares, professores e amigos.

AGRADECIMENTOS

Agradeço minha família por sempre ter dado o apoio e suporte necessários para conseguir chegar até aqui. Agradeço minha namorada Thaiza Wilwert por sempre ter me lembrado de fazer as matrículas para continuar como um aluno da UFSC, estar sempre me acompanhando e fazendo minha vida mais feliz. Todos os professores que participaram da minha formação, em especial os que compõem a banca avaliadora e o orientador José Eduardo De Lucca que fez muito mais do que suas obrigações como orientador, sem o suporte dele eu certamente não teria conseguido concluir o projeto, por isso serei eternamente grato! Agradecimentos ao Ronnis Martins da coordenação que sempre foi muito prestativo me auxiliado todas as várias vezes que precisei e aos amigos que fizeram companhia e ajudaram nos trabalhos e estudos necessários para a conclusão desta etapa.

"The blockchain cannot be described just as a revolution. It is a tsunami-like phenomenon, slowly advancing and gradually enveloping everything along its way by the force of its progression."

William Mougayar

RESUMO

O surgimento de uma nova tecnologia denominada *Blockchain*, uma espécie de registro criado de forma distribuída, vem permitindo o desenvolvimento de várias novas aplicações direcionadas a inúmeros nichos diferentes. Este trabalho tem como objetivo conhecer e avaliar essa tecnologia em profundidade - tanto em aspectos técnicos quanto em relação a seu impacto na sociedade - e aplicá-la em um problema de domínio específico. A solução foi modelada a partir das ferramentas e conceitos que essa tecnologia oferece para criar uma plataforma que realiza pagamentos automatizados em ambientes virtuais competitivos. Agora desenvolvedores possuem uma maneira simples e segura de monetizar suas plataformas.

Palavras-chave: chave 1. *Blockchain* chave 2. E-sports chave 3. Bitcoin chave 4. Ethereum

ABSTRACT

The emergence of a new technology called *Blockchain*, a kind of distributed created ledger, has allowed the development of several new applications in many different niches. This paper aims to study and evaluate this technology in depth - both in technical aspects and in relation to its impact on society - and apply it to a specific domain problem. The solution was modeled on the tools and concepts that this technology offers to create a platform that makes automated payments in competitive virtual environments. Now developers have a simple and secure way to monetize their platforms.
Keywords: key 1. *Blockchain* key 2. E-sports key 3. Bitcoin key 4. Ethereum

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Desde o começo da história da humanidade o ser humano interage através de trocas, seja de mercadorias, informações ou favores. Inicialmente as regras envolvidas nas trocas eram garantidas através da violência ou repercussões sociais, mas, com o avanço da sociedade, começaram a surgir formas mais complexas de controle (WARBURG, 2018). Surgiram então os órgãos reguladores e garantidores da ordem como bancos e instituições governamentais. Com o surgimento da Internet, essas regulações começaram a ser feitas online, agilizando o processo mas ainda havendo a necessidade de muita burocracia envolvida, para que o intermediário tivesse segurança para realizar as transações. A questão principal é que até pouco tempo atrás, não havia uma maneira confiável para que duas partes que não sabem se podem confiar uma na outra pudessem realizar trocas sem envolver uma terceira parte que ambas confiassem. Com o surgimento da *Blockchain*, existe a possibilidade de diminuir os riscos envolvidos em transações sem a necessidade de uma instituição política ou bancária fazendo essa intermediação, apenas usando tecnologia.

Da mesma forma que a internet não possui fronteiras geográficas aparentes, a descentralização inerente da rede do Bitcoin possibilita um grande potencial de inclusão financeira. Hoje no Brasil, praticamente metade da população não possui acesso a sistemas de pagamentos por não possuir conta em banco. Esta realidade pode mudar rapidamente, já que qualquer pessoa com um smartphone e internet consegue acessar plataformas de pagamentos com alcance global. Além de exigir menos burocracia, o serviço costuma ser mais ágil e custar menos que os meios de transferência monetários tradicionais (ULRICH, 2018).

Esse algoritmo de assinaturas encadeadas, por mais simples que pareça, provavelmente vai revolucionar as mais variadas áreas da sociedade, mesmo que elas não estejam diretamente ligadas à tecnologia. Isso ocorre pois o ativo não é controlado por nenhuma instituição centralizada, impedindo que governos ou instituições interfiram. Além disso seu algoritmo, embora simples, demonstra grande robustez e tem código aberto, permitindo a checagem de suas regras e garantindo que não exista nenhum tipo de backdoor. A empolgação com as criptomoedas foi tão grande que vários projetos utilizando a tecnologia ou variações dela estão sendo desenvolvidos, como é o caso do ODEM (ODEM, 2018), ETHEREUM (BUTERIN, 2018), IOTA (POPOV, 2018). Soluções para a área de registros e autenticação como as que são fei-

tas nos cartórios, identidade digital, moedas digitais, passe digital de voto, aplicação de contratos, histórico de procedência de produtos, histórico acadêmico, aplicação automática de contratos, as possibilidades são muitas! (TAPSCOTT, 2018)

1.1 JUSTIFICATIVA

O mercado de entretenimento mundial vem crescendo a passos largos, especialmente a parte que envolve jogos eletrônicos. Isso ocorre pois existe uma grande migração das mídias convencionais como o televisão aberta para mídias eletrônicas como plataformas de streaming e jogos eletrônicos (CAPSL, 2019), que estão cada vez mais acessíveis para a população com a difusão da internet, computadores e smartphones mais baratos.

No caso dos jogos eletrônicos a competitividade está muito presente e da mesma forma que ocorre nos esportes convencionais, existem competidores que se destacam e acabam se tornando cyberatletas. Eles entram para times e participam de competições, sendo que algumas delas chegam a rivalizar várias competições de esportes convencionais. Um jogo chamado de Dota 2 por exemplo, em 2018 distribuiu mais de 106 milhões de reais em um campeonato mundial, enquanto o campeonato brasileiro distribuiu "apenas" 63,7 milhões no mesmo ano. Para estressar como o crescimento está acelerado na categoria, o mesmo evento, no ano de 2019 distribuiu mais de 143 milhões de reais em premiações, 34% a mais do que o ano anterior (PRIZETRACKER, 2019).

Muitos dos cyberatletas, mesmo participando da elite em seus respectivos jogos, não são capazes de produzir renda diretamente da própria habilidade. Equipes que participam dos principais campeonatos e conseguem terminar bem posicionados conseguem fazer isso, mas times ou jogadores que participam do circuito e não ficam tão bem posicionados, acabam precisando recorrer a fontes secundárias de renda como patrocínios e participação de propagandas por visualizações em plataformas de streaming (SMITH, 2018).

Hoje existe um jogo de habilidade que é amplamente monetizado, ele se chama poker. Embora tenha florescido em ambientes relacionados com jogos de azar como cassinos, ele não é a mesma coisa que uma roleta por exemplo, pois os jogadores não jogam contra a casa, eles jogam contra outros jogadores e casa ganha um pequeno percentual por hospedar o jogo e criar toda a estrutura. Na prática, existem

jogadores apostando na própria habilidade de jogar melhor que outros jogadores e com isso, acabar ganhando dinheiro. Tal prática também é comum em outros esportes mais consolidados como o tênis. Existe a diferença que o jogador não precisa pagar uma entrada para participar do torneio, mas ele ainda precisa arcar com todos os custos de viagem e estadia. Neste caso a premiação é levantada através de patrocínios, mas em esportes menores ou classes mais amadoras, precisar pagar uma inscrição para participar de algum torneio não é incomum. Isso ocorre pois existem custos envolvidos em organizar um campeonato, desde estrutura necessária para o evento ocorrer, até a premiação que vai ser distribuída para os vencedores. Nem sempre o evento traciona mídia suficiente para angariar patrocínios que consigam pagar todos os custos, mas mesmo assim, existem muitas pessoas dispostas a competir entre elas para se consagrarem campeões e coletarem uma premiação.

Visto que a grande maioria dos atletas virtuais que, mesmo pertencentes da elite de seus respectivos *e-sports*, não conseguem promover uma renda consistente à partir de sua habilidade (TAYLOR, 2012), foi identificada a necessidade de uma solução que permita a monetização deste tipo de ecossistema digital. Com o surgimento das criptomoedas, *Blockchain* e os contratos inteligentes da rede Ethereum, este tipo de habilidade que exige altíssimo nível de especialização para ser desempenhada, poderá finalmente ter uma plataforma viabilizando a profissionalização, sem a necessidade de intermediários, para o ramo.

1.2 OBJETIVOS

Este trabalho tem como objetivo identificar e testar formas de integrar conceitos de *Blockchain* e *smart contracts* ao ecossistema de *e-sports*. Traçando uma estratégia de negócio adequada e escolhendo uma solução tecnológica para o desenvolvimento de uma criptomoeda e um serviço que sejam compatíveis com o mercado.

1.3 OBJETIVOS ESPECÍFICOS

1. Analisar a fundamentação teórica sobre *Blockchain*, *Smart Contracts* e *E-Sports*.

2. Modelar e implementar uma API que permita a automação de pagamentos utilizando *Blockchain* e *Smart Contracts* para o mercado competitivo eletrônico.

   (a) Criar uma *Blockchain* da rede Ethereum funcional.
   (b) Desenvolver o Smart Contract que vai fazer a custódia dos prêmios da competição e posteriormente realizar os pagamentos.
   (c) Desenvolver uma API que vai receber e enviar mensagens para os desenvolvedores e viabilizar o serviço.

3. Implantar e avaliar a solução desenvolvida.

2 REVISÃO BIBLIOGRÁFICA

2.1 BITCOIN

O Bitcoin foi a primeira moeda virtual criada utilizando a *Blockchain*, dando início a uma nova classe de ativo, as criptomoedas. Antes do Bitcoin o conceito de *Blockchain* não existia, ela foi desenvolvida justamente para viabilizar a possibilidade de uma moeda virtual que não necessitasse de alguma instituição (seja governamental ou privada) atuando como mediadora das transações e impedir a replicação indevida do ativo (AIR, 2016).

Ele(Bitcoin) foi criado em 2008 e começou a operar em 3 de janeiro de 2009, quando Satoshi Nakamoto minerou o "bloco gênese do Bitcoin" (bloco número 0) que o premiou com 50 Bitcoins. Essas foram as primeiras 50 moedas das mais de 18 milhões emitidas até hoje, onde a quantidade produzida máxima será de 21 milhões(CHOHAN, 2017). Acredita-se que Satoshi Nakamoto seja um pseudônimo para a pessoa ou equipe que desenvolveu o Bitcoin, pois até hoje o criador não foi identificado.

A moeda não possuía nenhum valor equivalente em dólares até maio de 2010, quando Laszlo Hanyecz realizou a primeira transação no mundo real ao comprar duas pizzas por 10 mil Bitcoins, dando uma cotação inicial de menos de 1 centavo de dólar por moeda (CHOHAN, 2017). Desde então o preço da criptomoeda subiu muito, chegando a cotação máxima de aproximadamente 20 mil dólares por unidade em dezembro de 2017.

Como foi mencionado anteriormente, a moeda funciona de forma distribuída e não possui nenhuma instituição que a controle, isso gerou alguns novos problemas para os governos. Como as transações de Bitcoin são feitas usando criptografia pela internet, é praticamente impossível que governos do mundo impeçam que as pessoas transacionem ela ou movimentem para dentro ou fora das nações de maneira ilegal. Antes das criptomoedas esse controle era feito através de bancos, mas como não existe uma entidade física real, é simplesmente impossível impedir que as moedas sejam transacionadas entre pessoas (TORPEY, 2019).

Outro problema que o Bitcoin cria para as autoridades acontece na hora do governo tentar tributar o dono dos recursos. Hoje o imposto de renda é recolhido na fonte e o trabalhador não tem outra opção além de aceitar que isto ocorra. Além disso, os bancos enviam

informações de todas as movimentações que ocorrem nas contas, permitindo assim que o governo saiba o quanto cada pessoa movimenta e com isso saber quanto cada uma deve ser tributada. No caso do Bitcoin e outras criptomoedas, por elas não serem controladas por alguma instituição específica, fica impossível para governos impor regras ou taxas no sistema (RATHS, 2013). Se alguma pessoa quiser esconder sua movimentação e sonegar impostos, não existe nem a possibilidade dos governos confiscarem os recursos, já que a única pessoa que consegue acessar e movimentar o dinheiro da carteira de Bitcoin é o portador da chave privada correspondente.

Antigamente era relativamente fácil para governos imporem regras para empresas que quisessem operar em seu território. Agora, com a internet, criptografia e a *Blockchain* viabilizando soluções descentralizadas, fica cada vez mais difícil para órgãos reguladores controlarem este tipo de operação. No caso do Bitcoin por exemplo, já ocorreram tentativas de proibição em corretoras, que são empresas especializadas em fazer esse tipo de câmbio entre Bitcoin e alguma moeda local, no entanto, fica virtualmente impossível de impedir que pessoas físicas estejam dispostas a fazer trocas de moedas entre si, ou realizar e aceitar pagamentos com ele.

2.2 BLOCKCHAIN

A tecnologia *Blockchain* (Cadeia de blocos) vem sendo considerada por muitos como *"The great new thing"* ("A grande novidade" - tradução livre) (ULRICH, 2018) (KOSBA, 2016). Isto acontece devido ao grande potencial que a tecnologia pode ter nas mais variadas áreas, sejam elas financeiras, jurídicas ou de segurança. Alguns autores e instituições especulam que em 20 anos o *Blockchain* vai revolucionar a sociedade da mesma forma que a internet revolucionou nos últimos 20 anos (HERNANDEZ, 2017). Uma de suas principais funcionalidades atuais consiste em uma solução descentralizada e virtual para o dinheiro conhecida como Bitcoin, permitindo o fácil armazenamento e transferência de moeda virtual, sem a necessidade de uma instituição intermediadora, para qualquer lugar do mundo que possua internet.

A cadeia de blocos funciona como um grande registro digital de transações que é distribuída, verificada e monitorada por múltiplas fontes de forma simultânea (GREENSTEIN, 2018). Ela permite que o registro de coisas que hoje é feito através de instituições reguladoras como bancos e cartórios, seja feito de modo confiável, seguro e com menos

margem para falsificação ou irregularidades. Uma outra característica interessante é a completa rastreabilidade desses registros, sejam eles referentes a uma transação monetária ou o registro de um carro ou imóvel por exemplo, garantindo coisas como: quem possui o quê, validando transações ou garantindo que alguma informação seja realmente verídica (BLENKINSOP, 2019).

A tecnologia que surgiu inicialmente como uma solução descentralizada para um tipo de dinheiro virtual, muitas vezes é desacreditada por alguns especialistas (NIKOLAEV, 2019) quanto a sua capacidade de se tornar uma moeda amplamente utilizada, seja pela variação dos preços, seja pelos custos das transações. Mas um fato interessante é que, mesmo os autores que vêem o Bitcoin como uma espécie de bolha de especulação financeira, reconhecem o poder disruptivo que a tecnologia *Blockchain* possui (CHEN, 2018). Isso reforça a necessidade de diferenciá-los claramente: uma é a tecnologia (Blockchain) outra é uma aplicação (Bitcoin).

Para entender o funcionamento da tecnologia *Blockchain*, será explicada a aplicação da mesma no mecanismo que sustenta a moeda Bitcoin. Segundo (NAKAMOTO, 2008) a ideia é que a moeda seja uma cadeia assinaturas eletrônicas. Cada dono de uma moeda faz a transferência assinando digitalmente o *hash* da transação anterior junto com a chave pública do próximo dono, adicionando elas no final da moeda. Um beneficiário pode verificar as assinaturas checando a cadeia de propriedade. O registro das transações é feito de forma descentralizada em uma cadeia de blocos, sendo que a maior cadeia acaba sendo a cadeia correta, que é seguida por todos os nodos da rede.

