**Capítulo 7: Bitcoin na Prática**

Neste capítulo, examinaremos como o protocolo do Bitcoin funciona, bem como alguns protocolos avançados e modernos de Bitcoin que foram desenvolvidos para resolver limitações do protocolo original do Bitcoin.

Neste capítulo, abordaremos:

* Bitcoin no mundo real
* Pagamentos com Bitcoin
* Inovação no Bitcoin
* Instalação do cliente Bitcoin
* Experimentando mais com bitcoin-cli
* Programação em Bitcoin

Antes de falarmos sobre os detalhes específicos do Bitcoin, vamos discutir brevemente a filosofia por trás dele, revisar a definição oficial de Bitcoin e considerar o Bitcoin da perspectiva de um usuário.

**Bitcoin no mundo real**  
Para pessoas com uma ideologia libertária, o Bitcoin é uma plataforma que pode ser usada em vez de bancos. No entanto, alguns acham que, devido às regulamentações, o Bitcoin pode se tornar outra instituição em que não se pode confiar. A ideia original por trás do Bitcoin era desenvolver um sistema de dinheiro eletrônico que não exigisse terceiros confiáveis e onde os usuários pudessem ser anônimos. Se as regulamentações exigirem verificações como Conheça Seu Cliente (KYC) e informações detalhadas sobre transações comerciais para facilitar o processo regulatório, então pode ser informação demais a ser compartilhada. Como resultado, o Bitcoin pode deixar de ser atraente para algumas entidades.

A regulamentação do Bitcoin é um assunto controverso. Por mais que seja o sonho de um libertário, agências de aplicação da lei, governos e bancos estão propondo várias regulamentações para controlá-lo.

Leitores interessados podem ler mais sobre a regulamentação do Bitcoin e outras notícias e atividades relevantes em:  
<https://cointelegraph.com/tags/Bitcoin-regulation>

**Bitcoin na prática**  
Neste ponto, surge a pergunta: se o Bitcoin está sob tanta pressão de órgãos reguladores, como conseguiu crescer tanto? A resposta simples é devido à sua natureza descentralizada e sem necessidade de confiança. Neste contexto, o termo *trustless* (sem necessidade de confiança) refere-se à distribuição de confiança entre os usuários, em vez de uma entidade central. Nenhuma entidade única pode controlar essa rede e, mesmo que algumas entidades tentem impor certas regulamentações, elas só conseguirão ir até certo ponto porque a rede é de propriedade coletiva de seus usuários, em vez de uma entidade única. Ela também é protegida por seu mecanismo de Prova de Trabalho (PoW), que impede ataques adversários à rede. Além disso, o anonimato do fundador do Bitcoin também desempenhou um papel em seu sucesso.

O crescimento do Bitcoin também se deve ao chamado efeito de rede. Também chamado de economias de escala do lado da demanda, esse é um conceito que significa que quanto mais usuários usam a rede, mais valiosa ela se torna. Com o tempo, foi observada uma expansão exponencial no crescimento da rede Bitcoin. Esse aumento no número de usuários é impulsionado principalmente por incentivos financeiros. Além disso, a escassez do Bitcoin e seu mecanismo de controle de inflação embutido lhe conferem valor, já que existem apenas 21 milhões de bitcoins que podem ser minerados. Também, a recompensa aos mineradores é reduzida pela metade a cada quatro anos, o que aumenta a escassez e, consequentemente, a demanda aumenta ainda mais.

Apesar de seu sucesso geral, há algumas preocupações em relação ao Bitcoin. O impacto ESG (Ambiental, Social e de Governança) do Bitcoin também é uma preocupação. Os principais argumentos incluem a centralização da mineração, onde alguns mineradores poderosos possuem a maior parte da rede (poder de hash), o alto consumo de energia, e o desenvolvimento e governança centralizados. Além disso, a volatilidade do Bitcoin é outra preocupação que é vista como uma barreira para sua adoção no dia a dia.

Contudo, apesar da pressão regulatória, de algumas limitações e mesmo de ter sido tornado ilegal em alguns países, o Bitcoin ganhou aceitação significativa em algumas partes do mundo. Por exemplo, o Bitcoin é moeda legal em El Salvador e na República Centro-Africana.

Na próxima seção, veremos como a rede Bitcoin se apresenta do ponto de vista do usuário—como uma transação é feita, como ela se propaga do usuário para a rede e como as transações são verificadas e, por fim, acumuladas em blocos.

**Pagamentos com Bitcoin**  
Tendo explorado os vários componentes da arquitetura e design do Bitcoin no capítulo anterior, o exemplo a seguir ajudará você a entender como uma transação de pagamento via a rede Bitcoin se apresenta da perspectiva do usuário final. Vários passos estão envolvidos nesse processo. Neste exemplo, estamos utilizando a carteira Blockchain para dispositivos móveis.

Os passos são descritos da seguinte forma:

1. Primeiro, ou o pagamento é solicitado por um usuário, ou o remetente inicia uma transferência para enviar dinheiro a outro usuário. Em ambos os casos, o endereço Bitcoin do beneficiário precisa ser enviado por um mecanismo apropriado de comunicação.
2. O remetente insere o endereço do destinatário ou escaneia o código QR gerado que contém o endereço Bitcoin, o valor e uma descrição opcional codificados. O aplicativo de carteira reconhece esse código QR e o decodifica em algo como:

"Por favor, envie <valor> BTC para o endereço <endereço do Bitcoin do destinatário>."

Com valores reais, isso se parecerá com o seguinte:

"Por favor, envie 0.00033324 BTC para o endereço  
1JzouJCVmMQBmTcd8K4Y5BP36gEFNn1ZJ3".

1. No aplicativo de carteira do remetente, essa transação é construída, assinada digitalmente usando a chave privada do remetente e, em seguida, transmitida para a rede Bitcoin.
2. As transações Bitcoin são serializadas para transmissão pela rede e codificadas em formato hexadecimal. Como exemplo, esta transação também é mostrada no formato hexadecimal serializado bruto da seguinte forma:

01000000017d3876b14a7ac16d8d550abc78345b6571134ff173918a096ef

90ff0430e12408b0000006b483045022100de6fd8120d9f142a82d5da9389

e271caa3a757b01757c8e4fa7afbf92e74257c02202a78d4fbd52ae9f3a00

83760d76f84643cf8ab80f5ef971e3f98ccba2c71758d012102c16942555f

5e633645895c9affcb994ea7910097b7734a6c2d25468622f25e12ffffff

ff022c820000000000001976a914c568ffeb46c6a9362e44a5a49deaa6ea

b05a619a88acc06c0100000000001976a9149386c8c880488e80a6ce8f18

6f788f3585f74aee88ac00000000

1. Uma vez que o código QR é decodificado, a transação aparecerá na carteira. Existem vários parâmetros exigidos para que uma transação funcione, como De (From), Para (To), BTC e Taxa (Fee). As taxas da rede Bitcoin asseguram que sua transação será incluída pelos mineradores no bloco.
2. Essa transação será capturada pelos mineradores para ser verificada quanto à legitimidade e incluída no bloco. Uma confirmação aparecerá assim que a transação for verificada, incluída no bloco candidato ou proposto e minerada.
3. Normalmente, neste ponto, os usuários esperam até seis confirmações antes que uma transação seja considerada final; entretanto, uma transação pode ser considerada final na etapa anterior. As confirmações servem como um mecanismo adicional para garantir que haja uma chance probabilística muito baixa de uma transação ser revertida, mas, de outro modo, uma vez que um bloco minerado é finalizado e anunciado, as transações dentro daquele bloco são finais naquele ponto.
4. A taxa apropriada será deduzida do valor original a ser transferido e será paga ao minerador que a incluiu no bloco para mineração.

Na transação descrita acima, um pagamento total de 0.00033324 BTC saiu do endereço do remetente, do qual 0.001267 BTC foi pago ao endereço do destinatário. Uma taxa de 0.00010622 foi deduzida da transação como taxa de mineração.

