Faculdades Integradas de Taquara - Faccat
Av. Oscar Martins Rangel, 4.500
Taquara, RS, CEP 95600-000

Curso de Sistemas de Informação

BLID: RESTFUL API PARA GERAÇÃO DE IDENTIDADE DIGITAL EM UM SISTEMA DISTRIBUÍDO BASEADO EM BLOCKCHAIN

Marcelo Cardoso da silva
Faculdades Integradas de Taquara – Faccat – Taquara – RS – Brasil
marcelosilva@sou.faccat.br

Francisco Assis Moreira do Nascimento
Faculdades Integradas de Taquara – Faccat – Taquara – RS – Brasil
assis@faccat.br

Resumo

Este artigo apresenta uma aplicação descentralizada para gerenciar identidade digital soberana para usuários de uma rede distribuída ponto-a-ponto (P2P) privada. O Blid pode gerar identidades para aplicações distribuídas baseadas em *blockchain* em diversas áreas. As identidades são fundamentadas na reputação do usuário perante os nodos da rede privada, selecionados para realizar o consenso voltado para a validação dos dados dos usuários submetidos ao Blid. A aplicação proporciona a segurança e a confiabilidade necessárias para utilização das identidades na rede distribuída privada.
Palavras-chave: identidade digital soberana; *blockchain*; contrato inteligente; rede distribuída.

BLID: RESTFUL API FOR DIGITAL IDENTITY GENERATION IN A BLOCKCHAIN-BASED DISTRIBUTED SYSTEM

Abstract

This paper presents a decentralized application to manage sovereign digital identity for users of a private peer-to-peer (P2P) distributed network. Blid can generate identities for distributed blockchain-based applications in many areas. The identities are based on the user's reputation provided by the private network nodes, which are selected to achieve the consensus for the data validation of the users submitted to the Blid. So the Blid application, provides the security and reliability necessary to be used in the private distributed network.
Keywords: *sovereign digital identity; blockchain; smart contract; distributed network.*

1 INTRODUÇÃO

Com a evolução e a grande disseminação do uso de tecnologias por todas as classes sociais, as pessoas passam a utilizar cada vez mais as tecnologias de acesso à *internet*, com diversos tipos de equipamentos para se conectar, tais como os *smartphones*. Estas pessoas permanecem muito tempo *online*, tendo acesso a bancos, redes sociais, jogos, lojas *online*, podendo realizar abertura de contas, empréstimos e fazer compras. Para ter acesso a essas plataformas e serviços, usa-se como identificação, em geral, um *email* e senha, que são muito fáceis de serem criados e recriados a qualquer momento, sem controle ou validação, inclusive existindo a possibilidade de se criar várias identidades para um único e mesmo indivíduo (BATISTA *et al.*, 2018).

Sem alguma forma de controle sobre as informações colocadas *online*, se está sujeito a uma série de problemas, como o ataque de *hackers*. Tais como os, que interferiram na eleição presidencial de 2016 nos Estados Unidos (BROOKER, 2018). Nas eleições de 2018 com 120 milhões de usuários no Brasil, onde a disseminação de notícias falsas pelas redes sociais levou o *WhatsApp* a banir milhares de contas no período das eleições (PIAZENTIN, 2018).

Existe um desejo crescente nas organizações, de pensar uma nova forma de como se fornecer segurança para que as transações com os clientes, funcionários, parceiros e fornecedores sejam mais ricas, seguras e flexíveis (WINDLEY, 2005).

A capacidade de provar quem você é passa a ser um componente fundamental da economia digital, para uso de serviços públicos ou privados na saúde, educação, serviços e justiça. De acordo com o banco mundial, estima-se que 1.1 bilhão de pessoas não podem provar quem são. Analisando este cenário, se torna muito importante que as pessoas possam ter uma identidade digital soberana (DOMINGO; ENRÍQUEZ, 2018), que pode ser criada digitalmente e então utilizada como única forma de confirmação do usuário ser quem diz ser, em qualquer plataforma ou serviço diferentes para transações envolvendo qualquer tipo de bens digitais, sem a possibilidade de violação e com total controle (BROOKER, 2018).

A identidade digital soberana é gerada com base em documentos do usuário, validados em termos da comprovação da veracidade deles, analisados em uma rede com participantes selecionados para confirmação dos dados (WINDLEY, 2005).