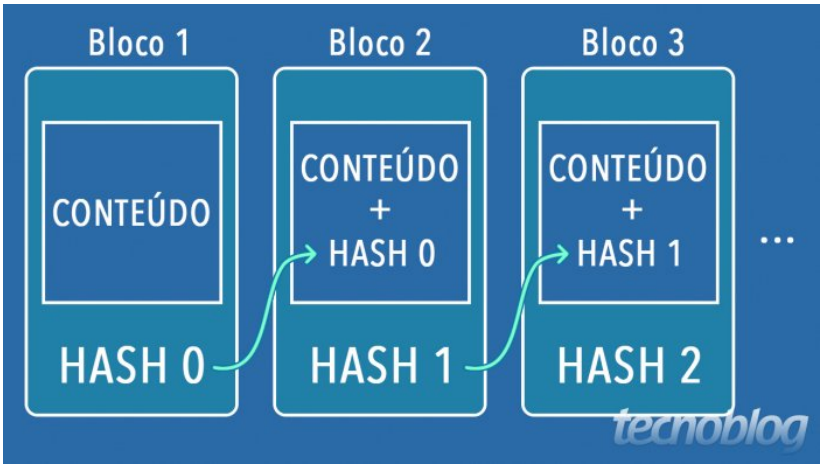


Figura 1 – Exemplo de como ocorre o encadeamento dos blocos na *Blockchain*.(PRADO, 2017)

Em outras palavras, quando alguém possui Bitcoins o que ela realmente possui é uma chave privada, que é a metade de uma assinatura digital que prova que ela possui algum Bitcoin específico. A outra metade é uma chave pública e é ela que fica registrada na *Blockchain*. Um *hash* desta chave pública é o que define o endereço da carteira de Bitcoin de um usuário. No momento que alguém que deseja transferir moedas de uma carteira para outra, ela faz uma requisição que é recebida por todos os computadores que compõem a rede P2P do Bitcoin (NAKAMOTO, 2008). Esses computadores e seus donos são chamados de mineradores. Eles organizam as transações recebidas em um bloco e adicionam este bloco à cadeia de blocos anteriores, sendo que quem finaliza a operação primeiro é premiado com novos Bitcoins em sua carteira. Quanto mais poder computacional o minerador possui, maior a chance dele realizar a operação primeiro. Assim que ele finaliza a operação e adiciona o bloco ao resto da cadeira, ele faz um broadcast para a rede, que checa o trabalho realizado e começa a considerar a nova versão da *Blockchain* como oficial, caso todas as operações tenham sido corretamente validadas. Interessante notar que, a cada novo bloco adicionado na *Blockchain*, mais segura ela fica, pois para alterar alguma transação registrada, seria necessário replicar todas as transações anteriores registradas, mais as transações do bloco atual, tudo isso antes de toda a rede validar as transações realizadas apenas no último bloco. Isso exigiria um poder computacional individual maior

que metade do poder da rede distribuída, o que torna essa possibilidade bastante improvável.

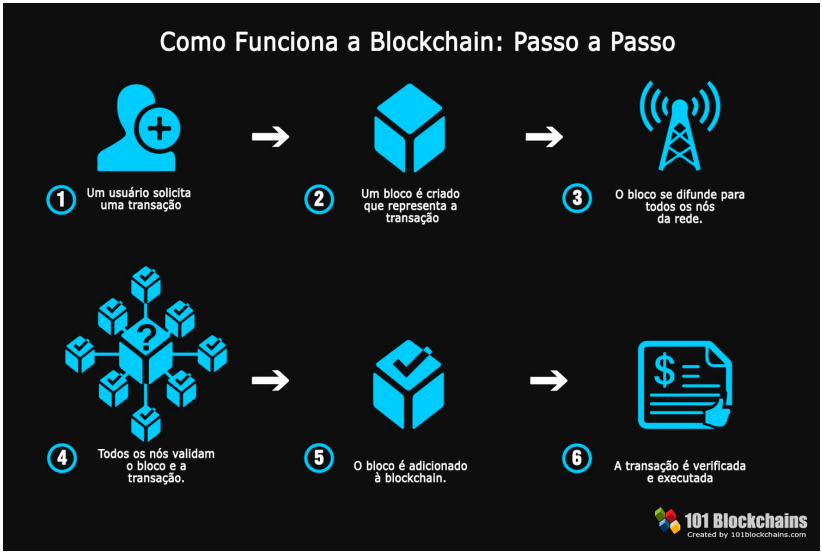


Figura 2 – Diagrama apresentando como ocorre uma transação na *Blockchain.*(LAMOUNIER, 2018)

Um Bitcoin não existe sem uma carteira, por isso que quando ele é criado ele é enviado ao minerador que finalizou o bloco. Ele existe como uma quantidade em algum endereço específico, por isso quando alguém envia moedas, sempre existe a referência de onde ela está saindo e para onde ela vai. Isto impede que moedas sejam duplicadas, pois as moedas são movidas apenas se a pessoa possui os fundos necessários e está enviando as moedas para outra carteira válida que vai recebê-las. Como existe o consenso da rede, quando um Bitcoin é movido de uma carteira para outra, não existe mais a possibilidade de reverter a operação, já que as moedas transferidas começam a pertencer a outra carteira, com sua própria chave privada.

2.3 ETHEREUM

Idealizado como um tipo de "computador mundial distribuído", o Ethereum é uma *Blockchain* pública de código aberto e uma plataforma

para computação distribuída que permite a criação de *Smart Contracts* (contratos inteligentes) com Turing-completude (BUTERIN, 2018).

Proposto inicialmente em 2013 em um *"white paper"* que teve Vitalik Buterin como autor, a ideia era disponibilizar uma plataforma que incrementaria o protocolo de *Blockchain* para armazenar e executar programas de computador através de uma rede de nodos distribuída internacionalmente. Em maio de 2015 foi lançada uma versão de teste junto com um anúncio solicitando que desenvolvedores testassem várias características da tecnologia em troca de uma recompensa de 25 mil ETH (abreviação da moeda que seria futuramente criada). Após alguns de testes, a rede oficial entrou em operação em 20 de julho de 2015 com a mineração do seu bloco gênese. Desde então a moeda vem crescendo em popularidade chegando na sua cotação máxima histórica de $1432,88 dólares em 13 de janeiro de 2018. Hoje, 23 de setembro, ela está na segunda colocação com uma capitalização de mercado de aproximadamente 22 bilhões de dólares e está cotada na faixa dos 200 dólares. Para méritos de comparação, hoje (23 de setembro) a moeda que possui maior capitalização de mercado é o Bitcoin (BTC), 170 bilhões de dólares e a terceira é a Ripple (XRP) com 13 bilhões (COINMARKETCAP, 2019).

Ocorreram alguns problemas pelo do caminho, como falhas na tecnologia que foram sendo ajustadas ao longo de sua história, sendo que a mais relevante delas foi um "Hard Fork" (quando um *Blockchain* é dividido em 2 distintos) criado para consertar um hack que havia desviado indevidamente 50 milhões de dólares em fundos do DAO (decentralized autonomous organization), fundo descentralizado que estava capitalizado em aproximadamente 150 milhões de dólares. Isso fez com que houvesse divergência entre os mineradores da rede da Ethereum e ela foi dividida em duas, a ETH (Ethereum) que arrumou o erro que permitia o hack e devolveu o dinheiro aos fundos e a ETC (Ethereum Classic) que manteve a falha e continuou com os fundos desviados indevidamente. Hoje a ETC está cotada na faixa dos 6 dólares, deixando claro que a rede que ficou com maior suporte foi a ETH.

O propósito da rede computacional distribuída da Ethereum é criar um novo ecossistema que permita o desenvolvimento de sistemas que até então não eram possíveis. Aplicativos distribuídos ("Dapps") agora possuem uma grande rede distribuída para operarem e permanecerem online. Até o momento já foram criadas soluções que tokenizam ativos, corretoras descentralizadas, sistemas de reputação em *Blockchain*, jogos de azar P2P, organizações autônomas descentralizadas (DAO), *Smart Contracts* e muitas outras coisas que ainda não

podem ser previstas.

2.4 *SMART CONTRACTS*

O termo Smart Contract (contrato inteligente) foi cunhado pelo cientista da computação norte americano Nick Szabo (CIEPLAK; LEEFATT, 2017). Embora em sua concepção inicial o termo estivesse mais ligado ao conceito clássico de contratos, hoje, com o surgimento da *Blockchain* o termo remete a um significado muito mais amplo, abrangendo qualquer aplicação onde sua computação acontece de maneira distribuída em uma *Blockchain*. Esta interpretação do termo começou ficar mais popular à medida que criptomoedas como a Ethereum foram ganhando espaço, já que uma de suas funcionalidades mais mencionadas é justamente a possibilidade de criação de contratos inteligentes. Eles são o principal recurso da moeda e funcionam como uma espécie de programa autoexecutável que facilita a troca de qualquer coisa de valor pela rede, de maneira irreversível na *Blockchain*. Eles são executados quando as condições especificadas no contrato são atendidas, não podem ser modificados ou influenciados por terceiros pois estão na *Blockchain* e estarão sempre ativos, desde que a rede da Ethereum continue operando.

A plataforma onde os contratos inteligentes são executados é chamada de EVM (Ethereum Virtual Machine). Ela é um programa com Turing-completude que roda na rede da Ethereum e permite que qualquer pessoa rode um programa escrito em qualquer linguagem na *Blockchain* da Ethereum. Isso cria um ambiente onde vários tipos de aplicativos descentralizados são viabilizados. Inclusive, foi desenvolvida uma linguagem de programação orientada a objeto especialmente direcionada para o desenvolvimento de *Smart Contracts* que rodam na EVM, o nome dela é Solidity. Ela é a linguagem primária do Ethereum e de várias outras *Blockchain*s privadas que estendem o código aberto da Ethereum e até chegam a competir com ela. Entre as principais linguagens influenciadoras estão JavaScript, C++, Phython e PowerShell. Embora o Solidity seja linguagem de programação principal utilizada no desenvolvimento de aplicativos descentralizados, também está sendo desenvolvida e em estágio de testes, uma nova linguagem de programação chamada de Vyper. O foco dela é criar uma maneira mais simples , segura e auditável para o desenvolvimento de contratos inteligentes, já que existem algumas deficiências referentes à complexidade desnecessária em alguns casos com o Solidity.

Na prática os contratos inteligentes são apenas programas de computador que são processados na máquina virtual da Ethereum (EVM). A execução deles ocorre sempre de maneira isolada, um contrato rodando dentro da EVM não tem acesso ao sistema de arquivos, rede ou quaisquer outros processos sendo executados no computador que hospeda a máquina virtual. Geralmente os contratos são escritos em linguagens mais alto nível e então são compilados para o bytecode EVM.

O processamento dos programas hospedados na *Blockchain* da Ethereum envolve custos, por isso eles precisam de "gas" (combustível). Ele é derivado do Ether, dinheiro utilizado na plataforma da Ethereum e serve para pagar por transações e processamento computacional através da rede. O Ether pode ser adquirido de várias formas, negociando ETH em corretoras em troca de outras moedas, através de doações ou minerando. Falando em mineração, é relevante mencionar que no caso da Ethereum os seus mineradores, além de completar e processar os blocos de transação, eles também auxiliam nos processamentos envolvidos nas aplicações, sendo recompensados pelo combustível pago pelo desenvolvedor para aplicação rodar.

A mecânica por trás do "gas" garante que qualquer contrato hospedado na rede tenha um custo para ser executado, isso é necessário pois o processamento envolvido será realizado por terceiros, que precisam ter alguma recompensa para ceder o poder computacional. Isso também aumenta o cuidado que os desenvolvedores precisam ter quando hospedam um contrato na rede, já que, caso mal redigido, o programa pode custar muito mais que o necessário para ser processado. De certa forma, pode-se dizer que isso também cria uma defesa contra loopings infinitos e coisas do tipo, já que se isso ocorrer, uma hora o programa fica sem "combustível" e tem processamento interrompido.

Todo contrato possui uma quantidade de combustível disponível e um valor de custo que especifica o quanto o usuário está disposto a pagar pelo processamento. Se o preço pago pelo gas for muito baixo, existe a possibilidade do programa nunca ser executado, já que os mineradores vão sempre dar prioridade para quem estiver disposto a pagar mais caro pelo seu processamento. O preço gas é definido de acordo com a demanda de recursos da rede, quanto maior o processamento necessário ou maior a necessidade de rapidez de processamento, maior será o custo.

3 ESTUDOS DE CASO

Nesta seção serão apresentados estudos de caso de algumas soluções de mercado que aplicam a monetização de jogos.

3.1 POKERSTARS

Poker hoje é o jogo mais monetizado do mundo, por padrão, o jogo não ocorre sem algum tipo de aposta, seja de dinheiro real ou fictício. As maiores empresas do ramo online são hoje PokerStars, Party Poker e 888poker (em ordem decrescente quanto ao tamanho de mercado) e o mercado vem crescendo bastante ano a ano. Para este estudo será discutido o exemplo da PokerStars (www.pokerstars.com), mas é importante ressaltar que todo o ecossistema do poker funciona de maneira idêntica.

A empresa desenvolve e hospeda seus próprios jogos, que consistem em vários formatos de jogo e diferentes variantes do poker como Texas Hold'em e Omaha. Não existe a possibilidade de terceiros hospedarem novos jogos em suas plataformas e toda a gestão do dinheiro envolvida e pagamentos das premiações é feita e garantida por seus respectivos sistemas. A monetização do sistema ocorre através de taxas cobradas dos usuários por utilizarem seus serviços. Por exemplo, um torneio com a entrada 10 dólares, custa na verdade 11 dólares, onde esse 1 dólar a mais é a taxa de 10% é retida pelo site e os 10 restantes são acumulados na premiação.

Figura 3 – Tela inicial da PokerStars na área de torneios de poker. (Grande variedade de opções)

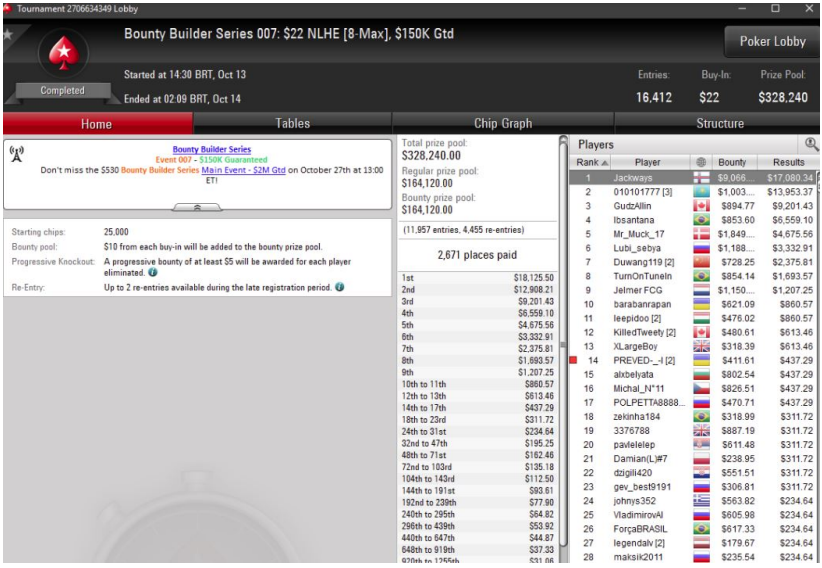


Figura 4 – Tela de um torneio de poker com 16.412 participantes, 22 dólares de entrada.(Muitos jogadores participando)

Figura 5 – Uma das mesas de torneio contendo 9 participantes. (exemplo de como ocorre a competição)

Last Updated: October 31, 2019 at 12:17 pm GMT

Poker Site	Online	Cash	24 H Peak	7 Day avg	Last Week	Play Now
PokerStars	55540	5689	9623	6000		Play Now

https://www.pokerscout.com/reviews/pokerstars/

Figura 6 – Número de usuários online no site no dia 31 de outubro de 2019.

Enquanto a indústria de jogos mundial cresceu 10% em 2018, a televisiva teve uma queda de 8% (SINGH, 2019). Isso é um forte indício que a população mundial está procurando formas de entretenimento mais interativas. Esse é um dos motivos do poker estar popularizando bastante na região asiática, em especial na Índia, onde vem perdendo a conotação de "jogo de azar" e começa a ser visto como um jogo de habilidade. Quando comparado a outros esportes, poker também possui certa vantagem, já que não precisa de muita estrutura física para que possa ocorrer. Isso fica ainda mais expressivo quando analisamos o

poker online, onde qualquer pessoa com um celular já consegue ter acesso ao recurso.

3.2 WORLDWINNER

O Worldwinner (www.worldwinner.com) é um site que permite a monetização de jogos para um único jogador. São jogos como "Paciência" onde cada jogador joga pelo navegador sozinho e de acordo com a maneira dele jogar, acumula pontos, seja por usar menos cliques, cometer menos erros, comprar menos cartas etc. No final da partida, são comparadas as pontuações atingidas pelos 2 jogadores que estão competindo entre si, quem teve a maior pontuação é o campeão e recebe o dinheiro da entrada decrescido das taxas envolvidas.

Figura 7 – Tela inicial do site.

Existem duas modalidades de competição, uma que envolve créditos fictícios e outra que envolve dinheiro real e não existe nenhuma maneira de transformar esses créditos em dinheiro real.