As transações não são criptografadas e são visíveis publicamente no blockchain. Podemos ver esses detalhes usando um explorador de blockchain, como o blockchain.info:  
<https://www.blockchain.com/btc/address/1JzouJCVmMQBmTcd8K4Y5BP36gEFNn1ZJ3>

Além disso, uma visualização de vários atributos da transação está disponível aqui:  
<https://www.blockchain.com/btc/tx/d28ca5a59b2239864eac1c96d3fd1c23b747f0ded8f5af0161bae8a616b56a1d>

O Bitcoin pode ser aceito como forma de pagamento utilizando várias técnicas. Está sendo cada vez mais aceito como método de pagamento por muitos comerciantes online e sites de comércio eletrônico. Existem várias maneiras pelas quais os compradores podem pagar a um negócio que aceita Bitcoin. Por exemplo, em uma loja online, podem ser utilizadas soluções de comerciantes Bitcoin, enquanto em lojas físicas tradicionais, podem ser usados terminais de Ponto de Venda (POS) e outros equipamentos especializados. Os clientes podem simplesmente escanear o código QR com o URI de pagamento do vendedor e pagar utilizando seus dispositivos móveis. URIs de Bitcoin permitem que os usuários façam pagamentos simplesmente clicando em links ou escaneando códigos QR. Um Identificador Uniforme de Recurso (URI) é uma string que representa as informações da transação. O código QR pode ser exibido próximo ao terminal de ponto de venda, podendo ser decodificado por carteiras.

Diversas soluções de pagamento, como o terminal POS Bitcoin 34 Bytes, estão disponíveis comercialmente. Geralmente, essas soluções funcionam seguindo os seguintes passos:

1. O vendedor insere o valor a ser cobrado em moeda fiduciária, por exemplo, dólares americanos.
2. Uma vez inserido o valor no sistema, o terminal imprime um recibo com um código QR e outras informações relevantes, como o valor a ser pago.
3. O cliente pode então escanear esse código QR utilizando sua carteira Bitcoin móvel para enviar o pagamento ao endereço Bitcoin do vendedor embutido no código QR.
4. Assim que o pagamento for recebido no endereço Bitcoin designado, um recibo é impresso como evidência física da venda.

Processadores de pagamento em Bitcoin são oferecidos por muitos provedores de serviços online. Isso permite a integração com sites de comércio eletrônico para facilitar pagamentos em Bitcoin. Esses processadores de pagamento podem ser usados para aceitar Bitcoin como forma de pagamento. Alguns provedores de serviços também permitem o armazenamento seguro de bitcoins, como, por exemplo, o BitPay (<https://bitpay.com>). Outro exemplo são as soluções para comerciantes em Bitcoin disponíveis em:  
<https://www.bitcoin.com/merchant-solutions>

O Bitcoin, e a tecnologia de blockchain em geral, está em constante evolução, e discutiremos a seguir algumas das ideias mais relevantes.

**Inovação no Bitcoin**  
O Bitcoin passou por muitas mudanças e continua evoluindo para se tornar um sistema cada vez mais robusto e melhor, abordando várias fraquezas do sistema. O desempenho tem sido um tema de intenso debate entre especialistas e entusiastas do Bitcoin por muitos anos. Assim, várias propostas foram feitas nos últimos anos para melhorar o desempenho do Bitcoin, resultando em maior velocidade de transação, aumento da segurança, padronização de pagamentos e melhoria geral de desempenho no nível do protocolo.

Essas propostas de melhoria geralmente são feitas na forma de Propostas de Melhoria do Bitcoin (*Bitcoin Improvement Proposals* – BIPs) ou versões fundamentalmente novas dos protocolos do Bitcoin, resultando em novas redes completamente diferentes. Algumas das mudanças propostas podem ser implementadas por meio de *soft forks*, mas algumas exigem um *hard fork* e, como resultado, dão origem a uma nova moeda.

**Propostas de melhoria do Bitcoin**  
Esses documentos, também chamados de BIPs, são usados para propor melhorias ou informar a comunidade do Bitcoin sobre as melhorias sugeridas, os problemas de design ou alguns aspectos do ecossistema do Bitcoin. Existem três tipos de BIPs:

* **BIP Padrão (Standard BIP):** Usado para descrever mudanças principais que têm grande impacto no sistema Bitcoin; por exemplo, mudanças no tamanho do bloco, mudanças no protocolo de rede ou mudanças na verificação de transações.
* **BIP de Processo (Process BIP):** Uma diferença importante entre os BIPs padrão e de processo é que os BIPs padrão cobrem mudanças no protocolo, enquanto os BIPs de processo geralmente tratam da proposição de uma mudança em um processo que está fora do protocolo principal do Bitcoin. Estes são implementados somente após consenso entre os usuários do Bitcoin.
* **BIP Informativo (Informational BIP):** Normalmente usados apenas para aconselhar ou registrar alguma informação sobre o ecossistema Bitcoin, como questões de design.

Diversos BIPs foram propostos e finalizados para introduzir e padronizar pagamentos em Bitcoin. Notavelmente, o **BIP70** (protocolo de pagamento seguro) descreve o protocolo para comunicação segura entre um comerciante e clientes. Esse protocolo utiliza certificados X.509 para autenticação e opera sobre HTTP e HTTPS. Há três mensagens nesse protocolo: *PaymentRequest*, *Payment* e *PaymentACK*. As principais características dessa proposta são defesa contra ataques do tipo *man-in-the-middle* e prova segura de pagamento.

Ataques do tipo *man-in-the-middle* podem resultar em um cenário no qual o atacante está entre o comerciante e o comprador, e parecerá ao comprador que está falando com o comerciante, mas, na verdade, é o intermediário que está interagindo com o comprador em vez do comerciante. Isso pode resultar na manipulação do endereço Bitcoin do comerciante para fraudar o comprador.

Vários outros BIPs, como o **BIP71** (Tipos MIME do protocolo de pagamento) e o **BIP72** (Extensões URI para o protocolo de pagamento), também foram implementados para padronizar os esquemas de pagamento a fim de suportar o **BIP70** (Protocolo de Pagamento).

Outro desenvolvimento inovador é a **Lightning Network**. Trata-se de uma solução para pagamentos instantâneos escaláveis fora da cadeia (*off-chain*). Foi introduzida no início de 2016 e permite pagamentos fora da blockchain. Esta rede faz uso de canais de pagamento que funcionam fora da blockchain, o que permite maior velocidade e escalabilidade para o Bitcoin.

Agora que fornecemos essa breve discussão sobre a melhoria e evolução do Bitcoin, vejamos algumas das excelentes ideias que surgiram a partir de esforços de pesquisa e inovação relacionados ao Bitcoin.

Um artigo está disponível em: <https://lightning.network/>  
Aqueles que tiverem interesse são incentivados a ler o artigo para entender a teoria e a lógica por trás dessa invenção.

**Protocolos avançados**

Nesta seção, apresentaremos vários protocolos avançados que foram sugeridos ou implementados para melhorar o protocolo Bitcoin. Por exemplo, a **taxa de transferência de transações** é uma das questões críticas que precisa de uma solução. A rede Bitcoin só pode processar aproximadamente de três a sete transações por segundo, o que é um número muito pequeno em comparação com outras redes financeiras. Por exemplo, a rede Visa pode processar aproximadamente, em média, **24.000 transações por segundo**. O PayPal pode processar aproximadamente **200 transações por segundo**, enquanto o Ethereum pode processar, em média, até **20**. À medida que a rede Bitcoin cresceu exponencialmente nos últimos anos, esses problemas começaram a crescer ainda mais. A diferença na velocidade de processamento também é mostrada no gráfico a seguir, que apresenta a escala da diferença entre a velocidade de transações do Bitcoin e de outras redes. O gráfico utiliza uma **escala logarítmica**, que demonstra a vasta diferença entre as velocidades de transação das redes:

<imagem\_7.1>

**Figura 7.1**: Velocidade de transação do Bitcoin comparada a outras redes (em escala logarítmica)

Além disso, questões de segurança, como **maleabilidade de transações**, são preocupações reais e podem resultar em negação de serviço. Várias propostas foram feitas para melhorar o protocolo Bitcoin e resolver várias de suas fraquezas. Uma seleção dessas propostas será apresentada a seguir.

