Blockchain e Games

Mark Joselli[1]*

[1]Escola Politécnica
PUCPR

RESUMO

Com o surgimento do Bitcoin e a popularização das utilizações de Blockchain criou novos negócios e novas aplicações para esta tecnologia. Uma blockchain é uma cadeia em bloco, organizada de forma descentralizada onde existe como um banco de dados distribuídos, onde todos os nós do bloco validam e sincronizam seus dados. Uma das aplicações mais interessantes em blockchain são os smart contracts, ou contratos inteligentes, que permitem a execução de códigos na estrutura da blockchain. Estes contratos permitem a utilização em diversos setores, sendo um deles os games. Este capitulo apresenta as bases por trás dessa tecnologia, bem como discute o processo de criação de contratos inteligentes e como interagir com eles em um game.

Keywords: Blockchain, Games, Smart Contracts, Ethereum, Unity3D.

1 INTRODUÇÃO

Blockchain é uma nova tecnologia, que surgiu com o bitcoin. O bitcoin foi criado como um meio de envio e recebimento de pagamentos para e de "endereços", sem uso de um banco ou outros meios de intermediação. É uma rede Open-Source Peer-to-Peer de pagamento. Essa rede usa assinaturas digitais, criptografia e a descentralização, como a base para a segurança e a liberdade de sua chain (cadeia), chamada de blockchain.

Este capitulo tem o objetivo de apresentar o Bitcoin e a tecnologia blockchain por trás disso, bem como os smart contracts, os contratos inteligentes que permite a execução de código na blockchain. Também será mostrado sua linguagem (Solidity) e as principais ferramentas que o desenvolvedor precisa conhecer para criar smart contracts, como funciona o consenso e bem como sua interação dentro de um game. Os potências usos da tecnologia com games será também discutido ao longo deste trabalho.

Este capitulo pretende, inicialmente, explicar os conceitos básicos da tecnologia utilizada na seção 2, apresentando como o blockchain está organizado, como suas transações ocorrem e como o consenso mantém tudo unido e seguro. Na seção 3 é abordado as principais vantagens, bem como as principais preocupações com a tecnologia. A seção 4 se concentra em apresentar os principais termos ou gírias utilizadas no meio, de forma a familiarizar o leitor. Os smart contract, bem como sua principal blockchain, a Ethereum são apresentadas na seção 5. Nesta seção também é apresentada o Solidity, que é a linguagem de programação mais popular para o desenvolvimento de smart contracts e também é discutidos o ERCs (Ethereum Request for Comments) que são protocolos oficiais que servem como base para criação de contratos. Na seção 6 games com blockchain são discutidos, pegando principalmente seu maior sucesso atualmente os "Criptokitites" [1]. Na seção 7 as principais ferramentas para o desenvolvimento de smart contracts são mostradas e discutidas e finalmente na seção 8 temos as conclusões deste trabalho.

*e-mail: mark.joselli@pucpr.br

[1] https://www.cryptokitties.co/

2 BLOCKCHAIN

A cadeia do bloco (blockchain) é a estrutura de dados fundamental do protocolo Bitcoin. É uma única estrutura de dados onde participantes passam de um para o outro, sendo assim todos possuem os mesmos dados, sem ter problemas caso algum nó dessa estrutura fique fora do ar. Isso lhes permite saber quem possui o quê, sendo que qualquer um pode alterá-lo para enviar dinheiro para alguém. Outros usuários matematicamente verificam a transação para garantir a sua validade, esse usuários são os chamados de mineradores. Esses mineradores têm de calcular um hash criptográfico do bloco que satisfaça certos critérios, e com isso ganham bitcoins ou outra criptomoeda.

O bitcoin foi o primeiro projeto em blockchain e foi inicialmente apresentado em 2008 como um white paper [20] por um programador ou grupo de programadores sob o pseudônimo Satoshi Nakamoto, que até hoje ninguém sabe quem seria. Ela é considerada a primeira moeda digital mundial descentralizada, e responsável pelo ressurgimento do sistema bancário livre. O bitcoin é baseado em:

- Governança - uma comunidade de código aberto de empreendedores apoiados pela Fundação Bitcoin discute e propõe a evolução do projeto.
- Democrático - Se alguém não gosta de uma das mudanças, ela é mais que bem-vindo para fazer um fork e implementar suas próprias regras. Isto já ocorreu diversas vezes, sendo as mais famosas o Bitcoin Cash, que aumentou a memória do bloco e o Bitcoin gold, que permitiu a mineração com uso de GPUs.
- Criação de dinheiro - é dada para os mineradores, que tem de gastar poder computacional, e não para os bancos centrais. Dessa forma, o dinheiro é produzido como contrapartida a um trabalho, e não a partir de uma organização governamental.
- E deflacionária por design - oferta de dinheiro não pode ser manipulado e é fixada em 21 milhões de moedas, cada uma divisível até 8 decimais. Dessa forma, depois de criados os 21 milhões de Bitcoin nenhum mais poderá ser criado.

Podemos definir o blockchain como uma estrutura de dados com seguintes elementos-chave [22]:

- Redundância de dados (cada nó tem uma cópia completa dos dados da blockchain). Dessa forma se algum nó cair ou parar de funcionar a rede continua a funcionar sem problemas.
- Rerificação dos requisitos de transação antes da validação.
- Registro de transações em blocos ordenados seqüencialmente, cuja criação é regida por um algoritmo de consenso.
- Transações baseadas em criptografia de chave pública;
- E possivelmente, uma linguagem de script de transação.

2.1 Transações (Transactions)

Uma transação em Bitcoin ou em blockchain nada mais é do que uma estrutura de dados que representa a transferência de um certo valor proveniente de uma fonte, chamada de entrada ou input, para um destino, chamado de saída ou output. Em Bitcoin essas transações são feitas entre duas carteiras, onde uma envia a criptomoeda e a outra recebe a criptomoeda através da rede. Para validar esta transação a carteira que esta enviando o valor deve assinar a transação com sua chave privada que será verificada junto com a sua chave publica e a carteira destino deve ter sua chave pública colocada como o destinatário do valor. Este processo tem um custo dado pela taxa de mineração ou também chamada de taxa de transação ou taxa da rede, que é calculada de acordo com o tamanho da operação e com o congestionamento da rede, que irá validar a transação através das confirmações. Um nó irá validar a transação, realizando um processo matemático, e irá transmitir para os outros nós confirmarem. Uma figura do white paper do Bitcoin ilustrando a transação pode ser visto na figura 1 [20].

A performance de uma rede blockchain é normalmente medida pelo TPS ou transações por segundo (transactions per second), que determina a velocidade da rede. Quanto maior o TPS, mais rápida as transações são executadas, validade e confirmadas na plataforma. O Bitcoin atualmente tem 7 TPS, o que é extremamente lento como um portal de pagamento, que seria sua principal função. Como critério de comparação a VISA atualmente processa 2.000 transações por segundo. Mas o bitcoin promete mudar isso com o novo protocolo chamado Lightning Protocol, que promete um limite de 60.000 TPS. Outros projetos, como o Ethereum é capaz de processar 30 TPS e o Ripple 1500 [2] (apesar de não ser um sistema totalmente descentralizado). Alguns projetos novos como o CREDITS [3] promote ser extremamente rápido e descentralizado, obtendo 488.403 TPS em sua versão alpha.