Reforçando esta ideia, Tim Berners-Lee, criador da WWW[1], já dizia que "tudo se baseava em não haver nenhuma autoridade central que você tivesse que pedir permissão", para que o usuário decida com quem e em que momento compartilhar suas informações, em um mundo inteiramente interconectado (BERNERS-LEE, 2018).

[1] World Wide Web

A tecnologia do *blockchain*[2] alterou a maneira de se realizar transações em um sistema distribuído, no qual se compartilham qualquer ativo de valor de uma pessoa (criptomoedas, documentos autenticados, produtos, votos, etc.), entre um conjunto de nós participantes de uma rede P2P[3]. Por ser rastreado, é possivel se reduzir os riscos, posto que um *blockchain* é um grande livro de registros de todos os ativos transacionados em uma escala global, podendo ser usado em qualquer área como finanças, médicas e políticas (SWAN, 2015).

Desta forma, com o uso de *blockchain* se elimina a necessidade de um agente fiscalizador, pois cada nó da rede concorda com a ordem de cada bloco adicionado na corrente, entrando todos em consenso sobre a sua validação.

Cada transação realizada adiciona um bloco a corrente (daí o termo *blockchain* em inglês), onde cada bloco, que traz consigo a chave do bloco anterior. Dessa maneira, tem-se um ambiente distribuído imutável, pois ao alterar o conteúdo de bloco são afetados todos os blocos anteriores a ele, e assim os invalidando. Por si só, esta caracteristica do *blockchain* já possibilita uma gama de aplicações em diversas áreas (SINGHAL *et al*.,2018).

Com o propósito de oferecer uma contribuição para a elevação da qualidade do projeto capaz de gerar uma identidade digital para um usuário, buscou-se identificar e observar trabalhos por empresas que possuem uma estrutura e resultados semelhantes como a Civic, a uPort, a Universal Login e a EduRoam, que serão descritas mais adiante.

Com base nestas informações, constatou-se que seria de grande valia o desenvolvimento de uma API (*Application Programming Interface*)[4] em um sistema distribuído utilizando a tecnologia do *blockchain*. O sistema é voltado ao público em geral, que faça parte de uma rede de organizações, instituições ou indivíduos e que queira vincular essa rede em um único sistema distribuído utilizando a segurança do *blockchain*.

O sistema fornece uma identidade única para o usuário do sistema, em um formato digital, válida e com a segurança de que não exista identidades falsas em uso no sistema, garantindo total segurança dos usuários para transacionarem na rede o que quiserem.

Dentro deste contexto, os objetivos do presente trabalho foram o desenvolvimento de uma *RESTful* API que possibilitasse o entendimento de uma nova tecnologia, de uso crescente no mundo atual (CONG; HE, 2019). E que permitisse aos usuários terem a garantia de estarem em um ambiente seguro com total confiança em todos os usuários de forma equivalente. Com a API desenvolvida, é possível se gerar uma identidade digital para o usuário de uma rede distribuída privada através da reputação do usuário candidato perante o restante da rede. Assim, esta aplicação serve como

[2] Corrente de blocos – livro de registros
[3] Ponto-a-ponto
[4] Interface de programação de aplicativos

ferramenta para se utilizar em qualquer tipo de sistema distribuído, que utiliza a tecnologia *blockchain*, para que se tenha certeza da honestidade de seus usuários.

Foi adotado, no presente trabalho, o padrão *JSON Web Token* para autenticação de grupos de usuários e se limitar os níveis de acesso. Foi realizado o desenvolvimento de uma rede distribuída privada para mineração e consenso dos nodos no *blockchain*, incluindo o desenvolvimento de um *smart contract* específico para a API, que realiza o armazenamento das transações na rede *blockchain* privada.

A estrutura deste artigo está organizada da seguinte forma: a Seção 2 apresenta a fundamentação dos temas abordados e das tecnologias utilizadas; em seguida, a Seção 3 aborda trabalhos correlatos; na Seção 4, apresenta-se detalhes da RESTful API; a Seção 5 descreve experimentos realizados, e, para finalizar, a Seção 6 apresenta as conclusões do artigo e as perspectivas de trabalho futuro.