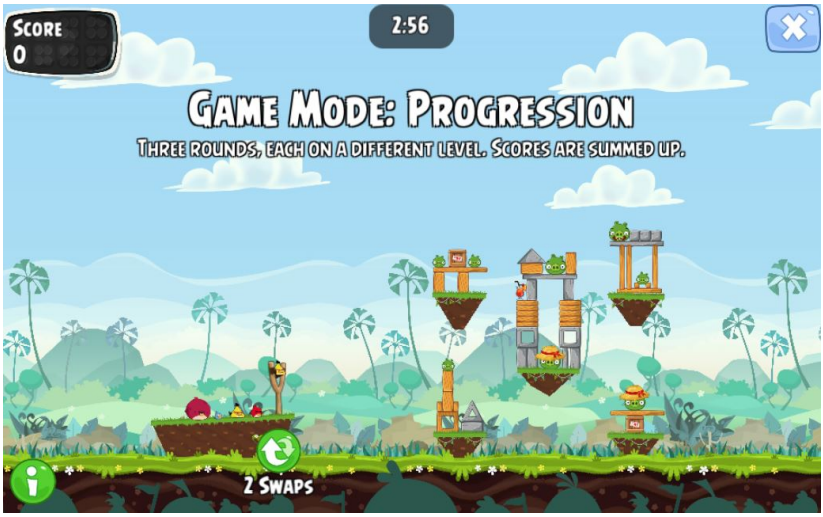


Figura 8 – Tela de um dos jogos disponíveis para competição. (exemplo de como ocorre a competição)

A interação é bem fraca, já que cada jogador não sabe de detalhes do desempenho do seu oponente durante a própria partida. Ele só fica sabendo se teve o melhor desempenho no final, quando as pontuações são comparadas. Não existem outras maneiras de competição onde mais de 2 jogadores possam participar, nem torneios. Também não foi identificado nenhum jogo multiplayer, que possua grande volume de jogadores, muito menos atletas virtuais de alto rendimento.

Figura 9 – Pontuação no final da partida.

Figura 10 – Momento onde o jogador descobre se ganhou. Neste caso estavam esperando algum oponente terminar seu jogo.

Pela proposta do site, é nítido o objetivo de criar um ambiente que viabiliza retorno financeiro através da disputa de algum jogo de

habilidade, mas a proposta de competitividade em si é muito fraca, não existindo nenhum tipo de ranking de desempenho ou interação real entre os usuários.

3.3 SITAGO

O Sitago (www.sitago.com) é um site que hospeda vários tipos de jogos básicos como sinuca, luta ou corrida. Todos eles são para apenas 2 jogadores e ao contrário do que ocorre no Worldwinner, em todos os jogos de sua plataforma o usuário consegue ver durante o andamento da partida se está indo melhor ou pior que seu oponente. Isso ocorre pois os jogadores de fato interagem durante o jogo, o que é um grande avanço quando comparado com a plataforma antes mencionada. No entanto, os jogos são muito simples, pouco comerciais, visualmente pobres e por isso são pouco atrativos para os jogadores. As taxas cobradas ficam na faixa dos 20%, muito acima do valor cobrado pelos concorrentes e a quantidade de jogadores também é bem baixa.

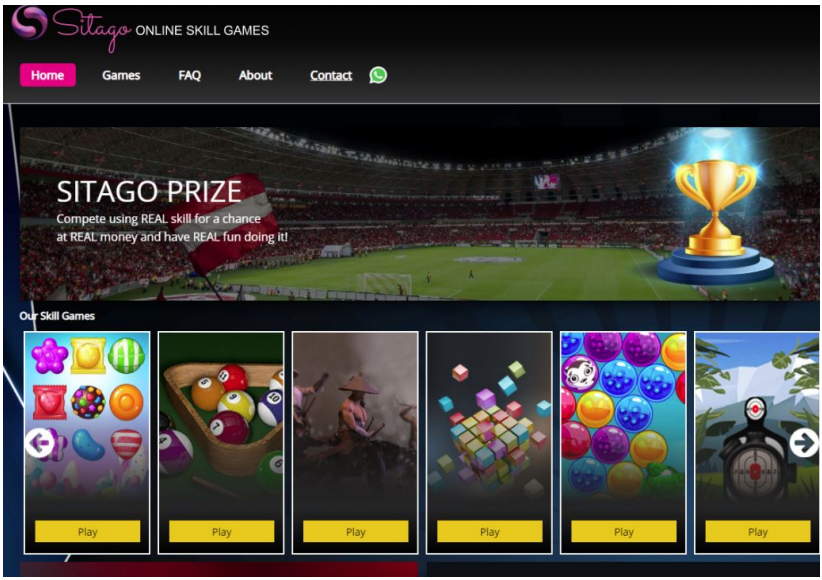


Figura 11 – Tela inicial do site.

Existem duas maneiras de interagir com o site, jogando a versão demonstração de cada jogo (o que permite que o usuário treine contra

uma inteligência artificial bem primitiva, sem poder escolher graus de dificuldade), ou competindo a dinheiro, onde cada jogador paga a entrada e quem ganhar a disputa recebe o prêmio. Não existe nenhum tipo de jogo para mais de dois jogadores, também nenhum formato que envolva alguma possibilidade de torneios.

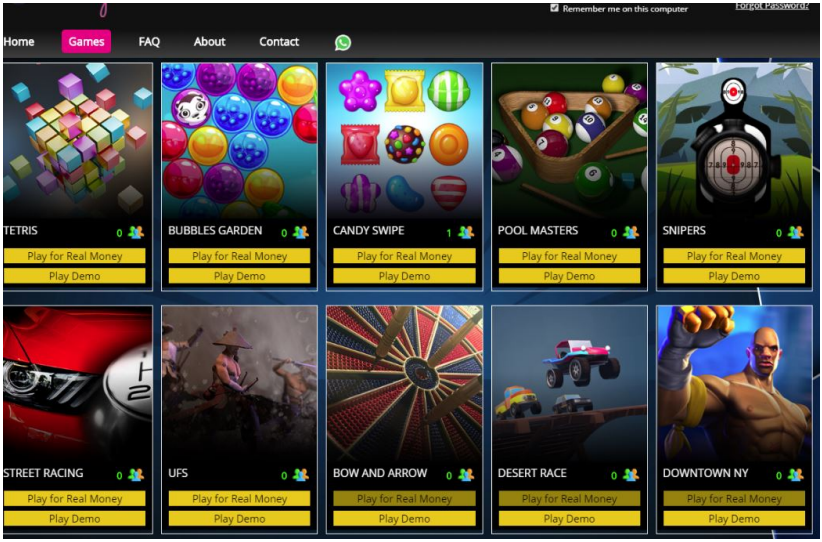


Figura 12 – Jogos disponíveis para competição.

Todos os jogos são desenvolvidos pela própria plataforma, então não possuem muitos jogadores, muito menos atletas de alto desempenho. Existe um sistema para que jogadores se comuniquem, mas aparentemente o público alvo do site são jogadores ocasionais que tenham vontade de competir por dinheiro usando sua habilidade em algum jogo pouco complexo.

Figura 13 – Partida de sinuca contra a inteligência artificial.

Figura 14 – Tela de registro para a disputa valendo dinheiro.

3.4 VISÃO GERAL

Visto que todas as soluções apresentadas possuem um modelo que permite a monetização da competição em algum tipo de jogo de habilidade, é importante comentar que em nenhuma delas há a possibilidade de criar interações mais complexas com o ecossistema, já que não são disponibilizados serviços externos para desenvolvedores. No caso do poker, a solução é focada apenas em viabilizar pagamentos em um ambiente interno e não passa de um meio para um fim. Já no caso das plataformas de monetização de jogos mais genéricos, não existe a possibilidade de acesso para desenvolvedores participarem dela, muito menos facilidade de grandes desenvolvedoras de jogos com maior apelo comercial participarem ou acessarem ao serviço.

4 MODELAGEM

4.1 DESCRIÇÃO GERAL

Criar um serviço que permita a monetização de jogos virtuais usando a tecnologia de *Blockchain* e *Smart Contracts*. Ele vai ser desenvolvido com o objetivo de facilitar a integração para desenvolvedores interessados em monetizar seus jogos, sem precisarem desenvolver toda a estrutura necessária para operar jogos a dinheiro.

4.2 DESCRIÇÃO ESPECÍFICA

Desenvolver e publicar uma API que recebe dados no formato JSON (especificando os participantes envolvidos na competição), mantém a custódia dos fundos, cria um contrato inteligente numa *Blockchain* e realiza os pagamentos ao receber a colocação que cada competidor terminou.

4.2.1 O Serviço

A proposta do serviço é ter uma API que facilita a monetização de jogos para seus desenvolvedores de forma que eles não precisem criar toda a estrutura monetária de seus jogos, desde a parte de gestão de fundos dos usuários até a parte de custódia e pagamentos das competições.

4.2.2 A Moeda

Será criada uma versão da *Blockchain* do Ethereum utilizando parte do seu código que é aberto, para a implementação dos *smart contracts* necessários para a viabilização das operações. A moeda desta nova *Blockchain* também será utilizada para validação das operações e como combustível (pagamento) necessário para que as validações e processamento distribuído ocorram.

4.2.3 A Competição

O ambiente onde as competições irão acontecer é irrelevante, a obrigação de garantir as regras do jogo disputado e a correta classificação dos participantes são do desenvolvedor do jogo que irá conectar seu jogo com o serviço. A API será desenvolvida para competições em formato de torneio, onde suas características como quantidade de participantes, valor da entrada, taxa de inscrição, prêmios e participantes envolvidos serão especificadas pelo criador do torneio através de uma requisição no formato JSON. A API ficará aguardando requisições de torneios e fará a custódia dos fundos até o momento que receber o JSON que indica o final do torneio.

4.2.4 Os Participantes

Os participantes dos torneios serão identificados pelo endereço de suas carteiras. Estes endereços são justamente para onde serão enviadas as premiações no momento que desenvolvedor enviar o JSON contendo o resultado final do torneio e a estrutura de premiação.

4.2.5 A Premiação

O valor da premiação ficará preso ao contrato que especifica o torneio e participantes envolvidos nele. A premiação consiste em valores arrecadados com entradas pagas pelos participantes reduzidos da taxa de prestação de serviço de hospedagem do torneio (2%).

4.2.6 Os Pagamentos

Quando a API receber a mensagem no formato JSON indicando o final do torneio junto com a colocação dos jogadores envolvidos e valor da premiação referente a cada posição, o contrato será ativado e irá realizar todos os respectivos pagamentos. Os pagamentos envolvem jogadores que atingiram a faixa de premiação, desenvolvedor criador do torneio que recebe as taxas de inscrição (valor recomendado de 10%) e a API que também recebe sua parte das taxas (2%).

4.2.7 Exemplo

Então neste caso, um torneio com 10 pessoas, com valor de entrada 100 e 10 de taxa, o desenvolvedor recebe todos os valores, fica com 80 para si e envia as 1020 moedas que vão ficar custodiadas pela API até ela receber a indicação da finalização do torneio. Neste caso, a mensagem indicou que o pagamento será de 500 para o primeiro colocado, 300 para o segundo e 200 para o terceiro. Os respectivos valores serão enviados para as carteiras dos jogadores de acordo com suas colocações e a API irá ficar com 20 pelo serviço prestado.

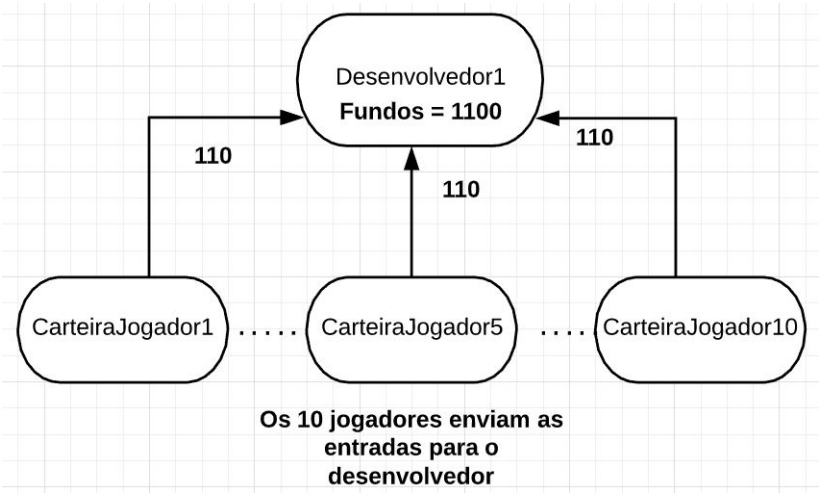


Figura 15 – O desenvolvedor recebe o valor de todas as entradas mais taxas de todos os jogadores participantes.

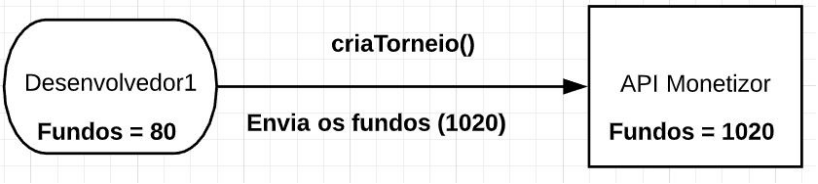


Figura 16 – O desenvolvedor inicia o torneio e envia os fundos junto com a taxa que vai ser necessária para o pagamento do serviço.

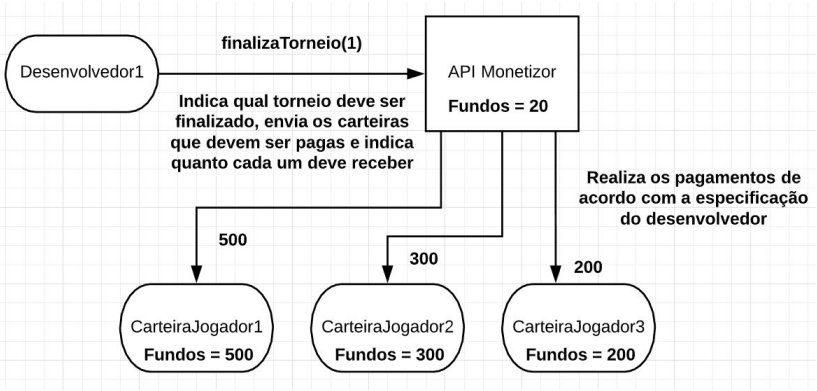


Figura 17 – O desenvolvedor envia uma mensagem indicando qual torneio deve ser finalizado, junto com a especificação de como os pagamentos devem ocorrer.

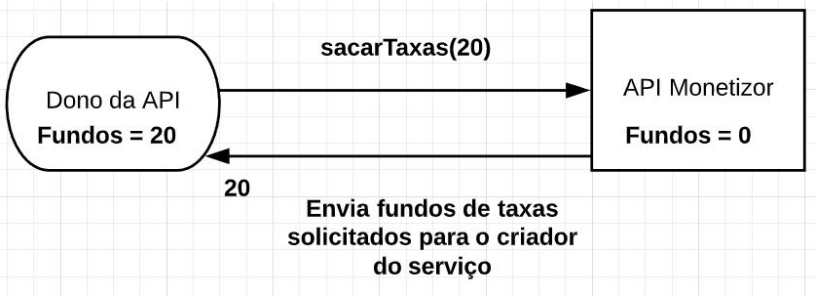


Figura 18 – O dono da API faz a solicitação de recebimento das taxas acumuladas sempre que achar necessário.

4.2.8 Requisitos Iniciais

1. Inicialização de torneios por parte do desenvolvedor.

2. Finalização de torneios por parte do desenvolvedor.

3. Pagamentos automatizados para os jogadores.

4.2.9 Métodos da API

Os métodos disponíveis da API e respectivos parâmetros são especificados nesta seção.

Método	Descrição
criarTorneio()	Inicia o torneio e recebe os fundos que vão ficar em custódia na API
finalizarTorneio()	Finaliza um torneio e indica como os pagamentos devem ser realizados
sacarTaxas()	Saca as taxas acumuladas na API

Figura 19 – Descritivo dos métodos que a API possui.

Métodos	Parâmetros
criarTorneio()	Identificador do Desenvolvedor, Valor envolvido do torneio, Identificador do serviço na Blockchain.
finalizarTorneio()	Identificador do Torneio, JSON especificando jogadores premiados e respectivos valores a serem pagos.
sacarTaxas()	Valor acumulado com taxas que deve ser sacado

Figura 20 – Descritivo dos parâmetros dos métodos.

5 DESENVOLVIMENTO E TESTES

Nesta seção estão relatados os momentos de desenvolvimento e detalhes das plataformas utilizadas no projeto. Houve a necessidade da configuração de dois nodos de uma *Blockchain* privada da Ethereum que permitissem o armazenamento e processamento de contratos inteligentes, conexão de múltiplas plataformas para interação com a *Blockchain* e a preparação do ambiente de desenvolvimento e testes.