As transações realizadas na rede também sofrem de latência de que pode demorar de dez minutos para ser confirmada, ou até mesmo horas, dependendo do tamanho do transação e do congestionamento da rede. No caso da rede da VISA a transação leva apenas alguns segundos antes de ter sua confirmação.

2.2 Consenso no Blockchain

Sendo um sistema descentralizado, o blockchain não precisa de uma autoridade externa. Ao invés disto, ele garante a confiabilidade e a consistência dos dados e das transações através de um mecanismo de consenso [18]. Isso é feito de forma a evitar o problema do gasto duplo. O gasto duplo é um dos maiores problemas em produtos digitais, que seria você gastar/utilizar/compartilhar duas vezes o mesmo produto. Por exemplo, se eu tenho um arquivo mp3 ou um e-book no meu computador, posso copiar esse arquivo milhares de vezes livremente e enviá-lo para milhares de pessoas diferentes. Para uma moeda digital, a possibilidade de cópia ilimitada significaria uma rápida morte hiperinflacionária.

O Bitcoin resolve o problema do gasto duplo mantendo uma rede peer to peer e registrando cada transação em único lugar chamado de blockchain. Se eu enviar 1 bitcoin do meu endereço de bitcoin para meu amigo João. A rede de bitcoins registra essa transação na cadeia de blocos e eu não tenho mais a posse desse bitcoin. A moeda "mudou"da minha carteira de bitcoins para a carteira de João.

Para burlar o blockchain e conseguir realizar o gasto duplo, somente se o atacante possuir mais de metade do poder de mineração. este ataque é chamado de ataque dos 51%. Nesse caso existe uma blockchain "fake"competindo com os reais e o fraudador tem que fazer tanto trabalho quanto o resto da rede para fazer a sua cadeia de bloco parecer tão confiável quanto a real. Nesse caso o fraudador pode:

- gastar o dobro: reverter transações que ele envia, enquanto ele está no controle.
- Impedir que algumas ou todas as transações de ganhar quaisquer confirmações.
- Impedir que alguns ou todos os outros geradores de obter quaisquer gerações.

Alguns ataques foram realizados com sucesso, como mostra a figura 2 [4] , mostrando também o gasto necessário para realizar este golpe. No caso, estas redes são menores, com pouco poder de processamento, caso o atacante quisesse realizar isso em uma blockchain mais madura, como o Bitcoin ou o Ethereum, não compensaria o gasto dele comparado com o poder de processamento e energia que ele iria gastar.

Atualmente existe quatro tipos principais de mecanismos de consenso [28]: prova de trabalho (Pow - proof of work) [20]; prova de participação (PoS - proof of stake) [16]; PNFT (Practical Byzantine Fault Tolerance) [4]; e prova de participação delegada (DPoS - delegate proof os stake) [17]. Existem outros tipos, como o prova de banda (PoB - proof of bandwith) [13]; prova de autoridade (PoA - proof of authority) [8]; prova de conceito (PoC - proof of concept) [15]; prova de tempo decorrido (poet - proof-of-elapsed-time) [5]; dentre outros. Os sistemas mais populares são a prova de trabalho, usado no Bitcoin e Ethereum e a prova de participação, usado na NEO e no Dash.

A prova de trabalho se utiliza de solução de processos matemáticos, de forma a comprovar a validade dos dados. Normalmente o processo matemático é um hash criptográfico, ou algum outro problema computacional difícil de ser resolvido, mas fácil de ser verificado. Quando um nó cria um bloco, ele deve resolver um processo matemático. Tendo isso resolvido ele faz a transmissão para os outros nós de forma a conseguir o consenso, como mostra a figura 3. Este processo é chamado de mineração.

Para entender a mineração, é preciso entender o que é uma função hash. Simplificando, uma função hash recebe uma entrada e cria uma saída aparentemente aleatória. Essa saída, apesar de parecer aleatória, é consistente, isto é, toda vez que você executa a função em uma determinada entrada você tem a mesma saída. Outro detalhe, o caminho inverso, determinar a entrada a partir da saída é muito difícil.

Como exemplo podemos criar uma função de hash que tem como saída os números de 4^o até o 9^o lugar do número. Como exemplo, no calculo da raiz quadrada de 3 temos 1.732050807568877293527446341 50. Pegando os dígitos do 4^o lugar após o decimal até o 9^o lugar após o decimal (050807). Agora calculando a raiz número primo, digamos 11 = 3.316624790355399849114932736 6707, que tem dígitos no lugar de 4^o a 9^o (624790). Esse é basicamente um hash muito simples (e fraco). Para qualquer dado número primo (neste caso, ele tem que ser primo), podemos encontrar um número (a saída resultante de 6 dígitos) que parece não ter nada a ver com a entrada, mas pode ser consistentemente e facilmente calculado.

Um outro parâmetro que existe na mineração é a dificuldade. A dificuldade é um dos critérios para o hash que é ajustado com base na frequência blocos estão sendo criados, de forma a ser atrativo para mineradores e também não ser inflacionário.

A mineração segue resumidamente segue o seguinte processo:

1. Recolhe as transações da rede.
2. Valida elas, e não permite transações conflitantes.
3. Coloca elas em grandes blocos.

[2] https://ripple.com/

[3] https://credits.com/

[4] https://www.coindesk.com/blockchains-feared-51-attack-now-becoming-regular/

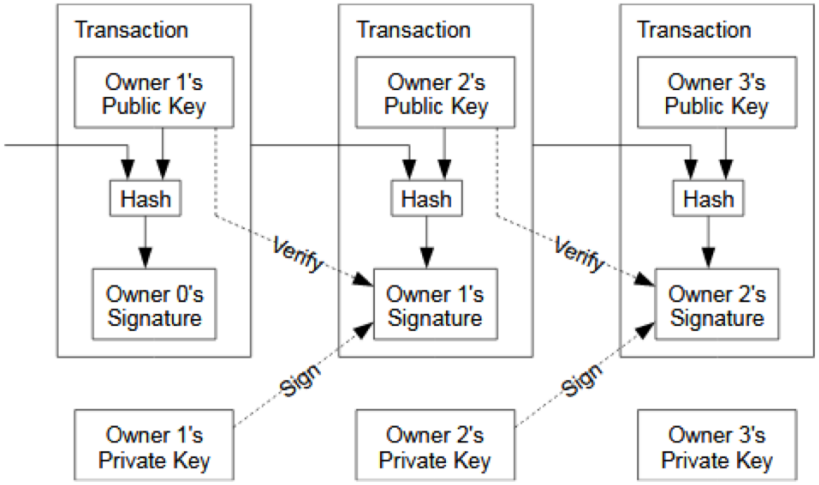

Figura 1: As Transações o blockchain [20].

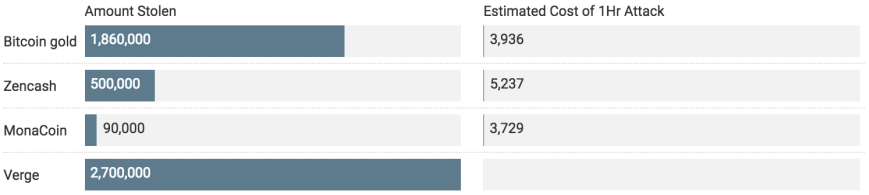

Figura 2: Gastos e golpes realizados com o ataque de 51 %.