**Segregated Witness**

O **SegWit** ou **Segregated Witness** é uma atualização baseada em *soft fork* do protocolo Bitcoin que aborda fraquezas como taxa de transferência e segurança no protocolo. O SegWit oferece várias melhorias, listadas a seguir:

* Uma correção para a **maleabilidade de transações** devido à separação dos dados de assinatura dos dados transacionais. Nesse caso, não é mais possível modificar o ID da transação, pois ele não é mais calculado com base nos dados de assinatura presentes na transação.
* Ao segregar os dados de assinatura e os dados de transação, **clientes leves** não precisam mais baixar transações com todas as assinaturas desnecessariamente. As transações podem ser verificadas sem assinaturas inúteis, o que permite maior eficiência de largura de banda.
* Redução no tempo de **assinatura e verificação de transações**, o que resulta em transações mais rápidas. Um novo algoritmo de *hashing* de transações para verificação de assinaturas foi introduzido e é detalhado no **BIP0143** (<https://en.bitcoin.it/wiki/BIP_0143>). Devido a essa mudança, o tempo de verificação cresce linearmente com o número de entradas, em vez de crescer quadraticamente, resultando em um tempo de verificação mais rápido.
* Uma capacidade de **versionamento de scripts**, que permite atualizações mais fáceis na linguagem de scripts. O número da versão é prefixado aos scripts de bloqueio para indicar a versão. Essa mudança permite atualizações e melhorias na linguagem de script, sem a necessidade de *hard fork*, apenas aumentando o número da versão do script.
* Aumento no tamanho do bloco por meio da introdução de um **limite de peso** em vez de um **limite de tamanho** no bloco e pela remoção dos dados de assinatura. Este conceito será explicado com mais detalhes em breve.
* Um formato de endereço aprimorado, também chamado de **endereço "bc1"**, que é codificado usando o mecanismo **Bech32** em vez de **base58**. Essa melhoria permite uma melhor detecção e correção de erros. Além disso, todos os caracteres são minúsculos, o que ajuda na legibilidade. Ademais, isso ajuda a distinguir entre transações legadas e transações SegWit. Mais informações estão disponíveis neste link: <https://en.bitcoin.it/wiki/Bech32>

O SegWit foi proposto nos BIPs **BIP141**, **BIP143**, **BIP144** e **BIP145**. Foi ativado na rede principal do Bitcoin em 24 de agosto de 2017, no bloco número **481824**. A ideia principal por trás do SegWit é a separação dos dados de assinatura dos dados de transação (isto é, da árvore de Merkle de transações), o que resulta na redução do tamanho da transação. Essa mudança permite que o tamanho do bloco aumente até **4 MB**. No entanto, o limite prático está entre **1,6 MB** e **2 MB**. Em vez de um limite fixo de tamanho de 1 MB para os blocos, o SegWit introduziu um novo conceito de **limite de peso do bloco**.

O peso do bloco é um novo mecanismo de restrição no qual cada transação possui um peso associado a ela. Esse peso é calculado com base na seguinte fórmula:

**Peso = (Tamanho da transação sem os dados de testemunha) × 3 + (Tamanho da transação)**

Os blocos podem ter no máximo **quatro milhões de unidades de peso**. Para comparação, um byte em um bloco legado de 1 MB equivale a 4 unidades de peso, mas um byte em um bloco SegWit pesa apenas 1 unidade de peso. Essa modificação resulta imediatamente em aumento da capacidade do bloco.

Para gastar uma **Saída de Transação Não Gasta (UTXO)** no Bitcoin, é necessário fornecer uma assinatura válida. No cenário pré-SegWit, essa assinatura é fornecida dentro do **script de bloqueio** (*locking script*), enquanto no SegWit essa assinatura **não** faz parte da transação e é fornecida separadamente.

Quatro tipos de transações foram introduzidos pelo SegWit. Esses tipos são:

1. **Pagamento para Hash de Chave Pública de Testemunha (P2WPKH)**:  
   Esse tipo de script é semelhante ao usual P2PKH (*Pay to Public Key Hash*), mas a diferença crucial é que a **assinatura da transação** usada como prova de propriedade no **ScriptSig** é movida para uma estrutura separada conhecida como **witness** (testemunha) da entrada. A assinatura é a mesma que no P2PKH, mas não faz mais parte do ScriptSig; ele fica simplesmente vazio. A chave pública (PubKey) também é movida para o campo witness.

Esse script é identificado por um hash de 20 bytes. O ScriptPubKey é modificado para um formato mais simples, como mostrado a seguir:

* + **ScriptPubKey P2PKH**:  
    OP\_DUP OP\_HASH160 <pubKeyHash> OP\_EQUALVERIFY OP\_CHECKSIG
  + **ScriptPubKey P2WPKH**:  
    OP\_0 <pubKeyHash>

1. **Pagamento para Script Hash – Pagamento para Hash de Chave Pública de Testemunha (P2SH-P2WPKH)**:  
   Este é um mecanismo introduzido para tornar as transações SegWit **compatíveis com versões anteriores**. Isso é possível aninhando o P2WPKH dentro do P2SH tradicional.
2. **Pagamento para Script Hash de Testemunha (P2WSH)**:  
   Esse script é semelhante ao legado P2SH, mas a assinatura e o script de resgate são movidos para o campo separado de witness. Isso significa que o ScriptSig fica simplesmente vazio. Este script é identificado por um hash SHA-256 de 32 bytes. O P2WSH é um script mais simples em comparação ao P2SH e possui apenas dois campos. O ScriptPubKey é modificado da seguinte forma:
   * **ScriptPubKey P2SH**:  
     OP\_HASH160 <pubKeyHash> OP\_EQUAL
   * **ScriptPubKey P2WSH**:  
     OP\_0 <pubKeyHash>
3. **Pagamento para Script Hash – Pagamento para Script Hash de Testemunha (P2SH-P2WSH)**:  
   Semelhante ao P2SH-P2WPKH, este é um mecanismo que permite a compatibilidade retroativa com os nós legados do Bitcoin.

A adoção do SegWit ainda está em andamento, pois nem todos os usuários da rede concordam ou começaram a utilizar o SegWit.

Em seguida, apresentaremos outras ideias inovadoras no espaço do Bitcoin. Não apenas o Bitcoin original evoluiu significativamente desde sua introdução, como também surgiram novas blockchains que são forks do Bitcoin ou implementações novas do protocolo Bitcoin com funcionalidades avançadas.

**Bitcoin Cash**

O **Bitcoin Cash (BCH)** aumenta o limite de tamanho do bloco para **8 MB**. Essa mudança imediatamente aumenta o número de transações que podem ser processadas em um bloco para um valor muito maior em comparação ao limite de 1 MB no protocolo Bitcoin original. Ele usa **Proof of Work (PoW)** como algoritmo de consenso, e o hardware de mineração ainda é baseado em ASIC. O intervalo entre blocos foi alterado de **10 minutos para entre 10 segundos e até 2 horas**. Ele também fornece **proteção contra repetição (replay protection)** e **proteção contra apagamento (wipe-out protection)**, o que significa que, como o BCH utiliza um algoritmo de hash diferente, isso impede que ele seja repetido na blockchain do Bitcoin. Ele também possui um tipo diferente de assinatura em comparação ao Bitcoin para diferenciar entre as duas blockchains.