Ambiente de Desenvolvimento	
Sistema Operacional	Windows 10 Enterprise, versão 1809
Processador	Intel Core i5-7600K 3,80GHz
Memória RAM	32 GB
Disco Rígido	2 TB

Figura 21 – Características do ambiente onde ocorreram o desenvolvimento e testes.

Programas, Recursos e Características	
Função	
Geth 1.9.6	Criação e interação com a Blockchain da Ethereum
Puppeth	Recurso do Geth para configuração e criação de uma Blockchain
Pré-mineração	Feita para que fundos estivessem disponíveis para realização de testes
2 Nós	Decisão de projeto para simular o processamento distribuído das transações na Blockchain
Remix	Plataforma para desenvolvimento e teste de contratos inteligentes
Solidity 0.5.12	Linguagem para desenvolvimento de contratos inteligentes

Figura 22 – Recursos e características do projeto.

5.1 CONFIGURAÇÃO DA *BLOCKCHAIN*

O primeiro passo necessário para o início do desenvolvimento do projeto era a configuração de uma *Blockchain* funcional, pois era através dela que tudo iria ocorrer, desde a criptomoeda que paga os jogadores até a plataforma que realiza os pagamentos. Foi selecionada a plataforma Geth (versão 1.9.6) para realizar a configuração. Ela é a implementação de um nodo da Ethereum e foi desenvolvida na linguagem GO (GETH, 2019). Através dela o usuário pode minerar Ether e desenvolver e testar programas que rodam na EVM. Durante a configuração, o usuário tem a opção de entrar em uma *Blockchain* já existente ou criar a sua própria. Para este estudo a opção escolhida foi a criar uma *Blockchain* privada. Após tentativas de configurar a *Blockchain* utilizando um JSON de configuração do bloco gênese da *Blockchain*, optou-se pelo uso de um recurso Geth chamado Puppeth que auxilia na criação e configuração de uma *Blockchain* da Ethereum. A ferramenta mostrou-se efetiva e facilitou esse procedimento inicial.

O Puppeth faz um passo-a-passo através de linha de comando onde o usuário pode definir quais as características desejadas na *Blockchain*. Para este projeto o modelo de consenso da rede selecionado foi "prova de trabalho" da Ethash e houve a seleção de uma opção que já adiciona fundos para alguns endereços de carteiras específicas, o que ajuda a acelerar o desenvolvimento, não havendo a necessidade de minerar com o computador as moedas iniciais.


```
{
  "config": {
    "chainId": 444,
    "homesteadBlock": 0,
    "eip150Block": 0,
    "eip150Hash": "0x0000000000000000000000000000000000000000000000000000000000000000",
    "eip155Block": 0,
    "eip158Block": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0,
    "petersburgBlock": 0,
    "ethash": {}
  },
  "nonce": "0x0",
  "timestamp": "0x5db7bcf6",
  "extraData": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "gaslimit": "0x47b760",
  "difficulty": "0x80000",
  "mixHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "coinbase": "0x0000000000000000000000000000000000000000",
  "alloc": {
    "0000000000000000000000000000000000000000": {
      "balance": "0x1"
```


Figura 23 – Detalhes do JSON de criação do bloco gênese.

Este foi o JSON do bloco gênese gerado no final do passo-a-passo. Pode-se notar a presença de características como a dificuldade de

mineração da rede, identificador da *Blockchain* e valor máximo possível gasto com "gas".

Antes da configuração da rede privada, também foram criadas as duas carteiras que posteriormente seriam pré capitalizadas. Elas foram geradas através do comando geth account new –datadir nodeA ou nodeB respectivamente.

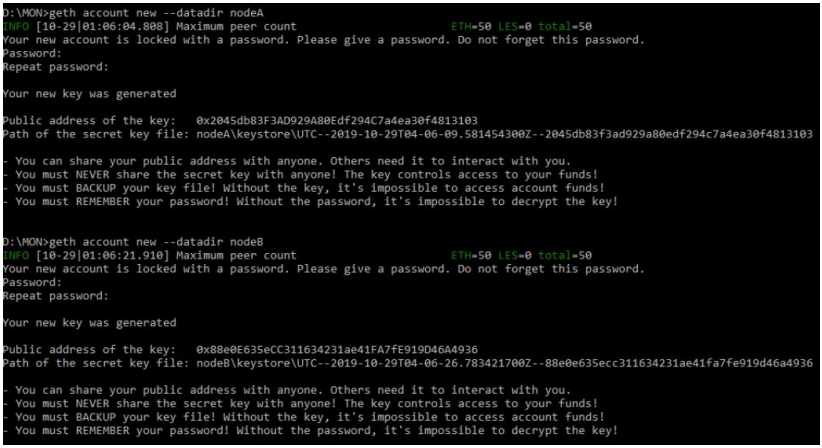


Figura 24 – Criação das carteiras em linha de comando.

Percebe-se que após a criação da carteira também são indicadas algumas recomendações de segurança para o usuário não perder acesso aos seus fundos nem ser roubado por alguém com más intenções.

Os 2 nós da *Blockchain* privada foram iniciados através dos respectivos comandos:

```
geth --datadir nodeA --networkid 444 --rpc --rpcport 8545 --port 8888  --rpcapi
"db,eth,net,web3,personal,miner,admin" --cache=128 --rpcaddr 0.0.0.0 --rpccorsdomain "*"
--allow-insecure-unlock console

geth --datadir nodeB --networkid 444 --rpc --rpcport 9545 --port 9999  --rpcapi
"db,eth,net,web3,personal,miner,admin" --cache=128 --rpcaddr 0.0.0.0 --rpccorsdomain "*"
--ipcdisable --allow-insecure-unlock console
```

Figura 25 – Comando de execução do Geth.

Tudo que vem escrito após "geth" são parâmetros que especificam

as configurações de execução do Geth. Ali estão especificadas características como a identificação da *Blockchain*, portas de comunicação, aspectos de segurança e a ativação de um console javaScript que ainda vai ser muito usado durante este projeto.

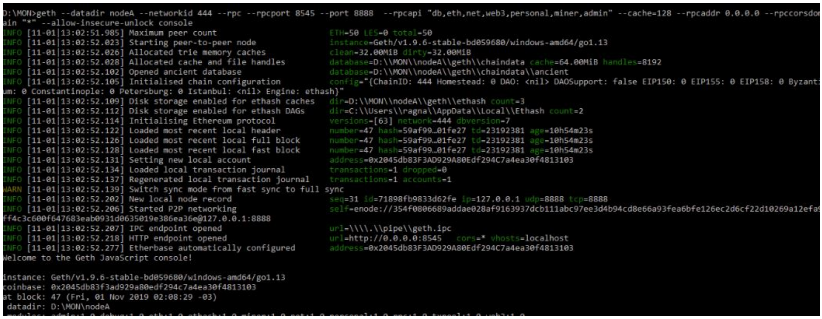


Figura 26 – Informações referentes ao nodeA após a execução dos comandos.

Os códigos de iniciação previamente mencionados foram abertos dois terminais diferentes, cada um referente a um nó distinto da mesma *Blockchain*. Através do comando "admin.nodeInfo.enode" conseguimos alguns identificadores globais que permitiram a conexão entre eles com o uso dos respectivos comandos:

NODE A

```
admin.addPeer("enode://7e25085da5dc812f2c343eeececf407a5fdf98ffa34
a9603f85ea83425f1a7b5d37ce47e2c1d6a8f1c888524eb7991dbca2faef34f
604630f70ace58a2602adc@189.4.82.51:9999?discport=1608")
```

NODE B

```
admin.addPeer("enode://354f0806689addae028af9163937dcb111abc97ee
3d4b94cd8e66a93fea6bfe126ec2d6cf22d10269a12efa9ff4c3c600f647683e
ab0931d0635019e386ea36e@127.0.0.1:8888")
```

Figura 27 – Comandos que realizam a conexão entre os nós.

À partir daí eles começaram a trabalhar como um único corpo, mesmo estando em diretórios diferentes eles ficavam sincronizados e

disputavam pela mineração dos blocos quando as funções de mineração de cada nó eram ativadas.

5.2 ESCOLHA DA LINGUAGEM DE PROGRAMAÇÃO

O Solidity é uma linguagem de programação orientada à objetos de alto nível que foi desenvolvida para a implementação de contratos inteligentes. Ela foi inicialmente proposta em 2014 por Gavin Wood e teve seu desenvolvimento feito pelo "Solidity Team" do projeto Ethereum com a liderança do Christian Reitwiessner. Sua semântica foi influenciada por linguagens como C++, Python, JavaScript e foi projetada para rodar na EVM (Ethereum Virtual Machine) após ser compilada para bytecode. Ela é estaticamente tipada, suporta herança e o uso de bibliotecas (SOLIDITY, 2019).

Por ser uma linguagem bastante nova, ela é atualizada com muita frequência, em atualizações que muitas vezes não são retrocompatíveis. Embora exista uma quantidade extensiva de documentação sobre como a linguagem funciona, existem poucos livros que abordam ela e boa parte dos que estão disponíveis no mercado estão ou desatualizados ou não se aprofundam muito na linguagem em si, o que dificulta o aprendizado para novos programadores.

5.3 CONFIGURAÇÃO DAS PLATAFORMAS DE DESENVOLVIMENTO

Com a *Blockchain* configurada foi a hora de prosseguir para o desenvolvimento dos primeiros *Smart Contracts*. Para isso foi utilizado o Remix, ele é uma plataforma WEB com uma série de ferramentas direcionadas para a interação com *Blockchain*s da Ethereum, sejam elas locais ou públicas (REMIX, 2019). Ela tem recursos para o desenvolvimento, compilação, testes genéricos, unitários e correção de bugs em contratos inteligentes. Suporta tanto o Solidity quanto o Vyper (já citados na seção 2.4), as duas principais linguagens para desenvolvimento de aplicativos descentralizados (ASHWIN, 2019). Para o desenvolvimento do projeto foi selecionado o Solidity (versão 0.5.12), que já está em operação a mais tempo, é mais estável e possui um mais documentação. Ainda sim foi desafiador realizar o desenvolvimento com a linguagem, já que muitos recursos comuns em linguagens comerciais como conversão tipo de variável, manipulação de estruturas e recebimento de chamadas externas de método com parâmetros enviados no

formato JSON ou arrays ainda não estão disponíveis.

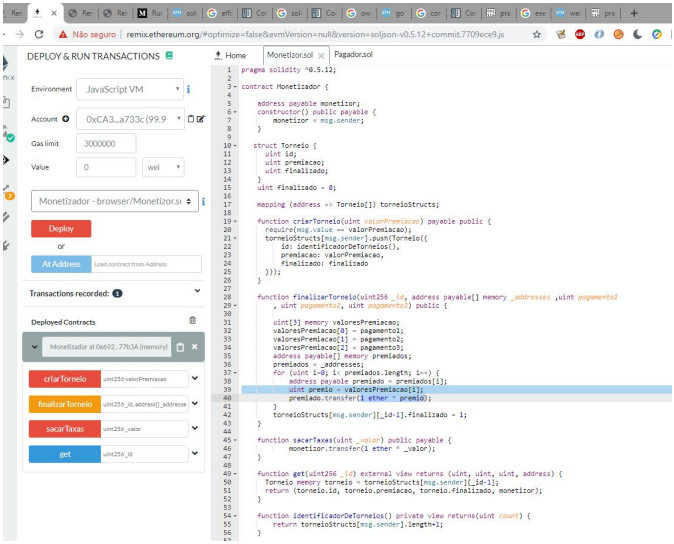


Figura 28 – Visualização da plataforma REMIX.

Outra plataforma que também teve que ser configurada para o desenvolvimento foi a MetaMask. Ela também é WEB e serve como uma carteira de criptomoedas, permitindo que o usuário possa utilizar várias contas, fazer transferências e acessar aplicativos descentralizados (dApps) (METAMASK, 2019). Sua principal função foi justamente fazer a transferência de moedas entre contas, para não haver a necessidade de realizar a tarefa por linha de comando.

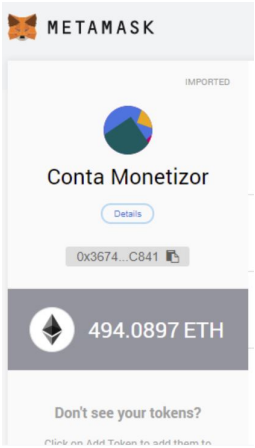


Figura 29 – Visualização da plataforma MetaMask com fundos disponíveis em uma das carteiras da *Blockchain* criada.

Também houve tentativas de configuração da Ethereum Wallet e Mist, plataformas que possuem recursos similares ao MetaMask (ETHWALLET, 2019), mas não houve sucesso.

5.4 DESENVOLVIMENTO

Este projeto foi a primeira experiência com o desenvolvimento real para *Blockchain* e contratos inteligentes. No início houve bastante dificuldade em compreender o Solidity e entender a melhor forma de trabalhar com as escassas ferramentas disponíveis. Embora exista documentação, informação em outras fontes eram difíceis de ser encontradas. Para auxiliar a familiarização com o novo ecossistema o desenvolvimento foi dividido em problemas menores e menos complexos.

5.4.1 Criação e Utilização de Um Contrato Ingeligente

A primeira tentativa foi focada em fazer um contrato inteligente interagir com a *Blockchain* e ser processado pela EVM. Para isso foi desenvolvido um pequeno programa que continha uma informação que seria gravada na *Blockchain* e poderia ser editada ou lida. Aí já ficou nítido como era a relação entre os contratos, o Ether, o "gas" (combustí-

vel), como eles interagiam entre si e como os mineradores são pagos por cederem o seu poder computacional para a rede. É bastante interessante ver o programa rodando de maneira distribuída, vê-lo executado em pequenos lotes e recebendo os retornos ao fim da mineração de cada bloco. Neste primeiro teste as interações com o contrato ocorriam através de botões que são criados para a execução dos contratos no Remix. Com eles é possível enviar parâmetros e chamar métodos. Sempre que um método é ativado aparece uma nova transação na *Blockchain* junto com o gasto de combustível e outras informações como o endereço do contrato.

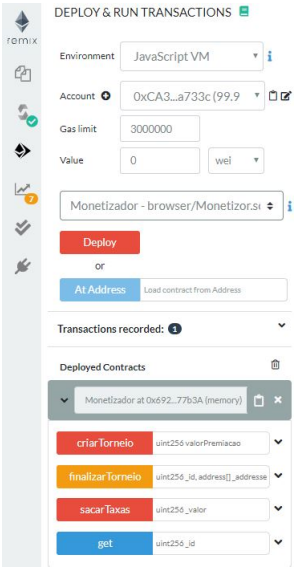


Figura 30 – Contrato sendo ativado e manipulado através da interface gráfica do REMIX. Na parte inferior da imagem estão os botões de ativação de funcionalidades do contrato.

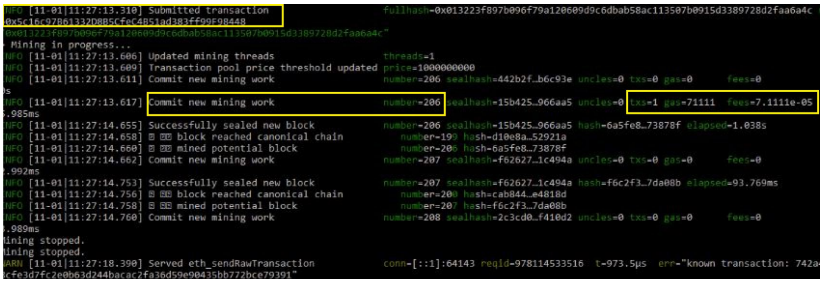


Figura 31 – Linha de comando do Geth apresentando o endereço da carteira emissora da transação na *Blockchain* e indicando novos blocos minerados pelos nós e taxas envolvidas com o processamento das transações (gas).

5.4.2 Interação com Contratos Inteligentes Através de Linha de Comando

O segundo passo foi conseguir interagir com o contrato de maneira mais complexa e através de código pelo console do Geth. Neste estágio o foco foi conseguir enviar moedas para os contratos e realizar chamada de métodos enviando alguns parâmetros em Array ou JSON, tarefa que se mostrou inviável devido a ausência de bibliotecas para manipulação de variáveis, tanto simples quanto complexas. Algumas bibliotecas até foram encontradas, mas eram de outras versões de compilador mais antigas e incompatíveis com a versão mais atual que estava sendo usada. A parte de envio de fundos foi mais fácil e foram identificadas várias maneiras de capitalizar os contratos, desde envios durante a construção, até envios posteriores.