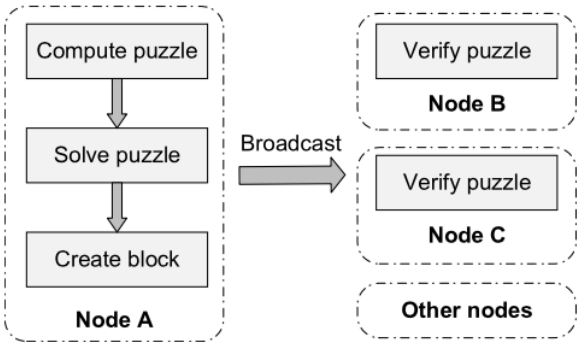

Figura 3: O processo de mineração.[18]

4. Calcula hashes criptográficos até encontrar um hash considerado bom o suficiente para ser valido.

5. Em seguida, envia o bloco para a rede, adicionando-o à blockchain e ganhar uma recompensa em troca.

3 VANTAGENS DA BLOCKCHAIN

O Bitcoin pode ser usado para comprar mercadorias anonimamente. O que pode ser bom ou ruim, pois ele também pode ser (e foi usado) usado desta forma para compras de produtos ilegais. Atualmente, se consegue seguir o lastro do Bitcoin para conseguir descobrir este anonimato. Dessa forma, algumas outras criptomoedas tentam ser totalmente anônimas, como a Monero [5] e a Verge [6].

Bitcoin não está vinculado a nenhum país ou sujeito a regulamentação, dessa forma ele não perde valor e não se inflaciona de acordo com planos de governo, sendo independente. Mas ele pode ser controlado por quem tem concentração de criptomoeda, fato que vem acontecendo [11].

Também seria possível a substituição das varias camadas de autenticação pela tecnologia Blockchain, o que daria também uma maior transparência a transações [3].

3.1 Vantagens em games

Normalmente games multiplayer requerem um servidor central, ou que alguma maquina dos jogadores atue como um servidor, que pode trazer problemas de confiança e de Denial-of-service (DoS) [2]. Com um formato de servidor central, o game pode ficar suscetível a hacks e a manipulações, influenciando no game. Além disto, existe o problema de DoS, onde o servidor pode ter tanto acessos que ele fica indisponível para todos. Isso pode ser causado por um jogador descontente dos resultados do jogo, fazendo com que ele não processe mais as requisições. E também o server pode ficar fora do ar, ou parar para manutenção, de forma que não está disponível para os jogadores.

A principal aplicação de blockchain é a descentralização, que faz com que o servidor esteja sempre disponível, não existe ataque por DoS, pois o servidor esta rodando em diversas máquinas diferentes e um ataque é bem mais caro em uma rede com muitos nós [1].

3.2 Preocupações com o Blockchain

As carteira são vulneráveis a roubo, com hackers e interceptação de pacotes (packet sniffing) usuários podem perder suas chaves privadas e roubarem todas as criptomoedas do usuário sem oportunidade de reave-las.

[5] https://getmonero.org/
[6] https://vergecurrency.com/

A blockchain também é suscetível a ataque Sybil [10]. O nome Sybil vem do livro Sybil de Flora Reta Schreiber, que retrata o estudo de uma mulher com transtorno dissociativo de identidade. O ataque Sybil tenta subverter a reputação do sistema forjando identidades em uma rede peer-to-peer para ter acesso à rede e a disseminação de dados. Dessa forma, poderia tentar o 51% roubando a identidade de 51% dos mineradores.

Outro tipo de ataque é Distributed denial of service (DDoS) attacks ou ataques distribuídos de negação de serviço é um dos desafios mais críticos em cybersecurity [25]. O DDoS é um ataque onde um grande numero de computadores acessam um determinado alvo com um numero alto de tráfego. Este alvo pode ser um servidor, um site, ou um banco, onde ele fica muito lento, ou ele para de funcionar por conta da quantidade de requisições que chegam até ele. Este problema é maior nas casas de cambio (que vendem cryptomoedas) e em ferramentas online, onde o atacante acaba conseguindo burlar o DNS e manda-lo para um site parecido com o original, onde ele consegue roubar suas credenciais ou chaves privadas.

Uma das maiores criticas ao Bitcoin é o seu consumo de energia [21]. Para minerar é preciso um alto poder de processamento, o que em contrapartida requer um alto consumo de eletricidade. As mineradoras da rede blockchain do Bitcoin estão tentando 450 mil trilhões de soluções por segundo nos esforços para validar as transações, usando quantidades substanciais de energia do computador. Isto faz com que a tecnologia seja altamente poluente, pois diversas fontes de energia são poluentes e este consumo faz com que estas fontes sejam utilizadas. Em contrapartida, esta procura por mineração esta fazendo com que a produção de energias renováveis como a éolica e a solar estejam se desenvolvendo e popularizando.

4 TERMOS IMPORTANTES NO BLOCKCHAIN

De forma a facilitar o entendimento pelo leitor dessa nova tecnologia, alguns termos devem ser compreendidos, são eles:

- Bagholder: pessoa que tem uma quantia considerável de investimento, sem vender sua posição.
- Whale/Baleia: um grande investidor que tem uma quantidade considerável de investimento, que caso resolva vender ou comprar vai influenciar os valores.
- Bloco (Block): uma coleção de transações registradas de forma permanente na blochchain registradas ao longo de um tempo.
- BTC: abreviação de Bitcoin.
- Carteira (Wallet): local que contém a chave privada. Podem ser um software, arquivo ou até mesmo em papel, chamada de paper wallet. Também tem a opção de ter um hardware separado, como um pendrive, especifico para guardar chaves privadas de forma segura.
- Chaves privada (Private Key): serie única e secreta de letras e números associadas a um endereço especifico utilizada para aprovar transações de envio.
- Chaves pública (Public Key): serie única de letras e números associadas a um endereço especifico utilizada para receber transações.
- Cold wallet: uma carteira não conectada a internet. Quando ela é conectada a internet é chamada de Hot Wallet.
- Consenso (consensus): quando todos os participantes da rede (blockchain) validam a transação, isto é, tem o mesmo registro.

- Contratos Inteligentes (Smart Contracts): códigos (algoritmos) que funcionam dentro da blockchain e são validados por ela.
- Criptomoeda (Cryptocoin): um tipo de moeda que se utiliza de criptografia e da blockchain para validar as transações;
- Endereço (Address): uma serie de caracteres alfanuméricos usado em conjunto com as chaves publica/privada para mandar ou receber transações na blockchain;
- Hash: uma função matemática baseada em criptografia, que os mineradores tem de resolver de forma a adicionar uma transação ou verificar um bloco no blockchain.
- Hodl: significa que o investidor não pretende vender a crytomoeda, apesar dela cair ou subir de valor. Esse termos vem de Hold, que significa segurar, mas em um forum foi escrito com um erro de digitação.
- Mineração (Mining): quando um computador ou conjunto de computadores (nesse caso o conjunto é chamado de pool) realizam processos matemáticos, adição de novas transações e verificação de blocos criados por outros mineradores, de forma a ganharem criptomoedas em contrapartida.
- Nó (node): um computador (ou conjunto) que participa de uma rede em blockchain e possui uma copia sincronizada dos registros de transações.
- Satoshi: a menor unidade divisível de um Bitcoin, equivale a 0,0000001 Bitcoins.
- Taxa de transação ou de mineração (Mining Tax): valor que deve ser pago para realizar a transação. Este valor é coletado pelos mineradores que processam o bloco, e é variável conforme o congestionamento da rede e da prioridade da transação.
- Transação: um transferencia entre partes registrada entre partes.