A carteira BCH e informações relevantes estão disponíveis no site:  
<https://www.bitcoincash.org>

**Bitcoin Unlimited**

O **Bitcoin Unlimited** aumenta o tamanho do bloco **sem definir um limite rígido**. Em vez disso, os mineradores chegam a um consenso sobre o limite de tamanho de bloco ao longo do tempo. Outros conceitos como **blocos extremamente finos** (*extremely thin blocks*) e **validação paralela** também foram propostos no Bitcoin Unlimited.

Blocos extremamente finos permitem **propagação mais rápida de blocos** entre nós do Bitcoin. Nesse esquema, o nó que solicita blocos envia uma solicitação getdata, junto com um **filtro de bloom**, para outro nó. O objetivo desse filtro de bloom é filtrar as transações que já existem no *mempool* do nó solicitante. O nó, então, envia de volta um bloco fino contendo apenas as transações ausentes. Isso corrige uma ineficiência no Bitcoin, onde transações são frequentemente recebidas duas vezes — uma quando são transmitidas pelo remetente, e novamente quando um bloco minerado é transmitido com a transação confirmada.

**Validação paralela** permite que os nós validem mais de um bloco, juntamente com novas transações recebidas, em paralelo. Esse mecanismo contrasta com o Bitcoin tradicional, onde um nó, durante seu período de validação após receber um novo bloco, **não pode retransmitir novas transações nem validar quaisquer blocos** até que tenha aceitado ou rejeitado o bloco recebido.

**Bitcoin Gold**

Essa proposta foi implementada como um **hard fork** desde o bloco **491407** da blockchain original do Bitcoin. Sendo um hard fork, resultou em uma nova blockchain, chamada **Bitcoin Gold**. A ideia central por trás desse conceito é **abordar a questão da centralização da mineração**, que prejudicou a ideia original do Bitcoin como dinheiro digital descentralizado, na qual mais poder de hash resultou em uma mudança de poder em direção aos mineradores com maior capacidade computacional. O Bitcoin Gold utiliza o algoritmo **Equihash** como seu algoritmo de mineração em vez do tradicional PoW; portanto, ele é inerentemente resistente a ASICs e utiliza **GPUs** para mineração.

O cliente Bitcoin Gold e as informações relevantes estão disponíveis em:  
<https://bitcoingold.org>

**Taproot**

O **Taproot** é uma atualização significativa do protocolo Bitcoin. Foi ativada no **bloco 709.632**. Ela melhora a **velocidade de transação**, a **privacidade** e a **escalabilidade**. Também permite **contratos inteligentes** na rede Bitcoin. O Taproot inclui vários componentes, que explicamos a seguir:

* **Assinaturas Schnorr**, que fornecem **agregação de assinaturas**, resultando em **privacidade e eficiência**. As assinaturas Schnorr são mais seguras do que o ECDSA.
* **Merkelized Alternative Script Tree (MAST)**, que compacta transações complexas do Bitcoin em um único hash, reduzindo a taxa de transação e o uso de memória. O MAST permite enumerar condições distintas de gasto separadamente. Isso permite que os fundos sejam gastos satisfazendo **qualquer** um dos scripts.

No MAST, cada script reside em uma folha da árvore de Merkle. Quando fundos são recebidos, eles são bloqueados para a raiz da árvore de Merkle. Para gastar os fundos, o script de uma única folha é revelado, satisfazendo as condições de gasto exigidas por aquela folha. Aqui, uma **prova de Merkle** comprova sua inclusão na árvore. Dessa forma, outras condições de gasto que não são relevantes são mantidas em sigilo. Isso também significa que múltiplas condições de gasto diferentes podem ser codificadas em várias folhas.

* **Pay2Taproot (P2TR)** é um novo tipo de script. Esse script **combina a assinatura Schnorr e o MAST em uma única transação**. A linguagem de script **Tapscript** é usada para permitir vários tipos de novas transações. Ela é semelhante à linguagem Script original do Bitcoin, mas com algumas mudanças. A mudança principal é a introdução do opcode **OP\_CHECKSIGADD**, que permite a **agregação de assinaturas** utilizando assinaturas Schnorr.

O Tapscript também permite atualizações futuras mais fáceis por *soft fork* por meio do novo opcode **OP\_SUCCESS**. Na prática, os endereços P2TR combinam assinaturas e scripts colocando o MAST de scripts em uma chave pública. Isso significa que os mesmos fundos podem ser gastos **ou** com uma assinatura comum correspondente àquela chave pública (o modo antigo), **ou** por meio de um dos scripts codificados no MAST.

A atualização Taproot é composta por três BIPs:

* **BIP340** (Schnorr),
* **BIP341** (Taproot),
* **BIP342** (Tapscript).

**Protocolos estendidos sobre o Bitcoin**

Vários protocolos, como discutido nas seções seguintes, foram propostos e implementados sobre o Bitcoin para **aprimorar e estender o protocolo** do Bitcoin, bem como para serem utilizados para outros propósitos além de simplesmente moeda virtual.

**Moedas coloridas (Colored Coins)**

As moedas coloridas são um conjunto de métodos desenvolvidos para **representar ativos digitais** na blockchain do Bitcoin. Colorir um bitcoin refere-se, coloquialmente, a atualizá-lo com alguns metadados que representam um ativo digital (*smart property*). A moeda ainda funciona e opera como um Bitcoin, mas adicionalmente **carrega metadados** que representam certos ativos. Isso pode ser alguma informação relacionada ao ativo, alguns cálculos relacionados a transações, ou quaisquer dados arbitrários. Esse mecanismo permite a **emissão e rastreamento de bitcoins específicos**. Os metadados podem ser registrados usando o opcode **OP\_RETURN** do Bitcoin ou, opcionalmente, em endereços multiassinatura (*multisig*). Os metadados também podem ser criptografados, se necessário, para tratar preocupações com privacidade. Algumas implementações também suportam o armazenamento de metadados em redes públicas de torrents, o que significa que **quantidades virtualmente ilimitadas de metadados** podem ser armazenadas. Normalmente, esses são objetos JSON representando vários atributos da moeda colorida. Além disso, **contratos inteligentes também são suportados**.

As moedas coloridas podem ser usadas para representar uma variedade de ativos, incluindo, mas não se limitando a:

* commodities,
* certificados,
* ações,
* títulos,
* votos.

Também deve ser observado que, para trabalhar com moedas coloridas, é necessária uma **carteira que interprete moedas coloridas**, e **carteiras Bitcoin normais não funcionam**, pois não conseguem diferenciar entre moedas coloridas e não coloridas.

A ideia de moedas coloridas é muito atraente, pois **não exige nenhuma modificação** a ser feita no protocolo existente do Bitcoin, e também pode aproveitar a rede segura já existente do Bitcoin. Além da representação tradicional de ativos digitais, também há a possibilidade de criar **ativos inteligentes** que se comportam de acordo com parâmetros e condições definidos para eles. Esses parâmetros incluem:

* validação temporal,
* restrições de transferibilidade,
* taxas.

Isso abre a possibilidade de criação de **contratos inteligentes**, os quais discutiremos no **Capítulo 8: Contratos Inteligentes**.

Um caso de uso significativo é a **emissão de instrumentos financeiros na blockchain**. Isso garantirá taxas de transação baixas, prova de propriedade válida e matematicamente segura, transferência rápida sem exigir intermediários e pagamento instantâneo de dividendos aos investidores.

**Counterparty**

Este é outro serviço que pode ser usado para criar **tokens personalizados** que atuam como uma criptomoeda e podem ser usados para diversos fins, como a emissão de **ativos digitais** sobre a blockchain do Bitcoin. Trata-se de uma plataforma bastante robusta que funciona com a blockchain do Bitcoin em sua base, mas que desenvolveu seu próprio cliente e outros componentes para dar suporte à emissão de ativos digitais. A arquitetura consiste nos seguintes componentes:

1. **Servidor Counterparty**:  
   Este é o cliente de referência e implementa o protocolo central do counterparty. É uma combinação de counterparty-lib e counterparty-cli.
2. **Counterblock**:  
   Este componente fornece serviços adicionais ao servidor Counterparty.
3. **Counterwallet**:  
   Esta é uma carteira web para Bitcoin e moedas Counterparty (XCPs).
4. **armory\_utxsvr**:  
   Este é um serviço usado para transações offline do Armory.