5.4.3 Manipulação de Recursos com o Solidity

O terceiro desafio foi utilizar os recursos enviados dentro dos contratos usando o Solidity, desde o envio dos fundos contidos no contrato para terceiros, até o encapsulamento das informações, procurando maneiras de garantir que os dados de um usuário do sistema não influenciassem nos dados do outro utilizando as poucas ferramentas disponíveis na linguagem. Os testes foram começando a ficar mais complexos e exigiam muitos passos para serem praticados, tomando muito tempo sempre que um novo teste fosse necessário. Para resolver isso foram

criados alguns Scripts.

A primeira tarefa excessivamente repetitiva era a parte de mineração, sempre que uma transação era enviada era necessário iniciar a mineração para que o processamento ocorresse na EVM, nesse momento que surge o primeiro script de mineração. Ele foi nomeado como "mining.js" e sempre ativava a mineração que surgia uma nova transação na *Blockchain*. Assim que o processamento era finalizado ele interrompia a mineração, que utilizava muito o processamento da CPU do computador, atrapalhando o uso e desenvolvimento da API.

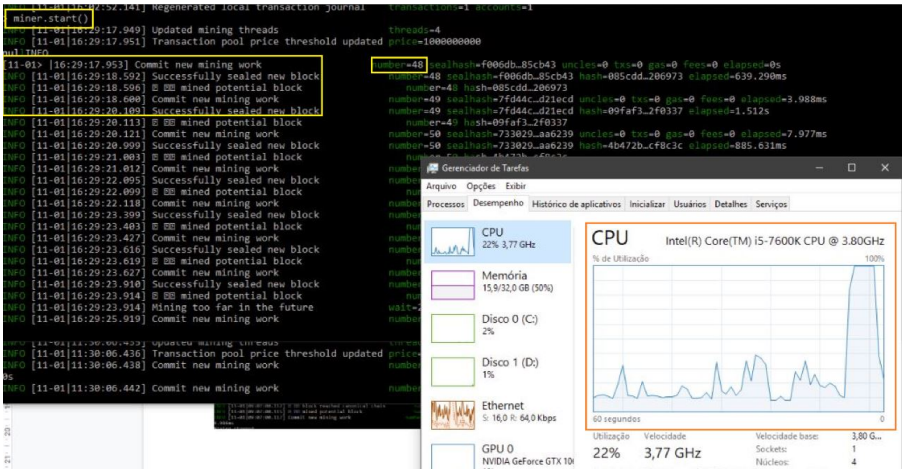


Figura 32 – Visualização da mineração sendo ativada em um dos nós através do comando "miner.start()"fazendo uso do processador ir a 100% da capacidade.

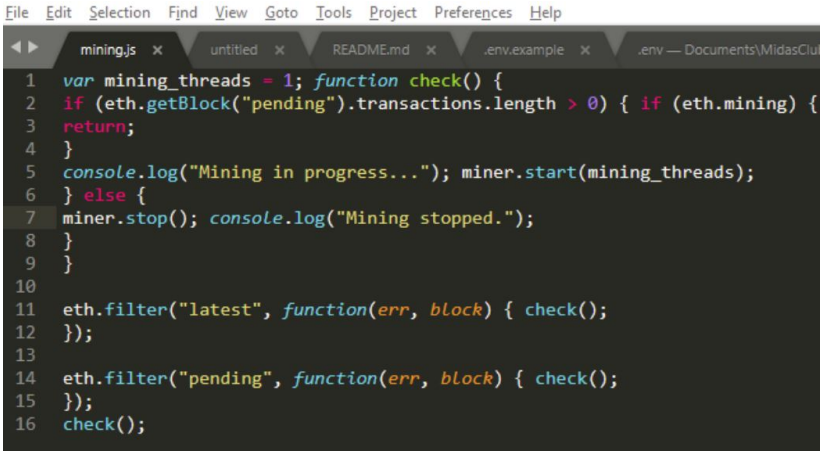


Figura 33 – Script mining.js.

É um código bem simples que especifica a quantidade de treads do processador que devem ser usadas na mineração (no caso apenas uma para não travar outros processos do computador) e fica constantemente checando se existe alguma transação pendente.

O próximo script foi nomeado apiMonetizor.js pois ele englobava boa parte das tarefas repetitivas necessárias para a execução de alguns métodos. Eles foram agrupados de acordo com suas funções e dependências. A primeira parte do código "destrava" algumas contas chave para que elas possam ser operadas livremente por linha de comando e em seguida ocorre a criação do contrato enviando todos os parâmetros necessários para que ele possa ser hospedado na *Blockchain*. A segunda parte envolve a função de criação de torneios que exigia uma sequência de passos muito grande para funcionar toda vez que era requisitada. Outro problema era que a linha de comando do windows 10 que foi utilizada no trabalho só permitia comandos de uma linha, todas as transações mais complexas precisavam ter todas as quebras de linha removidas antes de serem executadas, o que dificultava também a leitura e desenvolvimento.

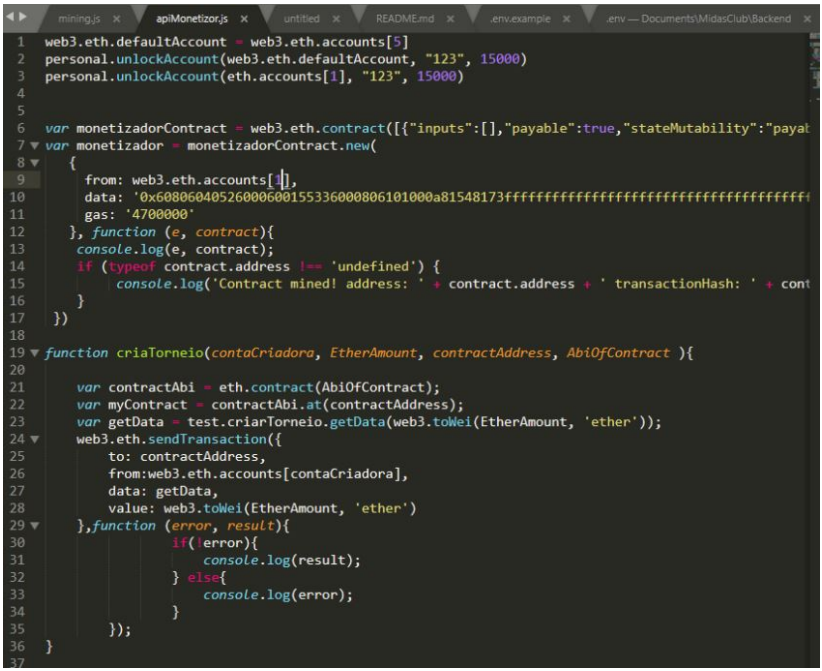


Figura 34 – Código do apiMonetizor.js

Quando o contrato é colocado na rede, ele fica aguardando ter alguma transação direcionada para ele. Quando isso ocorre, ele é ativado e ocorre o processamento. Isso nos leva à parte principal deste projeto que é o contrato inteligente que permite a operação das transações.

5.4.4 Contrato Inteligente

O Smart Contract é composto por 6 métodos principais (se considerarmos também o construtor) e uma estrutura de dados para armazenamento de algumas informações dos torneios.

No topo está a versão que o código escrito em Solidity será compilada seguida do nome do contrato, no nosso caso Monetizador. Ele tem um construtor do tipo *"payable"*, que permite que o contrato possa receber fundos durante sua instanciação. Dentro do construtor também já está sendo gravado o endereço da carteira que é dona deste contrato, é a mesma que vai receber as solicitações de saques das taxas pela hospedagem do serviço através do método "sacarTaxas".

Figura 35

```
pragma solidity ^0.5.12;

contract Monetizador {

    address payable monetizor;
    constructor() public payable {
        monetizor = msg.sender;
    }

   struct Torneio {
     uint id;
     uint premiacao;
     uint finalizado;
    }
    uint finalizado = 0;

    mapping (address => Torneio[]) torneioStructs;

    function criarTorneio(uint valorPremiacao) payable public {
      require(msg.value == valorPremiacao);
      torneioStructs[msg.sender].push(Torneio({
          id: identificadorDeTorneios(),
          premiacao: valorPremiacao,
          finalizado: finalizado
      }));
    }

    function finalizarTorneio(uint256 _id, address payable[] memory _addresses ,uint pagamento1
        , uint pagamento2, uint pagamento3) public {

        uint[3] memory valoresPremiacao;
        valoresPremiacao[0] = pagamento1;
        valoresPremiacao[1] = pagamento2;
        valoresPremiacao[2] = pagamento3;
        address payable[] memory premiados;
        premiados = _addresses;
        for (uint i=0; i< premiados.length; i++) {
            address payable premiado = premiados[i];
            uint premio = valoresPremiacao[i];
            premiado.transfer(1 ether * premio);
        }
        torneioStructs[msg.sender][_id-1].finalizado = 1;
    }

    function sacarTaxas(uint _valor) public payable {
            monetizor.transfer(1 ether * _valor);
    }

    function get(uint256 _id) external view returns (uint, uint, uint, address) {
      Torneio memory torneio = torneioStructs[msg.sender][_id-1];
      return (torneio.id, torneio.premiacao, torneio.finalizado, monetizor);
    }

    function identificadorDeTorneios() private view returns(uint count) {
        return torneioStructs[msg.sender].length+1;
    }

}
```

A estrutura que permite o armazenamento e leitura das informações dos torneios relacionados com o contrato é a “struct Torneio”, ela é composta pelo identificador do torneio, valor da premiação e um indicador que indica se ele já foi finalizado ou não. Ela fica armazenada em um array destas estruturas que está mapeada pelo endereço da carteira do usuário que utilizar o serviço de criação de torneios, isso

garante também a segurança dos dados de cada usuário, já que somente o dono da conta que solicita os torneios pode ter acesso e apenas aos dados ligados à própria conta. Vale comentar também que o contador de índice do torneio começa do 1 para todo mundo, sem que haja nenhuma confusão ou mistura de informação. Foi tomada a decisão de não armazenar os jogadores envolvidos nos torneios pois cada endereço de carteira de jogador é composto por um hexadecimal bem grande e começa a ficar caro hospedar toda essa informação na *Blockchain*. É necessário lembrar que qualquer processamento ou armazenamento de informação custa dinheiro quando estamos falando de aplicações distribuídas da rede Ethereum.

O método criarTorneio(valorPremiacao) também estende *"payable"* e ele exige que o desenvolvedor que solicitou a criação do torneio tenha enviado a mesma quantidade indicada no "valorPremiação" em fundos. Após a checagem ele cria a struct com os dados do torneio e adiciona ele no final do array de torneios daquele endereço. Os fundos ficam armazenados no endereço do contrato e são posteriormente manipulados pelos métodos, finalizarTorneio e sacarTaxas. O método sacarTaxas é bem simples e apenas envia o valor solicitado para o endereço criador do Monetizador. Já o método "finalizarTorneios" possui algumas peculiaridades. O único parâmetro do Solidity que aceita o recebimento de um array é o do tipo "address" por isso que foi o único que veio desta maneira. Todos os outros teriam que vir como inteiro ou como texto, sendo que não existe nenhum método para tratamento de string como "explode" ou "split" que quebram um string em arrays. Por não haverem esses métodos de tratamento e todas as soluções encontradas envolverem muitos passos, foram utilizados 3 parâmetros para indicar a distribuição dos prêmios. A ordem da premiação fica sendo a mesma ordem do array de endereços onde o "pagamento1" é o valor destinado ao primeiro lugar, "pagamento2" para o segundo e assim sucessivamente. Uma peculiaridade interessante é o envio do pagamento ter que ser daquela maneira específica "1 ether * premio", isso ocorre pois os valores especificados dentro dos contratos, quando falamos de pagamentos estão na unidade "Wei" que é a menor quantidade possível de divisão do Ether. 1 (um) Ether consiste em 1 seguido de 18 zeros de Wei. No fim o método edita o campo finalizado para 1 do torneio, indicando a finalização do mesmo.

O método "get" é uma view e permite a leitura de dados de algum torneio por parte do seu criador, sem a cobrança de nenhuma taxa e o método "IdentificadorDeTorneios" apenas indica quantos torneios já foram criados por aquele usuário.

5.4.5 Dificuldades e Soluções

Durante o desenvolvimento do projeto os conhecimentos técnicos sobre a tecnologia foram sendo aprofundados e foi percebido que alguns aspectos da modelagem inicial da API estavam longe do ideal e precisavam ser adaptados. Agora vão ser indicados os principais motivos causadores das modificações realizadas, junto com a alternativa escolhida.

O primeiro motivo é o fato do Smart Contract ser hospedado e executado de maneira distribuída na *Blockchain* da Ethereum. Só isso já permite chamadas criptografadas de qualquer lugar com acesso à internet, então criar uma API enviando mensagens em JSON pareceu bastante redundante. Isso nos leva ao segundo motivo. O Solidity era muito limitado e possuía pouquíssimos recursos para manipulação de variáveis, nem métodos comuns em outras linguagens como os que mudam a tipagem de variáveis estavam disponíveis, muito menos as que permitiam manipulação de Arrays ou JSON. Por isso foi adotada uma modelagem com menos mensagens em JSON e utilizando mais parâmetros simples, que a linguagem operava melhor. A ideia das mensagens de retorno também não se mostrou muito viável, já que enviar mensagens custa dinheiro. Por isso foi utilizado um único método que indica a situação geral do torneio, sem envolver custos para a leitura. Isso é possível pois funções de "view" do Solidity apenas fazem a leitura do estado de alguma característica do contrato e não necessitam de processamentos, por isso são operadas de graça.

Durante o desenvolvimento do projeto foram realizados vários testes para melhor compreensão da tecnologia e checagem de erros, não foram utilizados testes unitários, apenas aqueles Scprits previamente mencionados. Na próxima seção será apresentado um grande teste de validação que engloba tudo que foi desenvolvido.

5.5 PROVA DE CONCEITO

Foi feito um teste de uso que passa tudo que foi abordado neste projeto, seja de maneira direta ou indireta. Todo o processo de criação da *Blockchain* e ativação de Geth já foi previamente apresentado, então vamos avançar direto para a parte onde o contrato é utilizado por um desenvolvedor. Neste exemplo a quantidade de jogadores total envolvida no torneio não é importante, pois para o serviço só é necessário saber quanto dinheiro está envolvido na premiação e os jogadores que

foram premiados.

Primeiro foi levantado o saldo de todos os atores envolvidos nos testes. A conta número 5 é do desenvolvedor que vai utilizar os serviços de hospedagem de torneios. A número 1 é a conta que hospedou a API do Monetizor e as contas de 2 até 4 são de jogadores participantes dos torneios.

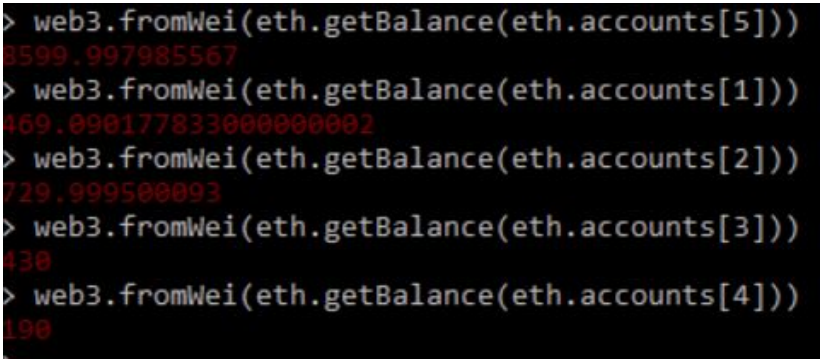


Figura 36 – Saldo inicial de todas as carteiras envolvidas nos testes.

Pode-se notar que o método que chama o balanço da carteira é englobado por outro "fromWei", ele é necessário pois, caso não fosse usado, todos os números listados seriam números de pelo menos 18 dígitos.

O script apiMonetizor.js é ativado criando o serviço de hospedagem de torneios pelo endereço da conta 1 (como apresentado anteriormente no descritor do Script).

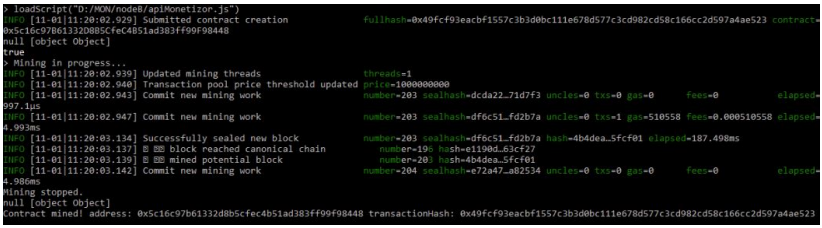


Figura 37 – Na primeira linha o Script é carregado iniciando a operação.