5 SMART CONTRACTS

Smart Contract ou em português contrato inteligente é um protocolo de computador que é executado pela blockchain. É como se fosse um contrato entre partes, com a diferença que ele é 100 % digital e uma vez publicado ele não pode ser modificado, perdido ou adulterado. As regras do contrato são definidas por uma linguagem de programação. Diversas criptomoedas brigam para se tornarem o padrão para esta tecnologia, sendo as principais: Ethereum [7], Cardano [8], EOS [9] e NEO [10].

Esses contratos podem ser parte ou blocos de uma aplicação descentralizada, que no caso é chamada de Dapp (decentralizer application), ou até mesmo de uma companhia autônoma e descentralizada, chamada de DAO (decentralized autonomous company).

Os smart contracts são chamados de segunda geração do Blockchain e recebem o nome de Blockchain 2.0 (sendo o Bitcoin e as criptomoedas o Blockchain 1.0) [12].

Smart contracts permitem diversas novas funcionalidades na blockchain, como:

- Funcionar como conta com múltiplas assinaturas, onde fundos só podem ser gastos quanto todos ou uma porcentagem concorda.
- Gerenciar acordos entre usuarios, como contratos, testamentos e seguros. Provendo funcionalidades extras, como acionamento automático.
- Guardar informações sobre a identidade de pessoas e como ela é utilizada.
- Verificar e aplicar regras de DRM (Digital Rights Management) de forma a impor que copias de forma não autorizada a musica e videos protegidos por direito autoral não seja possível.

Para executar smart contracts existem diversas blockchain que focam nisto, sendo a principal a Ethereum. Ethereum é uma plataforma descentralizada que executa contratos inteligentes: aplicativos que são executados exatamente como programados, sem qualquer possibilidade de tempo de inatividade, censura, fraude ou interferência de terceiros. Esses aplicativos são executados em um blockchain personalizado, que é uma infraestrutura global compartilhada extremamente poderosa que pode movimentar valor e representar a propriedade da propriedade. Isso permite que desenvolvedores criem mercados, armazenem registros de dívidas ou promessas, movimentem fundos de acordo com instruções dadas no passado (como um testamento ou um contrato futuro) e muitas coisas que ainda não foram inventadas, tudo isso sem um intermediário ou risco da contrapartida.

Para interagir com a blockchain o desenvolvedor pode-se utilizar de JavaScript com a Web3 [11] e até mesmo Unity3D [12]. A Unity3D é uma ferramenta que inclui o estado da arte no seu segmento, possuindo todos os módulos necessários para uma engine de games, mas para o desenvolvimento com o Blockchain ETH é necessário um plugin ou SDK como o Nethereum [13] ou o Loom [14].

Existem diversos ERCs (Ethereum Request for Comments) que o desenvolvedor deve se familiarizar caso queira desenvolver para a blockchain Ethereum. O ERC é um protocolo oficial para fazer sugestões para melhorar a rede Ethereum. Nele temos o ERC-721, que é um padrão aberto e gratuito que descreve como criar fichas/tokens não fungíveis ou exclusivas na blockchain Ethereum. Embora a maioria dos tokens seja fungível (cada token é o mesmo que qualquer outro token), os tokens ERC-721 são todos únicos. Pode-se pensar neles como itens colecionáveis raros e únicos, o que pode ter aplicação em games, como no caso do Criptokitties.

5.1 Ethereum

Ethereum é um blockchain feito de forma que desenvolvedores possam criar seus próprios smart contracts [27]. A linguagem é chamada de Solidity, e é Turing-complete, isto é, suporta um amplo conjunto de instruções computacionais. Turing-complete é um termo da teoria computacional para dizer que um sistema pode computar qualquer função de acordo com a maquina universal de Turing [14].

O processo de execução de um contrato parte de sua compilação, que transforma o Solidity em Solidity compilado, que por sua vez é transformado em bytecode e implantado em um nó da rede. Esse nó valida o contrato e sincroniza com o resto da rede. O processo completo, desde a compilação até a execução pode ser visto na figura 4.

Ethereum é uma plataforma descentralizada que executa contratos inteligentes: aplicativos que são executados exatamente como programados, sem qualquer possibilidade de tempo de inatividade, censura, fraude ou interferência de terceiros. Esses aplicativos são

[7] https://www.ethereum.org/
[8] https://www.cardano.org/
[9] https://eos.io/
[10] https://neo.org/
[11] https://github.com/ethereum/web3.js/
[12] https://unity3d.com/
[13] https://github.com/Nethereum/Nethereum
[14] https://loomx.io/developers/docs/en/unity-sdk.html

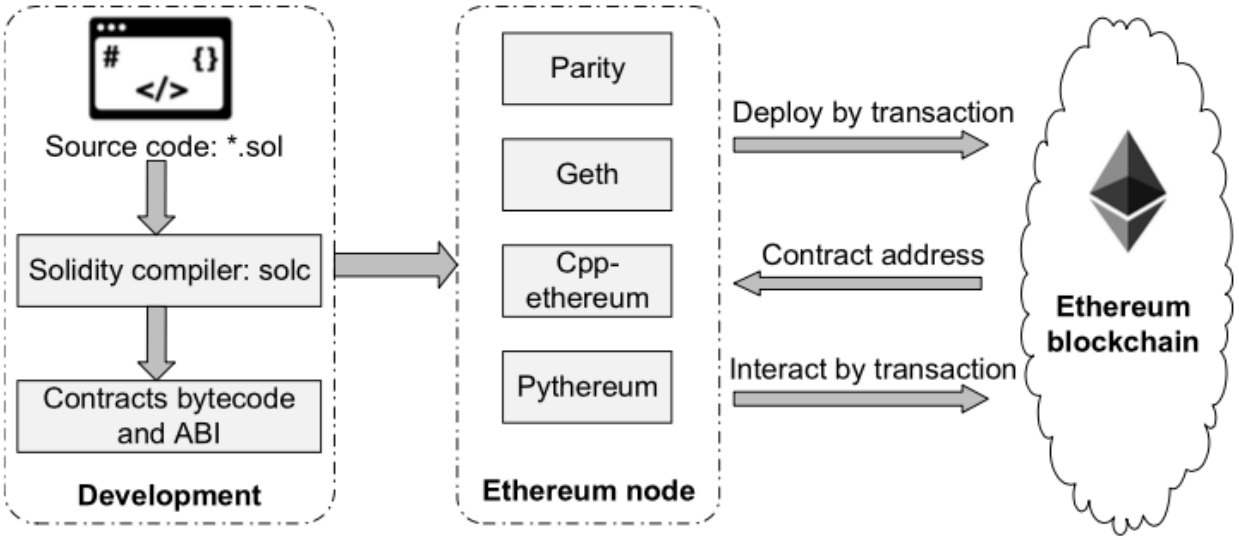

Figura 4: Processo de desenvolvimento de um smart contract [18].

executados em um blockchain personalizado, uma infra-estrutura global compartilhada extremamente poderosa que pode movimentar valor e representar a propriedade da propriedade. Isso permite que os desenvolvedores criem mercados, armazenem registros de dívidas ou promessas, movimentem fundos de acordo com instruções dadas no passado (como um testamento ou um contrato futuro) e muitas coisas que ainda não foram inventadas, tudo sem um intermediário ou risco da contrapartida.