O Counterparty funciona com base na mesma ideia das moedas coloridas, **incorporando dados em transações regulares do Bitcoin**, mas fornece uma biblioteca muito mais produtiva e um conjunto de ferramentas poderosas para dar suporte ao manuseio de ativos digitais. Essa incorporação também é chamada de **consenso embutido** (*embedded consensus*), porque as transações do Counterparty são **incorporadas dentro das transações do Bitcoin**. O método de incorporação dos dados é feito utilizando o opcode **OP\_RETURN** do Bitcoin.

A moeda criada e usada pelo Counterparty é chamada de **XCP** e é usada pelos contratos inteligentes como taxa para execução do contrato. No momento da escrita, seu preço era de **1,03 USD**. Os XCPs foram criados utilizando o método **PoB (Proof of Burn)** discutido anteriormente.

Havia alguns serviços disponíveis online que ofereciam suporte a moedas coloridas, mas eles **não estão mais ativos**. No entanto, consulte este link para obter mais informações:  
**Colu by Coinprism** – <https://en.bitcoin.it/wiki/Coinprism>

O Counterparty permite o desenvolvimento de **contratos inteligentes em Ethereum** usando a linguagem Solidity e permite a interação com a blockchain do Bitcoin. Para isso, é utilizado o **BTC Relay** para fornecer interoperabilidade entre Ethereum e Bitcoin. Este é um conceito engenhoso onde contratos do Ethereum podem se comunicar com a blockchain do Bitcoin e suas transações por meio do BTC Relay. Os *relayers* (nós que executam o BTC Relay) capturam os cabeçalhos de bloco do Bitcoin e os transmitem para um contrato inteligente na rede Ethereum que **verifica a Prova de Trabalho (PoW)**. Esse processo verifica que uma transação ocorreu na rede do Bitcoin.

Tecnicamente, trata-se de um contrato do Ethereum que pode armazenar e verificar cabeçalhos de blocos do Bitcoin, assim como os clientes leves de verificação de pagamento simples do Bitcoin (*SPV clients*) fazem, utilizando filtros de Bloom. Clientes SPV foram discutidos em detalhes no capítulo anterior. Essa ideia pode ser visualizada com o seguinte diagrama:

<imagem\_7.2>

**Figura 7.2**: Conceito de BTC Relay

Agora, passaremos para um tópico diferente, que explica como os altcoins são desenvolvidos, como funcionam e quão difícil é criar uma nova moeda.

**Altcoins a partir do Bitcoin**

Por definição, um **altcoin** é gerado no caso de um **hard fork**. Altcoins devem ser capazes de atrair novos usuários, transações e mineradores; caso contrário, a moeda **não terá valor**. A moeda ganha valor, especialmente no espaço de moeda virtual, devido ao **efeito de rede** e sua **aceitação pela comunidade**. Se uma moeda falhar em atrair usuários suficientes, em breve será esquecida. Usuários podem ser atraídos fornecendo uma quantidade inicial de moedas, o que pode ser feito por meio de diversos métodos. Há, contudo, o risco de que, se a nova moeda não tiver bom desempenho, seu investimento inicial possa ser perdido.

Os métodos para fornecer uma quantidade inicial de altcoins são os seguintes:

1. **Criar uma nova blockchain**:  
   Altcoins podem criar uma nova blockchain e alocar moedas aos mineradores iniciais, mas essa abordagem tornou-se impopular devido a muitos esquemas fraudulentos (*scam*) ou esquemas de manipulação (*pump-and-dump*), onde mineradores iniciais lucravam com o lançamento de uma nova moeda e depois desapareciam.
2. **Proof of Burn (PoB)**:  
   Outra abordagem para alocar fundos iniciais a um novo altcoin é o PoB, também chamado de **ponte unilateral** ou **teto de preço**. Nesse método, os usuários destroem permanentemente uma certa quantidade de bitcoins em proporção ao número de altcoins a serem reivindicados. Por exemplo, se 10 bitcoins forem destruídos, então os altcoins podem ter valor não superior ao dos bitcoins queimados. Isso significa que bitcoins são convertidos em altcoin ao serem queimados.
3. **Prova de posse (Proof of Ownership)**:  
   Em vez de destruir permanentemente bitcoins, um método alternativo é provar que os usuários possuem certa quantidade de bitcoins. Essa prova de posse pode ser usada para reivindicar altcoins ao **vincular blocos de altcoin a blocos do Bitcoin**. Por exemplo, isso pode ser feito por meio de **mineração combinada (merged mining)**, na qual, efetivamente, mineradores de Bitcoin podem minerar blocos de altcoin enquanto mineram bitcoins **sem qualquer trabalho extra**. A mineração combinada será explicada posteriormente neste capítulo.
4. **Sidechains vinculadas (Pegged Sidechains)**:  
   Sidechains, como o nome sugere, são blockchains separadas da rede Bitcoin, mas para as quais o Bitcoin pode ser transferido. Os altcoins também podem ser transferidos de volta para a rede Bitcoin. Este conceito é chamado de **ponte bidirecional (two-way peg)**.

Já cobrimos algumas novas blockchains e implementações do protocolo Bitcoin. A próxima seção introduz a **instalação do cliente Bitcoin** e uma visão geral básica de várias APIs e ferramentas que estão disponíveis para desenvolvimento de aplicações Bitcoin e interação com a blockchain do Bitcoin.

**Instalação do cliente Bitcoin**

O cliente **Bitcoin Core** pode ser instalado a partir de:  
<https://bitcoin.org/en/download>  
Está disponível para diferentes arquiteturas e plataformas, variando desde Windows x86 até Linux ARM.

Discutiremos alguns tópicos relacionados à instalação e configuração do Bitcoin. Começaremos discutindo os diferentes **clientes Bitcoin disponíveis** e suas **ferramentas associadas**, que permitem executar e gerenciar o cliente Bitcoin e interagir com a blockchain do Bitcoin.

**Tipos de clientes e ferramentas**

Há diferentes tipos de clientes Bitcoin Core e ferramentas relevantes. Um cliente Bitcoin é um software responsável por **gerar pares de chaves pública/privada** e facilitar **pagamentos em Bitcoin** usando a blockchain do Bitcoin. Além disso, um cliente pode implementar **sincronização completa com a blockchain** ou optar por implementar apenas a funcionalidade básica de carteira ou **verificação simples de pagamento (SPV)**. Um cliente também pode fornecer outras funções úteis, como monitoramento da rede, armazenamento seguro de chaves e interfaces amigáveis para interação com a blockchain do Bitcoin. Alguns dos elementos centrais do cliente Bitcoin Core e suas ferramentas associadas são os seguintes:

* **bitcoind**:  
  Este é o software cliente principal que roda como um daemon (como um serviço) e fornece a interface **JSON-RPC**.
* **bitcoin-cli**:  
  Esta é a ferramenta de linha de comando rica em recursos para interagir com o daemon do Bitcoin; o daemon então interage com a blockchain e executa várias funções. O bitcoin-cli apenas chama funções JSON-RPC e **não executa nenhuma ação diretamente na blockchain**.
* **bitcoin-qt**:  
  Esta é a interface gráfica (GUI) do cliente Bitcoin Core. Quando o software da carteira é iniciado, ele primeiro verifica os blocos no disco e então inicia o processo de sincronização com a blockchain. O processo de verificação **não é exclusivo do bitcoin-qt**; ele também é executado pelo cliente bitcoind.

