Capítulo 4

Visitando na teoria e na prática o Cartesi RollUps: para além das limitações da Blockchain, uma solução de futuro para aplicativos descentralizados

Antonio A. de A. Rocha, Arthur A. Vianna, Bruno T. Gondim,
Eduardo B. Loivos, Rayan G. O. J. Lima

Abstract

Polarizations, wars and the risk of arbitrary censorship are consequences of the centralization of decision-making in the hands of a few. Technology exists to solve human needs, such as civil liberties. In this sense, web 3.0 seeks in decentralization the neutralization of arbitrary actions. Within this perspective, this work proposes to expose one of the most promising technologies - Cartesi Rollups - capable of elevating Blockchain technology to the status of standard for future internet applications, minimizing its scalability limitations. Practical examples of how to develop and implement such decentralized applications will also be exposed.
Keywords*: Blockchain, Sharding, scaling, decentralized ledger, Cartesi, rollups.*

Resumo

Polarizações, guerras e riscos de censuras arbitrárias são consequências da centralização da decisão nas mãos de poucos. A tecnologia existe para resolver necessidades humanas, como a liberdade civil. Neste sentido, a web 3.0 busca na descentralização a neutralização de ações arbitrárias. Dentro desta perspectiva, este trabalho propõe expor uma das tecnologias mais promissoras - o Cartesi Rollups - capaz de alçar a tecnologia Blockchain ao status de padrão das aplicações na internet do futuro, minimizando suas limitações de escalabilidade. Também serão expostos exemplos práticos de como desenvolver e implementar tais aplicações descentralizadas.
Palavras-chaves*: Blockchain, Sharding, escalabilidade, razão descentralizada, Cartesi, rollups.*

4.1. Introdução

A expressão "*In Blockchain We Trust*" vem sendo utilizada para ressaltar a confiança na segurança dessa solução tecnológica. Isto porque, nela as transações entre indivíduos e a transferência de valores é garantida por meio de algorítimos matemáticos e criptográficos. A Blockchain é uma tecnologia de núcleo que possibilita que grandes grupos de pessoas cheguem a um acordo (também definido como, consenso) e registrem transações permanentes, sem uma autoridade central.

O termo (Bloco = block + chain = cadeia) tem origem no seu funcionamento, onde cada bloco validado é criptograficamente selado ao bloco anterior, formando uma cadeia de blocos cada vez maior, diretamente relacionado na construção de uma economia digital justa, inclusiva, segura e democrática. Por isso, a Blockchain é descrita como uma máquina de confiança, não é apenas um *ledger* distribuído em larga escala, mas também como uma trilha de auditoria imutável. Nela, cada bloco é incorporado em todos os seguintes, impossibilitando a alteração da história de seu conteúdo.

A tecnologia Blockchain também é, portanto, um banco de dados distribuídos, sendo praticamente invulnerável a falhas e adulterações. Suas múltiplas utilidades descolam-se da tecnologia de criptomoedas Bitcoin, para a qual foi criada. Antes do desenvolvimento da tecnologia Blockchain, os registros contábeis eram mantidos em bancos de dados centralizados e não públicos. As pessoas precisavam confiar na idoneidade do banco de dados para ter certeza de que não haveria nenhuma alteração nos registros (saldos e transações da conta, por exemplo). Com a Blockchain, os dados são distribuídos entre todos os participantes, com total transparência e descentralização. Logo, torna-se desnecessário confiar em uma terceira parte para que os dados contábeis sejam registrados corretamente e não haja perigo de fraudes. Podemos concluir que o modelo de solução Blockchain assemelha-se a um "livro-razão", ou seja, uma base de informações com entrada de diversos dados e transações. Todas as informações imputadas e contidas na ferramenta são compartilhadas entre vários usuários.

O processamento desta base de dados é feito em blocos, "de tempos em tempos", criando um código de verificação a cada bloco processado. Estes códigos de verificação são criados com base nos blocos processados anteriormente, fazendo com que a Blockchain seja uma solução de alta confiabilidade, pois, uma vez adulterado um bloco, isso impactará nos demais blocos processados. Por isso, tecnologias Blockchain encontram aplicações em soluções que existe desconfiança mútua pelas partes envolvidas, são exemplos delas: Estabelecimento de contratos; Registro de propriedade intelectual; Estabelecimento de Identidade digital; Prontuário médico; Cartórios digitais; Operações cambiais imediatas; Sistema de voto digital; e, Auditabilidade [Revoredo 2019].

4.2. Blockchain: conceitos fundamentais

A Blockchain oferece propriedades que proveem benefícios às aplicações e sistemas baseados nesta tecnologia. Conforme definido em [Rebello et al. 2019], as principais propriedades da Blockchain são:

- **Descentralização.** A Blockchain é executada de maneira distribuída, sem a necessidade de um intermediário confiável para a troca de ativos, através do estabeleci-

mento de consenso entre todos os participantes da rede;

- **Imutabilidade.** Os dados armazenados em uma corrente de blocos são imutáveis. Não é possível modificar ou recriar qualquer dado incluído na corrente de blocos. Toda atualização na corrente de blocos é realizada de forma incremental;
- **Irrefutabilidade.** Os dados são armazenados na corrente de blocos em forma de transações assinadas, que não podem ser alteradas devido à propriedade de imutabilidade da corrente de blocos. Portanto, o emissor de uma transação jamais pode negar sua existência;
- **Transparência.** Todos os dados armazenados na Blockchain são acessíveis por todos os participante da rede. Permitindo que todos os participantes possam verificar, auditar e rastrear os dados inseridos na corrente de blocos para encontrar possíveis erros ou comportamentos maliciosos;
- **Disponibilidade.** As correntes de blocos são estruturas replicadas em cada participante da rede e, portanto, a disponibilidade do sistema é garantida mesmo sob falhas, devido à redundância de informações;
- **Anonimidade.** Os usuários e nós mineradores de uma Blockchain são identificados por chaves públicas ou identificadores únicos que preservam suas identidades. Ainda, é possível utilizar uma chave pública em cada transação, evitando a rastreabilidade do usuário e conferindo um grau a mais de anonimidade.

4.2.1. Função Hash

A função hash é uma função matemática que tem como entrada uma string de tamanho variável e a mapeia em uma string de tamanho fixo. A conversão tem como objetivo gerar uma identidade única, resistente a colisões. Uma função hash é resistente à colisão, se houver baixa probabilidade de encontrar dois valores de entrada distintos, que possuam um mesmo valor de saída (Figura 4.1). Dado que mapear uma grande quantidade de dados para um universo menor, quanto menor a probabilidade de encontrar hash iguais de forma proposital, melhor a função de hash.

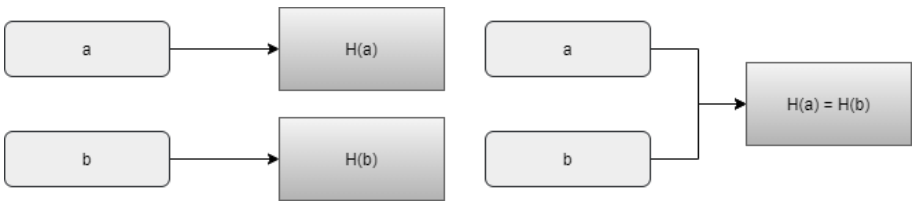

Figura 4.1. Exemplo de colisão entre dois hashs

Uma característica importante do hash é que, uma pequena variação na entrada resulta em uma grande variação na saída. Como pode ser visto na Figura 4.2, mudar uma uma letra de maiúscula para minúscula gera uma saída completamente diferente. Não deve ser possível adicionar conteúdo “a” um texto sem modificar sua saída. Se uma função H(a) resulta a string fixa “y”, encontrar uma string “b” tal que concatenada com “a”

resulte em H(a || b) = y, em um curto espaço de tempo, não deve ser possível. Da mesma forma, se uma entrada tem uma parte gerada de forma aleatória, do hash resultante não pode ser escolhido um hash específico. Outra propriedade da função hash é: Dada uma saída, não é possível determinar a entrada que a originou. Em um espectro pequeno ou repetido de entradas, uma vez vista a solução de para uma dessas entradas, passa a ser possível identificá-la através de seu hash. É possível ocultar a entrada usando um nonce, ou um valor secreto, concatenado a entrada original.

Figura 4.2. Exemplo hash usando SHA256

4.2.2. Estrutura e Cadeia de bloco

A estrutura de bloco consiste, principalmente, em um conjunto de dados. No universo das criptomoedas, os dados de um bloco representam as transações que ocorrem em um determinado período de tempo. Além dos dados, um bloco possui seu índice ou altura, e um campo de nonce. Uma vez formado um bloco, uma função de hash é usada para gerar uma assinatura para o mesmo, garantindo que os dados registrados nele não serão alterados. Uma função de hash comum para esta finalidade é a SHA de 256 bits usada pelo Bitcoin.

Ao criar um bloco e assiná-lo, os dados são verificados e apenas as transações válidas entram em um bloco. Para dificultar que um usuário mal-intencionado valide um bloco com dados forjados, é necessário um esforço para assinar o bloco. As formas mais comuns de esforço usadas são *Proof of Work (PoW)* e *Proof of Stake (PoS)*, mais detalhadas à frente.

A função hash garante que qualquer alteração em um bloco seja facilmente detectada. Para garantir a integridade dos dados, é necessário que um bloco esteja ligado ao próximo através do hash. Assim, para alterar um bloco, deve-se alterar todos os blocos posteriores. Os blocos são conectados através de um campo adicional contendo o hash do bloco anterior. O primeiro bloco, conhecido como bloco gênesis, possui esse campo com o valor 0 em todos os caracteres.

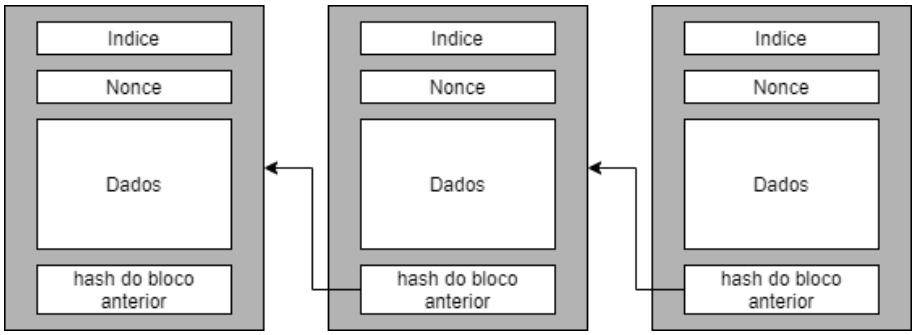

Figura 4.3. Modelo de uma Cadeia de Blocos (Blockchain)

Todos os usuários possuem uma cópia da Blockchain, e sempre que um minerador confirma um bloco ele transmite sua solução na rede. Devido a latência ou a ação de um nó mal intencionado, duas soluções podem ser aceitas em partes diferentes da rede gerando um *fork* na Blockchain. Uma vez identificado o *fork*, os nós consideram a cadeia mais longa como a verdadeira. Logo, para falsificar os dados de um bloco da cadeia, um usuário malicioso precisa, no caso de prova de trabalho, de poder computacional suficiente para tornar seu *fork* mais longo que a Blockchain verdadeira.

4.2.3. Consenso

Plataformas Blockchain usam consenso descentralizado para manter a consistência em uma máquina de estado distribuída. Esta pode ser usada para realizar pagamentos completamente descentralizados ou mesmo cálculos de Turing completos, tornando realidade a criação de aplicações descentralizadas. É papel do consenso em sistemas de Blockchain garantir que todos os nós confiáveis em uma rede Blockchain executem as mesmas atualizações de estado na mesma ordem [Muratov et al. 2018].

Há dois tipos bem comuns de algoritmo de consenso, os baseados em prova e os em voto. No conceito básico de algoritmo de consenso baseado em prova, entre os muitos nós que se juntam à rede, o nó que executa a prova terá o direito de acrescentar um novo bloco à corrente e receber a recompensa. Os principais algoritmos com base em prova são *Proof of Work* , *Proof of Stake* e alguma variação desses [Pahlajani et al. 2019]. Já os algoritmos de consenso baseado em votação, além de manter o livro-razão, todos os nós da rede teriam que verificar juntos as transações ou blocos. Eles se comunicam uns com os outros, antes de decidir anexar ou não os blocos propostos à cadeia. Dentro de quase todas essas variantes, os algoritmos requerem que um número mínimo de nós aceitem o mesmo bloco proposto para anexá-lo à cadeia [Pahlajani et al. 2019]. Alguns desses modos de consenso são descritos a seguir:

Proof of Work (PoW): O uso de um sistema como PoW impede que usuário utilize seu poder computacional para gerar um spam de determinada informação. No cenário da Blockchain, o PoW impede que diversos blocos sejam criados sequencialmente pelo mesmo usuário. O PoW envolve a busca de um valor de nonce que quando processado por um hash, como com SHA-256, o *output* começa com um determinado número de bits zero. O trabalho médio necessário é exponencial, de acordo com número de bits zero necessários, e pode ser verificado executando um único hash. [Nakamoto 2008]

Proof of Stake (PoS): Normalmente pergunta-se, se nós devem manter o consumo de energia para ter uma criptomoeda descentralizada. Portanto, demonstrar que a segurança de criptomoedas ponto a ponto não precisa depender de consumo de energia, é um marco importante tanto teórico quanto tecnologicamente [King and Nadal 2012]. O PoS aproveita o fato de que a posse de moeda é proporcional ao tempo que um nó participa da rede. Desta forma demonstra a sua confiabilidade e assegura a honestidade do mesmo.

Proof of Elapsed Time (PoET): O consenso PoET atua como uma loteria, na qual, cada nó deve esperar um período de tempo randômico, e o primeiro a concluir este tempo de espera ganha o bloco. O próprio nó deve gerar um valor de tempo pelo qual ele irá ficar em *sleep mode*, o primeiro nó a despertar, ou seja, aquele que tiver o menor período de tempo de espera, irá fazer o commit do bloco para a Blockchain e realizar o brod-

cast das informações relevantes [Jake Frankenfield 2020]. Geralmente, alguma prova de que ele não adulterou o processo é necessária. Para garantir que o nó não gere valores de tempo pequenos para sempre ganhar o bloco, os nós que participam deste consenso precisam executar esta geração de valores em um *TEE (Trusted Execution Environment)*. TEEs são ambientes de execução que têm uma superfície de ataque extremamente pequena e são equipados com procedimentos de verificação criptográfica que podem fornecer atestados verificáveis externamente e evidências de adulteração [Corso 2019].

Consenso baseado em voto: Nesse tipo de consenso, todos os nós mineradores devem votar pela aprovação ou rejeição de um bloco, e atingir um consenso antes de adicionar o novo bloco à corrente. Como consequência, todos os nós mineradores devem ser conhecidos e identificados. O protocolo de consenso garante que a adição de um bloco à corrente ocorre de forma sincronizada para todas as réplicas. Isto é, as réplicas adicionam a mesma sequência de blocos na corrente. Dentre os modelos baseados em voto, os tolerantes a falhas bizantinas (Byzantine Fault Tolerant - BFT) são particularmente interessantes. A ideia principal dos protocolos BFT é eleger um líder por uma rodada, que determina e propõe um novo bloco aos participantes do consenso. Os protocolos BFT promovem uma camada de confiança a mais em comparação a protocolos tolerantes apenas a falhas por parada, pois toleram comportamento malicioso na rede. No entanto, geralmente necessitam de mais réplicas, quando comparado com os protocolos que toleram falhas apenas por parada, para tolerar a mesma quantidade de falhas [Rebello et al. 2019].

4.2.4. Bitcoin x Ethereum

Como o protocolo Bitcoin é "Open Source", qualquer um poderia pegar seu protocolo, bifurcar (modificar o código) e iniciar sua própria versão de dinheiro eletrônico. Essa qualidade da Blockchain Bitcoin possuir um código aberto que contribuiu para que, ao longo dos anos, o protocolo Bitcoin fosse modificado centenas de vezes para criar versões alternativas do Bitcoin que são mais rápidas ou mais anônimas. Percebeu-se que o protocolo blockchain subjacente possibilitava que pessoas desconhecidas, ou que não confiavam entre si, realizassem qualquer tipo de transação de valor, e não apenas dinheiro, sem quaisquer intermediários.

Começam a surgir, então, projetos que buscavam usar a tecnologia Blockchain para transferências, sem intermediários, de outros tipos de valor. Também ganhou corpo a ideia de se afastar da blockchains de propósito único, para criar um protocolo onde qualquer tipo de transação, sem validadores tradicionais de confiança, fosse possível. Então, ao perceber que as adaptações da Blockchain Bitcoin não eram satisfatoriamente eficientes nem flexíveis, Vitalik Buterin introduziu a ideia de dissociar as funcionalidades blockchain e iniciou o projeto da Blockchain Ethereum [Wood 2014] em 2014.