2 REFERENCIAL TEÓRICO

2.1 Identidade Digital

Identidade é um termo complexo, definido em diferentes formas e contextos. No geral, se entende como um conjunto de características, que definem uma pessoa ou organização e pode ser usada para identificar um indivíduo como único (HALLER; MULLER, 2008). A forma que tem sido usada para gerar, comprovar e validar identidades de pessoas ou organizações, tem sido baseada sempre em interações de documentação na forma física (DOMINGO; ENRÍQUEZ, 2018).

Mas, o cenário atual, onde quase tudo é realizado online e praticamente todo serviço consumido pelo usuário demanda a criação de perfis e senhas para permitir o acesso a conteúdo, serviços, comunicação com outras pessoas, solicitação de documentos, serviços do governo e download de arquivos, produz assim uma quantidade enorme de identidades variadas para um único indivíduo (BASTISTA *et al.*, 2018).

A criação de uma identidade soberana para um mundo digital não significa apenas um número para identificar um indivíduo, uma organização e outras entidades, que existem em formato eletrônico. Mas, pode ser um conjunto de dados específicos para uma determinada situação, como todos os dados para uma transação bancária, número de cartão de crédito, conta ou documentos necessários para uma transação entre empresas. Ou seja, todo conjunto único de dados pode gerar uma identidade digital soberana (WINDLEY, 2005).

Segundo Domingo e Enríquez (2018), a identidade digital soberana gerada após o usuário provar através de documentos submetidos a uma rede validadora sobre a veracidade dos dados do usuário, se torna uma maneira de um usuário ou empresa poder provar que está *online* e com um nível de confiança apropriado para realizar quaisquer tipos de transação. Quanto mais confiável for uma

identidade digital, mais complexas serão as transações *online* que o usuário poderá realizar e o valor dos dados transacionados assim aumentar.

2.2 Blockchain

Em 2008, quando surgiu o *whitepaper* "*Bitcoin: A Peer to Peer Electronic Cash System*", criado por uma pessoa com o pseudônimo de Satoshi Nakamoto, se conheceu a tecnologia *blockchain*. Um livro de registro para um sistema econômico, descentralizado, sem nenhum intermediário e sendo possível a verificação de cada registro criptografado por todos os usuários da rede *peer-to-peer*[5] (DANG, 2019).

Os participantes da rede tem que verificar, auditar e concordar com as transações adicionadas a um *blockchain*, mantendo o consenso entre as partes, garantindo a confiança de que apenas o destinatário, que detenha a chave privada, possa ter acesso aos dados contidos nos blocos e dando a garantia da integridade dos dados, que não poderão ter sido forjados ou modificados por outros usuários não autorizados na rede de *blockchain* (SINGHAL *et al*., 2018).

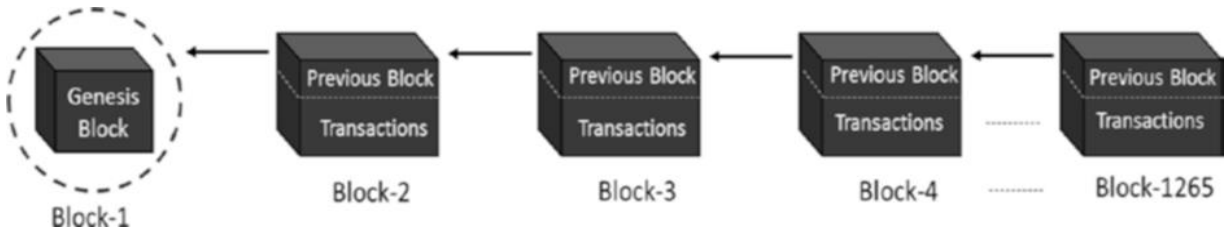
Este processo garante a integridade de um *blockchain*, e assim, ao gerar assinaturas digitais baseadas em criptografia de chave pública e privada, que são seguras, o destinatário pode então verificar a integridade de um *blockchain*, analisando se a transação foi ou não assinada pela chave privada correta, que é imutável (WINDLEY, 2005).

Um *blockchain* privado permite que uma rede descentralizada de diferentes nós, que já estão devidamente validados sobre a sua confiabilidade na rede, concordem sobre o estado verdadeiro dos dados compartilhados nos próximos blocos. Estes blocos contém todas as informações sobre o indivíduo ou empresa em uma forma digital, de maneira segura e incorruptível (TAPSCOTT; KAPLAN, 2019).