Também há outros clientes disponíveis, como o **btcd**, que é um cliente full-node Bitcoin escrito em Golang. Está disponível em:  
<https://github.com/btcsuite/btcd>

**Configurando um nó Bitcoin**

Nesta seção, exploraremos como configurar um **nó Bitcoin** para interagir com a rede Bitcoin e interagir com um nó Bitcoin usando a interface de linha de comando.

O software Bitcoin Core está disponível em:  
<https://bitcoin.org/en/download>  
Você pode baixar o software e instalá-lo de acordo com as instruções fornecidas, que são bastante simples.

Alternativamente, é possível baixar o código-fonte do Bitcoin e **compilá-lo manualmente** para produzir os binários, o que veremos a seguir.

**Configurando o código-fonte**

O código-fonte do Bitcoin pode ser baixado e compilado caso os usuários desejem usar o código do Bitcoin para fins de aprendizado ou simplesmente queiram produzir os binários manualmente. O comando git pode ser usado para baixar o código-fonte do Bitcoin:

$ git clone https://github.com/bitcoin/bitcoin.git

Cloning into 'bitcoin'...

Mude o diretório para a pasta bitcoin:

$ cd bitcoin

Depois que os passos anteriores forem completados, o código pode ser compilado com os comandos:

$ ./autogen.sh

$ ./configure.sh

$ make

$ sudo make install

**Nota**: o comando make mostrado aqui pode levar cerca de 30 minutos para ser concluído, dependendo da velocidade do seu computador.

**Configurando o bitcoin.conf**

O arquivo bitcoin.conf é um **arquivo de configuração** utilizado pelo cliente Bitcoin Core para salvar configurações. Todas as opções de linha de comando para o cliente bitcoind, exceto a opção -conf, podem ser definidas no arquivo de configuração, e, quando o bitcoin-qt ou o bitcoind for iniciado, ele obterá as configurações desse arquivo.

Em sistemas Linux, este arquivo geralmente é encontrado em $HOME/.bitcoin/, mas também pode ser especificado na linha de comando usando o parâmetro -conf=<arquivo> no cliente principal bitcoind. O arquivo de configuração pode ser gerado usando a ferramenta disponível aqui:  
<https://github.com/bitcoin/bitcoin/tree/master/contrib/devtools#gen-bitcoin-confsh>

Agora que configuramos o cliente Bitcoin, vejamos como iniciar o cliente Bitcoin para uso.

**Iniciando um nó na testnet**

O nó Bitcoin pode ser iniciado em uma rede de testes da blockchain (**testnet**) caso você queira testar a rede Bitcoin e realizar alguns experimentos. Esta é uma rede de teste completamente alternativa, usada para experimentação. É uma rede mais rápida em comparação com a principal (*mainnet*) e possui regras mais flexíveis para mineração e transações.

As principais diferenças entre a **mainnet** e a **testnet** são mostradas aqui:

| **Componente** | **Mainnet** | **Testnet** |
| --- | --- | --- |
| Porta de escuta | TCP 8333 | TCP 18333 |
| Porta de conexão RPC | TCP 8332 | TCP 18332 |
| Servidores DNS para bootstrap | Específicos da mainnet | Específicos da testnet |
| Campo ADDRESSVERSION | 0x00 | 0x6F |
| Bloco Gênese | Específico da mainnet | Específico da testnet |
| Verificação IsStandard() | Ativada | Desativada |

Vários **serviços de faucet** também estão disponíveis para a testnet do Bitcoin. Esses serviços são usados para obter alguns bitcoins de teste para contas da testnet. Uma lista de faucets está disponível na wiki do Bitcoin:  
<https://en.bitcoin.it/wiki/Testnet#Faucets>  
A disponibilidade de moedas de teste é muito útil para experimentação na testnet.

O comando de linha para iniciar a testnet do Bitcoin é o seguinte. Para executar o daemon Bitcoin na testnet:

$ bitcoind --testnet -daemon

Para executar a interface de linha de comando Bitcoin:

$ bitcoin-cli --testnet <comando>

Para executar a interface gráfica Bitcoin na testnet:

$ bitcoin-qt –testnet

**Um exemplo de execução:**

1. Inicie o nó Bitcoin em modo daemon (como processo em segundo plano) na testnet:

$ bitcoind --testnet -daemon

Bitcoin server starting

1. Verifique o número de blocos e a dificuldade. Note que há uma longa lista de comandos que o cliente Bitcoin suporta. Este é apenas um exemplo para mostrar como a interface de linha de comando funciona:

$ bitcoin-cli --testnet getmininginfo

{

"blocks": 566251,

"difficulty": 400.6820950060902,

"networkhashps": 572058533067.9225,

"pooledtx": 0,

"chain": "test",

"warnings": ""

}

1. Uma lista completa de comandos pode ser obtida com:

$ bitcoin-cli --testnet help

1. Podemos parar o daemon do Bitcoin com o seguinte comando:

$ bitcoin-cli --testnet stop

Bitcoin server stopping

Com isso, completamos uma introdução básica à testnet do Bitcoin. Em breve faremos mais experimentações com isso, mas antes vamos ver outro modo no qual o nó Bitcoin pode ser executado e que é especialmente útil para fins de teste.

**Iniciando um nó em modo regtest**

O modo **regtest** (*regression testing mode*) pode ser usado para criar uma **blockchain privada local para fins de teste**. Neste modo, o usuário pode **controlar a geração de blocos** para experimentação e testes, e uma quantidade de blocos sem valor pode ser gerada. Ele cria, efetivamente, uma nova blockchain Bitcoin local e isolada para propósitos de teste.

A saída do comando anterior mostra várias opções de linha de comando disponíveis no bitcoin-cli, a interface de linha de comando do Bitcoin. Esses comandos podem ser usados para consultar a blockchain, enviar transações e controlar o nó local.

Os comandos a seguir podem ser usados para iniciar um nó no modo regtest:

1. Inicie o daemon do Bitcoin no modo regtest:

$ bitcoind -regtest -daemon

Bitcoin server starting

1. Verifique o saldo:

$ bitcoin-cli -regtest getbalance

0.00000000

1. Gere blocos e endereços:

$ bitcoin-cli -regtest generatetoaddress 200 $(bitcoin-cli -regtest getnewaddress)

Saída:

[

"366fce3c35031eaa3b085ae7d2631cb5b212bac7e3447bd8ffddb17ef97569c4",

"...",

"33361a74d2586259d69a724921ff7b931cc6c95bd52f09fc05a4b8905695384f"

]

1. Agora podemos verificar o saldo executando:

$ bitcoin-cli -regtest getbalance

5000.00000000

1. Execute um comando, por exemplo, getmininginfo:

$ bitcoin-cli -regtest getmininginfo

{

"blocks": 200,

"currentblockweight": 4000,

"currentblocktx": 0,

"difficulty": 4.656542373906925e-10,

"networkhashps": 12,

"pooledtx": 0,

"chain": "regtest",

"warnings": ""

}

O motivo pelo qual geramos 200 blocos no comando anterior é que, no regtest, um bloco deve ter **100 confirmações** antes que a recompensa associada possa ser utilizada. Portanto, é necessário gerar **mais de 100 blocos** para ter acesso a essa recompensa. Neste comando, geramos 200 blocos, o que resulta em **5.000 bitcoins**, devido à recompensa fixa de 50 bitcoins por bloco.

1. Também podemos obter informações sobre a blockchain com o seguinte comando:

$ bitcoin-cli -regtest getblockchaininfo

{

"chain": "regtest",

"blocks": 200,

"headers": 200,

"bestblockhash": "1cafd1e540b6772f4fe4ab561def0de69945f84e701e5fffa8426ea572d3769b",

"difficulty": 4.656542373906925e-10,

"mediantime": 1577225980,

...

}

1. Pare o daemon do Bitcoin:

$ bitcoin-cli -regtest stop

Bitcoin server stopping

Se quiser deletar o nó anterior do regtest e iniciar um novo, simplesmente delete o diretório chamado **regtest** sob o diretório $HOME do seu computador. No macOS, ele está localizado em:  
/$HOME/Library/Application Support/Bitcoin.