Ao contrário da Blockchain Bitcoin, que é uma Blockchain de propósito único com um único contrato inteligente, a Blockchain Ethereum é projetada como uma rede de computadores descentralizada na qual qualquer tipo de contrato inteligente pode ser programado, permitindo qualquer tipo de troca direta de valor. Outras Blockchains surgiram como solução, pois o surgimento da Ethereum inspirou projetos de Blockchain mais recente (como NEO, EOS, CARDANO, CHAINLINK, QTUM, STELLAR, GOCHAIN, entre outros. Além do aspecto tecnológico, fatores técnicos, econômicos e legais também

serão relevantes para avaliar a viabilidade de uma Blockchain [Revoredo 2019].

4.2.5. Um novo mundo para aplicações da Blockchain

Acredita-se que, as máquinas serão os clientes do futuro, presumindo que, mais cedo ou mais tarde, terão carteiras integradas. As primeiras tentativas nesse sentido já foram feitas e com o Ethereum, em que é possível implantar carteiras com contratos inteligentes. Embora a aplicação como moeda e sistema financeiro sem intermediários seja a aplicação mais famosa da tecnologia de Blockchain, várias outras possibilidades e aplicações podem ser vislumbradas.

Uma Organização Autônoma Descentralizada, ou DAO, é uma organização ou empresa teórica operada por código em vez de pessoas. Os DAOs criam uma maneira de as organizações ou empresas serem estruturadas de forma menos hierárquica, argumentam os defensores, com os investidores direcionando diretamente a direção das empresas, em oposição aos líderes designados. Os defensores do DAO acreditam que o Ethereum pode dar vida a essa ideia futurista. Ethereum é a segunda maior criptomoeda por capitalização de mercado e é a maior plataforma para usar a tecnologia por trás da criptomoeda - Blockchain - para usos além do dinheiro. A ideia é que, se o Bitcoin pode eliminar os intermediários nos pagamentos online, a mesma tecnologia ou uma tecnologia comparável pode fazer o mesmo pelos intermediários nas empresas? E se organizações inteiras pudessem existir sem um líder central ou CEO comandando o show?

Com isso, será possível criar múltiplas cópias de segurança: Redundância de informações é algo fundamental para evitar perdas de dados, e nada tem mais backups do que algo controlado por um Blockchain. Afinal, cada um dos mineradores e até mesmos outros membros da rede (como o caso dos fullnodes do Bitcoin) possuem uma cópia completa, atualizada e integral de todos os blocos e, portanto, todas as informações registradas por ele, tais como:

- Registros virtualmente inalteráveis: o hash que valida um bloco é formado pelas informações registradas em um bloco e hash do bloco anterior, que foi gerado pelos dados dele e seu antecessor, e por aí vai. Assim sendo, as informações em um Blockchain são incrementais e acumulativas, armazenadas de forma a não ser alteradas ou apagadas.

- A autenticidade de documentos a garantir a autenticidade de documentos é algo custoso, especialmente no Brasil, e ser caro não garante que o processo é à prova de falhas ou fraudes, pois sempre que houver um fator humano que possa ser corrompido e subverter o sistema, a fraude será possível. A startup brasileira OriginalMy surgiu com uma proposta: registrar documentos em um Blockchain, beneficiando-se de, pelo menos, duas características importantes: o fato de que o registro do documento não possa ser alterado ou removido e a transparência que um Blockchain público provê, permitindo a verificação deste registro sem burocracias.

Também será possível uma maior proteção dos direitos autorais, caso um sistema notarial de algum país fosse implementado integralmente em um único Blockchain. Além

de mitigar fraudes por adulteração ou remoção de informações, todos os documentos poderiam ser verificados e confrontados, evitando a fraude do "vidente que registra sua previsão em cartório": algumas pessoas fazem dezenas de previsões aleatórias, como "prever" celebridades mortas em acidentes aéreos, por exemplo. Ao registrar tais previsões em dezenas de cartórios diferentes (cada previsão em um cartório diferente), quando o previsto se torna um fato, basta apresentar a previsão que "acertou" o acontecimento, desprezando todas as demais. Algo semelhante aconteceu na Copa do Mundo de 2014, quando uma conta de Twitter chamada "@fraudefifa" quis provar que a FIFA manipulava os resultados da competição. Curiosa, no entanto, foi a maneira pela qual a conta decidiu "provar" sua tese, realizando postagens com todas as possibilidades de confrontos e vencedores, dias antes das partidas. Quando dois times realmente jogavam e um destes ganhava, bastava remover do Twitter todas as postagens desfavoráveis.

Surgiram outras iniciativas de autenticação de documentos na Internet, como é o caso do site LexisNexis, que também registra os documentos de maneira a comprovar sua autoria futura. Nesse contexto, a gigante Kodak decidiu não investir na tecnologia em razão da grande receita que a empresa adquiria na revelação de filmes fotográficos tradicionais. A ideia é criar uma plataforma de gerenciamento de direitos de imagem da qual os fotógrafos possam registrar seus trabalhos em um Blockchain, que impedirá que tal registro seja alterado ou apagado. Desta maneira, o registro da foto em um Blockchain poderia comprovar um eventual plágio fotográfico, permitindo identificar o autor da obra com facilidade.

Existem estudos de Implementação do Smart Governance, votação 2.0. Além de garantir a privacidade, o contrato traz transparência ao processo de apuração pois, ao verificar as condições para as quais os votos foram apurados, é possível garantir que 1 pessoa = 1 voto. Ao utilizar um Blockchain como infraestrutura para que o sistema de votação seja realizado, reduzem - se drasticamente quaisquer fraudes que possam hoje ser realizadas em sistemas eleitorais tradicionais.

As Blockchains e smart contracts são tecnologias em ascensão. Enquanto os ativos digitais são usados por uma parcela muito pequena da população mundial, a adoção da Blockchain é inicial em outros campos e ainda certamente encontrarão inúmeros casos de uso da tecnologia ainda desconhecidos ou com desenvolvimento ainda muito incipientes [Eichmann 2018].

4.3. Aplicações Descentralizadas

A criação de contratos inteligentes trouxe consigo uma nova proposta de arquitetura para aplicações, chamada de DApp. A sigla DApp vem do inglês "Decentralized Application", que tem como tradução "Aplicação Descentralizada". Ao contrário do que pode parecer a princípio, o termo "descentralizada" presente na sigla DApp não diz respeito a um back-end ou processamento descentralizado. Afinal, uma aplicação não precisa rodar na Blockchain para ser descentralizada. Há algum tempo já existem bancos de dados distribuídos. Nesse caso, a palavra descentralizada remete a ideia de que a aplicação não possui uma única entidade que controla o seu funcionamento. Não apenas isso, para que uma aplicação seja considerada um DApp, ela deve atender os seguintes critérios, segundo [Johnston et al. 2014]:

- A aplicação deve ser completamente open-source, deve operar de forma autônoma, sem que nenhuma entidade controle a maioria de seus tokens, e seus dados e registros de operação devem ser armazenados criptograficamente em uma cadeia de blocos pública e descentralizada, ou seja, armazenada em uma Blockchain pública;
- A aplicação deve gerar tokens de acordo com um algoritmo padrão ou conjunto de critérios e possivelmente distribuir alguns ou todos os tokens no início de sua operação. Esses tokens devem ser necessários para o uso da aplicação e qualquer contribuição dos usuários deve ser recompensada com pagamento de tokens da aplicação;
- A aplicação pode adaptar o seu protocolo em resposta às melhorias propostas e feedback do mercado, mas todas as mudanças devem ser decididas por consenso da maioria de seus usuários.

Dado que tokens constituem o critério do segundo ponto, é importante termos uma definição para tal. A finalidade de um token é permitir o acesso a uma aplicação. Por exemplo, um indivíduo deve possuir um número de Bitcoins para poder realizar qualquer transação na rede Bitcoin. Os tokens nos DApps não representam nenhum ativo subjacente, não dão direito a um dividendo e nenhum patrimônio é representado por meio deles. Embora o valor de um token no DApp possa aumentar ou diminuir ao longo do tempo, os tokens não são títulos de capital.

Com base na definição apresentada para tokens e nos critérios definidos para que uma aplicação seja considerada um DApp, vemos que o conceito de "descentralizada" é mais amplo do que pode parecer a princípio.

4.3.1. Bitcoin visto como um DApp

Além de ser um sistema de dinheiro eletrônico ponto a ponto, o Bitcoin também é um aplicativo com o qual os usuários podem interagir. Por conta disso e com base nos critérios definidos, nós podemos afirmar que o Bitcoin é um DApp porque:

- O Bitcoin é open-source, nenhuma entidade (governo, empresa ou organização) controla o Bitcoin e todos os registros relacionados ao uso do Bitcoin são abertos.
- O Bitcoin gera seus tokens, os Bitcoins, com um algoritmo predeterminado que não pode ser alterado, e esses tokens são necessários para o funcionamento do Bitcoin. Os mineradores de Bitcoin são recompensados com Bitcoins por suas contribuições na segurança da rede Bitcoin.
- Todas as alterações no Bitcoin devem ser aprovadas por um consenso majoritário de seus usuários por meio do mecanismo de Proof of Work.

4.3.2. Classificação

A forma padrão usada para se classificar um DApp é de acordo com a Blockchain utilizada por ele, uma Blockchain própria ou de outro DApp. A partir dessa forma de classificação, surgem três tipos:

- **Tipo 1:** DApps que possuem a sua própria Blockchain. Bitcoin é o exemplo mais famoso de um DApp tipo I, mas Litecoin e outras "alt-coins" também são do mesmo tipo.
- **Tipo 2:** DApps que utilizam a Blockchain de um DApp do tipo 1. DApps do tipo 2 são protocolos e possuem tokens que são necessários para o seu funcionamento. Um exemplo de DApp desse tipo é o Omni Layer [OmniLayer 2020], que é uma plataforma para troca de assets ou moedas digitais que faz uso da Blockchain do Bitcoin. Dessa forma o Omni Layer busca expandir o Bitcoin criando novas funcionalidades, mas no fundo todas as transações do Omni são transações Bitcoin.
- **Tipo 3:** DApps que utilizam o protocolo de um DApp do tipo 2. DApps do tipo 3 também são protocolos e possuem tokens que são necessários para o seu funcionamento. Um caso seria uma aplicação que faz uso de um DApp tipo 2 para emitir suas moedas, como ocorre com o SAFE Network (SafeCoin) e o Omni.

4.3.3. Smart Contracts

O termo "Smart Contract" foi criado por Nick Szabo em 1996, antes mesmo do surgimento da primeira Blockchain, o Bitcoin, em 2008. Eles não tinham nada a ver com DAOs, ou trocas descentralizadas ou qualquer um desses tipos de construções que as pessoas tendem a pensar quando ouvem o termo. O conceito é na verdade bem mais simples do que os sistemas e aplicações que vem à mente quando pensamos em contratos inteligentes, uma vez que ele é derivado da ideia de contrato em si, que é: um conjunto de promessas acertadas em um "encontro de mentes", é a forma tradicional de formalizar um relacionamento. Note que, este conceito vai além de relações comerciais, podendo se estender à política e também a relacionamentos pessoais, como o casamento.

A partir do conceito de contrato, os contratos inteligentes são definidos da seguinte forma, de acordo com o criador do termo [Szabo 1996]. "Novas instituições e novas formas de formalizar as relações que compõem essas instituições são agora possibilitadas pela revolução digital. Eu chamo esses novos contratos de inteligentes, porque eles são muito mais funcionais do que seus ancestrais inanimados baseados em papel. Nenhum uso de inteligência artificial está implícito. Um contrato inteligente é um conjunto de promessas, especificadas em formato digital, incluindo protocolos dentro dos quais as partes cumprem essas promessas."

Dada a definição, ao desenvolver contratos inteligentes é necessário cumprir 4 objetivos para garantir que o contrato é funcional. [Szabo 1996]

- **Observabilidade:** As pessoas (ou coisas) envolvidas no contrato devem ser capazes de ver que a outra parte está cumprindo corretamente os termos do contrato e ser capazes de provar que elas mesmas estão cumprindo corretamente com a outra parte.
- **Verificabilidade:** Todas as partes de um contrato precisam da capacidade de provar ao árbitro escolhido do contrato que ele foi executado corretamente ou que uma parte (ou partes) violou suas obrigações no contrato. O árbitro em questão pode ou não ser uma terceira parte confiável.

- **Privacidade:** O contrato deve ser estruturado da forma mais privada possível. A quantidade de informações privadas sobre o contrato ou as partes envolvidas que são divulgadas além delas para o público ou outros terceiros deve ser mantida no mínimo necessário para a execução do contrato.

- **Aplicabilidade:** Deve haver algum mecanismo para garantir que as coisas sejam executadas corretamente, mesmo no caso de uma ou mais partes violar suas obrigações sob os termos do contrato e, também, o contrato deve ser estruturado de forma a tornar a necessidade de coerção para seu cumprimento muito improvável. Os contratos devem encorajar as partes a cumprir voluntariamente suas obrigações sob os termos.

4.3.4. Bitcoin Smart Contracts

A rede Bitcoin funciona como um árbitro distribuído gigante, impondo a execução adequada de contratos inteligentes sem depender de uma única autoridade central para fazê-lo. Ele fornece um mecanismo para que os contratos sejam observáveis, verificáveis e aplicáveis. A única qualidade de um contrato que ficou aquém historicamente é a privacidade - todos os termos dos contratos inteligentes do Bitcoin são públicos para todos verem. No entanto, preserva as identidades reais daqueles envolvidos em contratos. Em novembro de 2021 houve um soft-fork do Bitcoin onde foi ativado o Taproot, "raiz principal" em tradução livre, é uma melhoria proposta por Gregory Maxwell em 2018, e visa ocultar as cláusulas de um contrato, a menos que seja necessário aplicá-las, para isso faz uso de um protocolo de assinatura chamado MuSig [Maxwell et al. 2019].

Os contratos inteligentes que podem ser desenvolvidos no Bitcoin são escritos na linguagem Script, muitas vezes chamada de Bitcoin Script, que é uma linguagem do paradigma orientada a pilha(stack-oriented ou stack-based). Com isso, essa linguagem além de possuir um paradigma pouco usual os contratos desenvolvidos utilizando-a são pouco expressivos devido à limitações dela.

4.3.5. Ethereum Smart Contracts

No Ethereum, existem dois tipos de contas: contas de usuário e contas de contrato inteligente. As contas de usuário representam participantes, incluindo callers (que chamam funções de contratos inteligentes), deployers (que implantam contratos inteligentes no Ethereum) e miners (cujos nós trabalham para contribuir para o livro-razão). As contas de contrato inteligente representam o contrato inteligente que é um tipo de programa que é salvo e pode ser executado em Blockchains, também chamado de chaincode (código na corrente). Ethereum é a primeira Blockchain que fornece linguagem de programação Turing-completa para desenvolver contratos inteligentes [Wu et al. 2021].

As transações podem ser classificadas em duas dimensões, a do dado enviado ou do iniciador da transação. Na dimensão do dado, temos dois tipos, transferência de Ether, que representa a transferência de uma conta de usuário para outra, e execução de contrato, que representa a chamada de um método do contrato com seu input e Ether para pagar a taxa de execução de tal método. Na dimensão do iniciador, também temos dois tipos, transações externas, que são aquelas iniciadas por contas de usuários, e transações

internas, que são aquelas iniciadas por contas de contratos inteligentes. A Figura 4.4 resume os tipos de contrato.

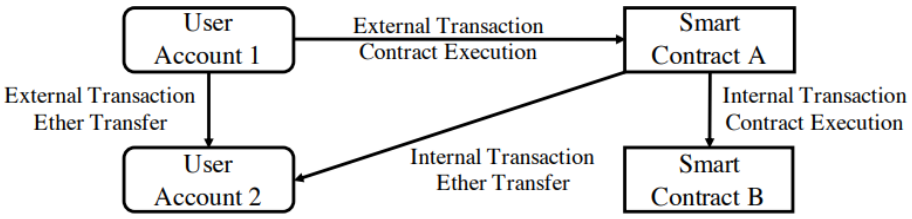

Figura 4.4. Tipos de Contrato no Ethereum [Wu et al. 2021]

O Blockchain Ethereum fornece recursos de computação e armazenamento por meio do mecanismo de contratos inteligentes. Portanto, os DApps que utilizam o Ethereum como Blockchain podem implantar contratos inteligentes para usar tais recursos. Em teoria, todos os processos e dados de um DApp baseado em Blockchain devem ser manipulados e armazenados na Blockchain para pura descentralização. No entanto, devido ao gargalo de desempenho da Blockchain, que será estudado em detalhes na próxima seção, os DApps atuais geralmente implementam apenas partes de sua funcionalidade na Blockchain. Como resultado, três tipos de arquiteturas são adotados pelos DApps Ethereum, tais arquiteturas estão dispostas na Figura 4.5.