O indivíduo deve ter total controle sobre seus dados, comtendo o direito garantido de possuir seus dados, independentemente de qual entidade os coletou e que podendo ter acesso no momento em que desejar, além de poder encerrar contas e remover seus dados, descartar ou distribuir a qualquer momento (SHRIER *et al*., 2016).

A Figura 1 mostra a estrutura dos blocos em um *blockchain*.

Figura 1 - Estrutura de dados do blockchain.



Fonte: Singhal *et al*., (2018).

[5] Ponto-a-ponto

Cada nodo da rede tem uma cópia idêntica do *blockchain* inteiro, onde cada bloco é uma coleção de transações. Existem duas partes principais em cada bloco: a parte do cabeçalho, que faz o *link* para o bloco anterior, contendo o seu *hash* para que ninguém possa alterar qualquer transação nos blocos, pois para isto teria que gerar novamente todos os *hashes* dos blocos existentes. A outra parte de um bloco é o conteúdo do corpo, que tem uma lista validada de transações, seus valores e os endereços das partes envolvidas. Então, dado o último bloco, é possível acessar todos os blocos anteriores em um *blockchain* (SINGHAL *et al*., 2018).

2.3 Carteiras Digitais

As carteiras digitais são similares às carteiras tradicionais de dinheiro, que guardam cédulas. Uma carteira digital armazena os dados financeiros, documentos digitais e de identidade do usuário no âmbito virtual. Em um sistema distribuído baseado na tecnologia *blockchain*, todos os participantes têm carteiras digitais usadas para armazenar as chaves privadas, os documentos digitalizados e as assinaturas digitais, que representam os bens digitais, que os usuários possuem (HERBERT; LITCHFIELD, 2015).

Podem ser armazenadas de forma privada ou *online*, dependendo do perfil do usuário e do uso da carteira, tornando possível realizar uma série de operações de maneira mais cômoda e rápida entre usuários (CHEN, 2017).

2.4 Smart Contracts

O contrato inteligente é o que transforma um *App* (do termo em inglês, *application*) em um *DApp* (do termo em inglês, *decentralized application*) realmente descentralizado. Ele que consiste em um conjunto de códigos, que pode ser executado em um *blockchain*. Contratos inteligentes geralmente suportam lógica programável e armazenamento de dados de forma imutável. Para armazenar dados no *blockchain*, é implantado um contrato inteligente específico para cada tipo de *DApp*, e a API de identidade digital também utiliza um contrato inteligente específico (NEO-PROJECT, 2019).

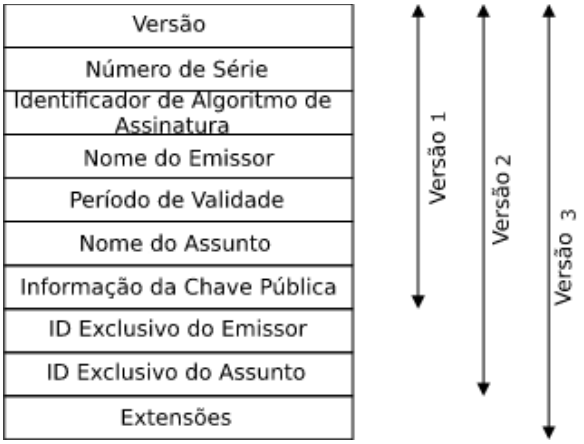
O sistema de contrato inteligente da *NEO Smart Economy*, criado em 2014 e aberto no *GitHub* em junho de 2015 é um projeto sem fins lucrativos, que utiliza a tecnologia *blockchain* e identidade digital para digitalizar e automatizar o gerenciamento de ativos digitais para qualquer finalidade, usando contratos inteligentes em uma rede distribuída.

Os contratos inteligentes podem ser desenvolvidos usando uma variedade de linguagens de programação, que incluem C#, VB.Net, F#, Java, Kotlin e Python. O *NEO* utiliza o padrão X.509 para identidade digital (NEO, 2019).

Devido à grande quantidade de linguagens de alto nível, a capacidade de criar contratos inteligentes torna cada vez mais popular a plataforma *NEO*, por não se precisar aprender uma linguagem específica como a *Solidity* do *Ethereum*. O padrão PKI (*Public Key Infrastructure*)[6] X.509 voltado para certificação digital baseada em chaves, que são geradas para um usuário, envolve um conjunto de requisitos para se gerar um certificado, que vincula uma chave pública a um indivíduo (NEO, 2019).