O projeto Ethereum foi inicializado por meio de uma pré-venda de éter durante agosto de 2014 por fãs de todo o mundo. É desenvolvido pela Ethereum Foundation, uma organização sem fins lucrativos suíça, com contribuições de indivíduos e organizações em todo o mundo.

Várias vantagens de Ethereum que incluem:

- A Ethereum Wallet, que é uma porta de entrada para aplicativos descentralizados na blockchain da Ethereum, permitindo que os usuários mantenham e protejam seus éters e outros ativos de criptografia criados na Ethereum, bem como escrevam, implantem e usem contratos inteligentes
- Projetar e emitir seu próprio token criptomoeda / rastreável
- Kickstart um projeto com Crowdsale (ICO)

Os smart contract são executados pela Ethereum Virtual Machine (EVM) em 'bytecode', que são como os contratos ficam depois de compilados para a EVM como uma serie de 0s e 1s, e são interpretados pela rede. A EVM mantém o contrato inerte até que alguma transação faz o contrato ser executado, realizando as tarefas definidas no código. Um contrato pode ler e modificar seus estados internos ou até mesmo chamar funcionalidades de outros contratos e executa-los. Quanto todas as ações do contrato foram executadas ele volta ao estado inerte até receber mais uma transação.

Para interagir com o contrato normalmente é utilizado o ABI, que significa interface binária de aplicativo. Em geral, o ABI é a interface entre dois módulos de programa, um dos quais é o nível de código de máquina e o ABI é forma de interagir com ele.

No Ethereum, é basicamente como você pode codificar chamadas de contratos de Solidity para o EVM e, de trás para frente, como ler os dados de transações.

Os contratos na rede Ethereum são atualmente escritos em Solidity, que é uma linguagem orientada a objetos parecida com o JavaScript [24]. Além disto, podem ser escritos em Serpent, que lembra Python e LLL, que lembra LISP, mas estas ultimas duas linguagens estão com pouco foco pela comunidade, com o repositórios sem atualização.

Toda transação efetuada na rede Ethereum tem um custo, dado pelo éter (ether). O éter é um elemento necessário - um combustível - para operar a plataforma de aplicação distribuída Ethereum. É uma forma de pagamento feita pelos clientes da plataforma para as máquinas que executam as operações solicitadas, funcionando como o incentivo que garante que os desenvolvedores escrevam aplicativos de qualidade e que a rede permaneça saudável. A oferta total de éter e sua taxa de emissão foram decididas pelas doações arrecadadas na pré-venda de 2014. Os desenvolvedores que pretendem criar aplicativos que usarão o blockchain Ethereum precisam de éter. Os usuários que desejam acessar e interagir com contratos inteligentes no blockchain Ethereum também precisam de éter.

5.2 Solidity

Solidity é uma Linguagem Orientada a Objetos de Alto Nível para Smart Contracts parecida com o Javascript. O Solidity foi inicialmente proposta em agosto de 2014 por Gavin Wood [7], e posteriormente passou a ser desenvolvido pelo time do Ethereum. O Solidity permite programar no Ethereum, que pode ser considerada uma máquina virtual baseada em blockchain.

Sua linguagem de programação com tipagem estática com suporte a diferentes tipos básicos, como ints, booleanos e strings. Um tipo especifico que ele permite é o address que pode guardar endereços de usuários e de outros contratos. Os recursos específicos do contrato incluem cláusulas modificadoras (guard), notificadores de eventos e variáveis globais personalizadas.

Solidity suporta herança, bibliotecas e tipos complexos definidos pelo usuário. A ideia do solidity é permitir ao desenvolvedor codificar aplicações que implementar lógica de negócios autocontrolado

pelo próprio contrato. Um exemplo de aplicação exemplo em Solidity que implementa a criação e troca de uma moeda pode ser vista abaixo.

```
pragma solidity ^0.4.21;

contract Coin {
    // The keyword "public" makes those variables
    // readable from outside.
    address public minter;
    mapping (address => uint) public balances;

    // Events allow light clients to react on
    // changes efficiently.
    event Sent(address from, address to, uint
        amount);

    // This is the constructor whose code is
    // run only when the contract is created.
    function Coin() public {
        minter = msg.sender;
    }

    function mint(address receiver, uint amount)
        public {
        if (msg.sender != minter) return;
        balances[receiver] += amount;
    }

    function send(address receiver, uint amount)
        public {
        if (balances[msg.sender] < amount) return;
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        emit Sent(msg.sender, receiver, amount);
    }
}
```

5.3 ERC-20 - Tokens

Um uso mais amplo é suportado pela infraestrutura digital introduzida pelo Bitcoin, são representada por tokens. Um token pode ser definido como um ativo digital escasso baseado na tecnologia subjacente inspirada pelo Bitcoin, ou seja, pela blockchain. Um token também pode ser chamado de criptomoeda. Os tokens podem usar bases de código similares, mas diferentes bancos de dados blockchain.

O Ethereum foi inspirado pelo Bitcoin, mas tem seu próprio blockchain e foi projetado para ser mais programável. Tokens podem ser emitidos no topo do blockchain Ethereum, e com o ERC-20 ele popularizou a emissão desse tipo de moeda.

Os compradores de tokens estão na verdade comprando chaves privadas, que são semelhantes às chaves da API, mas podem ser transferidas para outras partes sem o consentimento, somente com o uso do contrato. Os tokens têm um valor e, portanto, um preço, e normalmente são comercializados em casas de cambio de criptomoedas. Os tokens são um novo modelo de tecnologia e podem ser uma alternativa ao financiamento baseado em capital.

Lançamentos de tokens são chamados de ICOs (initial coin offering) e são quando as companhias ou projetos realizam a criação dos tokens e normalmente os vendem em troca de ETH de forma a financiar o desenvolvimento do projeto.

O padrão ERC-20 [26] define um conjunto de regras que devem ser atendidas para que um token seja aceito e capaz de interagir com outros tokens na rede. Os próprios tokens são ativos de bloco, que podem ter valor, e podem ser enviados e recebidos como qualquer outra criptomoeda.de blocos Ethereum. Alguns tokens famosos são: Binance Coin (BNB) [15], Walton (WTC) [16], OmiseGO (OMG) [17], VeChain (VEN) [18] e SUBstratum (SUB) [19].

O padrão ERC-20 fornece seis parâmetros obrigatórios e três opcionais (mas recomendados) para qualquer contrato inteligente, sendo opcionais: Nome do Token, Símbolo e Decimal (até 18); e obrigatórias: totalSupply; balanceOf; transfer; transferFrom; approve e allowance. Sua interface pode ser vista abaixo.

```

// ERC Token Standard #20 Interface
// https://github.com/ethereum/EIPs/blob/master/
    EIPS/eip-20-token-standard.md

contract ERC20Interface {
    function totalSupply() public constant returns
        (uint);
    function balanceOf(address tokenOwner) public
        constant returns (uint balance);
    function allowance(address tokenOwner, address
        spender) public constant returns (uint
        remaining);
    function transfer(address to, uint tokens)
        public returns (bool success);
    function approve(address spender, uint tokens)
        public returns (bool success);
    function transferFrom(address from, address to
        , uint tokens) public returns (bool
        success);

    event Transfer(address indexed from, address
        indexed to, uint tokens);
    event Approval(address indexed tokenOwner,
        address indexed spender, uint tokens);
}
```

O ERC-20 pode ser aplicado dentro de games, criando dinheiro virtual, que pode ser trocado e utilizado, dentro e fora do jogo, em compra de assets e outros itens do game..