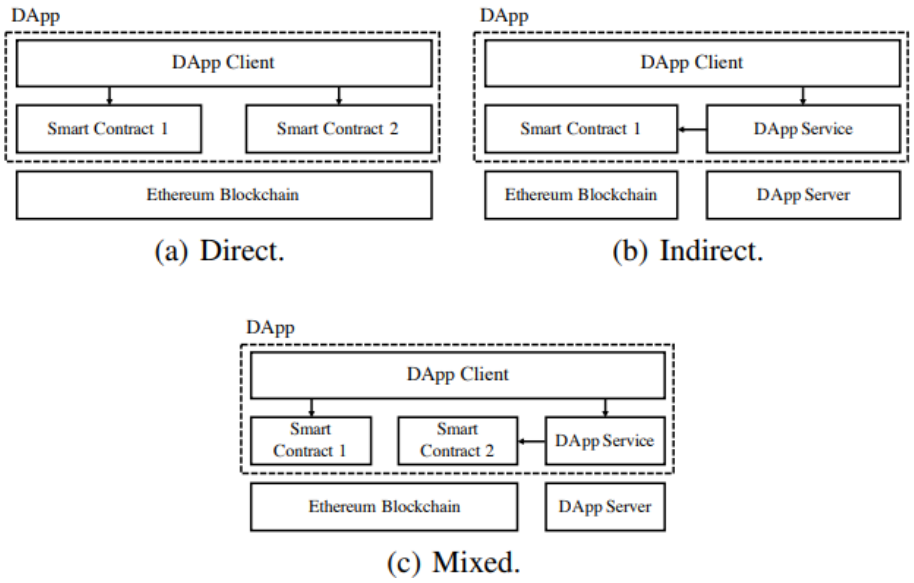

Figura 4.5. Tipos de Arquiteturas Ethereum DApp [Wu et al. 2021]

No Ethereum os contratos são escritos na linguagem Solidity, na qual definimos contratos, que é basicamente uma classe, funções e eventos. Uma vez pronto, o código-fonte do contrato inteligente é compilado em bytecode para ser implantado no Ethereum. Após a implantação, o contrato inteligente receberá um endereço. Qualquer conta pode realizar a implantação de um contrato, inclusive uma conta de contrato pode implantar

outros contratos, chamados de contratos filhos, a implantação feita dessa forma recebe o nome de “factory patterns”.

Todas as transações no Ethereum acarretam em uma taxa, chamada “gás”, usada para custear a operação da rede, em outras palavras, recompensar os mineradores pelo poder computacional cedido. No contexto dos contratos inteligentes o custo se dá de duas maneiras, na implantação e na execução, que no final das contas também são transações. A implantação do contrato pode ser vista como uma chamada a uma função especial construtora do contrato, portanto, o custo de implantação pode representar a complexidade do contrato. Já o custo da execução depende das instruções a serem executadas e se existem ou não chamadas a métodos de outros contratos, isto é, transações internas.

Visando reduzir os custos de implantação e execução dos contratos, na Blockchain Ethereum, o gás total de um bloco é limitado. Um contrato inteligente complexo pode custar muito gás e inviabilizar a sua implantação, ou seja, o bloco não conterá a transação. Além disso, quanto maiores forem os custos de execução do contrato, menor será o rendimento das execuções dele e mais tempo os usuários terão que aguardar pelas confirmações das execuções. Porém, tal abordagem não barateou o custo de operação, ela apenas impediu que ele atingisse determinados valores, sendo assim, o custo continua alto e supondo que haja muito recurso disponível para manter um determinado contrato inteligente, ele será limitado pelo total de gás permitido no bloco.

4.4. Soluções de escalabilidade na Blockchain

Escalabilidade é um termo utilizado em sistemas, que diz respeito à capacidade de um sistema crescer, tendo como intenção atender mais usuários ou adicionar mais funcionalidades. Sendo assim, um sistema é dito escalável quando o seu desempenho aumenta proporcionalmente com o seu poder computacional.

Podemos dividir escalabilidade em duas vertentes, escalabilidade horizontal e escalabilidade vertical. Na escalabilidade horizontal aumentamos o poder computacional do sistema adicionando nós ao sistema, ou seja, adicionando uma nova máquina. Já na escalabilidade vertical aumentamos o poder computacional do sistema melhorando um nó existente, como por exemplo, adicionando mais memória RAM. Porém há sistemas em que o aumento no poder computacional não acarreta um aumento no desempenho do sistema, nestes casos é necessário rever a arquitetura.

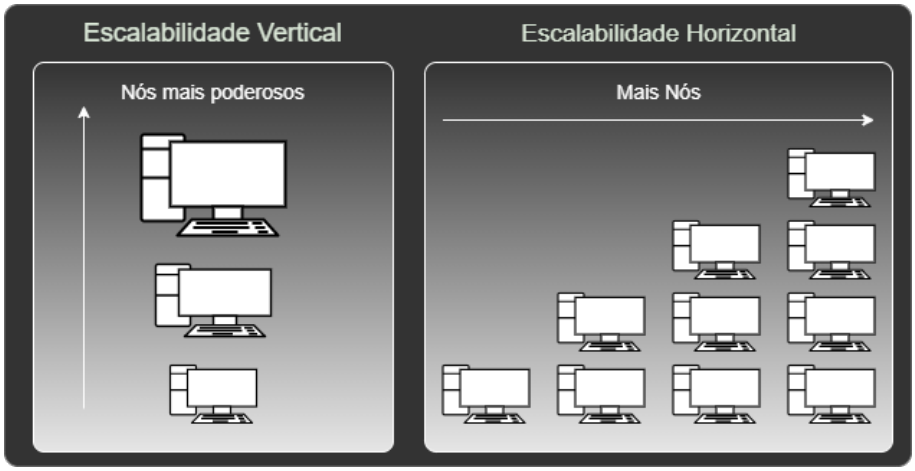

Figura 4.6. Tipos de Escalabilidade

Soluções de Blockchain tradicionais, como por exemplo Bitcoin e Ethereum, apresenta problemas de escalabilidade, no que diz respeito ao tempo para confirmar uma transação e vazão (quantas transações são confirmadas por segundo). Em especial, quando comparado a sistemas de cartões de crédito. Esses, por por sua vez, são capazes de processar mais de duas mil transações por segundo. Nos sistemas de cartões de crédito, a escalabilidade pode ser tanto vertical, quanto horizontal, pois trata-se de um sistema centralizado. Portanto, a entidade central pode analisar e investir recursos para melhorar o desempenho da forma que julgar conveniente.

Como a Blockchain é uma rede P2P, a sua escalabilidade vai se dar principalmente da forma horizontal, no qual novos validadores (mineradores) vão se juntar à rede e o poder computacional do sistema irá aumentar. Em se tratando de prova de trabalho, o modelo que é diretamente afetado pelo poder computacional, aumentar a capacidade de um nó não gera ganho para a rede, apenas uma vantagem para este nó em detrimento dos demais. Para os demais modelos, essa desigualdade é irrelevante para a escalabilidade, uma vez que um nó possui apenas um voto ou uma maior probabilidade de ser sorteado.

Nas Blockchains tradicionais, apesar de termos esse aumento de poder computacional, nós não temos um aumento proporcional no desempenho do sistema. No caso do Bitcoin, nem mesmo há um aumento, o desempenho da rede se mantém constante, independentemente do poder computacional. Levanta-se, então, um questionamento sobre o porque a Blockchain tradicional não escalar. Para responder esta pergunta, devemos observar as decisões de arquitetura desta classe de Blockchain. Decisões na arquitetura de sistemas são feitas com base em tradeoff. Tratando-se de Blockchain e performance, devemos olhar basicamente três pontos principais: a quantidade de blocos que cada nó armazena, o algoritmo de consenso e o tamanho dos blocos da cadeia. Para a análise dos pontos mencionados, teremos como caso de estudo a Bitcoin, dado que, outras Blockchains que se encaixam na categoria de Blockchain tradicional foram muito influenciadas por ela.

O primeiro ponto diz respeito à quantidade de blocos que cada nó armazena, ou seja, o quanto da cadeia de blocos cada nó guarda. No caso da Bitcoin, cada nó armazena toda a cadeia de blocos. Isto traz como benefícios, uma maior disponibilidade das

informações, uma vez que ela está replicada em todos nós da rede. Porém, traz também desvantagens como uma maior dificuldade de manter a coerência destes dados e um desafio de performance, pois toda vez que for feita uma consulta estaremos consultando a base de dados completa.

Como segundo ponto temos o algoritmo de consenso. Como mencionado anteriormente, o algoritmo de consenso é responsável por manter a consistência dos dados armazenados pelos nós da rede. Essa tarefa se torna consideravelmente mais lenta uma vez que cada nó armazena a cadeia completa. No caso da Bitcoin o algoritmo de consenso utilizado é o Proof of Work, que é executado por máquinas que são chamadas de mineradores. Fundamentalmente, ele funciona como um puzzle computacional, no qual deve-se simplesmente buscar por um valor que faça parte do conjunto de soluções do puzzle. Esta ideia que faz com que o algoritmo de consenso funcione, mas também é o seu maior problema, pois acabamos tendo um desperdício de poder computacional.

Quando um minerador na Bitcoin descobre a solução para uma prova, ele envia uma mensagem *Broadcast* para avisar aos outros nós da rede que aquela prova foi resolvida e o bloco pode ser fechado. Entretanto, existe um tempo de propagação dessa mensagem e, enquanto isso, um outro nó pode encontrar uma outra solução para a prova. A partir daí podemos acabar gerando duas cadeias de blocos diferentes, dado que os mineradores podem pegar um conjunto de transações diferentes para tentar formar um bloco. Este cenário é chamado de *fork*. É importante ressaltar que, o *fork* pode ser enxergado como uma anomalia, pois a existência de nós com cadeias de blocos diferentes vai contra a ideia de consenso. Esta situação é resolvida simplesmente adotando a maior cadeia como a correta. Assim temos novamente um desperdício computacional, pois, apenas um *fork* é escolhido para representar a cadeia, desperdiçando todo o poder computacional investido na construção dos outros *forks*.

Por fim, o terceiro ponto temos que, na Bitcoin foi definido um tamanho máximo de 1MB para os blocos. Este tamanho para o bloco parecia viável quando a Bitcoin foi concebida. Com o passar dos anos, a criptomoeda se popularizou muito e por conta disso a quantidade de transações aumentou também, fazendo com que haja uma demora ainda maior ao confirmar as transações. É necessário que um bloco seja fechado contendo todas as transações, e como os blocos são muito pequenos se comparados ao volume de transações disponíveis, acaba formando-se um grande volume de transações à espera de um bloco para incorporá-las.

Com base nesses três pontos, é possível entender porque a Bitcoin não escala. Nesta temos um atraso de confirmação das transações, que se dá justamente por conta do tempo necessário para os mineradores solucionarem o puzzle do Proof of Work. Também por conta do tamanho dos blocos, esse tempo de confirmação é maior que 10 minutos. Além da demora para a primeira confirmação da transação, temos a situação dos *forks*, que podem fazer com que o *fork* que já tinha processado determinada transação seja descartado em detrimento de um que ainda não a processou. No fim das contas a Bitcoin só é capaz de processar apenas aproximadamente 4 transações por segundo[L. 2019]. Isso destoa muito, se comparado a um sistema de cartão de crédito que atualmente é capaz de processar aproximadamente 1700 transações por segundo [L. 2019], como no caso de sistemas de cartão de crédito atuais. Além de uma vazão muito maior, sistemas de cartão

de crédito também possuem um tempo de confirmação de transação menor, apresentando, então, alta vazão e baixa latência. Tratando-se de criptomoedas e Blockchain tem-se como meta um desempenho tão bom quanto o dos sistemas de cartões de crédito.

Para reduzir a disparidade com os sistemas de cartão de crédito, processar transações com uma taxa próxima a eles, soluções de Blockchain usam uma combinação de técnicas *On-chain* ou *layer-1* e *Off-chain* ou *layer-2*. As soluções *On-chain* são propostas que alteram o protocolo padrão da Blockchain visando a escalabilidade. Um exemplo desse modelo de proposta é a técnica de *Sharding*. As soluções *Off-chain* são implementadas fora da rede principal adicionando camadas de processamento que independem do núcleo da Blockchain. Os Rollups são tecnicas *Off-chain* [Zhou et al. 2020a, Tianchen et al. 2021].

4.4.1. *Sharding*

À medida que a popularidade da Blockchain cresce, o mesmo acontece com o volume transacional gerenciado pela rede e a carga de trabalho é ajustada para manter a dificuldade. Se pensarmos em uma Blockchain como um banco de dados compartilhado, à medida que mais e mais dados são adicionados à rede, precisa-se encontrar novas maneiras de processar todos esses dados com eficiência e rapidez. É aí que o sharding pode ajudar. [Mearian 2019]

O *sharding* não é uma técnina nova, ele foi originalmente desenvolvido para melhorar a performance de bancos de dados muito grandes, e somente, recentemente, passou a ser explorado como solução para escalabilidade na Blockchain. A técnica consiste na fragmentação ou divisão horizontal de bancos de banco de dados, permitindo que processem mais transações por segundo. O *sharding* divide o sistema em partições menores, conhecidas como "*shards*". Cada fragmento é composto por seus próprios dados, tornando-o distinto e independente quando comparado a outros fragmentos, permitindo um melhor manuseio dos mesmos, tornando-os menos pesados e mais fáceis de operar.

O *sharding* pode ajudar a reduzir a latência ou lentidão de uma rede, pois divide uma rede Blockchain em fragmentos independentes. No entanto, existem algumas questões de segurança em torno do *sharding* que podem ser exploradas por adversários. Ao subdividir os nós em *shards*, o poder de *hashing* de cada grupo diminuirá consideravelmente. Isso gera um problema de segurança ao permitir que um agente mal-intencionado execute um ataque com mais facilidade. Uma situação que coloca em risco a segurança e a integridade das informações.[bit2me 2020]

Figura 4.7. Tradeoff em Sistemas Distribuídos

Devemos ter em mente o tradeoff que estamos fazendo ao implementar uma Blockchain que faz uso de *sharding*, pois como ilustrado na Figura 4.7, é impossível atingir os três extremos. O *sharding* por sua vez, tende à escalabilidade e descentralização, introduzindo assim, desafios de segurança à Blockchain. Temos então, dois principais desafios introduzidos pelo *sharding* [Wang et al. 2019]:

1. Distribuir os nós de maneira uniforme nos *shards*, de forma a garantir que a maioria deles seja honesta com alta probabilidade;

2. Como garantir que um adversário não obtenha vantagem significativa, enviesando as operações ou criando identidades Sybil, que é quando um único nó consegue burlar a política de identidade e se passar por vários simultaneamente [Douceur 2002].

Outro problema inserido na Blockchain tradicional com a implementação de partições ocorre porque os nós, quando atribuídos a um subgrupo, não têm acesso ao saldo dos nós que pertencem aos outros subgrupos. Tornando necessário criar um protocolo para resolver as transações entre os subgrupos.

Portanto, para uma Blockchain se manter segura e funcional, enquanto faz *sharding*, ela deve, em geral, ser composta por cinco componentes [Wang et al. 2019], que são:

1. **Identity establishment and committee formation**: Para se juntar ao protocolo, cada nó precisa estabelecer uma identidade, como uma chave pública, um endereço IP e uma solução de prova de trabalho (PoW). Cada nó é então atribuído a um comitê correspondente à sua identidade estabelecida. Nesse processo, o sistema precisa prevenir a identidade Sybil. No entanto, uma Blockchain permissionada não requer este processo;

2. **Overlay setup for committees**: Uma vez que os comitês são formados, cada nó se comunica para descobrir as identidades de outros nós em seu comitê. Para uma Blockchain, uma sobreposição de um comitê é um subgrafo totalmente conectado contendo todos os membros do comitê. Normalmente, este processo pode ser feito com um protocolo de fofoca(*gossip protocol*);

3. **Intra-committee consensus**: Cada nó dentro de um comitê executa um protocolo de consenso padrão para concordar com um único conjunto de transações. Nesse

processo, todos os membros honestos devem concordar com o bloco proposto em seu comitê;

4. **Cross-shard transaction processing**: A transação deve ser confirmada atomicamente em todo o sistema. Para transações cross-shard, os shards relacionados precisam obter consistência. Normalmente, esse processo requer um tipo de transação de "retransmissão" para sincronizar entre os fragmentos relacionados;

5. **Epoch reconfiguration**: Para garantir a segurança dos shards, os shards precisam ser reconfigurados, a cada período de tempo chamado epoch, exigindo uma aleatoriedade. Essa aleatoriedade será usada na próxima epoch.

As soluções que utilizam Sharding podem ainda ser divididas em duas categorias, Partial Sharding e Complete Sharding, dependendo de como a proposta aborda o processamento, armazenamento e comunicação [Hong et al. 2021]. No modelo de partição parcial, os grupos trabalham em conjunto para gerar uma cadeia única partilhada por todos. Enquanto no completo, cada grupo possui uma cadeia de blocos distinta. Por fim, na categoria outras soluções, iremos analisar propostas que destoam dos padrões vistos nas outras duas. A Figura 4.8 ilustra a diferença entre partial sharding e complete sharding.