A Figura 2 mostra a estrutura do certificado X.509 que já teve três evoluções desde sua criação em 1998, onde foram inseridos campos para se ter um aumento na segurança e confiabilidade do *token* gerado. Como *Issuer Unique* ID[7] que contém um valor exclusivo que pode ser usado para tornar o nome X.509 da autoridade de certificação não-ambíguo quando utilizado por entidades diferentes ao longo do tempo, inserido na versão 2 e *Extensions*[8] que contém uma lista de extensões e suas descrições suportadas pela *microsoft* como *Authority Key Identifier*, *Key Usage* e *Certificate Policies*, inserido na versão 3.

Figura 2 - Certificado X.509.



Fonte: *Microsoft* (2019).

2.5 Plataformas para Blockchain

O *blockchain* teve origem no protocolo do *Bitcoin* em 2009, voltado principalmente para transações de moedas digitais. Com a experiência da utilização da tecnologia *blockchain* no *bitcoin*, surgiram várias plataformas, que procuravam contribuir com melhorias.

2.5.1 Ethereum

[6] Infraestrutura de chave pública
[7] ID exclusivo do emissor
[8] Extensões

A plataforma *Ethereum*[9], idealizada pelo Vitalik Buterin, adota o conceito de contratos inteligentes com uma lógica de transação melhorada, possibilitando a implementação de aplicações descentralizadas e autônomas.

Para realizar as transações, o *Ethereum* utiliza ativos criptográficos chamados de Ether e, para possibilitar o consenso, usa um mecanismo de prova de trabalho (*proof-of-work*) ou prova de participação (*proof-of-stake*).

O *PoW* recompensa os mantenedores da rede que resolvem as criptografias complicadas para validar as transações e criar blocos. E garante que, após validar as transações, não possam ser negadas, porque isso exige que a entidade mal-intencionada tenha poder computacional para competir com toda a rede existente. Alcançando um consenso robusto e à prova de adulteração sobre a validade dessas transações. Ao contrário do *PoW*, no *PoS*, o criador do próximo bloco é escolhido de maneira determinística, a chance de um nodo ser escolhida depende de quantas de criptomoedas ele tem. Em ambos casos o objetivo é incentivar manutenção da rede ao mesmo tempo em que reduz manipulação e adulteração de dados (CONG; HE, 2019).

Os contratos da plataforma *Ethereum* são escritos na linguagem *Solidity*. O usuário precisa utilizar a extensão *Metamask* no navegador para executar as transações ou utilizar aplicações de terceiros, como o *Universal Login* ou *uPort*, como soluções *on board*[10], descritos mais adiante.

O *Ethereum* tem um tempo alto para realizar transações, resultando em um problema de escalabilidade, ou seja, não facilita o aumento desta taxa de processamento de transações. Para lidar com esse problema, uma solução tem sido usar *side-chains* (DANNEN, 2017)

Segundo Johnson *et al*. (2019), os *side-chains* são redes *blockchain* secundarias, ligadas a rede principal de *blockchain Ethereum*, que realizam o mesmo *smart contract* da rede principal, visando se ter ganho na escalabilidade. Exemplos de *side-chains* incluem as redes *Ethereum* privadas fornecidas por *Kaleido*[11] e *Quorum*[12], ou mesmo aplicações que provem esse tipo de serviço como Plasma[13], POA *Network*[14] ou *Liquid*[15].

2.5.2 NEO - Smart Economy

A plataforma *NEO Smart Economy*[16] utiliza tecnologia *blockchain* e identidades digitais para digitalizar e gerenciar os ativos digitais usando contratos inteligentes. Para realizar as transações, o

[9] https://www.ethereum.org/
[10] Soluções que são aderidas a aplicação que utiliza a plataforma *Ethereum.*
[11] https://kaleido.io/
[12] https://www.goquorum.com/
[13] https://github.com/omisego/plasma-mvp
[14] https://github.com/poanetwork
[15] https://www.blockstream.com/
[16] https://neo.org/

NEO utiliza ativos criptografados, chamados de *GAS* e, com um mecanismo de consenso, chamado dBFT2.0 (*Delegated Byzantine Fault Tolerant*)[17], atualizado em março de 2019. *NEO* precisa de cerca de 15 a 20 segundos para gerar um bloco. A taxa de transferência de transação é medida em cerca de 1.000 TPS, o que é um excelente desempenho entre as redes públicas.