Logo que o contrato é submetido, o script mining.js (que já havia sido carregado anteriormente) inicia a mineração. O contrato pode ser identificado pelo endereço nos cantos esquerdos, tanto superiores

quanto inferiores. Na imagem é visível que apenas 1 thread está envolvida com a mineração. O bloco inicial sendo minerado é o 203 e inclusive aparece a quantidade de gas envolvida junto com o valor das taxas na mineração deste bloco.

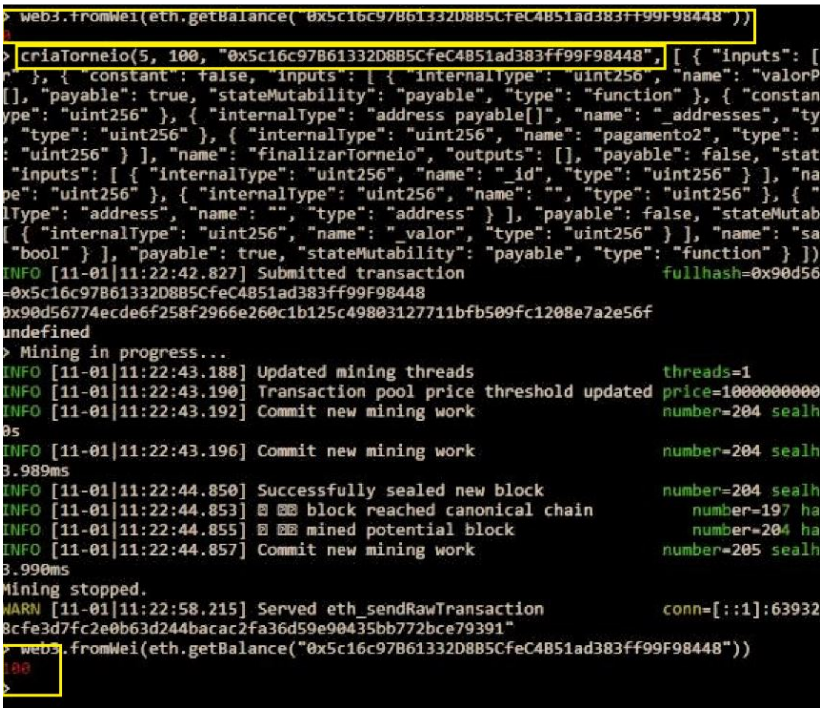


Figura 38 – Torneio sendo criado via linha de comando.

Na primeira linha da imagem checamos os fundos existentes no endereço do contrato criado pelo script apiMonetizor.js, que estão zerados, já que a função de criação do serviço utilizada não estava enviando nenhum valor. Em seguida é utilizada a função "criaTorneio" do mesmo script, enviando todos os parâmetros necessários e sem quebras de linha. Podemos ver que o criador do torneio é o endereço 5 e que o valor do torneio é de 100 Ethers. Quando a transação é submetida o minerador identifica que existem pendências e inicia a mineração novamente, realizando novamente o processamento da transação (vamos ignorar esse passo daqui para frente pois ele sempre ocorre da mesma maneira). No final da imagem, no canto inferior esquerdo, é checada a

quantidade de fundos do contrato, que agora é 100.

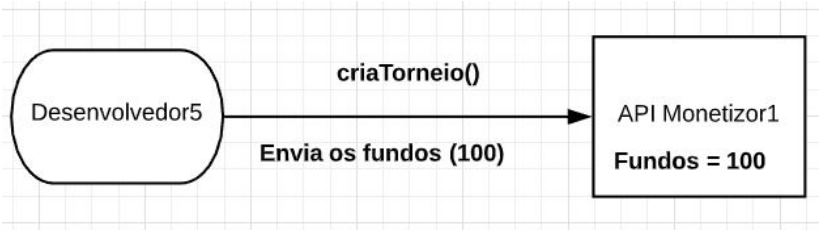


Figura 39 – Representação visual do que ocorreu.

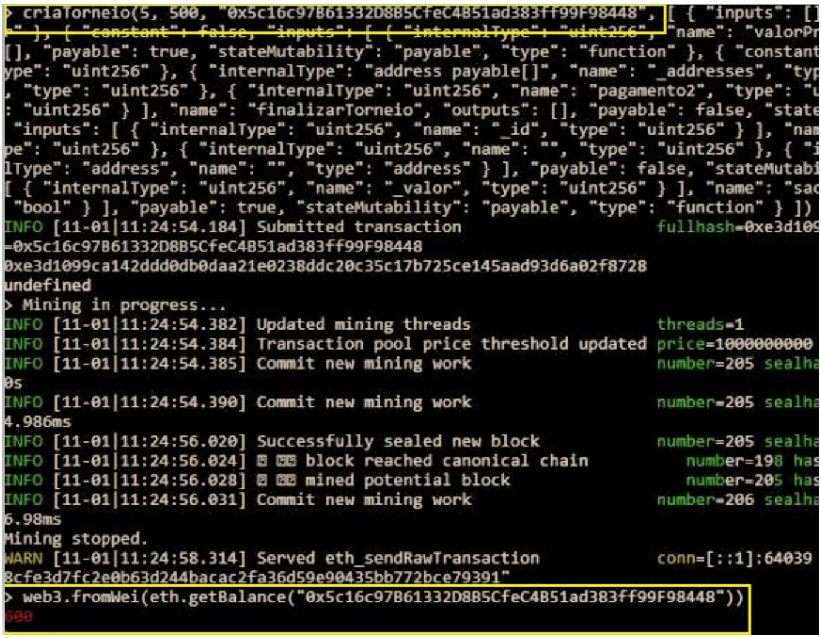


Figura 40 – Segundo torneio sendo criado.

Agora o método criaTorneio é ativado novamente pelo endereço 5, criando outro torneio com valor de 500 Ethers. No fim podemos ver que o saldo do contrato está em 600 agora.

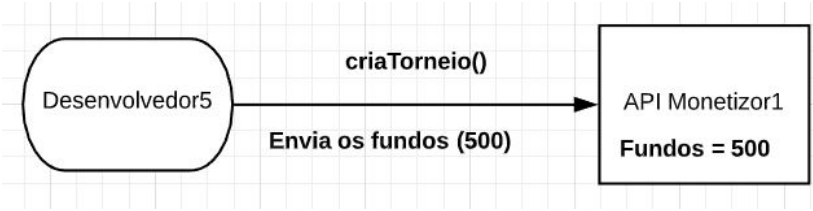


Figura 41 – Representação visual do que ocorreu.

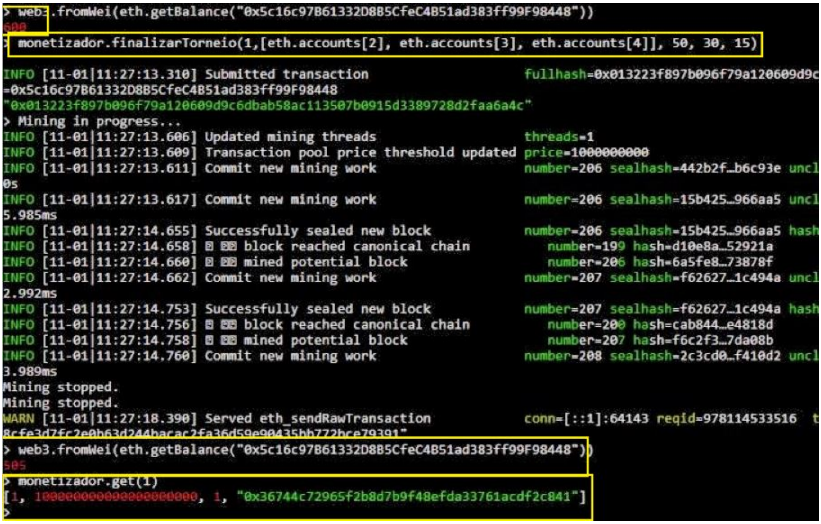


Figura 42 – Torneio 1 sendo finalizado.

Nesta imagem é solicitada a finalização do torneio com o identificador número 1 (o que tinha 100 de fundos). Ao checarmos o valor do contrato no final da imagem vemos que ainda existem 505 Ethers ligados a ele. Também é executado o método get para checagem dos dados do torneio número 1. Repare que, por ser uma view, o método não necessitou de nenhum processamento. Ali também está o saldo do contrato em Wei (com todos os zeros)! O terceiro parâmetro retornado é referente ao status de finalizado, seguido do endereço que está hospedando o Monetizor.

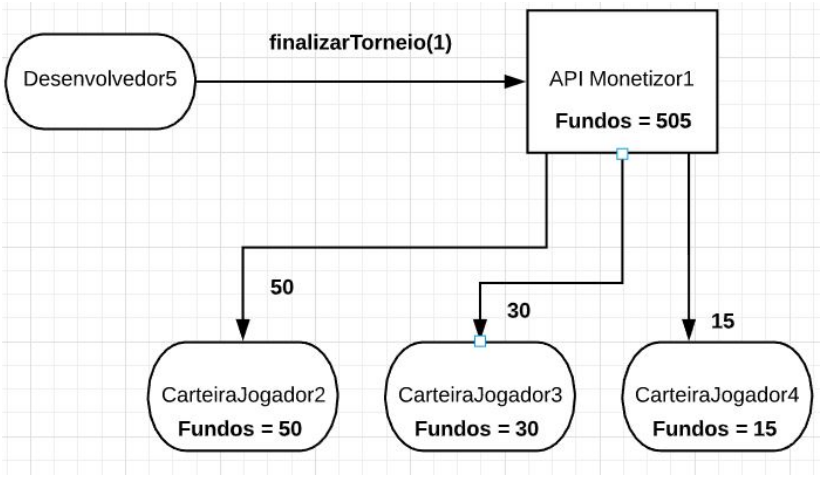


Figura 43 – Representação visual do que ocorreu.

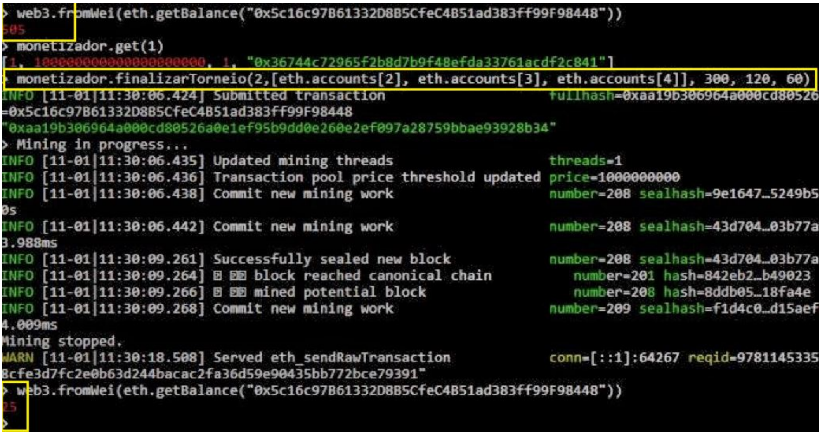


Figura 44 – Comando de finalização do torneio 2.

Agora o segundo torneio é finalizado, os pagamentos são realizados restando 25 Ethers no contrato.

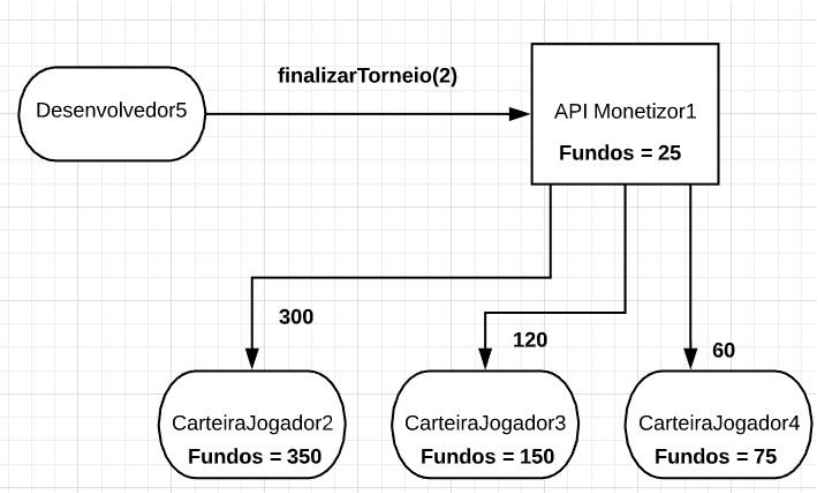


Figura 45 – Representação visual do que ocorreu.

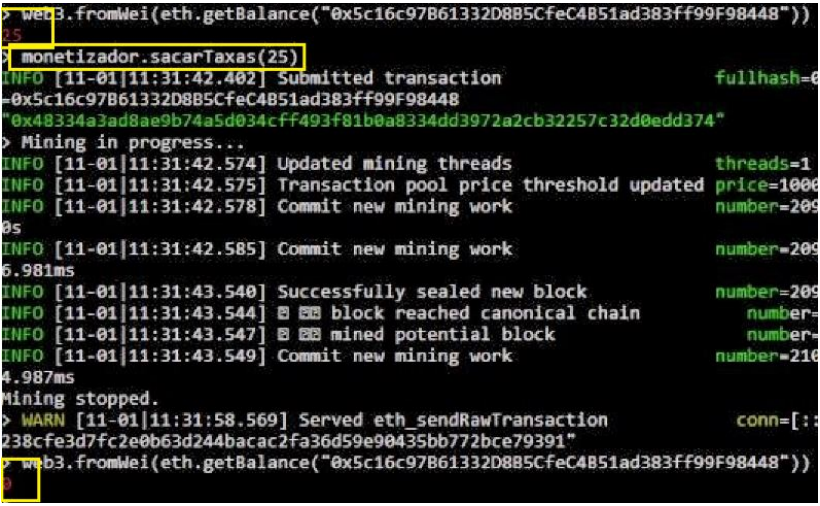


Figura 46 – Saque de taxas efetuado.

Agora é solicitado o saque das taxas, finalmente zerando o valor disponível no contrato.

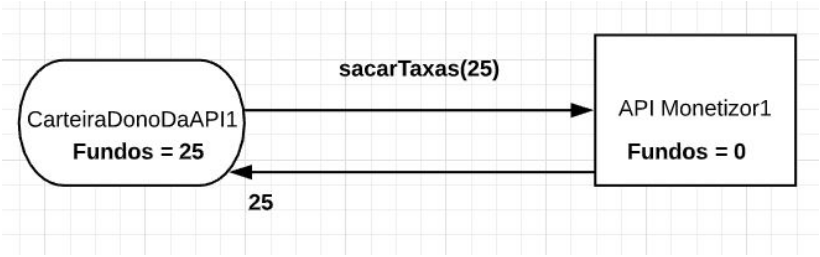


Figura 47 – Representação visual do que ocorreu.

```
> eth.accounts
["0x88e0e635ecc311634231ae41fa7fe919d46a4936", "0x36744c
eecd30ec1a8f4e7a982ef8fb286a77", "0xdd0ef1bd179f1679a2c0
> web3.fromWei(eth.getBalance(eth.accounts[5]))
7999.997652409
> web3.fromWei(eth.getBalance(eth.accounts[1]))
494.089667275000000002
> web3.fromWei(eth.getBalance(eth.accounts[2]))
1079.999500093
> web3.fromWei(eth.getBalance(eth.accounts[3]))
580
> web3.fromWei(eth.getBalance(eth.accounts[4]))
265
>
```

Figura 48 – Balanço das contas no final dos testes.

Agora podemos verificar que as contas envolvidas nas transações tiveram seus fundos modificados de acordo com o que deveria ser cada pagamento. A conta 5 que hospedou 600 Ethers em torneios por exemplo, está com 600 a menos (menos taxas envolvidas). A conta 2 ,que era a do jogador que ficou na primeira colocação em todos os casos, subiu bastante também e a conta 1, que ficava com as taxas menores também subiu de acordo.

Agora será demonstrado como a plataforma MetaMask também está conectada com a *Blockchain*.