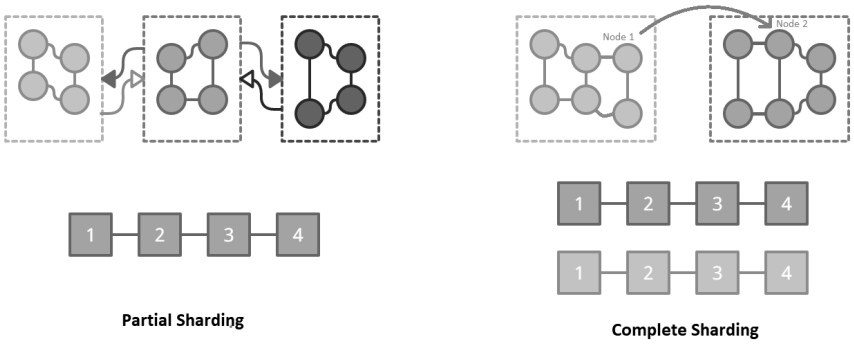

Figura 4.8. Estrutura dos modelos de *sharding*

4.4.1.1. Partial Sharding

As primeiras propostas de *sharding* tentam resolver o problema da escalabilidade dividindo o poder computacional da rede, criando grupos capazes de minerar blocos de forma independente. Porém mantêm a estrutura base da Blockchain onde todos os nó da rede possuem uma cópia exata da Blockchain. Uma vez que cada nó recebe e armazena informações completas do sistema, não existem transações *cross-shard* no sistema, mas os nós ainda sofrem de armazenamento pesado e sobrecarga de largura de banda [Hong et al. 2021]. Uma vez que a fragmentação ocorra em apenas uma das três dimensões processamento,armazenamento e comunicação esse modelo é classificado com partial *sharding*.

4.4.1.2. Complete Sharding

Sharding completo são aqueles que atingem a fragmentação nas bases computação, armazenamento e comunicação. A divisão do poder computacional permite, assim como no modelo parcial, aumentar o número de blocos minerados ao permitir que as partições minerem simultaneamente. Para o armazenamento o objetivo é diminuir o tamanho da cadeia armazenada em cada nó pois a mesma cresce indefinidamente. O consenso baseado em BFT é muito presente nos modelos que visam a escalabilidade pois as provas demandam tempo e esforço, impedindo que muitos blocos sejam minerados. Esses modelos de consenso requerem um grande volume de comunicação e começam a perder sua eficiência em redes maiores. A fragmentação da comunicação busca resolver esse problema.

4.4.2. Soluções *Off-chain*

A estrutura descentralizada da Blockchain oferece benefícios para algumas aplicações do cotidiano. Por exemplo um serviço de carona, como o Uber, contaria com uma redução nas taxas devido a ausência de um intermediário [Sá 2022]. As soluções de *layer-2* são um conjunto de técnicas com objetivo de aumentar a velocidade e vazão das transações. Além disso essas técnicas reduzem o gas das transações que seria um grande empecilho para o surgimento de aplicações integradas à Blockchain.

O Rollup é uma técnica que consiste em empacotar e compactar dados de transações, chamados de lote (batch), off-chain e, em seguida, armazená-los on-chain, Rollup é um termo geral para a tecnologia de Layer-2 que adota esse framework. Esta é uma solução que teve como origem o Sidechain e foi evoluindo até culminar no estado atual, como mostra a Figura 4.9 [Tianchen et al. 2021].

Figura 4.9. Evolução do Rollup [Tianchen et al. 2021]

Side Chains são cadeias que processam dados em uma velocidade maior ao reduzir a descentralização e segurança. Uma vez que a side chain atua de forma independente, a segurança da rede principal é preservada. Rootstock [Lerner 2015] é uma side chain associadas ao Bitcoin responsável por habilitar smart contractsna rede. Polygon usa Proof of Stake para processar transações de forma mais rápida e guarda snapshots dos resultados no Ethereum.

A Lightning Network é uma rede executada sobre a Bitcoin. Essa rede permite que dois usuários abram um canal de troca par a par ao aportarem moedas na rede principal. Os usuários passam a poder trocar livremente até o limite aportado. No Bitcoin são registrados somente as transações de abertura e fechamento do canal.

Plasma é um framework de cadeias secundárias que atuam sobre a rede Ethereum. O Plasma difere das Side Chains pois é contruida sobre um contrato inteligente. Cada cadeia da Plasma é um contrato customizado para um proposito. [Binance 2020]

4.4.2.1. Rollups

A técnica de rollups consiste em acumular as transações em lotes antes de adiciona-los a Blockchain. As transações são processadas fora da rede principal, compactadas em um lote e então adicionadas a Blockchain. Um smart contract registrado na rede mantém os lotes em uma árvore Merkle. Toda vez que um novo lote é adicionado, caso o lote gere uma nova árvore, novo hash, a árvore é registrada na Blockchain.

Os dados processados fora da rede principal devem ser verificados antes de serem adicionados para preservar a confiabilidade. A forma de verificar os dados gerou duas categorias para a técnica de Rollups, uma baseada em Fraud Proof (Optimistic Rollups) e outra baseada Validity Proofs (Zero Knowledge Rollups) [Buterin 2021].

No Optimistic Rollup, o contrato de rollup acompanha todo o histórico de estado das raízes e o hash de cada lote. Se alguém descobrir que um lote gerou uma raiz incorreta, eles podem publicar uma prova na Blockchain, provando que o lote foi calculado incorretamente. O contrato verifica a prova e reverte esse lote e todos os lotes posteriores.

No Zero Knowledge Rollup, cada lote inclui uma prova criptográfica chamada ZK-SNARK (por exemplo, usando o protocolo PLONK), que prova que a raiz gerada pelo lote é o resultado correto da execução do lote. Não importa quão grande seja o cálculo, a prova pode ser verificada muito rapidamente na cadeia.

4.5. Cartesi

Continuando com soluções de escalabilidade na Blockchain, nesta seção, nos aprofundaremos em uma delas, a Cartesi. A solução é apresentada e são detalhadas suas principais tecnologias, a Cartesi Machine e o Cartesi Rollups, que serão utilizadas na criação dos DApps na seção seguinte. A principal referência para esta seção é a documentação oficial, disponível no site oficial da Cartesi [Cartesi 2022].

A Cartesi é uma plataforma de layer-2 para o desenvolvimento de aplicações descentralizadas. Ela permite que DApps sejam desenvolvidos utilizando linguaguens de programação convencionais ao oferecer um sistema operacional Linux acoplado a uma infraestrutura Blockchain. Por exemplo, contratos inteligentes para a Ethereum devem ser escritos utilizando Solidity e, ao utilizarmos a Cartesi podemos o escrever com linguagens mais familiares, como Python ou C++. Além disso, diversos softwares, ferramentas, bibliotecas e serviços utilizados para a criação de aplicações tradicionais ficam a disposição do desenvolvedor, garantindo um ambiente de desenvolvimento familiar, diminuindo a barreira de entrada para a criação de aplicações descentralizadas e livrando os desenvolvedores das limitações e especificidades impostas por cada Blockchain.

Na Cartesi, tem-se duas tecnologias principais. A primeira é a Cartesi Machine, uma máquina virtual que permite computação verificável usando um sistema operacional Linux. A segunda é o Cartesi Rollups, que fornece uma estrutura geral para criação de DApps com a combinação da Cartesi Machine com Optimistic Rollups.

Assim, podemos dizer que a Cartesi busca:

- Oferecer um sistema operacional completo para aplicativos Blockchain;

- Solucionar o problema de escalabilidade usando o Cartesi Rollups para suportar cálculos complexos;
- Possibilitar o desenvolvimento de DApps de qualquer complexidade usando ferramentas de desenvolvimento convencionais em cima de redes Blockchain estabelecidas, como Ethereum, Polygon, Avalanche e BNB Smart Chain.

4.5.1. Cartesi Machine

O componente mais importante para a solução Cartesi, a Cartesi Machine foi criada para que desenvolvedores possam utilizar ferramentas tradicionais para a criação de aplicativos descentralizados escaláveis.

Diferentemente das VM tradicionais, a Cartesi Machine não se destina a ser interativa. Em vez disso, ela deve inicializar, abrir o sistema operacional, executar algum software predefinido e depois parar [Cartesi 2022].

Os desenvolvedores costumam trabalhar com cadeias de ferramentas que operam num nível de abstração mais alto. Essas cadeias de ferramentas os isolam de detalhes de hardware irrelevantes e até mesmo dos detalhes de um determinado sistema operacional. Inventar uma nova arquitetura ad-hoc exigiria a portabilidade de uma cadeia de ferramentas e sistema operacional. Em vez disso, as Cartesi Machines são baseadas em uma arquitetura comprovada para a qual uma cadeia de ferramentas padrão e um sistema operacional já são acessível.

Para que a Cartesi Machine seja verificável, a Blockchain deve, portanto, hospedar uma implementação de referência de toda a arquitetura. Se for confiável, essa implementação deve ser facilmente auditável. Para isso, tanto a arquitetura quanto a implementação devem ser abertas e relativamente simples. Juntos, esses requisitos apontam para o RISC-V [Teixeira and Nehab 2018]. O emulador de Cartesi Machine implementa o ISA RV64IMASU da RISC-V e é escrito em C/C++ com dependências POSIX restritas às instalações de terminal, processo e mapeamento de memória. O RISC-V garante estabilidade simulando hardware real, podendo trabalhar em um nível mais baixo, com mais ferramentas de desenvolvimento.

Escalabilidade

A Escalabilidade nas Cartesi Machines existe, uma vez que elas podem processar quantidades praticamente ilimitadas de dados e em um ritmo 4 ordens de magnitude mais rápido. Isso é possível porque as Cartesi Machines são executadas off-chain, livres da sobrecarga imposta pelos mecanismos de consenso usados pelas Blockchains. Em um cenário típico, uma das partes envolvidas em um DApp executará a Cartesi Machine off-chain e relatará seus resultados à Blockchain. Diferentes partes não precisam confiar umas nas outras porque a plataforma Cartesi inclui um mecanismo automático de disputa para as Cartesi Machines. Todos os interessados repetem o cálculo off-chain e, caso seus resultados não concordem, entram em uma disputa, que o mecanismo garante ser sempre vencida por uma parte honesta contra qualquer parte desonesta, isso só é possível pois:

- As Cartesi Machines são independentes — elas funcionam isoladas de qualquer influência externa na computação;
- As Cartesi Machines são reprodutíveis — Duas partes realizando a mesma compu-

tação sempre obtêm exatamente os mesmos resultados;
- As Cartesi Machine são transparentes — Elas expõem todo o seu estado para inspeção externa.

Produtividade

O desenvolvimento de software moderno envolve a combinação de dezenas de componentes prontos para uso. A criação desses componentes exigiu o esforço conjunto de uma comunidade mundial ativa ao longo de várias décadas. Todos eles foram desenvolvidos e testados usando conjuntos de ferramentas bem estabelecidos (linguagens de programação, compiladores, linkers, profilers, depuradores, etc.), e contam com vários serviços fornecidos por sistemas operacionais modernos (gerenciamento de memória, multitarefa, sistemas de arquivos, rede, etc). Ao disponibilizar um sistema operacional com suporte a tais ferramentas, a Cartesi Machine é capaz de preservar essa produtividade no ambiente de desenvolvimento de DApps.

Caso fosse necessária a utilização massiva de ferramentas desenvolvidas especificamente para desenvolvimento em Blockchain e o abandono de ferramentas já conhecidas, o fator econômico seria negativo a ponto de possivelmente tornar inviável a adoção de nova tecnologia. Além da economia em se utilizar ferramentas que já se tenha domínio, também obtém-se ganho no quesito segurança, utilizando tecnologias exaustivamente testadas por diversos desenvolvedores.

Arquitetura

A arquitetura da Cartesi Machine pode ser vista de diferentes perspectivas, dependendo de qual nível estamos, são elas:

- **Host Perspective:** Este é o ambiente fora do emulador Cartesi Machine. É mais relevante para desenvolvedores configurando Máquinas Cartesi, executando-as ou manipulando seu conteúdo. Inclui a API do emulador em: C++, Lua, gRPC e a interface de linha de comando.
- **Target Perspective:** Este é o ambiente dentro da Cartesi Machine. Ele engloba a arquitetura RISC-V da Cartesi, bem como a organização do sistema operacional Linux embutido que é executado em cima dela. É mais relevante para os programadores responsáveis pelos componentes do DApp que são executados off-chain, mas devem ser verificáveis. A cadeia de ferramentas de compilação cruzada e as ferramentas usadas para construir o kernel do Linux e os sistemas de arquivos raiz do Linux incorporados também são importantes nessa perspectiva, embora sejam usadas no host;
- **Blockchain Perspective:** Esta é a visão que os contratos inteligentes têm das Máquinas Cartesi. Consiste quase exclusivamente na manipulação de hashes criptográficos do estado das Máquinas Cartesi e partes das mesmas. Em particular, usando apenas operações de hash, a Blockchain pode verificar asserções sobre o conteúdo do estado e obter o hash de estado que resulta de modificações no estado.

4.5.2. Cartesi Rollups

Os Cartesi Rollups são a versão da Cartesi de Optimistic Rollups. O objetivo é mover a maior parte da computação para fora da Blockchain, usando-a como provedora de

dados para a execução off-chain das aplicações. Pode-se encontrar mais detalhes em [Moura 2021, Argento 2021]. Essa solução possui componentes em layer-1 e em layer-2. Na layer-1, temos os contratos inteligentes dos Cartesi Rollups. Eles fazem a mediação dos contratos inteligentes de propriedade externa com os componentes da layer-2, onde a computação é realizada pelos Cartesi Nodes.

O Cartesi Node é o componente de layer-2 responsável por executar os cálculos das aplicações utilizando uma Cartesi Machine. O Cartesi Node faz o uso de um middleware para ler dos dados dos contratos inteligentes em layer-1 e fornecê-los à Cartesi Machine, dessa forma este middleware estabelece a comunicação entre os layers 1 e 2. Temos dois tipos de Cartesi Nodes e eles diferem na forma como interagem com os contratos de rollup em layer-1, são eles: validadores e usuários. Um Cartesi Node validador é responsável por garantir que os dados enviados para a Blockchain sejam confiáveis, lutando contra validadores desonestos para assegurar a que reivindicações honestas sejam cumpridas. Já um Cartesi Node usuário não realiza essas validações, ele se preocupa apenas em avançar o estado de sua máquina, consumindo os dados da Blockchain.

Ao invés de verificar cada nova atualização do estado da Cartesi Machine, os Cartesi Nodes validadores fazem isso fim de cada *epoch*. Um *epoch* é um conjunto de entradas que sejam agrupadas que seguem o mesmo ciclo. Com a verificação dessa forma, evitamos uma interação excessiva com a Blockchain. O back-end da aplicação lê a carga de entrada e a processa.

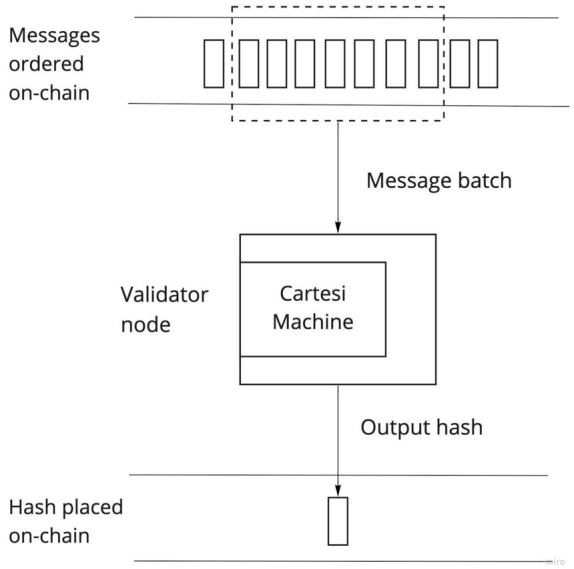

Figura 4.10. Agrupamento das transações e armazenamento do hash de saída [Moura 2021]

Seguindo o protocolo do Cartesi Rollups, quando é processado o conteúdo de cada epoch, um Cartesi Node validador insere na Blockchain o hash que representa o estado do contrato de layer-2 e, então inicia-se o período de desafio. Caso não ocorram disputas o estado representado pelo hash é assumido. Caso contrário, o mecanismo de disputa é acionado para garantir que a reivindicação correta seja aplicada.

Devido a natureza determinística das Cartesi Machines, qualquer Cartesi Node interessado no cálculo da aplicação pode perceber um comportamento desonesto de um outro Node alegando uma saída falsa para a computação.

Diferentemente de como ocorre as transações envolvendo contratos inteligentes de layer-1 na Ethereum, no Cartesi Rollups, as Cartesi Machines são criadas especificamente para executar contratos únicos de layer-2, ao invés de processar todas as transações como faz a EVM. Assim, cada contrato tem seu conjunto especifico de Cartesi Nodes validadores, permitindo um ganho de escalabilidade e mantendo a segurança desde que tenhamos pelo menos um validador honesto.