Os contratos inteligentes no *NEO* são escritos e compilados em C# e Java. Para realizar as transações em plataforma *Windows* pode ser utilizado o *NEO GUI* como cliente da rede e para *Linux* o *NEO CLI* (NEO-PROJECT, 2019).

2.5.3 Bigchain DB

O Bigchain DB[18] é um banco de dados *blockchain* que possui algumas características de banco de dados comum e algumas características de *blockchain*, incluindo descentralização, imutabilidade e suporte nativo a ativos. Em alto nível, é possível se comunicar com uma rede distribuída Bigchain DB, usando a API com requisições *HTTP/HTTPS* do Bigchain DB ou um *wrapper*[19] para essa API, como o *Driver Python* do Bigchain DB. Cada nó do Bigchain DB executa o Bigchain DB Server e vários outros softwares (BIGCHAINDB, 2019).

Dividido em 3 tipos de redes diferentes: uma com nó de desenvolvimento criado no Bigchain DB Server para testar código novo ou alterado, executado na máquina local do desenvolvedor; Uma rede com nó básico, implantado na nuvem, como parte de uma rede de teste; E uma rede de produção, que contém o nó de produção que faz parte da rede Bigchain DB em uso, e que possui mais componentes e requisitos.

Para realizar o consenso nas transações, a mineração utiliza a função de tolerância a falhas bizantinas (BFT[20]), que se trata de algoritmo para tratar falhas no consenso dos blocos, onde calcula-se uma aceitação máxima de falhas. O Bigchain DB não executa contratos inteligentes, mas pode ser conectada a outras redes *blockchain* para executar contratos inteligentes e utilizado como um banco de dados *blockchain* armazenando metadados encriptados de arquivos por exemplo (BIGCHAINDB, 2019).

2.6 RESTFul API

Richardson e Amundsen (2013) com o intuito de criar uma aplicação flexível, para que uma enorme e variada quantidade de desenvolvedores e inúmeros sistemas tenham acesso ao conteúdo, tem sido adotada a utilização de APIs baseada no padrão REST (*Representational State Transfer*[21]).

[17] Tolerante a falhas bizantinas delegadas
[18] https://www.bigchaindb.com/
[19] Função para chamar sub-rotinas, expondo funcionalidades do sistema
[20] Byzantine fault tolerance -tolerancia a falhas bizantina
[21] Transferência de Estado Representacional

As práticas recomendadas para o projeto de uma *RESTful* estão implícitas no próprio padrão *HTTP*, e que consistem em especificar as *URLs*, os métodos utilizados para as requisições e atualização de estado de recursos, além especificar os códigos de estados de respostas e a estrutura do *JSON* a ser adotada. O protocolo *REST* é utilizado como solução para programar e integrar a comunicação entre sistemas *web*, ouvindo diretamente e respondendo a solicitações de clientes. REST utiliza o padrão de arquitetura *HTTP* com foco nas requisições e respostas entre o cliente e servidor através do *header* (MASSE, 2012).

Segundo Costa *et al.* (2014), uma das maiores vantagens para o uso desse *design* na criação de aplicações é a agilidade. Pelo fato de ser um protocolo sem estado e com o mesmo padrão, as integrações são mais rápidas. Além disso, tem-se a flexibilidade de se poder escolher o formato de resposta mais adequado e a praticidade de ter um protocolo bem simples, que pode ser implantado em qualquer cliente/servidor com suporte a *HTTP/HTTPS*[22].

Analisando as requisições enviadas pelo cliente, a API faz uma chamada para o banco de dados com o modelo de dados da aplicação, realizando inserção de dados, alteração, exclusão ou apenas solicitação de dados.

A troca de informações entre o cliente e a API é feita através de um arquivo com formato *JSON*[23]. No *JSON*, existem três tipos de dados: escalar (número, texto, booleano e nulo), matriz e objeto, que descrevem praticamente todos os dados para a serem expostos como recurso. Em geral todas as linguagens de programação modernas oferecem suporte embutido para estes tipos de dados (HOPE *et al*., 2014).

2.7 JSON Web Token

O *JSON Web Token* (*JWT*) define uma maneira compacta e autocontida para transmitir informações com segurança (JONES, 2011). Essas informações podem ser verificadas e são confiáveis porque são assinadas digitalmente. Os *JWTs* podem ser assinadas usando um par de chaves pública/privada. Os *tokens* assinados *JWT* podem verificar a integridade das reivindicações contidas nele. Quando os *tokens* são assinados usando pares de chaves pública/privada, a assinatura também certifica que somente a parte que detém a chave privada é aquela que a assinou (ETHELBERT *et al*., 2017).