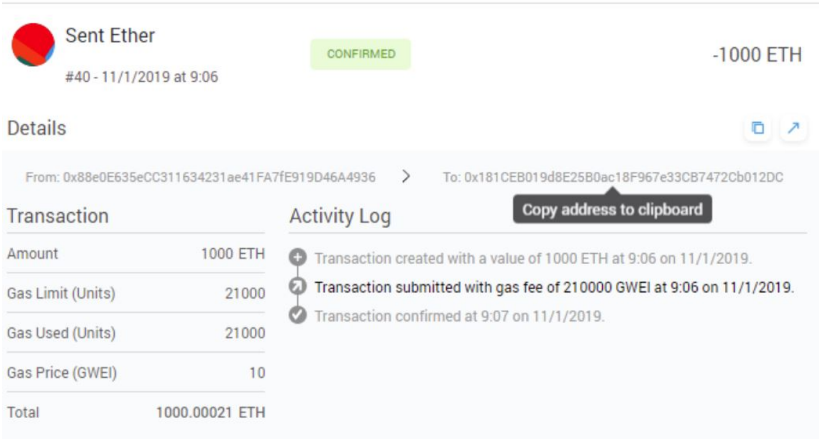


Figura 49 – Transação realizada para a carteira especificada no canto superior direito.

É solicitado, através da plataforma o envio de 1000 Ethers para outra carteira. Note a hora que a solicitação ocorreu, 9:06 e que ela foi confirmada 9:07. Repare também nas taxas envolvidas e qual a carteira destino.

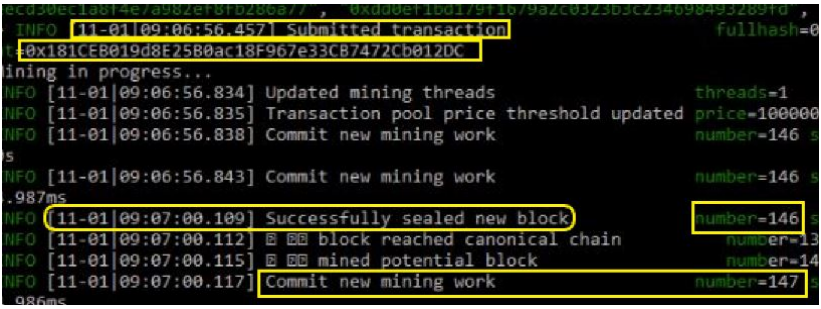


Figura 50 – Registro temporal da transação.

Agora vemos no console do Geth a transação para a mesma carteira sendo enviada 9:06:56 e sendo confirmada 9:07:00 quando o bloco 146 foi minerado. Pode-se notar também que a carteira receptora da transação é a mesma da imagem anterior.

5.6 CONCLUSÕES

O desenvolvimento desse projeto envolvendo *Blockchain* e *Smart Contracts* foi o primeiro contato mais baixo nível com o assunto. Antes havia uma noção conceitual sobre a tecnologia e possibilidades que ela viabiliza, mas ter esse contato foi muito gratificante. Acredito que entender tudo estava acontecendo foi um pouco mais fácil mim, por já ter lido bastante sobre o tópico.

Fiquei muito satisfeito com o produto final deste estudo pois ele atinge todos os objetivos esperados. A solução funciona de maneira distribuída por estar hospedada em múltiplos nós da *Blockchain* o que aumenta a disponibilidade do serviço. O código do contrato também não tem como ser alterado já que ele está hospedado na *Blockchain*, não havendo a possibilidade de existirem modificações na forma do serviço operar, garantindo para o desenvolvedor usuário a confiabilidade necessária para operar. A única forma de criar uma nova versão do serviço seria criando um novo contrato, que no final das contas acaba sendo um serviço novo, com um novo endereço, sem nenhuma ligação com o antigo. O serviço antigo vai continuar operando indefinidamente e o desenvolvedor só vai precisar modificar sua integração com a API se sentir a necessidade de usar algum dos novos recursos disponíveis. Além disso existe a garantia para o desenvolvedor de que o serviço vai fazer exatamente o que se propõe, pois o código fonte que gera o contrato pode ser comparado com a versão compilada gravada na *Blockchain*.

O dinheiro movimentado ser uma versão de criptomoeda também resolve alguns problemas de segurança que poderiam existir. Já que os fundos ficam na carteira de cada usuário, não há a necessidade do desenvolvedor precisar criar toda a estrutura necessária para gerenciar o dinheiro deles, nem correr o risco de ficar com a custódia dos fundos dos seus usuários. É menos risco para o usuário e para o desenvolvedor.

O fato de um smart contract ser utilizado na prestação do serviço acabou criando algumas camadas de segurança muito relevantes que inicialmente não haviam sido identificadas. Já que os fundos envolvidos na competição ficam sob custódia do contrato, algum tipo de ataque ao desenvolvedor não coloca em risco o dinheiro dos competidores. Além disso, os fundos da competição ficam muito pouco tempo sob custódia do desenvolvedor, já que ele recebe o dinheiro dos competidores, tira as suas taxas e envia logo em seguida para o serviço, fazendo com que haja uma janela muito pequena para que algum ataque ocorra. Pela maneira que o contrato foi programado, apenas a carteira que iniciou um torneio pode finalizar ele, então somente o dono da chave privada

poderá interagir com os torneios em aberto, impedindo que alguém mal intencionado possa fazer algo.

Embora existam algumas limitações do Solidity que acarretaram em modificações no projeto, os benefícios de usar a *Blockchain* e contratos inteligentes para a prestação desse serviço são tantos justificam a complexidade envolvida na mudança de paradigma e no desenvolvimento da solução. Pretendo estender a tecnologia criada neste projeto para também receber e realizar pagamentos em outras criptomoedas como o Bitcoin, além de desenvolver mais recursos para o serviço como outros formatos de competição e outros serviços utilizando a API.

Eu pessoalmente considero a *Blockchain* uma tecnologia tão ou mais disruptiva que a internet. Acredito que ainda vamos ver a mudança das reservas de valor mundiais para alguma alternativa em cripto, que governos vão precisar mudar, pois cada vez terão menos poder sobre a vida das pessoas. Tudo isso graças aos novos aplicativos descentralizados que virão, diminuindo a quantidade de intermediários das operações, diminuindo taxas e aumentando a segurança, já que o código é incorruptível. Hoje temos uma oportunidade única de poder conhecer, estudar, empreender e investir em algo que é tão revolucionário e ainda está só começando. O futuro é agora!

REFERĘNCIAS

AIR. Air applied innovation review. *Berkeley*, v. 2, n. 3, p. 19, 2016.

ASHWIN, C. *Ethereum ditching Solidity for Vyper?* 2019. Disponível em: <https://geth.ethereum.org/>. Acesso em: 22/10/2019.

BLENKINSOP, C. *GTracking Bitcoin Transactions, Explained*. 2019. Disponível em: <https://cointelegraph.com/explained/tracking-bitcoin-transactions-explained>. Acesso em: 31/10/2019.

BUTERIN, V. *A Next Generation Smart Contract Decentralized Application Platform*. 2018. Disponível em: <https://whitepaperdatabase.com/ethereum-eth-whitepaper/>. Acesso em: 17/07/2018.

CAPSL, C. *History of Esports: From PC to Mobile Tournaments*. 2019. Disponível em: <http://capsl.cc/history-of-esports/>. Acesso em: 14/05/2019.

CHEN, L. Y. *Jack Ma Embraces Blockchain for Ant But Warns of Bitcoin Bubble*. 2018. Disponível em: <https://www.bloomberg.com/news/articles/2018-06-25/jack-ma-embraces-blockchain-for-ant-but-warns-of-bitcoin-bubble>. Acesso em: 25/06/2018.

CHOHAN, M. U. W. A history of bitcoin. *Discussion Paper Series: Notes on the 21st Century*, v. 1, p. 10, 2017.

CIEPLAK, J.; LEEFATT, S. Smart contracts: A smart way to automate performance. *1 GEO. L. TECH. REV. 417*, v. 1, n. 1, p. 11, 2017.

COINMARKETCAP, c. 2019. Disponível em: <https://coinmarketcap.com/pt-br/currencies/ethereum/historical-data/>. Acesso em: 11/09/2019.

ETHWALLET, E. *Ethereum Wallet and Mist Beta 0.11.1*. 2019. Disponível em: <https://github.com/ethereum/mist/releases>. Acesso em: 29/09/2019.

GETH. 2019. Disponível em: <https://geth.ethereum.org/>. Acesso em: 20/10/2019.

GREENSTEIN, S. *The Paradox of Technological Déjà Vu*. 2018. Disponível em: <https://www.computer.org/csdl/mags/mi/2018/01/mmi2018010118.pdf>. Acesso em: 06/05/2018.

HERNANDEZ, K. *Blockchain for Development – Hope or Hype?* 2017. Disponível em: <https://opendocs.ids.ac.uk/opendocs/handle/123456789/12945>. Acesso em: 25/04/2018.

KOSBA, A. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. *IEEE*, v. 1, p. 20, 2016.

LAMOUNIER, L. *O Guia Definitivo da Tecnologia Blockchain: Uma Revolução Para Mudar o Mundo*. 2018. Disponível em: <https://101blockchains.com/pt/tecnologia-blockchain-guia/>. Acesso em: 19/07/2019.

METAMASK, M. *Brings Ethereum to your browser*. 2019. Disponível em: <https://metamask.io/>. Acesso em: 25/09/2019.

NAKAMOTO, S. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008. Disponível em: <https://bitcoin.org/bitcoin.pdf/>. Acesso em: 15/04/2018.

NIKOLAEV, K. *Bitcoin Is 'The Most Extraordinary Bubble of Our Generation' Says Trader*. 2019. Disponível em: <https://www.ccn.com/bitcoin-is-the-most-extraordinary-bubble-of-our-generation-says-trader/>. Acesso em: 18/09/2019.

ODEM, . *Program Staking Token Architecture*. 2018. Disponível em: <https://odem.io/wp-content/uploads/2019/01/ODEM.IO-Technical-Whitepaper.pdf>. Acesso em: 17/07/2018.

POPOV, S. *The Tangle*. 2018. Disponível em: <https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae336
12/06/2019.

PRADO, J. *O que é blockchain? - indo além do bitcoin*. 2017. Disponível em: <https://tecnoblog.net/227293/como-funciona-blockchain-bitcoin/>. Acesso em: 20/07/2019.

PRIZETRACKER. 2019. Disponível em: <https://dota2.prizetrac.kr/international2019>. Acesso em: 20/08/2019.

RATHS, M. *Government Harasses Bitcoins Because It Can't Tax Them*. 2013. Disponível em: <https://www.atr.org/government-harasses-bitcoins-cant-tax-them-a7710>. Acesso em: 19/04/2018.

REMIX, D. *Welcome to Remix documentation!* 2019. Disponível em: <https://remix-ide.readthedocs.io/en/latest/>. Acesso em: 29/09/2019.

SINGH, N. *5 Reasons to Invest in the Online Poker Gaming Industry*. 2019. Disponível em: <https://www.entrepreneur.com/article/326194y>. Acesso em: 20/07/2019.

SMITH, N. *Its not as awesome as people imagine: Esports players say 'dream job' is more than fun and games*. 2018. Disponível em: <https://www.washingtonpost.com/sports/2018/12/13/its-not-awesome-people-imagine-esports-players-say-dream-job-is-more-than-fun-games/>. Acesso em: 17/04/2019.

SOLIDITY, S. 2019. Disponível em: <https://solidity.readthedocs.io/en/v0.5.12/>. Acesso em: 11/10/2019.

TAPSCOTT, D. *How blockchains could change the world*. 2018. Disponível em: <https://www.mckinsey.com/industries/high-tech/our-insights/how-blockchains-could-change-the-world>. Acesso em: 17/07/2018.

TAYLOR, T. L. *Raising the Stakes: E-Sports and the Professionalization of Computer Gaming*. London, England: The MIT Press, 2012. 304 p.

TORPEY, K. *U.S. Lawmakers Are Realizing They Can't Ban Bitcoin*. 2019. Disponível em: <https://www.forbes.com/sites/ktorpey/2019/07/30/us-lawmakers-are-realizing-they-cant-ban-bitcoin>. Acesso em: 12/08/2019.

ULRICH, F. *Por que o bitcoin é a invenção mais importante desde a internet*. 2018. Disponível em: <https://www.youtube.com/watch?v=NB4FwKOY4Bk>. Acesso em: 17/07/2018.

WARBURG, B. *How the Blockchain Will Radically Transform the Economy*. 2018. Disponível em:

<https://www.ted.com/talks/bettina$_w arburg_h ow_t he_b lockchain_w ill_r adically_t ra
17/07/2018.

APÊNDICE A – Artigo

Monetizer: API de pagamentos automatizados utilizando *Blockchain* para *e-sports*

Luis Felipe Brasil Corrêa[1]

[1]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC)
– Florianópolis – SC – Brasil

luisfelipebrasil@gmail.com

Abstract. *The emergence of a new technology called Blockchain, a kind of distributed created ledger, has allowed the development of several new applications in many different niches. This paper aims to study and evaluate this technology in depth - both in technical aspects and in relation to its impact on society - and apply it to a specific domain problem. The solution was modeled on the tools and concepts that this technology offers to create a platform that makes automated payments in competitive virtual environments. New developers have a simple and secure way to monetize their platforms.*

Resumo. *O surgimento de uma nova tecnologia denominada Blockchain, uma espécie de registro criado de forma distribuída, vem permitindo o desenvolvimento de várias novas aplicações direcionadas a inúmeros nichos diferentes. Este trabalho tem como objetivo conhecer e avaliar esta tecnologia em profundidade - tanto em aspectos técnicos quanto em relação a seu impacto na sociedade - e aplicá-la em um problema de domínio específico. A solução foi modelada a partir das ferramentas e conceitos que esta tecnologia oferece para criar uma plataforma que realiza pagamentos automatizados em ambientes virtuais competitivos. Agora desenvolvedores possuem uma maneira simples e segura de monetizar suas plataformas.*

1. Introdução

Desde o começo da história da humanidade o ser humano interage através de trocas, seja de mercadorias, informações ou favores. Inicialmente as regras envolvidas nas trocas eram garantidas através da violência ou repercussões sociais, mas, com o avanço da sociedade, começaram a surgir formas mais complexas de controle [Warburg 2018]. Surgiram então os órgãos reguladores e garantidores da ordem como bancos e instituições governamentais. Com o surgimento da Internet, essas regulações começaram a ser feitas online, agilizando o processo mas ainda havendo a necessidade de muita burocracia envolvida, para que o intermediário tivesse segurança para realizar as transações. A questão principal é que até pouco tempo atrás, não havia uma maneira confiável para que duas partes que não sabem se podem confiar uma na outra pudessem realizar trocas sem envolver uma terceira parte que ambas confiassem. Com o surgimento da *Blockchain*, existe a possibilidade de diminuir os riscos envolvidos em transações sem a necessidade de uma instituição política ou bancária fazendo essa intermediação, apenas usando tecnologia.

O mercado de entretenimento mundial vem crescendo a passos largos, especialmente a parte que envolve jogos eletrônicos. Isso ocorre pois existe uma grande migração das mídias convencionais como o televisão aberta para mídias eletrônicas como plataformas de streaming e jogos eletrônicos [CAPSL 2019], que estão cada vez mais acessíveis para a população com a difusão da internet, computadores e smartphones mais baratos.

Muitos dos cyberatletas, mesmo participando da elite em seus respectivos jogos, não são capazes de produzir renda diretamente da própria habilidade. Equipes que participam dos principais campeonatos e conseguem terminar bem posicionados conseguem fazer isso, mas times ou jogadores que participam do circuito e não ficam tão bem posicionados, acabam precisando recorrer a fontes secundárias de renda como patrocínios e participação de propagandas por visualizações em plataformas de streaming [Smith 2018].

Este trabalho tem como objetivo identificar e testar formas de integrar conceitos de *Blockchain* e *smart contracts* ao ecossistema de *e-sports*. Traçando uma estratégia de negócio adequada e escolhendo uma solução tecnológica para o desenvolvimento de uma criptomoeda e um serviço que sejam compatíveis com o mercado.

2. *Blockchain*

A cadeia de blocos (Blockchain) funciona como um grande registro digital de transações que é distribuída, verificada e monitorada por múltiplas fontes de forma simultânea [Greenstein 2018]. Ela permite que o registro de coisas que hoje é feito através de instituições reguladoras como bancos e cartórios, seja feito de modo confiável, seguro e com menos margem para falsificação ou irregularidades. Uma outra característica interessante é a completa rastreabilidade desses registros, sejam eles referentes a uma transação monetária ou o registro de um carro ou imóvel por exemplo, garantindo coisas como: quem possui o quê, validando transações ou garantindo que alguma informação seja realmente verídica [Blenkinsop 2019].

A tecnologia surgiu inicialmente como uma solução descentralizada para um tipo de dinheiro virtual chamado Bitcoin e segundo [Nakamoto 2008] a ideia é que a moeda seja uma cadeia assinaturas eletrônicas. Cada dono de uma moeda faz a transferência assinando digitalmente o *hash* da transação anterior junto com a chave pública do próximo dono, adicionando elas ao final da moeda. Um beneficiário pode verificar as assinaturas checando a cadeia de propriedade. O registro das transações é feito de forma descentralizada em uma cadeia de blocos, sendo que a maior cadeia acaba sendo a cadeia correta, que é seguida por todos os nodos da rede.