Vouchers e Notices

A Cartesi Machine pode responder e interagir com contratos em layer-1 com os chamados *vouchers*. Um voucher é uma combinação de um endereço de destino e uma carga em bytes. Em sua execução, um voucher envia uma mensagem para o endereço de destino com a carga como parâmetro. Além dos vouchers, a Cartesi Machine também pode produzir *notices*. Uma notice é uma caraga em bytes que é enviada para fins informativos.

4.5.2.1. Componentes em layer-1

Na Blockchain, o Cartesi Rollups utiliza contratos inteligentes para mediar a relação dos componentes em layer-2 com outros contratos inteligentes e contas externas. A seguir, apresentamos uma breve descrição de cada dos componentes de layer-1.

Contrato de Entrada

Este contrato recebe as entradas do usuário para a execução da computação. O conteúdo das entradas é sempre guardado em *calldatas* na Blockchain, mantendo um único hash para cada entrada de uma epoch ativa. Este hash resume tanto a entrada em si quanto seus metadados. As entradas não são interpretadas em layer-1, é papel da Cartesi Machine, em layer-2, detectar se há algum problema no conteúdo de cada entrada. Dessa forma, os recursos de execução em layer-1 são econimizados.

Contrato de Saída

O contrato de saída pode tanto verificar a validade do conteúdo de uma notice, quanto executar um voucher. Este módulo garante que apenas as saídas sugeridas pela layer-2 e finalizadas na layer-1 possam ser executadas. Ele faz isso exigindo que uma prova seja enviada com a chamada de execução.

Portal

O Portal é a forma de transferir ativos da layer-1 para a layer-2. Com os ativos em layer-2, pode-se transasioná-los de forma menos custosa, utilizando entradas simples que são entendidas pela lógica Linux. Pode-se pensar no Portal como uma conta bancária, de propriedade da máquina off-chain.

Validator Manager

Este componente é responsável pelo gerenciamento das reivindicações, conceder

permissões e aplicar punições.

Dispute Resolution

Este módulo tem o papel de mediar as interações entre os Cartesi Nodes validadores interessados na disputa. Uma disputa ocorre quando dois os mais validadores reivindicam atualizações de estados diferentes para uma mesma epoch.

Cartesi Rollups Manager

Este componente realiza a sincronização entre os demais componentes. Ele é responsável pela mediação entre o componente Dispute Resolution e o Validator Manager. Além disso, ele notifica todos os outros módulos qualquer alteração de fase.

4.5.2.2. Componentes em layer-2

Como citado anteriormente, a layer-2 do Cartesi Rollups é constituída de Cartesi Nodes. Agora exibiremos uma pequena descrição de alguns componentes internos de um Cartesi Node.

State Server

Este é um serviço com papel de assegurar que os outros componentes tenham acesso a uma visão consistente da Blockchain. Ele monitora todas as atividades relevantes dos contratos inteligentes do Cartesi Rollups, guardando as informações que são emitidas pelas Blockchain.

Server Manager

Este componente administra a Cartesi Machine, enviando entradas e obter as saídas produzidas. É responsável por iniciar e parar a máquina conforme apropriado, bem como fornecer uma API para os outros módulos para consultar o estado da máquina.

Rollups Dispatcher

Este módulo é responsável por interpretar o estado atual dos contratos inteligentes da Cartesi Rollups, ele informa o Gerenciador de Servidores sobre quaisquer entradas recebidas e, no caso do Validator Nodes, também envia transações correspondentes a reclamações de atualização do estado. Ele também lidará com quaisquer disputas caso surjam.

Rollups Indexer

Este serviço obtém informações da layer-1 pelo State Server e os dados produzidos pelo back-end da aplicação para consolidar o estado de um Cartesi Node.

Query Server

Para que usuários e clientes recuperem vouchers e notices produzidos pelo back-end da aplicação, este componente provê uma API GraphQL para consulta do estado do Cartesi Node.

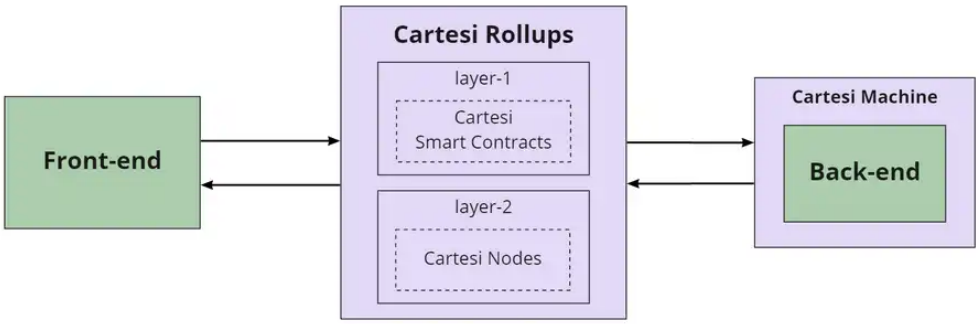

Figura 4.11. Componentes dos Cartesi Rollups [Cartesi 2022]

Perceba, na Figura 4.11, que, de forma distinta à aplicações convencionais, em um DApp Cartesi, não temos a comunicação direta entre front-end e back-end. Ao invés disso, o front-end envia as entradas para o framework Cartesi Rollups, que as dispõe para as instâncias de back-end executadas em cada Cartesi Node. Após o processamento do back-end, as saída retornam ao Cartesi Rollups que as verifica e, possivelmente, as corrige e as disponibiliza para o front-end. Essa comunicação é realizada por um conjunto de APIs HTTP da Cartesi.

Utilizando uma série de *endpoints* HTTP o back-end comunica ao Cartesi Rollup que qualquer processamento ou inicialização anterior foi concluída, e que o back-end está pronto para lidar com a próxima solicitação, que por sua vez, pode envia uma consulta sobre o estado atual do aplicativo ou fornecer uma entrada que será processada pelo back-end e atualizará seu estado. Assim, o back-end pode acessar endpoints HTTP para informar os resultados da computação realizada.

O front-end da aplicação também precisa acessar o Cartesi Rollups para enviar solicitações do usuário e recuperar as respostas correspondentes computadas pelo back-end. Essa comunicação pode ocorrer de diferentes formas:

- **AddInput:** uma transação regular na layer-1 para enviar os dados de entrada para os contratos inteligentes de rollup.
- **QueryOutputs:** envia uma consulta à layer-2 para recuperar notices e vouchers.
- **InspectState:** envia uma consulta a um Cartesi Node para receber o estado de um aplicativo.
- **ExecuteVoucher:** uma transação regular na layer-1 solicitando a execução de contrato inteligente de rollup.

4.6. Cartesi Rollups DApp: Uma Abordagem Hands On

Com base no que foi visto a respeito do Cartesi Rollups, a arquitetura de um DApp Cartesi e como ocorre a comunicação do front-end com o back-end, vamos para o desenvolvimento prático de uma aplicação. Parte importante deste desenvolvimento é a idealização do DApp e, para tal, utiliza-se o modelo de desenvolvimento exposto na Figura 4.12, seguindo a linha de desenvolvimento da esquerda para a direita. [Cartesi 2022]

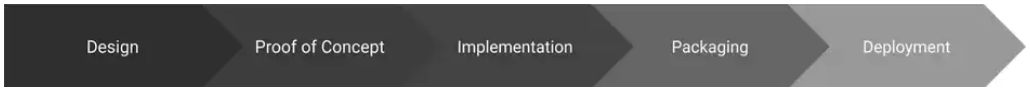

Figura 4.12. Ciclo de vida do DApp [Cartesi 2022]

4.6.1. Modelo de Desenvolvimento

O modelo proposto visa aproximar o desenvolvimento de um DApp do de uma aplicação "tradicional", ou seja, não descentralizada. O objetivo de tal abordagem é tornar o desenvolvimento familiar, mesmo para aqueles que nunca desenvolveram um DApp, permitindo a utilização de ambientes de desenvolvimento mainstream na maior parte do processo. Com isso em mente, nos aprofundaremos em cada uma das etapas deste modelo.

Design

Inicialmente, define-se o design conceitual da aplicação, especificando o escopo principal da lógica de front-end e back-end, bem como os requisitos gerais de tecnologia.

Proof of Concept

Etapa na qual implementa-se uma prova de conceito visando garantir a viabilidade de execução da lógica de back-end pretendida e pilhas de tecnologia dentro de uma Cartesi Machine. Este estágio pode envolver a compilação cruzada de uma biblioteca para o sistema operacional Linux RISC-V da Cartesi e a verificação de que o desempenho da execução do software pretendido na Cartesi Machine é satisfatório.

Implementation

Este é o estágio em que a lógica de front-end e back-end é realmente implementada, representando a maior parte do trabalho no ciclo de vida de desenvolvimento do DApp. Nesta fase, o desenvolvedor do DApp fará uso do Cartesi Rollups Host Environment para implementar os módulos front-end e back-end do DApp. Ele executará a aplicação em Modo Host, onde a lógica do back-end é executada nativamente em localhost. Este modo de execução é importante para que os desenvolvedores possam testar e depurar as aplicações utilizando as ferramentas na qual já estão acostumados.

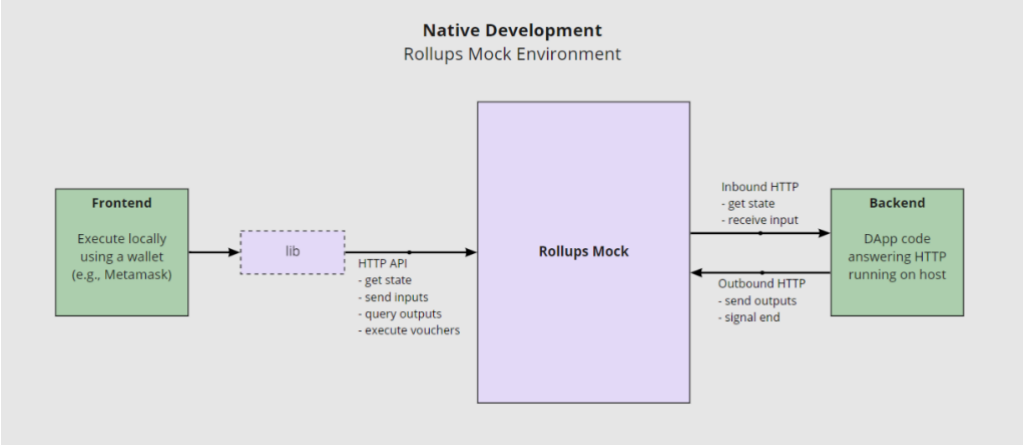

Figura 4.13. Modo Host

Packing

Nesta etapa, a parte de back-end é empacotada para rodar dentro de uma Cartesi Machine, para ser executada por um nó real da Camada 2 (ou seja, não uma versão de desenvolvimento na qual o back-end é executado diretamente no host). A viabilidade de executá-lo dentro de uma Cartesi Machine já deve ter sido acessada durante o Estágio 2, e agora veremos apenas se a implementação completa do DApp também roda satisfatoriamente. Este modo de execução é chamado Modo de Produção. A execução neste modo é fundamental, pois emula a execução da aplicação da forma como será quando for implementada. Isso significa que qualquer código escrito em linguagens compiladas, como C++ ou Rust, deve ser compilado para a arquitetura RISC-V da Cartesi Machine. Nesse modo, a computação realizada pelo back-end é reproduzível e, portanto, verificável, permitindo uma execução verdadeiramente confiável e descentralizada. [Cartesi 2022]

Como o back-end continuará usando a mesma API HTTP de antes, ela permanece inalterada. A única diferença é que desta vez ele irá interagir com um serviço rodando dentro da própria Cartesi Machine, ao invés de se comunicar diretamente com a infraestrutura do Rollups Host como no Estágio 3.

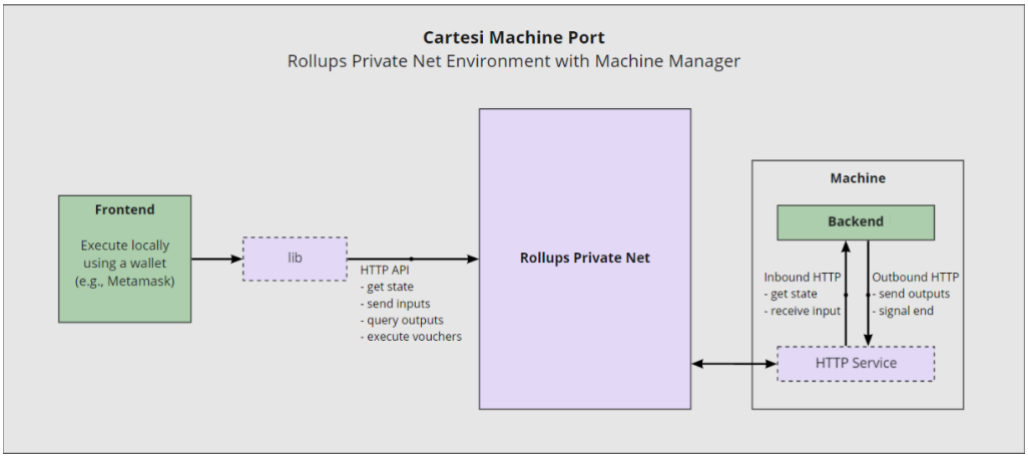

Figura 4.14. Modo de Produção

Deployment

Nesta etapa, a parte de back-end é empacotada para rodar dentro de uma Cartesi Machine, para ser executada por um nó real da Camada 2 (ou seja, não uma versão de desenvolvimento na qual o back-end é executado diretamente no host). A viabilidade de executá-lo dentro de uma Cartesi Machine já deve ter sido verificada durante o Estágio 2, e agora veremos apenas se a implementação completa do DApp também roda satisfatoriamente.

Como o back-end continuará usando a mesma API HTTP de antes, ela permanece inalterada. A única diferença é que desta vez ele irá interagir com um serviço rodando dentro da própria Cartesi Machine, ao invés de se comunicar diretamente com a infraestrutura do Rollups Host como no Estágio 3.

4.6.2. Ferramentas Necessárias

Antes de iniciar a criar aplicações vamos instalar algumas ferramentas úteis que seram necessárias para o desenvolvimento dos DApps Cartesi.

Docker

Docker é uma plataforma de código aberto utilizada para criar e administrar containers (ambientes isolados), onde suas aplicações podem ser executadas. Além disso, podemos compartilhar estes containers com outros usuários de Docker. A Cartesi o utiliza para distribuir a estrutura do Cartesi Rollups. Para instalá-lo siga os passos descritos no site oficial.

Docker Compose

Docker Compose é um gerenciador de containers da Docker. Com ele, podemos iniciar múltiplos containers simultaneamente e definir a forma como eles interagem uns com os outros. Com o Compose, podemos instanciar localmente um ambiente completo de Cartesi Rollups, para que possamos testar os DApps usando apenas sua máquina de desenvolvimento. Para adicionar o Docker Compose à sua máquina siga as instruções do site ofical.

Node.js e NPM

Node.js é um ambiente de execução JavaScript com inspirações no interpretador V8 do Google que nos permite executar códigos JavaScript sem depender de um navegador web. Ele normalmente é distribuído com o gerenciador de pacotes NPM. Pode-se encontrar os passos para a instalação no site oficial do Node.js.

Yarn

Yarn é um gerenciador de pacotes e o utilizamos para adicionar os pacotes aos nossos DApps Cartesi. Para instalá-lo acesse o site oficial do Yarn.

4.6.3. Criando o ambiente

A seguir são expostas as instruções para baixar e iniciar a operação de um Dapp. O primeiro passo é baixar o repositório e reconstruir o back-end, executando os seguintes comandos em um terminal Linux.

```
git clone <endereco_do_repositorio>
cd rollups-examples/<nome_do_Dapp>
make machine
```

O primeiro comando baixa uma cópia do repositório no diretório atual, o segundo comando altera o diretório atual para o diretório do Dapp, e o terceiro comando executa uma série de Makefiles e Shellscripts para construir o sistema de arquivos do Dapp e move-lo para dentro da Cartesi Machine, que é criada na pasta também recém criada chamada **machine**.

Como as interações com a Cartesi Machine serão por meio de comandos Node.js, é preciso operar a partir da pasta **contracts**.

```
cd contracts/
yarn
```

O Yarn irá instalar o HardHat e demais pacotes necessários para interagir com o contrato, que está na estrutura de simulação do HardHat.

4.6.3.1. Escolhendo o Modo de Execução

Para simular a arquitetura Cartesi, um conjunto de containers docker entra em operação. Há 2 modos de operação. O Modo host, utilizado para desenvolvimento, e o modo de produção. Os comandos para aciona-los, respectivamente, são os 2 a seguir.