Após excluir o diretório regtest, execute novamente o comando do primeiro passo nesta seção para criar um novo ambiente regtest.

Nesta seção, abordamos como iniciar um nó Bitcoin nos modos de teste (**testnet**) e desenvolvimento (**regtest**) e como interagir com a blockchain do Bitcoin usando o bitcoin-cli, a ferramenta de linha de comando do cliente Bitcoin. A seguir, faremos mais experimentações com alguns comandos e interfaces do Bitcoin.

**Experimentando mais com o bitcoin-cli**

Como vimos até agora, o bitcoin-cli é uma interface de linha de comando poderosa e rica em funcionalidades disponível com o cliente Bitcoin Core e pode ser usada para executar várias funções usando a interface **RPC** fornecida pelo cliente.

Agora veremos como enviar bitcoins para um endereço utilizando a linha de comando. Para isso, usaremos a interface de linha de comando do Bitcoin na rede **regtest**:

1. **Gerar um novo endereço** usando o seguinte comando:

$ bitcoin-cli -regtest getnewaddress

2NC31WFFRwRkwd3S4TpyjN5GGDY7E63GSVd

Nota: A saída completa não é mostrada aqui devido ao seu comprimento, mas é suficiente para explicar o conceito.

1. **Enviar 20 BTC** para o endereço recém-gerado:

$ bitcoin-cli -regtest sendtoaddress \

2NC31WFFRwRkwd3S4TpyjN5GGDY7E63GSVd 20.00

A saída deste comando mostrará o ID da transação, por exemplo:

a83ff460a32f29387d531f19e7092a5dcf6ce52d20931227447c0b9b7a5f2980

1. Agora podemos **gerar mais alguns blocos** para obter confirmações:

$ bitcoin-cli -regtest generatetoaddress 7 $(bitcoin-cli -regtest getnewaddress)

1. Também podemos **consultar as informações da transação** com o seguinte comando:

$ bitcoin-cli -regtest gettransaction \

a83ff460a32f29387d531f19e7092a5dcf6ce52d20931227447c0b9b7a5f2980

Isso mostrará uma saída semelhante à exibida na Figura 7.3. Note que usamos o mesmo hash do ID da transação que foi gerado anteriormente no passo 2.

<imagem\_7.3>

**Figura 7.3**: saída do comando gettransaction

Até agora, usamos o bitcoin-cli no modo **regtest**. No entanto, também podemos usar o bitcoin-cli em **qualquer rede do Bitcoin**, por exemplo, a **testnet** ou a **mainnet**. Basta usar o comando bitcoin-cli sem especificar a rede para consultar a blockchain principal (mainnet). A seguir, mostraremos um exemplo rápido de consulta à blockchain da mainnet do Bitcoin. O cliente Bitcoin fornece **três métodos** para interagir com a blockchain, listados a seguir:

* **Interface de linha de comando (CLI)** do Bitcoin
* Interface **JSON-RPC**
* Interface **HTTP REST**

Primeiro, veremos um exemplo do bitcoin-cli consultando a blockchain usando o método getblock. Em seguida, veremos como o mesmo método pode ser invocado usando a interface JSON-RPC e a interface HTTP REST.

Queremos consultar o **bloco 100** da blockchain do Bitcoin, com o seguinte hash:

000000007bc154e0fa7ea32218a72fe2c1bb9f86cf8c9ebf9a715ed27fdb229a

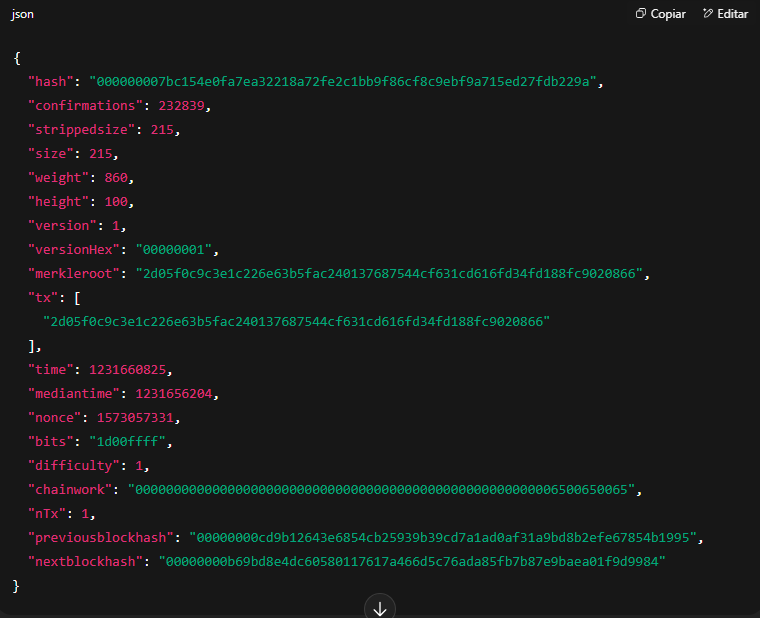
**Usando a ferramenta de linha de comando do Bitcoin**

Podemos usar o bitcoin-cli da seguinte forma:

$ bitcoin-cli getblock \

"000000007bc154e0fa7ea32218a72fe2c1bb9f86cf8c9ebf9a715ed27fdb229a"

A saída do comando anterior mostra os detalhes do bloco com o hash especificado:



**Usando a interface JSON-RPC**

Agora, executaremos o mesmo comando, mas usando a interface **JSON-RPC**. Observe que estamos utilizando a **mainnet do Bitcoin** para este exemplo.

No mínimo, para usar a interface JSON-RPC, precisamos configurar o nome de usuário e a senha do RPC no arquivo bitcoin.conf. Isso pode ser feito facilmente. Um exemplo de configuração que será utilizada neste exemplo é mostrado abaixo:

$ cat bitcoin.conf

rpcuser=test1

rpcpassword=testpassword

Podemos utilizar a ferramenta de linha de comando **curl** para interagir com a API JSON-RPC, como mostrado a seguir:

$ curl --user test1 --data-binary '{"jsonrpc": "1.0",

"id":"curltest", "method": "getblock", "params":

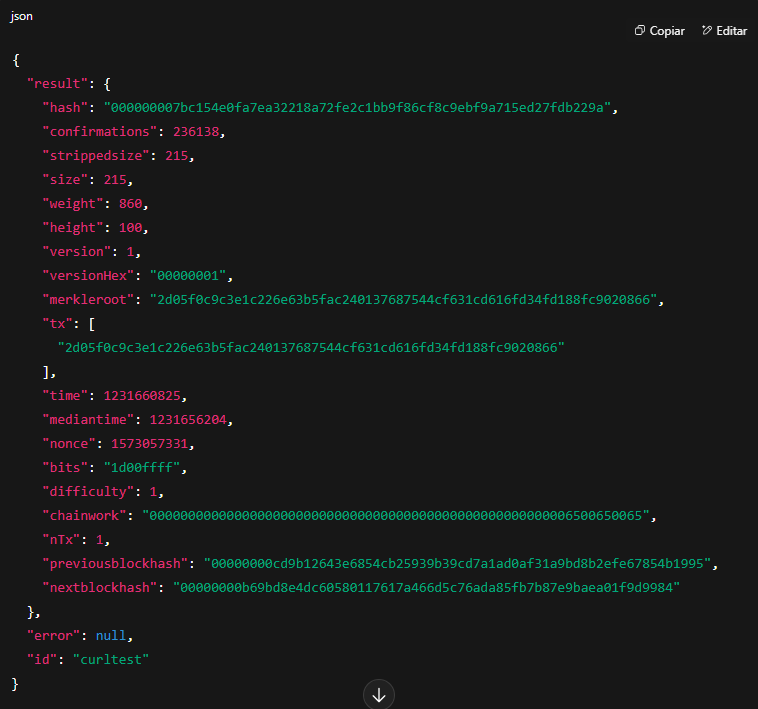
["000000007bc154e0fa7ea32218a72fe2c1bb9f86cf8c9ebf9a715ed27fdb229a"] }' -H