Figura 1. Exemplo de como ocorre o encadeamento dos blocos na *Blockchain*.[Prado 2017]

3. Ethereum

Idealizado como um tipo de "computador mundial distribuído", o Ethereum é uma *Blockchain* pública de código aberto e uma plataforma para computação distribuída que permite a criação de *Smart Contracts* (contratos inteligentes) com Turing-completude [Buterin 2018]. O propósito da rede computacional distribuída da Ethereum é criar um novo ecossistema que permita o desenvolvimento de sistemas que até então não eram possíveis. Aplicativos distribuídos ("Dapps") agora possuem uma grande rede distribuída para operarem e permanecerem online. Até o momento já foram criadas soluções que tokenizam ativos, corretoras descentralizadas, sistemas de reputação em *Blockchain*, jogos de azar P2P, organizações autônomas descentralizadas (DAO), *Smart Contracts* e muitas outras coisas que ainda não podem ser previstas.

4. *Smart Contracts*

Contratos inteligentes são programas de computador que são processados na máquina virtual da Ethereum (EVM). A execução deles ocorre sempre de maneira isolada, um contrato rodando dentro da EVM não tem acesso ao sistema de arquivos, rede ou quaisquer outros processos sendo executados no computador que hospeda a máquina virtual. Geralmente os contratos são escritos em linguagens mais alto nível e então são compilados para o bytecode EVM.

O processamento dos programas hospedados na *Blockchain* da Ethereum envolve custos, por isso eles precisam de "gas" (combustível). Ele é derivado do Ether, dinheiro utilizado na plataforma da Ethereum e serve para pagar por transações e processamento computacional através da rede. Todo contrato possui uma quantidade de combustível disponível e um valor de custo que especifica o quanto o usuário está disposto a pagar pelo processamento. Se o preço pago pelo gas for muito baixo, existe a possibilidade do programa nunca ser executado, já que os mineradores vão sempre dar prioridade para quem estiver disposto a pagar mais caro pelo seu processamento. O preço gas é definido de acordo com a demanda de recursos da rede, quanto maior o processamento necessário ou maior a necessidade de rapidez de processamento, maior será o custo.

5. Modelagem

O serviço criado permite a monetização de jogos virtuais usando a tecnologia de *Blockchain* e *Smart Contracts*. Ele foi desenvolvido com o objetivo de facilitar a integração para desenvolvedores interessados em monetizar seus jogos, sem precisarem desenvolver toda a estrutura necessária para operar jogos a dinheiro. Suas principais características serão especificadas nessa sessão.

5.1. A Moeda

Foi criada uma versão da *Blockchain* do Ethereum utilizando parte do seu código que é aberto, para a implementação dos *smart contracts* necessários para a viabilização das operações. A moeda desta nova *Blockchain* também é utilizada para realizar os pagamentos, validação das operações e como combustível (pagamento) necessário para que as validações e processamento distribuído ocorram. Além disso é na blockchain dela que fica hospedado o serviço.

5.2. O Serviço

Recebe um comando de início de torneio por parte do desenvolvedor junto com os fundos envolvidos na competição. A custódia dos fundos fica no contrato até que o desenvolvedor envie uma requisição indicando a finalização do torneio e indicando os valores que cada jogador deve receber de acordo com sua colocação. Recebe os fundos envolvidos em competições e fica com a custódia deles até receber o sinal do desenvolvedor indicando a finalização.

Método	Descrição
criarTorneio()	Inicia o torneio e recebe os fundos que vão ficar em custódia na API
finalizarTorneio()	Finaliza um torneio e indica como os pagamentos devem ser realizados
sacarTaxas()	Saca as taxas acumuladas na API

Figura 2. Descritivo dos métodos que a API possui.

5.3. A Competição

O ambiente onde as competições acontecem é irrelevante, a obrigação de garantir as regras do jogo disputado e a correta classificação dos participantes são do desenvolvedor do jogo que irá conectar seu jogo com o serviço. A API foi desenvolvida para competições em formato de torneio, onde suas características como quantidade de participantes, valor da entrada, taxa de inscrição, prêmios e participantes envolvidos são especificadas pelo criador do torneio através de uma requisição. A API fica aguardando requisições de torneios e faz a custódia dos fundos até o momento que receber a requisição que indica o final do torneio.

5.4. Os Participantes

Os participantes dos torneios são identificados pelo endereço de suas carteiras. Estes endereços são justamente para onde serão enviadas as premiações no momento que desenvolvedor enviar a mensagem contendo o resultado final do torneio e a estrutura de premiação.

5.5. A Premiação

O valor da premiação fica preso ao contrato que especifica o torneio e participantes envolvidos nele. A premiação consiste em valores arrecadados com entradas pagas pelos participantes reduzidos da taxa de prestação de serviço de hospedagem do torneio (2%).

5.6. Os Pagamentos

Quando a API recebe a mensagem indicando o final do torneio junto com a colocação dos jogadores envolvidos e valor da premiação referente a cada posição, o contrato é ativado e realiza todos os respectivos pagamentos. Os pagamentos envolvem jogadores que atingiram a faixa de premiação, desenvolvedor criador do torneio retém as taxas de inscrição antes de utilizar o serviço e a API que também fica com sua parte das taxas.

6. Exemplo de Uso

Então neste caso, um torneio com 10 pessoas, com valor de entrada 100 e 10 de taxa, o desenvolvedor recebe todos os valores, fica com 80 para si e envia as 1020 moedas que vão ficar custodiadas pela API até ela receber a indicação da finalização do torneio. Neste caso, a mensagem indicou que o pagamento será de 500 para o primeiro colocado, 300 para o segundo e 200 para o terceiro. Os respectivos valores serão enviados para as carteiras dos jogadores de acordo com suas colocações e a API irá ficar com 20 pelo serviço prestado.

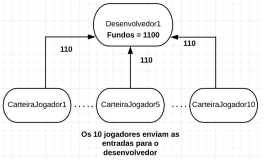


Figura 3. O desenvolvedor recebe o valor de todas as entradas mais taxas de todos os jogadores participantes.

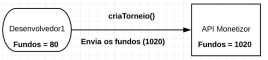


Figura 4. O desenvolvedor inicia o torneio e envia os fundos junto com a taxa que vai ser necessária para o pagamento do serviço.

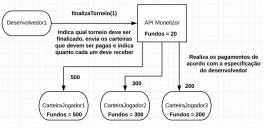


Figura 5. O desenvolvedor envia uma mensagem indicando qual torneio deve ser finalizado, junto com a especificação de como os pagamentos devem ocorrer.

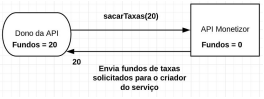


Figura 6. O dono da API faz a solicitação de recebimento das taxas acumuladas sempre que achar necessário.

7. Conclusões

O desenvolvimento desse projeto envolvendo *Blockchain* e *Smart Contracts* foi o primeiro contato mais baixo nível com o assunto. Antes havia uma noção conceitual sobre a tecnologia e possibilidades que ela viabiliza, mas ter esse contato foi muito gratificante. Acredito que entender tudo estava acontecendo foi um pouco mais fácil mim, por já ter lido bastante sobre o tópico.

Fiquei muito satisfeito com o produto final deste estado pois ele atinge todos os objetivos esperados. A solução funciona de maneira distribuída por estar hospedada em múltiplos nós da *Blockchain* o que aumenta a disponibilidade do serviço. O código do contrato também não tem como ser alterado já que ele está hospedado na *Blockchain*, não havendo a possibilidade de existirem modificações na forma do serviço operar, garantindo para o desenvolvedor usuário a confiabilidade necessária para operar. A única forma de criar uma nova versão do serviço seria criando um novo contrato, que no final das contas acaba sendo um serviço novo, com um novo endereço, sem nenhuma ligação com o antigo. O serviço antigo vai continuar operando indefinidamente e o desenvolvedor só vai precisar modificar sua integração com a API se sentir a necessidade de usar algum dos novos recursos disponíveis. Além disso existe a garantia para o desenvolvedor de que o serviço vai fazer exatamente o que se propõe, pois o código fonte que gera o contrato pode ser comparado com a versão compilada gravada na *Blockchain*.

O dinheiro movimentado ser uma versão de criptomoeda também resolve alguns problemas de segurança que poderiam existir. Já que os fundos ficam na carteira de cada usuário, não há a necessidade do desenvolvedor precisar criar toda a estrutura necessária para gerenciar o dinheiro deles, nem correr o risco de ficar com a custódia dos fundos dos seus usuários. É menos risco para o usuário e para o desenvolvedor.

O fato de um smart contract ser utilizado na prestação do serviço acabou criando algumas camadas de segurança muito relevantes que inicialmente não haviam sido identificadas. Já que os fundos envolvidos na competição ficam sob custódia do contrato, algum tipo de ataque ao desenvolvedor não coloca em risco o dinheiro dos competidores. Além disso, os fundos da competição ficam muito pouco tempo sob custódia do desenvolvedor, já que ele recebe o dinheiro dos competidores, tira as suas taxas e envia logo em seguida para o serviço, fazendo com que haja uma janela muito pequena para que algum ataque ocorra. Pela maneira que o contrato foi programado, apenas a carteira que iniciou um torneio pode finalizar ele, então somente o dono da chave privada poderá interagir com os torneios em aberto, impedindo que alguém mal intencionado possa fazer algo.

Embora existam algumas limitações do Solidity que acarretaram em modificações no projeto, os benefícios de usar a *Blockchain* e contratos inteligentes para a prestação desse serviço são tantos justificam a complexidade envolvida na mudança de paradigma e no desenvolvimento da solução. Pretendo estender a tecnologia criada neste projeto para também receber e realizar pagamentos em outras criptomoedas como o Bitcoin, além de desenvolver mais recursos para o serviço como outros formatos de competição e outros serviços utilizando a API.

Eu pessoalmente considero a *Blockchain* uma tecnologia tão ou mais disruptiva que a internet. Acredito que ainda vamos ver a mudança das reservas de valor mundiais para alguma alternativa em cripto, que governos vão precisar mudar, pois cada vez terão menos poder sobre a vida das pessoas. Tudo isso graças aos novos aplicativos descentralizados que virão, diminuindo a quantidade de intermediários das operações, diminuindo taxas e aumentando a segurança, já que o código é incorruptível. Hoje temos uma oportunidade única de poder conhecer, estudar, empreender e investir em algo que é tão revolucionário e ainda está só começando. O futuro é agora!

Referências

Blenkinsop, C. (2019). Gtracking bitcoin transactions, explained.

Buterin, V. (2018). A next generation smart contract decentralized application platform.

CAPSL, C. (2019). History of esports: From pc to mobile tournaments.

Greenstein, S. (2018). The paradox of technological déjà vu.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

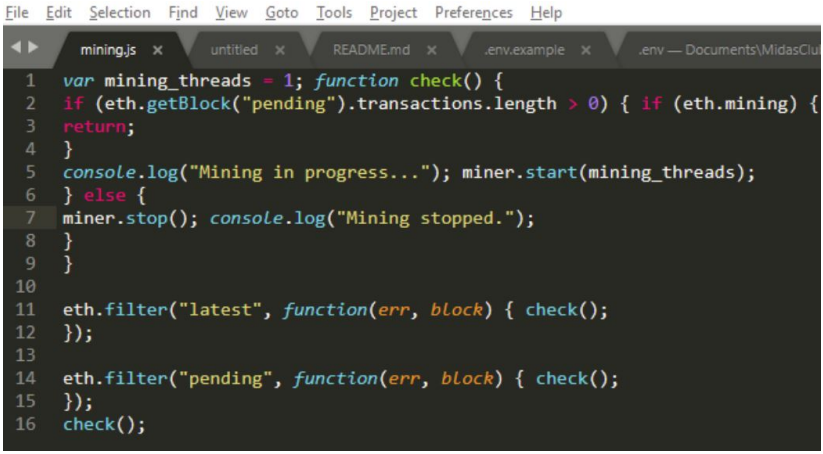
Prado, J. (2017). O que é blockchain? - indo além do bitcoin.

Smith, N. (2018). Its not as awesome as people imagine: Esports players say 'dream job' is more than fun and games.

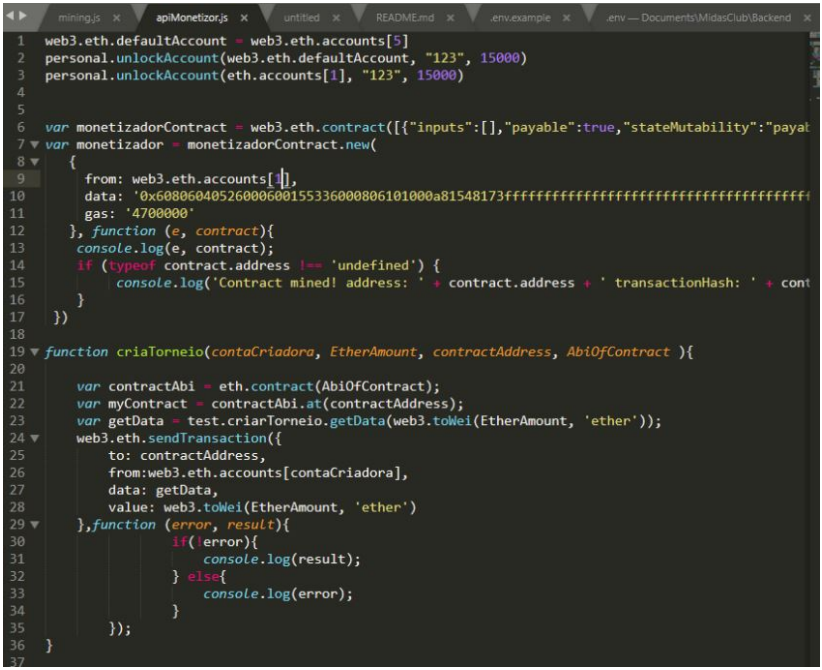
Warburg, B. (2018). How the blockchain will radically transform the economy.

APÊNDICE B – Código Fonte

B.1 MINING.JS


File Edit Selection Find View Goto Tools Project Preferences Help
mining.js
untitled
README.md
.env.example
.env — Documents\MidasClu
1 var mining_threads = 1; function check() {
2 if (eth.getBlock("pending").transactions.length > 0) { if (eth.mining) {
3 return;
4 }
5 console.log("Mining in progress..."); miner.start(mining_threads);
6 } else {
7 miner.stop(); console.log("Mining stopped.");
8 }
9 }
10
11 eth.filter("latest", function(err, block) { check();
12 });
13
14 eth.filter("pending", function(err, block) { check();
15 });
16 check();

B.2 APIMONETIZOR.JS



B.3 MONETIZOR.SOL

```
pragma solidity ^0.5.12;

contract Monetizador {

    address payable monetizor;
    constructor() public payable {
        monetizor = msg.sender;
    }

   struct Torneio {
     uint id;
     uint premiacao;
     uint finalizado;
    }
    uint finalizado = 0;

    mapping (address => Torneio[]) torneioStructs;

    function criarTorneio(uint valorPremiacao) payable public {
      require(msg.value == valorPremiacao);
      torneioStructs[msg.sender].push(Torneio({
          id: identificadorDeTorneios(),
          premiacao: valorPremiacao,
          finalizado: finalizado
      }));
    }

    function finalizarTorneio(uint256 _id, address payable[] memory _addresses ,uint pagamento1
        , uint pagamento2, uint pagamento3) public {

        uint[3] memory valoresPremiacao;
        valoresPremiacao[0] = pagamento1;
        valoresPremiacao[1] = pagamento2;
        valoresPremiacao[2] = pagamento3;
        address payable[] memory premiados;
        premiados = _addresses;
        for (uint i=0; i< premiados.length; i++) {
            address payable premiado = premiados[i];
            uint premio = valoresPremiacao[i];
            premiado.transfer(1 ether * premio);
        }
        torneioStructs[msg.sender][_id-1].finalizado = 1;
    }

    function sacarTaxas(uint _valor) public payable {
            monetizor.transfer(1 ether * _valor);
    }

    function get(uint256 _id) external view returns (uint, uint, uint, address) {
      Torneio memory torneio = torneioStructs[msg.sender][_id-1];
      return (torneio.id, torneio.premiacao, torneio.finalizado, monetizor);
    }

    function identificadorDeTorneios() private view returns(uint count) {
        return torneioStructs[msg.sender].length+1;
    }

}
```