É importante ressaltar que eles não operam simultaneamente. Ou se esta em um ambiente ou em outro. E para aciona-los pelo comando docker deve-se estar no diretório **<nome do Dapp>**.

```
docker-compose up --build
```

```
docker-compose -f docker-compose.yml -f docker-compose-host.yml up --
    build
```

Ao final, ou antes de alternar entre ambientes, deve-se pressionar as teclas Ctrl+C e esperar que os containers sejam desativados, feito isso, é possível também utilizar os comandos abaixo para deletar os volumes criados pelos containers. O primeiro comando é para o modo de produção e o segundo é para o modo host.

```
docker-compose down -v
```

```
docker-compose -f docker-compose.yml -f docker-compose-host.yml down -v
```

4.6.4. LGPD-Compliant Datasets Dapp

Este exemplo, que chamamos de LGPD Datasets Dapp, expõe como implementar uma plataforma de provisionamento de modelos de aprendizado de máquinas ciente da procedência dos dados, com base na plataforma Amnesia.[Stach. et al. 2020]. Para alcançar este objetivo, é utilizado um banco de dados já criado com uma tabela chamada medical no SQLite, dentro da Cartesi Machine. O passo a passo da instalação e operação do Dapp mostra como construir e interagir com o respectivo aplicativo Cartesi Rollups que atua de

forma transparente sobre os dados pessoais, buscando cumprir requisitos da LGPD - Lei Geral de Proteção de dados, ou da sua equivalente internacional, a GDPR - General Data Protection Regulation.

Por um lado é possível realizar operações predefinidas para criar, atualizar e excluir registros, atuando como titular dos dados. Por outro lado, também se pode operar como controlador/processador de dados para realizar consultas e construir novos conjuntos de dados para distribuição ou processamento específico. O titular dos dados e o controlador/processador de dados são atores dentro da definição da LGPD/GDPR.

O exemplo destaca alguns aspectos da LGPD/GDPR, como a garantia do titular dos dados de não ter mais seus dados em um conjunto de dados ou a garantia de seu direito de saber como seus dados estão sendo usados, bem como a garantia dos consumidores finais de estarem utilizando dados de origem legítima. Direito ao esquecimento e outros aspectos de segurança não estão presentes neste Dapp conceitual. Este Dapp foi elaborado a partir do SQLite Dapp, disponivel em https://github.com/cartesi/rollups-examples/tree/main/sqlite.

Ao final, restará claro que através das propriedades do Cartesi Rollups de registrar todos os estados da máquina, é possível verificar quais dados foram utilizados em cada subconjunto de dados disponibilizados para propósitos específicos, como, por exemplo, o treinamento de um modelo de aprendizado de máquina que precisa ser retreinado caso um dos registros de sua base de dados de treinamento seja alterado por solicitação do usuário titular dos dados.

Contexto

Ao se demonstrar que a viabilidade técnica de determinado projeto respeita exigências jurídicas, a adoção da tecnologia é revestida de legitimidade e ganha mais notoriedade de empresas e governos [Governo Federal 2021] que possuem rígidas regras de controle, conformidade(*compliance*) e transparência. Este diferencial pode ser uma vantagem competitiva frente a tecnologias que ignoram estes aspectos.

Escalabilidade de Blockchains é o principal desafio encarado por esta tecnologia atualmente. [Zhou et al. 2020b] A Cartesi oferece à sociedade uma solução do tipo *Layer-2* para este desafio. Otimiza-se a computação com operações *off-chain* entre usuários e otimiza-se o armazenamento de dados *on-chain* através de extensivo uso de *hashes*.

Com a Cartesi Machine como base de dados é possível obter pleno controle sobre inserções, atualizações e exclusões. Utilizando o próprio sistema de arquivos da máquina virtual, arquivos .txt ou .csv, ou algum banco de dados leve, como o SQlite, é possível construir uma Cartesi Machine que opere conforme as necessidades da aplicação atendendo aos requisitos da LGPD/GDPR.

Na máquina *off-chain* estará a maior parte possível da lógica de operação e controle, bem como a base contendo dados pessoais do(s) usuário(s).

As entradas de dados serão requisições para inclusão, leitura, atualização ou exclusão de dados, e os dados propriamente ditos. Embora operações de autenticação, autorização e controle de acesso sejam necessárias para uma implementação em um ambiente

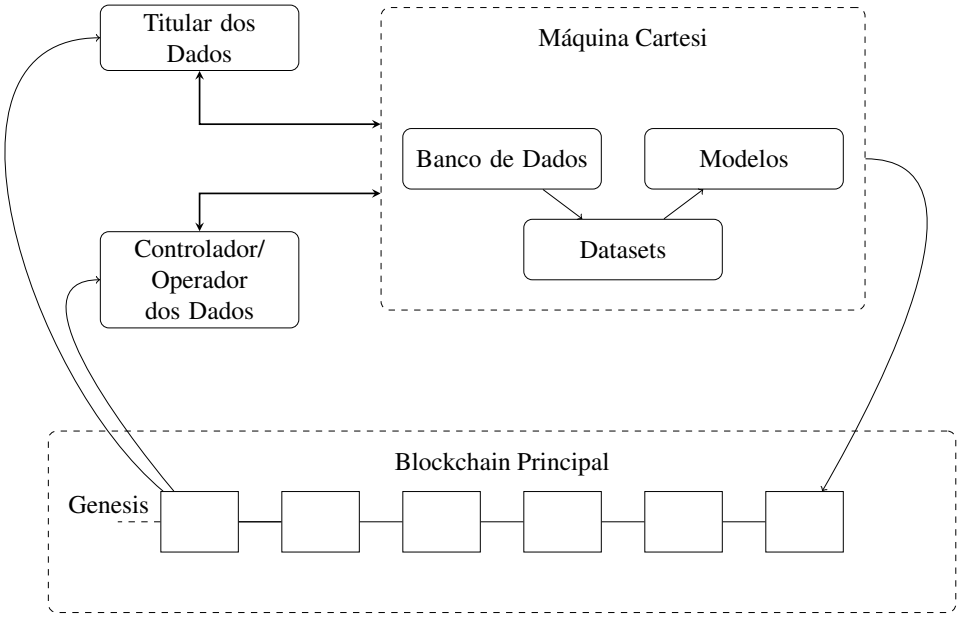

Figura 4.15. LGPD-Compliant Datasets Dapp

de produção real, suas dinâmicas não são tratadas no presente exemplo.

A saída de dados compreende a confirmação da operação e os dados afetados. Após a cada operação, o novo estado é salvo, e novas operações serão sobre o novo estado salvo. A não utilização um estado antigo é o que vai ao encontro do direito ao esquecimento do usuário em um sistema totalmente descentralizado da forma mais segura possível atualmente.

Mais especificamente no contexto do presente exemplo, a Cartesi Machine armazena dados pessoais para fins de treinamento, validação e teste de modelos de aprendizado de máquina. O diferencial desta implementação é o caráter sob-demanda em que o modelo pode ser treinado. Em respeito ao direito do usuário de ser esquecido, o modelo poderia ser totalmente retreinado quantas vezes fossem necessárias nos casos em que um dos titulares dos dados pessoais revogasse o direito de operação de terceiros sobre eles.

Esta abordagem contextualizada pode dar origem a uma nova arquitetura de armazenamento de dados descentralizados de longo prazo e processamento em tempo real em conformidade com requisitos legais de privacidade, como é proposto na Amnésia [Stach. et al. 2020].

Além de propiciar transparência à todos os envolvidos, esta abordagem também fornece maior garantia de que os dados revogados não estão sendo disponibilizados aos antigos interessados. Quem cede os dados tem a garantia das suas corretas utilizações, e quem usa os dados tem garantias de procedência e consentimento atualizado do titular dos dados.

Na Figura 4.15, o **titular dos dados** e o **controlador/operador dos dados** leem na Blockchain a referencia ao contrato que permite interagir com os dados.

Caso a ação seja desencadeada pelo **titular dos dados**, este poderá adicionar,

atualizar e deletar seus dados no **banco de dados**, e o **controlador/operador dos dados** terá garantia de procedência dos dados atualizados, bem como seus **modelos** atualizados.

Caso a ação seja desencadeada pelo **operador dos dados**, este poderá ler os conjuntos de dados pertinentes (**conjunto de dados, ou *datasets***) e utiliza-los para treinar novos **modelos** de aprendizado de máquina caso o **titular dos dados** tenha concedido autorização para os novos propósitos do treinamento.

Em ambos os casos, ao final, após consenso entre as partes, o estado da **Cartesi Machine** é atualizado e seu hash é salvo na Blockchain.

O presente exemplo tem como foco a implementação focada nos componentes **banco de dados** e **datasets** da **Cartesi Machine**.

Por motivos de praticidade e pelo próprio proposito do atual construtor de Dapps da Cartesi, as ações das partes interessadas são realizadas de forma assíncrona, ou seja, em uma operação o titular dos dados atual, em outra, o controlador/operador atua. Cada operação corresponde a uma transação completa na Cartesi Machine, auditável e verificável.

Após o **banco de dados/datasets** serem atualizados, a leitura para utilização dos dados pode ser precedida de 2 dos 4 filtros de seleção de dados, a zona de filtro horizontal e a vertical, conforme arquitetura proposta pela Amnesia. [Stach. et al. 2020]

A zona de filtro horizontal equivale a um operador de seleção(σ), removendo origens ou limitando por período de tempo. Este filtro remove registros inteiros.

A zona de filtro vertical equivale a um operador de projeção (π), removendo atributos dos registro(colunas).

Os dados gerados a partir da aplicação dos filtros são armazenados nos **datasets**, que serão utilizados para treinar os **modelos**.

Limitações do direto ao esquecimento

A busca pela conformidade à LGPD/GDPR encontra uma barreira em particular quando se trata da tecnologia Blockchain. Por ser um livro razão, os dados nela inseridos são "eternos" - o que exclui o direito do usuário de exigir o esquecimento dos seus dados por quem os controla e opera. [Truong et al. 2020] [Daudén-Esmel et al. 2021]

Assim, atualmente, a única alternativa para buscar garantir o direito do usuário da Blockchain ao esquecimento é alocar os dados pessoais fora da Blockchain. Dentro desta alternativa, há 2 caminhos possíveis. Ou se armazena os dados pessoais em recipiente centralizado com opção de exclusão - comprometendo a descentralização; ou se armazena em outro sistema descentralizado, como o IFPS - InterPlanetary File System [Benet 2014] que, devido ao seu protocolo de funcionamento, não é possível garantir que todos os nós do sistema deletem determinado dado eventualmente já adquirido.

Uma possível futura alternativa seria a utilização de criptografia completamente homomórfica para realizar queries e outras operações sobre dados criptografados dentro da Cartesi Machine [Gentry 2009]. Como os dados estariam sempre criptografados,

a sua remoção de conjunto de dados para usos específicos poderia ser entendido como esquecimento de tais dados, para fins de processamentos posteriores.

Direito ao esquecimento em Blockchain é um assunto controverso e pouco definido, e não será tratado neste exemplo. Embora haja debates sobre o que é ou não um dado pessoal, ou quando um dado deixa de ser pessoal, o fato é que o quanto mais difícil é recuperar um dado que se deseja esquecer, mais garantida está a satisfação do usuário ao olhos da LGPD/GDPR, que transfere controle absoluto ao usuário sobre seus dados pessoais[1]. Dados cifrados cuja chave privada não esteja disponível para uso podem ser considerados dados inacessíveis ou esquecidos.[Stach. et al. 2020] Em contrapartida, é possível que um dia haja possibilidade de recuperar as informações criptografadas mesmo sem conhecimento da chave privada. [Shor 1994] [Shor 1997]

4.6.4.1. Front-End

Nesta aplicação, o front-end tem o papel de oferecer uma melhor interface para o usuário e uma forma de inserir os comandos SQL mais confortável.

Ele foi desenvolvido utilizando Flask, um microframework, escrito em Python, para aplicações web. Além disso, utilizamos alguns módulos do Flask e a biblioteca web3.py.

A biblioteca web3.py é utilizada no desenvolvimento de DApps em Python. Ela é utilizada para facilitar a interação com contratos inteligentes, obter informações da Blockchain, criar transações e muito mais.

O front-end segue o padrão de arquitetura MVC (Model, View e Controller). Nesta implementação, o diretório `templates` é responsável pela componente View da arquitetura. A árvore abaixo exibe os principais arquivos criados para o funcionamento do front-end da aplicação.

```
front_end
├── config
│   └── connection.py
├── controllers
│   ├── addInput.py
│   └── getNotice.py
├── models
│   └── forms.py
├── static
│   ├── ABI
│   └── bootstrap-5.1.3-dist
└── templates
    ├── addinput.html
    ├── getnotice.html
    └── index.html
```

[1] salvo exceções de interesse público

```
└── dapp.py
```

A seguir, descrevemos o papel de cada componente da árvore para a aplicação.

Models

Na pasta `models` temos apenas um arquivo, `forms.py`. Nele, utilizamos o módulo WTForms do Flask para criar os campos para o preenchimento de dados do usuário, que serão utilizados no arquivo `sqlite.py`. O campo Statement é onde o usuário insere o código SQL e o campo Epoch é onde ele define o epoch no qual quer obter as informações registradas.

View: Templates

Neste diretório, temos os arquivos HTML para exibição das páginas da aplicação. O arquivo `base.html` serve como um template base para os outros arquivos, que são uma extensão dele. Os arquivos `addinput.html` e `getnotice.html` são para a exibir das páginas onde o usuário insere, respectivamente, o statement SQL e o epoch. O arquivo `index.html` exibe a página inicial da aplicação.

Controllers

Na pasta `controllers` estão os arquivos que executam a lógica da aplicação. O arquivo `addInput.py` é responsável por executar o método addInput, que envia o statement SQL para o back-end da aplicação. O arquivo `getNotice.py` é usado para realizar consultas utilizando o GraphQL, ele recebe o epoch dado pelo usuário e envia uma query para o GraphQL, e seu resultado é, em seguida, usado para exibir uma resposta ao usuário.

Além disso, temos o arquivo `connection.py` que estabelece a conexão com o back-end da aplicação. E no diretório `static`, temos a pasta com as ABIs dos contratos de Input do Cartesi Rollups e a pasta `bootstrap-5.1.3-dist` com distribuições *bootstrap* para a estilização das páginas da aplicação.

4.6.4.2. Back-End

A Cartesi atualmente possui alta frequência de atualizações significativas de seu código-fonte e documentação, incluindo *scripts* para criação de modelo de Dapp com nome personalizado e respectivo conjunto de pastas e arquivos. Este e demais exemplos estão na url *https://github.com/cartesi/rollups-examples*. O LGPD Datasets está disponível em *https://github.com/brunogondim/Dapp-Cartesi-sqlite_extended*.

Em versões mais recentes do kit de desenvolvimento Cartesi é possível criar um Dapp modelo básico a partir da execução de um script shell. Entretanto, nada impede o desenvolvedor de seguir a partir de um dos exemplos já prontos, como foi este caso. Enquanto o desenvolvimento do Front-End permite maior liberdade ao desenvolvedor, o Back-End possui um conjunto de pastas e arquivos padrão.

Partindo da estrutura básica, onde o que muda de uma aplicação para a outra é basicamente o arquivo python, sqlite.py, dentro do diretório **server**, altera-se o código

python para atender o proposito da nova solução. Deve-se ressaltar aqui que, embora o exemplo seja sobre a linguagem de programação *python*, é possível teoricamente utilizar qualquer linguagem que rode sobre Linux.

Quando o servidor, seguindo a logica contida no arquivo .py, recebe uma requisição *POST* na url terminada em *advance/*, executa o conteúdo da mensagem como comando SQL, abrindo uma conexão com o banco de dados *SQLite* para efetuar uma operação comum de banco de dados.

Caso os dados no banco de dados sejam alterados, eles ficarão salvos no estado final da Cartesi Machine, quando o processamento for finalizado. É esta evolução de estados da máquina que a permite funcionar como banco de dados persistente.

Caso seja uma mera consulta, o estado da máquina também será atualizado, pois ocorrem modificações no arranjo da memória e no processador, bem como a marca do tempo. Os dados consultados são transmitidos por uma requisição *POST* assíncrona enviada do servidor para a porta padrão de saída da Cartesi Machine, com a url contendo */notice* no final. Esta ação ocorre antes da resposta final à requisição inicial. O servidor encerra sua ação respondendo por *POST* a url de final */finish*.

Os dados estarão disponíveis para consulta na Blockchain em momento posterior à finalização da transação, ou seja, o recebimento dos dados depende de um novo comando.

O banco de dados

Para este exemplo, o banco de dados foi criado e populado fora da Cartesi Machine. Isto não é um requisito, mas caso o banco seja previamente criado, assim como qualquer outro eventual arquivo ou pasta que faça parte do back-end, deve ser referenciado nos arquivos Makefile e shell script, caso contrario, no momento da construção da máquina, eles não serão copiados para dentro da máquina. Dentro do diretório *server*, a linha `$(DAPP_FS_BIN): sqlite.py run.sh data.db` do arquivo *Makefile* deve ser alterada e a linha `cp ./data.db $DAPP_FS` do arquivo *build-dapp-fs.sh* deve ser incluida para que o script de criação da máquina considere os novos arquivos.