5.4 ERC-721

O ERC-721 [9] é um padrão aberto e gratuito que descreve como criar fichas não fungíveis ou exclusivas na blockchain Ethereum. Embora a maioria dos tokens seja fungível (cada token é o mesmo que qualquer outro token), os tokens ERC-721 são todos únicos. Podemos pensar neles como itens colecionáveis raros e únicos.

Tokens do tipo ERC-721 tokens podem ser comercializados em casas de cambio de cripto moedas ou em markets exchanges, mas seu valor ao contrário dos ERC-20, vem da suas características únicas e a raridade associada a cada token. O padrão define as funções: name , symbol , totalSupply , balanceOf , ownerOf , approve , takeOwnership , transfer , tokenOfOwnerByIndex , and tokenMetadata . E também a dois eventos: Transfer and Approval.

Pode-se ver que o ERC-721 define funções de acordo com o padrão ERC-20, o que permite que carteiras possam mostrar as funcionalidades básicas do token. A interface do ERC-721 pode ser vista abaixo.

```
        contract ERC721 {
  // ERC20 compatible functions
  function name() constant returns (string name);
  function symbol() constant returns (string
      symbol);
  function totalSupply() constant returns (
      uint256 totalSupply);
```

[15] https://www.binance.com/en
[16] https://www.waltonchain.org/
[17] https://omisego.network/
[18] https://www.vechain.com/
[19] https://substratum.net/

```
6     function balanceOf(address _owner) constant
          returns (uint balance);
7     // Functions that define ownership
8     function ownerOf(uint256 _tokenId) constant
          returns (address owner);
9     function approve(address _to, uint256 _tokenId)
          ;
10    function takeOwnership(uint256 _tokenId);
11    function transfer(address _to, uint256 _tokenId
          );
12    function tokenOfOwnerByIndex(address _owner,
          uint256 _index) constant returns (uint
          tokenId);
13    // Token metadata
14    function tokenMetadata(uint256 _tokenId)
          constant returns (string infoUrl);
15    // Events
16    event Transfer(address indexed _from, address
          indexed _to, uint256 _tokenId);
17    event Approval(address indexed _owner, address
          indexed _approved, uint256 _tokenId);
18 }
```

Os tokens ERC-721 podem ser utilizados em games para criar itens únicos que podem ser trocados ou vendidos pelos jogadores, como armas especiais, cartas do jogo e itens em geral. Esse token pode gerar uma nova forma de monetização.

5.5 ERC-998

O token ERC-998 [19] é similar aos tokens ERC-721, isto é, os dois são fungíveis, ou seja únicos, mas no caso do ERC-998 ele é composto, isto é, ele pode ter na sua composição tokens ERC-721 junto com tokens ERC-998 e tokens ERC-20. Dessa forma, o ERC-998 funciona como um token composto de outros tokens. Como exemplo em um game, pode ser visto em uma espada que é composta de aço (um ERC-20) e de uma esmeralda magica única (um ERC-721).

Esses tokens que fazem parte da composição entram como filhos do ERC-998. Na implementação deste ERC ele se utiliza do ERC-165, que detecta nos filhos que interfaces ele implementa, de forma que o ERC-998 pode se utilizar de outros ERCs no futuro, desde que eles implementem certas interfaces.

5.6 ERC-1155

O ERC-1155 [23] permite criar um contrato único, infinitos números de itens, que podem ser únicos (fungíveis) ou não. Dessa forma, existe economia de espaço e também de custo de transação, pois varias trocas podem ser efetuadas pelo mesmo contrato. Este ERC foi proposta pela Enjin [20] empresa de games focada em blockchain, que possui seu próprio token baseado no ERC-1155.

6 Games com Blockchain

Para o uso em games o mais interessante são os contratos inteligentes, ou smart contracts. Os smart contracts são fundamentais para determinar a sequência de ações resultantes do código do contrato. Isso permite que negociação de tudo usando peer-to-peer, desde energia renovável até reservas automatizadas de quartos de hotel. São protocolos de computador que facilitam, verificam ou reforçam a negociação ou a execução de um. Podem ajudar a trocar dinheiro, propriedade, ações ou qualquer coisa de valor de uma forma transparente e livre de conflitos, evitando os serviços de um intermediário. Nele as regras e penalidades são definidas em torno de um contrato da mesma maneira que um contrato tradicional, mas também automaticamente imponha essas obrigações (código é lei).

[20] https://enjincoin.io/

6.1 Games colecionáveis - CryptoKitties

CryptoKitties é um jogo virtual baseado em blockchain desenvolvido pela Axiom Zen que permite aos jogadores comprar, coletar, reproduzir e vender vários tipos de gatos virtuais, baseados no ERC-721. Representa uma das primeiras tentativas de implantar a tecnologia blockchain para fins recreativos e de lazer. A popularidade do jogo em dezembro de 2017 congestionou a rede Ethereum, fazendo com que ela alcançasse um recorde de transações e desacelerasse significativamente. Em 20 de março de 2018, foi anunciado que a CryptoKitties seria desmembrada em sua própria empresa e arrecadou US $ 12 milhões de várias empresas de capital de risco e investidores-anjo. Em dezembro de 2017, um CryptoKitty foi vendido por US $ 100.000,00 [29][6].

Figura 5: CriptoKitties.

No contrato as principais funcionalidades do Criptokitties são:

- KittyBreeding: Este arquivo contém os métodos necessários para criar gatos juntos, incluindo manter as ofertas e depende de um contrato externo de combinação genética.
- KittyAuctions: Aqui temos os métodos públicos para leiloar ou licitar em gatos ou serviços. A funcionalidade real do leilão é tratada em dois contratos (um para vendas e outro para contratação), enquanto a criação de leilões e os lances são mediados principalmente por essa faceta do contrato principal.
- KittyMinting: Esta última faceta contém a funcionalidade que usamos para criar novos gatos gen0. Podemos criar até 5000 gatos promocionais que podem ser doados (especialmente quando a comunidade é nova), e todos os outros só podem ser criados e colocados imediatamente em leilão através de um preço inicial determinado por algoritmos. Independentemente de como eles são criados, há um limite rígido de 50k de kitties. Depois disso, depende da comunidade criar, criar, criar!

7 Ferramentas pra desenvolvimento de smart contracts

Existem diversas ferramentas que podem ser utilizadas para o desenvolvimento de smart contracts.

7.1 Metamask

MetaMask [21] é a forma mais fácil de interagir com a rede Ethereum sem necessitar de rodar um nó do sincronizado com a rede Ethereum, que consome memória e tempo. Através dele que podemos fazer o deploy de contratos, ou validar transações de contratos, além de poder enviar e receber tokens ou ether. Ele funciona como um add-on para os navegadores: Chrome, Firefox, Opera e o Brave. A sua interface básica pode ser vista na figura 6.