'content-type: text/plain;' <http://127.0.0.1:8332/>

O comando pedirá a senha definida no arquivo bitcoin.conf:

Enter host password for user 'test1':

Se a senha estiver correta, após executar o comando, o resultado será exibido no formato **JSON**, semelhante ao seguinte:



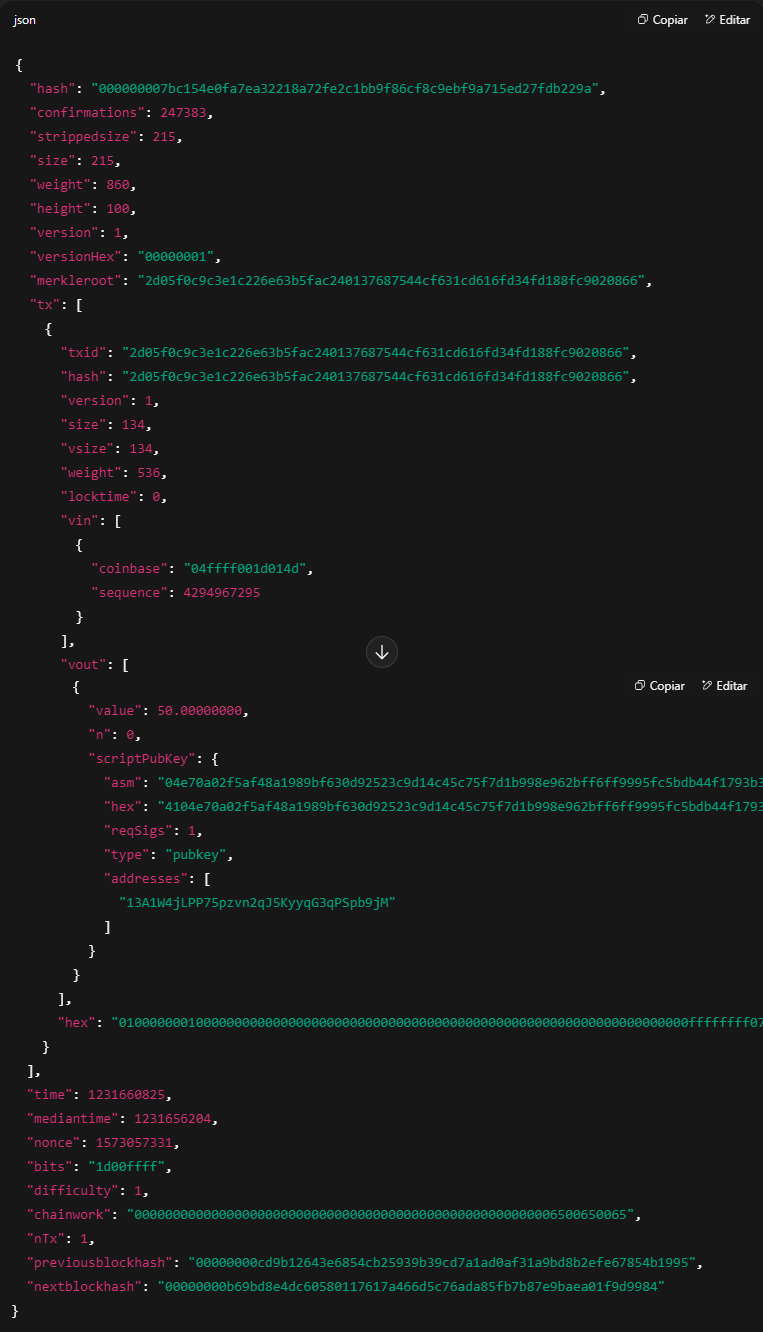
**Usando a interface HTTP REST**

A partir do cliente Bitcoin Core versão **0.10.0**, a interface **HTTP REST** também está disponível. Por padrão, ela é executada na mesma porta TCP (**8332**) que a interface JSON-RPC e **não requer autenticação**. Ela é ativada adicionando a opção rest=1 no arquivo bitcoin.conf, ou na linha de comando do bitcoind por meio do parâmetro -rest.

Podemos usar o curl novamente para este propósito:

$ curl <http://localhost:8332/rest/block/000000007bc154e0fa7ea32218a72fe2c1bb9f86cf8c9ebf9a715ed27fdb229a.json>

A saída do comando acima será:



O curl é uma excelente ferramenta de linha de comando usada para transferir dados usando URLs. É comumente utilizado para interagir com APIs REST por meio de HTTP. Mais informações sobre o curl estão disponíveis em: <https://curl.haxx.se>

**Programação em Bitcoin**

A programação em Bitcoin é um campo muito vasto. O cliente Bitcoin Core expõe vários **comandos JSON-RPC** que podem ser usados para construir transações brutas (*raw transactions*) e executar outras funções por meio de **scripts personalizados ou programas**. Também está disponível a ferramenta de linha de comando bitcoin-cli, que utiliza a interface JSON-RPC e fornece um conjunto de ferramentas rico para trabalhar com Bitcoin.

Essas APIs também estão disponíveis por meio de muitos provedores de serviços online, na forma de **APIs Bitcoin**, e fornecem uma interface simples via **HTTP REST**. APIs de Bitcoin como:

* [blockchain.info](https://blockchain.info/api)
* [BitPay](https://bitpay.com/api)
* [block.io](https://www.block.io)

oferecem uma infinidade de opções para desenvolver soluções baseadas em Bitcoin.

Diversas bibliotecas estão disponíveis para programação com Bitcoin. A lista a seguir apresenta algumas delas. Aqueles que tiverem interesse podem explorar mais profundamente:

* **Libbitcoin**: Disponível em <https://libbitcoin.dyne.org/> — fornece utilitários de linha de comando poderosos e clientes.
* **Pycoin**: Disponível em <https://github.com/richardkiss/pycoin> — uma biblioteca em Python.
* **Bitcoinj**: Disponível em <https://bitcoinj.github.io/> — uma biblioteca implementada em Java.

Existem muitas APIs de Bitcoin disponíveis online; algumas APIs comumente utilizadas são listadas a seguir:

* <https://bitcore.io/>
* <https://bitcoinjs.org/>
* <https://blockchain.info/api>

Como todas as APIs oferecem tipos similares de funcionalidade, pode ser confuso decidir qual usar. Também é difícil recomendar qual API é a melhor, pois todas são ricas em funcionalidades de forma semelhante. No entanto, **uma coisa a ter em mente é a segurança**. Portanto, sempre que avaliar uma API para uso, além de considerar os recursos oferecidos, avalie também o **quão segura é a arquitetura da API**.

**Resumo**

Neste capítulo, analisamos os **pagamentos com Bitcoin** e **processadores de pagamento**, juntamente com algumas **inovações importantes no Bitcoin**, que incluíram tópicos como **BIPs** e **protocolos avançados** do Bitcoin.

O capítulo prosseguiu com uma introdução à **instalação do cliente Bitcoin**, seguida por uma discussão sobre a **configuração do código-fonte** e como configurar **clientes Bitcoin para várias redes**. Após isso, examinamos várias opções de linha de comando disponíveis nos clientes Bitcoin. Por fim, vimos quais **APIs estão disponíveis para programação em Bitcoin** e os principais pontos a serem considerados ao avaliar APIs para uso.

Na próxima capítulo, apresentaremos os **contratos inteligentes**, que são um elemento fundamental de blockchains programáveis e permitem que programadores escrevam programas armazenados na blockchain e que implementem **lógica de negócios na cadeia**.

Para se juntar à comunidade Discord deste livro — onde você pode compartilhar feedback, fazer perguntas ao autor e ficar por dentro dos lançamentos — acesse:

🔗 <https://packt.link/ips2H>