Um aspecto crucial para possibilitar utilização de banco de dados na Cartesi Machine é o tamanho do sistema de arquivos *.ext2* criado durante o comando *makefile*. Seu tamanho deve ser proporcional ao tamanho que se espera armazenar na máquina. Desta forma, o sistema de arquivos foi customizado para 16MB, alterando-se o parametro *-b* para o valor 16384 no arquivo *build-dapp-fs.sh* a linha:

```
genext2fs -f -i 512 -b 16384 -d $DAPP_FS $DAPP_FS_BIN
```

4.6.4.3. Interagindo com a aplicação

A interação com o Dapp pode ocorrer por linha de comando ou por seu Front-End. De preferência por este ultimo.

O comando a seguir representa uma possível requisição à rede Blockchain, pelo HardHat, que por sua vez aciona o contrato referente ao Cartesi *on-chain*, que, em se-

guida, aciona a Cartesi Machine *off-chain*. É na parte *off-chain* que o núcleo do Dapp opera. Os comandos Hardhat devem ser acionados dentro do diretório **contracts**.

```
npx hardhat --network localhost sqlite:addInput --input "INSERT INTO
    Medical VALUES ('35', 'male', '32.4', '0', 'no', 'southeast',
    '10000.0000' )"
```

Neste Dapp conceitual, todas as operações possíveis se resumem a operações CRUD. Elas representam as ações dos titulares dos dados e dos controladores/operadores dos dados. O resultado das queries não são disponibilizadas na hora. Após a execução, deve-se aguardar alguns minutos e executar o comando abaixo, para retornar o resultado das queries.

```
npx hardhat --network localhost sqlite:getNotices --epoch 0 --payload
    string
```

O que vai disponibilizar os *datasets* em conformidade com a LGPD são os filtros sobre a tabela principal, com seu resultado salvo em novas tabelas especificas, que poderão ser usadas para treinar e validar modelos de aprendizado de máquina e outros tipos de processamento. A ciência do titular dos dados sobre quais dados estão sendo usados para cada proposito, praticamente em tempo real, bem como seu controle de exclusão e alteração são as principais vantagens dessa abordagem. A seguir um exemplo de filtro vertical. No primeiro comando cria-se a tabela. No segundo, ela é preenchida.

```
npx hardhat --network localhost sqlite:addInput --input "CREATE TABLE
    Vertical_Filter (age text, bmi text, charges text)"
npx hardhat --network localhost sqlite:addInput --input "INSERT INTO
    Vertical_Filter (age, bmi, charges) SELECT age, bmi, charges FROM
    Medical"
```

Lembrando que tais queries podem tanto ser comandadas pelo comando HardHat quanto pelo Front-End.

Para executar o front-end, com o back-end em operação, navegue até o diretório `front_end` e instale as dependências com o comando a seguir.

```
pip install -r requirements.txt
```

Prosseguindo, definimos a variável de ambiente `FLASK_APP` e rodamos o front-end. Para isso, execute os comandos abaixo.

```
export FLASK_APP=dapp
flask run
```

Agora, com o front-end em operação, vamos utilizá-lo para enviar os comando SQL para o back-end, ao invés de utilizarmos a linha de comando do terminal. Ele roda na página `localhost:5000`, acesse-a e use a barra de navegação para acessar a página AddInput. Nessa página, temos o campo de preenchimento Statement, que o usuário utiliza para inserir os comandos SQL. Por exemplo, ao interagirmos com a aplicação diretamente pela linha de comando, podemos usar o comando a seguir.

```
npx hardhat --network localhost sqlite:addInput --input "INSERT INTO
    Medical VALUES ('35', 'male', '32.4', '0', 'no', 'southeast',
    '10000.0000' )"
```

Utilizando o front-end, ao invés do comando acima, podemos simplesmente inserir o comando SQL a seguir no campo Statement da página AddInput.

```
INSERT INTO Medical VALUES ('35', 'male', '32.4', '0', 'no', 'southeast
    ', '10000.0000' )
```

Para obtermos os resultados das queries, pelo front-end, acessamos a página GetNotice e inserimos o epoch correspondente a notice que queremos no campo de preenchimento Epoch.

Limitações do Dapp

Embora a tecnologia Cartesi disponha de um sistema operacional Linux como base para a pilha tecnológica do contrato inteligente a ser desenvolvido, propiciando o uso de qualquer tecnologia que opere sobre o Linux, hoje ainda não é possível se utilizar de todo o arsenal tecnológico compatível com Linux na Cartesi de maneira prática. Cada componente que necessite de instalação deve estar presente no sistema de arquivos raiz previamente desenvolvido pela Cartesi. Atualmente a alteração do sistema de arquivos raiz da Cartesi Machine para ser utilizado com a solução de Roll-ups ainda não está disponível ao público, impossibilitando o uso de demais pacotes que precisam ser compilados para a arquitetura da Cartesi.

A obtenção dos dados consultados é realizada em uma operação posterior a solicitação. A solicitação altera o estado da máquina, mas a obtenção dos dados consultados não. Desta forma não é possível registrar dentro da máquina por quem e quando os dados são lidos.

Garantir a privacidade dos dados é um dos principais desafios de se utilizar este banco de dados distribuídos. Como todos os estados anteriores são inspecionáveis, eventuais chaves privadas salvas na máquina podem ser acessadas, mesmo se excluídas.

Como todos os dados inseridos na máquina ficarão inspecionáveis para sempre, os dados pessoais devem ser inseridos já anonimizados na Cartesi Machine, dificultando a confirmação da procedência e controle para posteriores alterações.

A efetiva garantia de que o controlador/operador dos dados irá respeitar a vontade do titular dos dados de ter seus dados alterados ou excluídos, ou apenas utilizados para um proposito específico previamente autorizado está além da capacidade de controle sobre a Blockchain e sua 2ª camada Cartesi. Ela depende de procedimentos extra-sistema. Mas esta limitação tecnológica não é exclusividade nem da tecnologia Blockchain nem da camada 2. Um dado, uma vez visto, não pode mais ser "desvisto". Qualquer dado que seja inserido em um sistema centralizado ou trafegue em rede publica pode nunca mais ser "esquecido", mesmo que seja excluído, pois pode ter sido copiado para outro sistema.

Possíveis soluções às limitações

Uma possível solução para garantir que os dados pessoais serão utilizados conforme acordado é implementar o treinamento do modelo e sua utilização operando dentro da própria Cartesi Machine. Teoricamente é viável treinar modelos dentro da máquina, já

que esta opera Linux, e Linux possui diversas bibliotecas para treinamento de modelos. A utilização efetiva dos modelos vai depender da dinâmica de cada negócio.

O desafio da autenticação do titular dos dados anonimizados para futuras alterações ou exclusões, e até mesmo para evitar duplicidade de registros, por quem de fato é o titular dos dados pode ser encarado utilizando identidades auto-soberanas. Embora atualmente haja diferentes noções do seu conceito e imprecisão sobre o alcance das suas aplicações, este tipo de identidade vem se mostrando cada vez mais uma forma promissora de interação entre as partes interessadas nos dados pessoais na internet do futuro [Ferdous et al. 2019].

4.6.5. IoT Rollups DApp

Como mencionado anteriormente, Blockchain é especialmente útil quando se envolve desconfiança mútua, por conta disso, seu uso é muito vantajoso no contexto de serviços públicos. O transporte público é um destes serviços que carece de confiança, basta se perguntar: como saber se o serviço está cumprindo com o cronograma acordado?

O cronograma mencionado diz respeito a rota que o veículo deve percorrer e o horário que ele deve estar em cada ponto de parada para pegar ou deixar passageiros. Portanto, esse cronograma pode ser visto como um contrato entre a prefeitura e a empresa que fornece o serviço. Para que seja possível verificar que as empresas estão fazendo a sua parte no acordo, ou seja, cumprindo o cronograma, os veículos devem ser capazes de fornecer dados geoespaciais (GPS).

Para que os veículos forneçam dados de GPS, eles devem estar equipados com dispositivos IoT (Internet of Things). Esses dispositivos, usualmente pequenos, são equipamentos que se conectam à internet e juntos formam um ecossistema no qual determinada aplicação roda fazendo uso deles de alguma forma. Eles geralmente funcionam como sensores ou atuadores. Como sensores, esses dispositivos têm o objetivo de gerar dados de medição, por exemplo, temperatura, umidade e proximidade. Como atuadores, esses dispositivos tem o objetivo de realizar mudanças de estado, por exemplo, abrir uma porta ou janela, quando solicitado. No contexto em questão, eles seriam sensores e forneceriam a coordenada do veículo.

Um fato relevante a ser destacado é que dispositivos IoT geralmente funcionam bem com Blockchain, devido a natureza descentralizada inerente ao uso de tais dispositivos, sendo assim, a criação de um Cartesi DApp que envolve dispositivos IoT é um passo natural. Desse modo, a aplicação proposta a seguir é um Cartesi DApp que faz uso de dispositivos IoT para solucionar o problema levantado a respeito do transporte público.

Visando solucionar o problema, o DApp a ser desenvolvido tem que ser capaz de verificar se o serviço de transporte público, nesse caso o ônibus, está cumprindo o seu cronograma. Caso seja comprovado que determinado veículo está descumprindo o cronograma, seja por desvio de rota ou por atraso, deve-se gerar uma multa. As multas geradas devem estar disponíveis em um dashboard online para que o público possa consultá-las.

Para simplificar a implementação, considera-se que os dispositivos que estão nos veículos fornecem dados confiáveis. A informação fornecida é composta pela coordenada do veículo acompanhada da timestamp do momento da medição. O DApp recebe essa

informação e no back-end consulta um banco de dados que possui o cronograma daquele veículo, utilizando essas duas informações é feita a verificação. As multas são geradas na forma de "/notice", que posteriormente é consultado usando GraphQL para gerar o dashboard público. A Figura 4.16 contém um diagrama que representa o DApp e as subseções a seguir detalham o front-end, back-end e como interagir com a aplicação.

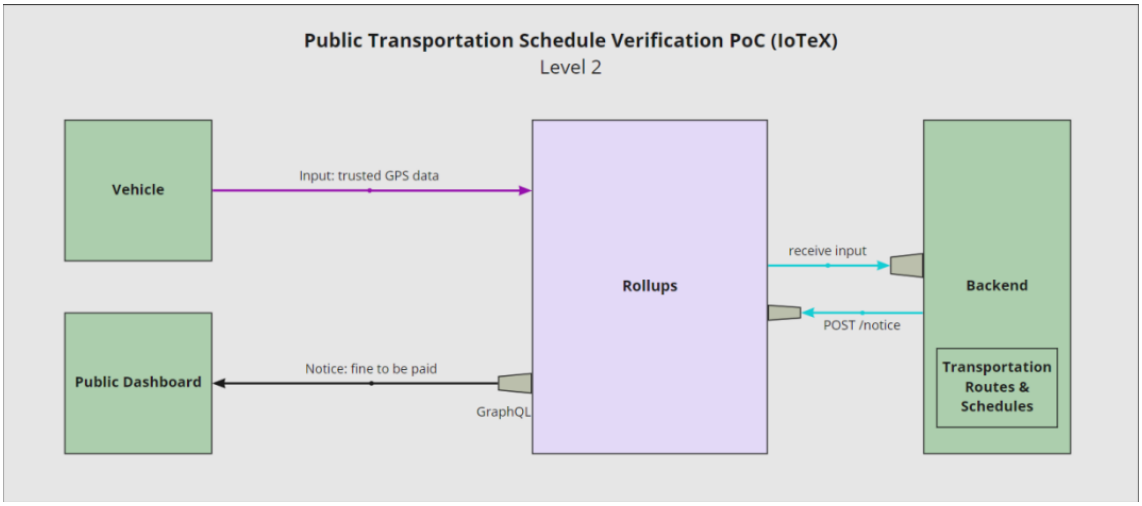

Figura 4.16. Arquitetura do Dapp

4.6.5.1. Front-End

Como mencionado anteriormente, a aplicação deve fornecer um dashboard para consulta do público, mas esse não é o único papel do front-end nesse DApp. O front-end é fornecido por um servidor web desenvolvido em NodeJS, e é responsável também por receber os inputs e repassá-los à Blockchain utilizando Web3, que é um novo padrão de interação na internet que se apoia em tecnologias descentralizadas como a Blockchain.

Para a construção do front-end a biblioteca Web3 utilizada foi a Web3.js, devido ao javascript ser a linguagem nativa do NodeJS. O servidor Web desenvolvido segue a arquitetura MVC, que é composta por 3 componentes (**M**odel, **V**iew e **C**ontroller) que dão nome ao padrão. A seguir é explicado, de forma sucinta, o papel de cada componente.

- Model: É usado pelo controller para acessar o banco de dados. Contém a lógica de negócio, especificações de determinado dado e a lógica que será aplicada a ele.
- View: É por onde os usuários interagem com a aplicação, sendo composto apenas por templates HTML que são preenchidos dinamicamente de acordo com o dado que é passado para este template.
- Controller: Interage com o Model e fornece as respostas, oriundas de algum back-end, e funcionalidade para o View. O papel do controller é basicamente realizar o "meio de campo" entre o View, que é por onde o usuário interage com a aplicação, e o Model, que contém a lógica de negócio.

Cada componente corresponde a um diretório, como mostra a estrutura de diretórios abaixo, onde tem-se código javascript. Nessa estrutura observa-se também a presença de outros 3 diretórios. Em "config", tem-se a configuração de acesso ao back-end da aplicação, em "routes" são definidas as rotas e em "public" temos assets usados no view, bibliotecas, arquivos estáticos, entre outros. Agora, tendo ciência deste padrão de design, visitaremos os componentes criados para o front-end utilizado neste DApp.

```
front_end
├── config
│   └── eth-connection.js
├── controllers
│   └── home.js
├── models
│   └── blockchain-model.js
├── public
│   ├── ABI
│   ├── assets
│   └── bootstrap-5.1.3-dist
├── routes
│   └── home.js
├── views
│   ├── form.ejs
│   └── index.ejs
└── app.js
```

Model

No model temos o "blockchain-model.js", onde são definidas 3 funções:

- getNoticePage: Realiza consultas usando graphql para o container "query_server", com o objetivo de recuperar as multas geradas (/notices), o resultado é posteriormente exibido no dashboard.
- getAccounts: Utilizando o módulo de conexão à Blockchain Hardhat definido em "config", realiza uma consulta para recuperar as contas existentes.
- addInput: Utilizando o módulo de conexão à Blockchain Hardhat definido em "config", executa o método addInput do contrato de input para enviar o input para o back-end do DApp.

View

No View temos os dois templates HTML, "index.ejs" e "form.ejs". O "index.ejs" é a página que possui o Dashboard para a consulta das multas, o "form.ejs" é a página por onde pode-se enviar o cronograma de uma linha de ônibus para o back-end do DApp.

Controller

No controller temos o “home.js”, onde são definidas 3 funções:

- homePage: Usando a função getNoticePage do model, recupera os dados e encaminha para a view “index.ejs”. É acessado através da rota “/” e renderiza o dashboard.
- formPage: Usando a função getAccounts do model, recupera os dados e encaminha para a view “form.ejs”. É acessado através da rota “/form” e renderiza a página do formulário.
- submit: Executa a função addInput do model. É acessado através da rota “/submit” e envia um JSON de resposta para o cliente que acessou a rota informando se o addInput foi executado com sucesso ou não.

4.6.5.2. Back-End

O back-end da aplicação, que é o código executado dentro da Cartesi Machine em Production Mode, foi desenvolvido em Python e seus arquivos encontram-se no diretório *server*. O back-end recebe dois tipos de input, um deles é o cronograma de uma linha de ônibus e o outro é o dado enviado pelo dispositivo IoT em um veículo. Dependendo do input recebido, o back-end deverá processá-lo de forma diferente.

Caso o input recebido seja um cronograma de uma linha de ônibus, o processamento consiste em salvar o cronograma em um banco de dados SQLite do DApp. Este input possui o identificador da linha de ônibus, a rota que é feita por ele, a coordenada das paradas e o cronograma de cada viagem que aquela linha faz. Vale lembrar também que se o back-end estiver sendo executado em Production Mode, o banco de dados ficará dentro da Cartesi Machine.

Caso o input recebido seja um dado enviado por um veículo, o processamento consiste em a partir do identificador da viagem, que também é fornecida pelo veículo, consultar o banco de dados de cronogramas para recuperar a rota e o cronograma daquela viagem. Com os dados recebidos pelo input e os dados recuperados do banco é possível verificar se o veículo está na rota e se o veículo está atrasado ou não.

O código Python foi dividido em três módulos que têm seus nomes e funcionalidades mostrados a seguir.

- db_manager.py: É o módulo que gerencia a conexão ao banco de dados SQLite, é responsável pela criação das tabelas utilizadas, caso não existam, inserções e consultas diversas.
- util.py: É o módulo que contem funções de cálculo e conversão. Conversão de string para hexadecimal, cálculo da distância entre duas coordenadas e funções que fazem uso do cálculo de distância para verificar a rota e o atraso.
- iot_dapp.py: É a main do back-end e importa os outros dois módulos. Seu papel é receber os inputs verificando qual é o tipo, um cronograma ou um dado enviado por um veículo, em seguida utilizar as funções dos outros dois módulos para processá-lo corretamente.

O fluxograma da Figura 4.17 resume o fluxo do processamento realizado pelo back-end.