[21] https://metamask.io/

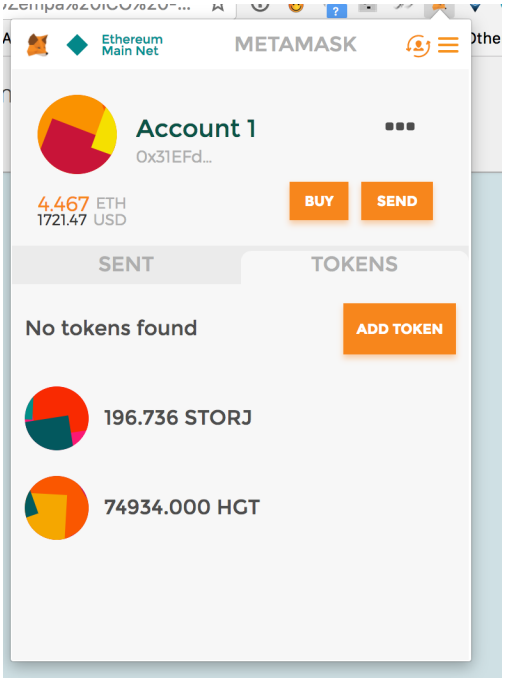

Figura 6: Tela do metamask.

7.2 Remix IDE

Remix [22] é uma IDE para o Solidity, que permite a interação com a blockchain do Ethereum, de forma a criar, debugar e testar Dapps. Ele está disponível online, rodando direto do navegador, ou para baixar de forma a rodar offline. Ele permite a interação direta com git, de forma a facilitar o desenvolvimento e o deploy. Ele também permite o deploy e o testes de transações interagindo diretamente com o Metamask. A sua interface com um contrato pode ser vista na Figura 7.

Existem outras ferramentas, como o Truffle e o testRPC, que simulam uma blockchain inteira e permite o teste em um ambiente mais realístico.

8 Ferramentas para interação com smart contracts

8.1 Web3

A biblioteca web3.js [23] é uma coleção de módulos em JavaScript que contem funcionalidades especificas para o ecossistema Ethereum. Essa biblioteca está já portada para outras linguagens e permite a sua utilização como cliente (direto no navegador com HTML) e como servidor (em um node.js por exemplo).

Os nós Ethereum conversam através do JSON-RPC, que não é fácil de ser entendido por humanos. O Web3.js pode ser visto como um wrapper para a API de JSON-RPC, e pode ser utilizada para realizar as chamadas sem a necessidade de escrever o JSON-RPC na mão. Uma chamada em JSON-RPC para pegar o saldo do endereço 0x407d73d8a49eeb85d32cf465507dd71d507100c1 pode ser vista abaixo.

```
{
"jsonrpc":"2.0",
"method":"eth_getBalance",
"params":["0x407d73d8a49eeb85d32cf465507dd71d507100c1", "latest"],
"id":1
}
```

O mesmo processo pode ser feito com a Web3.js com apenas uma chamada de função, como é mostrado abaixo.

```
web3.eth.getBalance('0x407d73d8a49eeb85d32cf465507dd71d507100c1')
```

O web3 também pode ser utilizado para interagir com um contrato utilizando o ABI e o endereço do smart contract. Um exemplo de código HTML interagindo com um contrato pode ser visto abaixo.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Front-end</title>
    <!-- Include web3.js here -->
    <script language="javascript" type="text/javascript" src="./web3.min.js"></script></script>
    <script language="javascript" type="text/javascript" src="./abi.js"></script>
  <script type="text/javascript">
        var myContract;

    var userAccount;

    var web3js;


        function startApp() {
          var address = "0xaeAD2C7d3a1ed6378A4D067bA7F04498Ad6716dA";
          //myContract = web3js.eth.contract(abi, address);
          myContract = new web3js.eth.Contract(abi, address);
          userAccount = web3.eth.defaultAccount;

        }


        function getWeaponDetails(id) {
        return myContract.methods.weapons(id).call()
    }


        window.addEventListener('load', function() {

            if (typeof web3 !== 'undefined') {
              web3js = new Web3(web3.currentProvider);
            } else {
              console.log('No Web3 Detected... ERROR!!!')
            }
            startApp()
        })

        function BuyWeaponTest(){
```

[22] https://remix.ethereum.org/
[23] https://web3js.readthedocs.io/en/1.0/

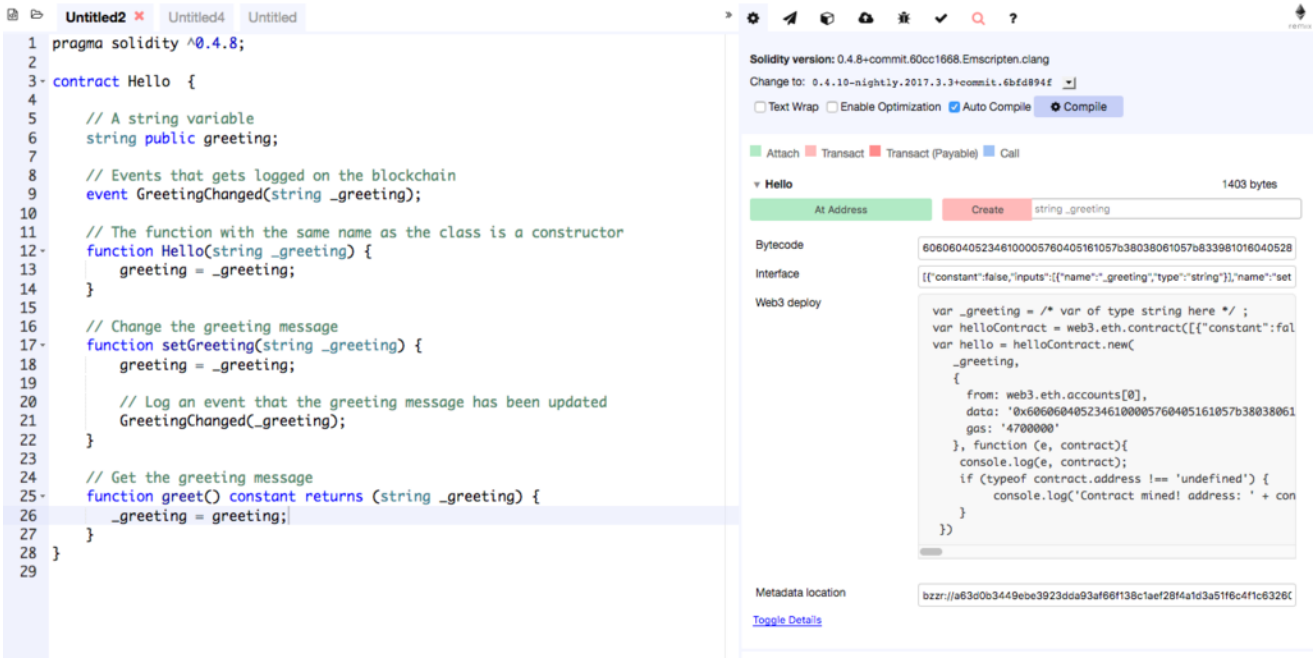

Figura 7: Tela do Remix IDE.