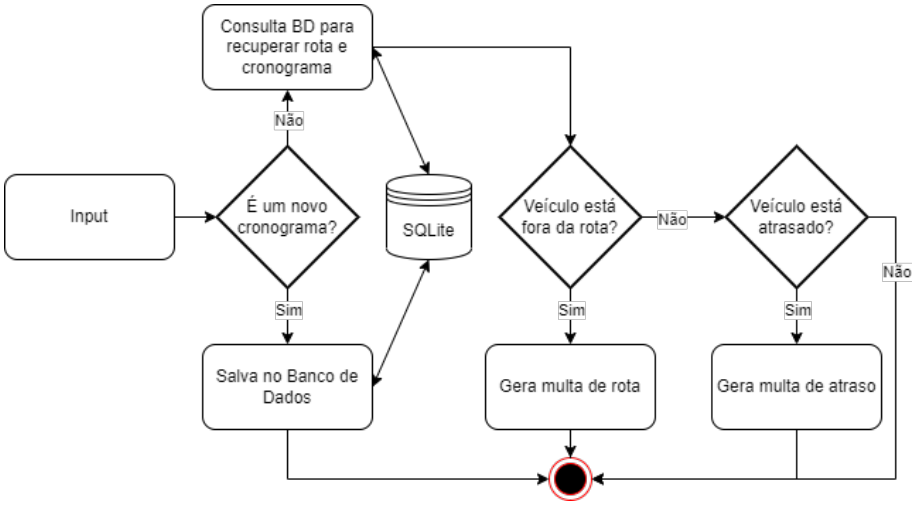

Figura 4.17. Fluxograma do Back-End

4.6.5.3. Interagindo com a aplicação

Antes de interagir com a aplicação, precisamos executá-la e isso envolve rodar o back-end e o front-end. Como já foi visto anteriormente o procedimento para executar o back-end, veremos nesta subseção apenas como executar o front-end antes de interagir com a aplicação. Para o front-end deste DApp é necessário executar os seguintes passos:

1. Navegar até o diretório *front_end*;
2. Instalar as dependências utilizadas executando o comando **npm install**;
3. Rodar o back-end em Host Mode ou Production Mode em outro terminal;
4. Executar o front-end após o back-end estar rodando, utilizando o comando node **app.js**.

Uma vez cumpridas todas as etapas, o DApp estará rodando e o front-end pode ser acessado em `http://localhost:3000`, nessa página inicial temos o dashboard, que no momento deve estar vazio. Vimos na seção back-end que o DApp trabalha com dois tipos de input, o cronograma e o dado enviado por um veículo. Sabendo que não faz sentido processar o dado de um veículo que não possui um cronograma cadastrado, é mostrado primeiro como cadastrar um cronograma e somente depois como enviar o dado de um veículo.

Cadastrando um cronograma

Para cadastrar um cronograma devemos acessar a página de formulário fornecido pelo front-end, para acessá-la basta clicar no link “form” na barra de navegação ou acessar `http://localhost:3000/form`. Uma vez na página de formulário, escolha o endereço de onde a transação deve ser enviada, em seguida carregue um arquivo que contenha o cronograma desejado, o diretório *data_demo* possui dois arquivos de exemplo, por fim clique em submit.

Enviando dados de um veículo

Para o envio de dados de um veículo não existe uma página com interface para isso, apenas uma rota para a qual deve-se enviar tais dados, a rota é `http://localhost:3000/submit`. Portanto, deve-se fazer um POST para essa rota, tento no corpo da requisição um JSON com os dados. Abaixo temos exemplos de envios usando o comando *curl*.

```
curl -H "Content-Type: application/json" -d '{
"fromAddress":"0x14dC79964da2C08b23698B3D3cc7Ca32193d9955",
"data": { "bus_id":  "18C", "trip_id":"18C;1",
"lat": 57.828261, "lon": 26.535419,
"ts": "2022-03-25 07:45:50" }
}'
http://localhost:3000/submit
```

Listing 4.1. Enviando dados da linha 18C (Veículo fora da rota)

```
curl -H "Content-Type: application/json" -d '{
"fromAddress":"0x14dC79964da2C08b23698B3D3cc7Ca32193d9955",
"data": { "bus_id":  "18C", "trip_id":"18C;1",
"lat": 57.82847892, "lon": 26.53362055,
"ts": "2022-05-04 07:48:30" }
}'
http://localhost:3000/submit
```

Listing 4.2. Enviando dados da linha 18C (Veículo atrasado)

Caso deseje-se enviar outros dados, é importante ressaltar que o JSON enviado no corpo da requisição deve possuir os seguintes campos:

- fromAddress: O endereço de onde a transação deve ser enviada
- data: O payload que será processado pelo back-end do DApp
  - bus_id: Identificador da linha de ônibus.
  - trip_id: Identificador da viagem, é o bus_id concatenado com o inteiro correspondente ao número da viagem.
  - ts: momento da medição, está no formato %Y-%m-%d %H:%M:%S
  - lat: latitude medida no momento "ts".
  - lon: longitude medida no momento "ts".

4.7. Conclusões e considerações finais

Nesta seção iremos revisitar diversos dos conceitos vistos até o momento, procurando ressaltar os pontos mais importantes a respeito dos mesmos, seguindo a ordem em que aparecem no texto.

4.7.1. Blockchain

Podemos resumir Blockchain como um livro razão digital, ou seja, uma ferramenta de auditabilidade que veio para oferecer segurança e transparência em transações envolvendo duas partes que não confiam uma na outra, sem a necessidade de uma terceira parte confiável. Sobre essa definição destacam-se os termos "transparência" e "transação", a Blockchain é capaz de oferecer transparência porque é auditável, mas ao mesmo tempo mantém um certo grau de privacidade, pois uma pessoa ou empresa não precisa se identificar ao criar uma carteira no Bitcoin, por exemplo, e ainda pode criar quantas quiser. Já o termo transação, é genérico e pode assumir diversos papéis, pode simplesmente representar uma transação monetária, ou a execução de um complexo método de um contrato inteligente com diversas transações internas. Um ponto a ser destacado é que existe uma terceira parte confiável na Blockchain, mas esse papel não cabe a um indivíduo ou entidade, a rede como um todo realiza esse papel através de algoritmos de consenso.

Mas a Blockchain, tal como qualquer sistema distribuído, não consegue alcançar os três vértices do triângulo de tradeoffs exibido na Seção 4.4. E como a Blockchain envolve criptoativos na forma de tokens, as soluções não desejam abrir mão da descentralização, visando evitar que determinada entidade seja capaz de manipular tais ativos, e nem da segurança, este por motivos ainda mais óbvios. Sendo assim, a primeira limitação que nos deparamos na tecnologia Blockchain é a escalabilidade, vale ressaltar que isso ocorre apenas para Blockchains públicas, como o Ethereum e o Bitcoin, uma vez que uma Blockchain privada está abrindo mão da descentralização ela deve ser capaz de obter um desempenho melhor nos outros dois pontos.

4.7.2. Escalabilidade

Por conta da dificuldade mencionada em abrir mão da descentralização e da segurança, os olhos e esforços dos desenvolvedores daqueles que contribuem com a tecnologia Blockchain voltaram para a escalabilidade. No texto destacamos duas das principais linhas de proposta nesse segmento, Sharding e Rollups. Porém, o sharding acarreta em alguns desafios difíceis de serem resolvidos de forma ótima:

- Distribuir os nós nos Shards de forma uniforme e imprevisível, para evitar que um shard fique com poucos nós e também evitar que algum atacante mal intencionado preveja os nós que compõem os Shards.
- Criar um controle de identidade com o objetivo de impedir ataques sybil.
- No caso do Complete Sharding também surge a necessidade de executar algoritmos de consenso intra-shard e inter-shard, tendo como objetivo manter a coerência das transações entre shards.

Dado estes pontos, o Rollups se mostra uma boa alternativa devido a sua premissa de acumular as transações em lotes e não gerar problemas com o nível de complexidade

daqueles presentes no sharding.

4.7.3. DApps

O problema de escalabilidade mencionado por si só acaba inviabilizando diversos tipos de DApps que envolvem a necessidade de um tempo de resposta rápido ou que devem atender um número de usuários muito grande.

Outro fator que muitas das vezes inviabiliza a criação de determinados DApps é o custo, o custo de implantação e execução dos contratos inteligentes também inviabiliza a criação de DApps que envolvem uma computação muito complexa.

Como DApps são constituídos de um ou mais contratos inteligentes, é importante ressaltar a dificuldade de desenvolvê-los, uma vez que possuem linguagem própria, variando de Blockchain para Blockchain, como visto para o Ethereum e o Bitcoin, e ainda a ausência de um ambiente de desenvolvimento adequado com debuggers e outras funcionalidades.

4.7.4. Cartesi Rollups

A estrategia de Optimistic Rollups implementada pela Cartesi aborda as questões de escalabilidade e custo. As transações podem ser processadas em uma taxa maior fora da Mainnet e custo de transação e diluído entre todas as transações do lote. A segurança da Cartesi Rollups é garantida por fraud proofs, as transações são confirmadas se não forem contestadas. A Cartesi Machine oferece computação verificável, duas execuções com os mesmas entradas geram sempre o mesmo resultado.

Além de habilitar a verificação do fraud proofs a Cartesi Machine possui uma arquitetura RISC-V com sistema operacional Linux. Ao mover a maior parte de sua lógica DApp para rodar dentro de Máquinas Cartesi, os desenvolvedores podem usar as linguagens e ferramentas de sua escolha.

Assim, podemos resumir o Cartesi Rollups como uma solução para o problema de escalabilidade das Blockchains, enquanto oferece um ambiente familiar para o desenvolvimento de DApps.

Referências

[Argento 2021] Argento, F. (2021). Rollups: On-chain. https://medium.com/cartesi/rollups-on-chain-d749744a9cb3.

[Benet 2014] Benet, J. (2014). Ipfs - content addressed, versioned, p2p file system.

[Binance 2020] Binance, A. (2020). What is ethereum plasma? https://academy.binance.com/en/articles/what-is-ethereum-plasma. Accessed on 2022-05.

[bit2me 2020] bit2me (2020). O que é sharding? https://academy.bit2me.com/pt/o-que-é-sharding/.

[Buterin 2021] Buterin, V. (2021). An incomplete guide to rollups. https://vitalik.ca/general/2021/01/05/rollup.html. Accessed on 2022-04.

[Cartesi 2022] Cartesi (2022). Cartesi Documentation. https://www.cartesi.io/en/docs/.

[Corso 2019] Corso, A. (2019). *Performance analysis of proof-of-elapsed-time (poet) consensus in the sawtooth blockchain framework*. PhD thesis, University of Oregon.

[Daudén-Esmel et al. 2021] Daudén-Esmel, C., Castellà-Roca, J., Viejo, A., and Domingo-Ferrer, J. (2021). Lightweight blockchain-based platform for gdpr-compliant personal data management. In *2021 IEEE 5th International Conference on Cryptography, Security and Privacy (CSP)*, pages 68–73.

[Douceur 2002] Douceur, J. R. (2002). The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer.

[Eichmann 2018] Eichmann, K. (2018). The future client of an energy utility company will be a machine. https://medium.com/future-energy-ventures/machine-economy-a-decentralized-future-that-is-enabled-by-autonomous-machine-to-machine-e497b90f13c1.

[Ferdous et al. 2019] Ferdous, M. S., Chowdhury, F., and Alassafi, M. O. (2019). In search of self-sovereign identity leveraging blockchain technology. *IEEE Access*, 7:103059–103079.

[Gentry 2009] Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. In *STOC '09*.

[Governo Federal 2021] Governo Federal, M. d. E. (2021). Guia de requisitos mínimos de segurança e privacidade para aplicativos móveis - lgpd. https://www.gov.br/governodigital/pt-br/seguranca-e-protecao-de-dados/guias/guia_seguranca_apps.pdf.

[Hong et al. 2021] Hong, Z., Guo, S., Li, P., and Chen, W. (2021). Pyramid: A layered sharding blockchain system. *IEEE INFOCOM*.

[Jake Frankenfield 2020] Jake Frankenfield, S. G. A. (2020). Proof of elapsed time (poet) (cryptocurrency). https://www.investopedia.com/terms/p/proof-elapsed-time-cryptocurrency.asp.

[Johnston et al. 2014] Johnston, D., Yilmaz, S. O., Kandah, J., Bentenitis, N., Hashemi, F., Gross, R., and Mason, S. (2014). The general theory of decentralized applications. *DApps, URL-https://cryptochainuni. com/wp-content/uploads/The-General-Theory-of-Decentralized-Applications-DApps. pdf*.

[King and Nadal 2012] King, S. and Nadal, S. (2012). Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19:1.

[L. 2019] L., K. (2019). The blockchain scalability problem & the race for visa-like transaction speed. https://towardsdatascience.com/the-blockchain-scalability-problem-the-race-for-visa-like-transaction-speed-5cce48f9d44.

[Lerner 2015] Lerner, S. D. (2015). Rsk. Technical report, Tech. Rep.

[Maxwell et al. 2019] Maxwell, G., Poelstra, A., Seurin, Y., and Wuille, P. (2019). Simple schnorr multi-signatures with applications to bitcoin. *Designs, Codes and Cryptography*, 87(9):2139–2164.

[Mearian 2019] Mearian, L. (2019). Sharding: What it is and why many blockchain protocols rely on it. https://www.computerworld.com/article/3336187/sharding-what-it-is-and-why-so-many-blockchain-protocols-rely-on-it.html.

[Moura 2021] Moura, E. (2021). Cartesi rollups. https://medium.com/cartesi/scalable-smart-contracts-on-ethereum-built-with-mainstream-software-stacks-8ad6f8f17997. Accessed on 2022-02.

[Muratov et al. 2018] Muratov, F., Lebedev, A., Iushkevich, N., Nasrulin, B., and Takemiya, M. (2018). Yac: Bft consensus algorithm for blockchain. *arXiv preprint arXiv:1809.00554*.

[Nakamoto 2008] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260.

[OmniLayer 2020] OmniLayer, C. (2020). Omnilayer documentation. https://github.com/OmniLayer/spec/blob/master/OmniSpecification-v0.6.adoc.

[Pahlajani et al. 2019] Pahlajani, S., Kshirsagar, A., and Pachghare, V. (2019). Survey on private blockchain consensus algorithms. In *2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT)*, pages 1–6. IEEE.

[Rebello et al. 2019] Rebello, G., Camilo, G., Silva, L., Souza, L., Guimarães, L., Alchieri, E., Greve, F., and Duarte, O. (2019). Correntes de blocos: Algoritmos de consenso e implementação na plataforma hyperledger fabric. *Sociedade Brasileira de Computação*.

[Revoredo 2019] Revoredo, T. (2019). Blockchain: tudo que você precisa saber (potencial e realidade).

[Shor 1994] Shor, P. (1994). Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134.

[Shor 1997] Shor, P. W. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509.

[Stach. et al. 2020] Stach., C., Giebler., C., Wagner., M., Weber., C., and Mitschang., B. (2020). Amnesia: A technical solution towards gdpr-compliant machine learning. In *Proceedings of the 6th International Conference on Information Systems Security and Privacy - ICISSP,*, pages 21–32. INSTICC, SciTePress.

[Szabo 1996] Szabo, N. (1996). Smart contracts: building blocks for digital markets. *EXTROPY: The Journal of Transhumanist Thought,(16)*, 18(2):28.

[Sá 2022] Sá, C. (2022). Escalabilidade para ethereum com rollups. https://goblockchain.io/escalabilidade-para-ethereum-com-rollups/. Accessed on 2022-05.

[Teixeira and Nehab 2018] Teixeira, A. and Nehab, D. (2018). The core of cartesi. Whitepaper, Cartesi.

[Tianchen et al. 2021] Tianchen, W., Miaoyan, X., Han, C., and Yuming, Y. (2021). An in-depth look at rollup's tech, application, and data. https://medium.com/huobi-research/an-in-depth-look-at-rollups-tech-application-and-data-eb8f91bf369b. Accessed on 2022-03.

[Truong et al. 2020] Truong, N. B., Sun, K., Lee, G. M., and Guo, Y. (2020). Gdpr-compliant personal data management: A blockchain-based solution. *IEEE Transactions on Information Forensics and Security*, 15:1746–1761.

[Wang et al. 2019] Wang, G., Shi, Z. J., Nixon, M., and Han, S. (2019). Sok: Sharding on blockchain. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 41–61.

[Wood 2014] Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151.

[Wu et al. 2021] Wu, K., Ma, Y., Huang, G., and Liu, X. (2021). A first look at blockchain-based decentralized applications. *Software: Practice and Experience*, 51(10):2033–2050.

[Zhou et al. 2020a] Zhou, Q., Huang, H., Zheng, Z., and Bian, J. (2020a). Solutions to scalability of blockchain: A survey. *IEEE Access*, 8:16440–16455.

[Zhou et al. 2020b] Zhou, Q., Huang, H., Zheng, Z., and Bian, J. (2020b). Solutions to scalability of blockchain: A survey. *IEEE Access*, 8:16440–16455.