```
44            myContract.methods.buyRandomWeapon("
                  teste")
45          .send({ from: userAccount, value: web3js.
                  utils.toWei("0.001", "ether") })
46          }
47
48
49          function getWeaponTest(){
50            myContract.methods.weapons(0).call()
51          .then(function(result) {
52              console.log("0: " + JSON.stringify(
                    result));
53          });
54          }
55
56
57      </script>
58    </head>
59    <body>
60    <button type="button" onClick="getWeaponTest();"
          >Get Weapon</button>
61    </br>
62    <button type="button" onClick="BuyWeaponTest();"
          >Buy Weapon</button>
63    </body>
64  </html>
```

No código acima pode ser visto como pegar dados do contrato e como interagir com ele. A tela gerada pelo código acima é uma pagina com dois botões que interagem com o contrato, como mostra a imagem 8.

No caso o contrato utilizado na interação é o abaixo, que define um ERC-721 com armas únicas.

```
1  pragma solidity ^0.4.21;
2
3  import "github.com/OpenZeppelin/zeppelin-solidity/
       contracts/token/ERC721/ERC721Token.sol";
```

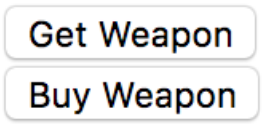

Figura 8: Pagina Web rodando o código.

```
4  import "github.com/OpenZeppelin/zeppelin-solidity/
       contracts/ownership/Ownable.sol";
5
6  contract WeaponToken is ERC721Token("WeaponToken",
       "WEAPON"), Ownable {
7    event NewWeapon(uint weaponId, string name, uint
         dna);
8
9    uint dnaDigits = 16;
10   uint dnaModulus = 10 ** dnaDigits;
11
12   uint creationUpFee = 0.001 ether;
13
14   struct Weapon {
15     string name;
16     uint dna;
17   }
18
19   Weapon[] public weapons;
20   mapping (uint => address) public weaponToOwner;
21   mapping (address => uint) ownerWeaponCount;
22
23   function _createWeapon(string _name, uint _dna)
         internal {
24     uint id = weapons.push(Weapon(_name, _dna)) -
```

```
        1;
25      weaponToOwner[id] = msg.sender;
26      ownerWeaponCount[msg.sender]++;
27      emit NewWeapon(id, _name, _dna);
28    }
29    function _generateRandomDna(string _str) private
          view returns (uint) {
30      uint rand = uint(keccak256(abi.encodePacked(
            _str)));
31      return rand \% dnaModulus;
32    }
33    function buyRandomWeapon(string _name) external
          payable {
34      require(msg.value == creationUpFee);
35      _createWeapon(_name,_generateRandomDna(_name))
            ;
36    }
37
38    function getWeaponByOwner(address _owner)
          external view returns(uint[]) {
39      uint[] memory result = new uint[](
            ownerWeaponCount[_owner]);
40      uint counter = 0;
41      for (uint i = 0; i < weapons.length; i++) {
42        if (weaponToOwner[i] == _owner) {
43          result[counter] = i;
44          counter++;
45        }
46      }
47      return result;
48    }
49
50    function getWeaponData(uint weaponId) external
          view returns(string, uint) {
51      return (weapons[weaponId].name,weapons[
            weaponId].dna);
52    }
53
54  }
```

9 Conclusões

As aplicações de games utilizando o blockchain são algo ainda novo e que ainda tem muito espaço para ser explorado. Este capitulo não pretendeu cobrir por completo todas as funcionalidades que um game poderia se utilizar da Blockchain, mas pelo menos mostrar ao leitor os conceitos básicos de forma a fomentar no leitor as ferramentas para que ele possa desenvolver seu próprios contratos e aplicações com Blockchain.

Referências

[1] N. Atzei, M. Bartoletti, and T. Cimoli. A survey of attacks on ethereum smart contracts (sok). In *Principles of Security and Trust*, pages 164–186. Springer, 2017.

[2] G. Barshap. {\em Crypto-Battleships} or how to play battleships game over the blockchain? *arXiv preprint arXiv:1807.08142*, 2018.

[3] M. A. Bovério and V. A. F. da Silva. Blockchain. *Revista Interface Tecnológica*, 15(1):109–121, 2018.

[4] C. Cachin. Architecture of the hyperledger blockchain fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*, volume 310, 2016.

[5] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi. On security analysis of proof-of-elapsed-time (poet). In *International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 282–297. Springer, 2017.

[6] U. Chohan. The leisures of blockchains: Exploratory analysis. 2017.

[7] C. Dannen. *Introducing Ethereum and Solidity*. Springer, 2017.

[8] S. De Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone. Pbft vs proof-of-authority: applying the cap theorem to permissioned blockchain. 2018.

[9] W. Entriken, D. Shirley, J. Evans, and N. Sachs. Erc-721 non-fungible token standard. *Ethereum Foundation (Stiftung Ethereum), Zug, Switzerland*, 2015.

[10] I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, 61(7):95–102, 2018.

[11] N. Gandal, J. Hamrick, T. Moore, and T. Oberman. Price manipulation in the bitcoin ecosystem. *Journal of Monetary Economics*, 95:86–96, 2018.

[12] M. Gates. *Blockchain: Ultimate guide to understanding blockchain, bitcoin, cryptocurrencies, smart contracts and the future of money.* CreateSpace Independent Publishing Platform, 2017.

[13] M. Ghosh, M. Richardson, B. Ford, and R. Jansen. A torpath to torcoin: proof-of-bandwidth altcoins for compensating relays. Technical report, NAVAL RESEARCH LAB WASHINGTON DC, 2014.

[14] R. Herken. The universal turing machine. a half-century survey. 1992.

[15] S. Higgins. Ibm reveals proof of concept for blockchain-powered internet of things. *CoinDesk, January*, 17:2015, 2015.

[16] S. King and S. Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19, 2012.

[17] D. Larimer. Delegated proof-of-stake (dpos). *Bitshare whitepaper*, 2014.

[18] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen. A survey on the security of blockchain systems. *Future Generation Computer Systems*, 2017.

[19] M. Lockyer, N. Mudge, and J. Schalm. Erc-998 composable non-fungible token standard. *Ethereum Foundation (Stiftung Ethereum), Zug, Switzerland*, 2015.

[20] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[21] K. J. O'Dwyer and D. Malone. Bitcoin mining and its energy footprint. 2014.

[22] S. Porru, A. Pinna, M. Marchesi, and R. Tonelli. Blockchain-oriented software engineering: challenges and new directions. In *Proceedings of the 39th International Conference on Software Engineering Companion*, pages 169–171. IEEE Press, 2017.

[23] W. Radomski and A. Cooke. Erc-998 crypto item standard. *Ethereum Foundation (Stiftung Ethereum), Zug, Switzerland*, 2015.

[24] N. P. Triantafyllidis and T. Oskar van Deventer. Developing an ethereum blockchain application, 2016.

[25] M. Vasek, M. Thornton, and T. Moore. Empirical analysis of denial-of-service attacks in the bitcoin ecosystem. In *International Conference on Financial Cryptography and Data Security*, pages 57–71. Springer, 2014.

[26] F. Vogelsteller and V. Buterin. Erc 20 token standard. *Ethereum Foundation (Stiftung Ethereum), Zug, Switzerland*, 2015.

[27] G. Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32, 2014.

[28] Z. Zheng, S. Xie, H.-N. Dai, and H. Wang. Blockchain challenges and opportunities: A survey. *Work Pap.–2016*, 2016.

[29] Q. T. Zhong and Z. Cole. Analyzing the effects of network latency on blockchain performance and security using the whiteblock testing platform.