A aplicação do cliente solicita a autorização para o servidor de autorização, que é realizada através de um fluxo do código de autorização. Quando a autorização é concedida, o servidor de autorização retorna um token de acesso ao cliente, para que possa utilizar um recurso protegido da *RESTful* API (JONES, 2011).

[22] Hyper Text Transfer Protocol Secure - protocolo de transferência de hipertexto seguro
[23] JavaScript Object Notation

Tokens Web JSON consistem em três partes: a primeira parte é o *header*, que normalmente consiste no tipo do *token* e do algoritmo de *hashing*, usados como *HMAC SHA256* ou *RSA* (ETHELBERT *et al*., 2017).

A segunda parte do *JWT*, que contém as declarações com os dados do cliente, é de três tipos: registradas, públicas e privadas. Sendo que as registradas são um conjunto de declarações pré-definidas, que não são obrigatórias, mas recomendadas para fornecer um conjunto de declarações úteis e interoperáveis; as públicas, podem ser definidas à vontade pelo usuário do *JWT*; e, as privadas, são criadas para compartilhar informações entre as partes que concordam em usá-las e não são registradas ou públicas (JONES, 2011).

A terceira parte é a assinatura usada para verificar se a mensagem não foi alterada ao longo do caminho e, no caso de *tokens* assinados com uma chave privada, ela também pode verificar o remetente e a saída contém três *strings Base64-URL* separadas por pontos que podem ser facilmente transmitidos em ambientes *HTML* e *HTTP* (ETHELBERT *et al*., 2017).

2.8 Off-Chain

O *off-chain* funciona como um armazenamento secundário. Dados fora do *blockchain* podem ser usados para verificação sem custos de alguma transação encaminhada para a rede enquanto aguardam confirmação de *blockchain* e para manter alguns dados essenciais das transações para uma busca mais rápida. Na API de identidade digital, desenvolvida no presente trabalho, é adotado o sistema gerenciador de banco de dados *MongoDB*, que "é um banco orientado a documentos, de código aberto, que fornece alto desempenho, disponibilidade e dimensionamento automático" (MAKSIMOVIC, 2017).

O *MongoDB* é um banco *NoSQL* disponível gratuitamente, sob a Licença Pública Geral GNU (GPL – GNU: *General Public License*)[24] e está disponível sob licença comercial, como parte da oferta comercial da própria empresa *MongoDB*[25].

A principal razão para se afastar do modelo relacional é facilitar o dimensionamento, que, em um modelo de dados orientado a documentos, permite dividir os dados em vários servidores e controlar o balanceamento de dados e de carga com muito mais facilidade.

Um documento é a unidade básica do *MongoDB* que é equivalente a uma linha de dados relacional. Da mesma forma, uma coleção pode ser considerada como um esquema dinâmico. Uma única instância do *MongoDB* pode hospedar vários bancos de dados independentes com suas próprias coleções (MONGODB, 2018).

[24] Licença Pública Geral
[25] https://www.mongodb.com/

Segundo MongoDB (2018), os aplicativos para dispositivos móveis, *web* e na nuvem mudaram as expectativas dos usuários, como plataforma de dados distribuída, o MongoDB oferece aos desenvolvedores quatro recursos essenciais para atender às necessidades modernas de aplicativos: disponibilidade, isolamento de carga de trabalho, escalabilidade e localidade de dados.

3 TRABALHOS CORRELATOS

Com o surgimento da tecnologia *blockchain* e, especialmente, da tecnologia de contratos inteligentes se abrem novas expectativas para trabalhos sobre integridade de dados e, com a união destas tecnologias, surgem as aplicações descentralizadas conhecidas como *DApps* (KLEINAKI *et al.*, 2018).

A tecnologia *Blockchain* foi adotada em muitos setores para diferentes deficiências nos sistemas existentes. O *Ethereum* hospeda diferentes categorias de *DApps*, incluindo intercâmbio, energia, finanças, saúde, identidade, seguro, entre outros (CAI *et al*., 2018).

Atualmente pode-se encontrar diferentes *DApps* em várias áreas de atuação, que possuem diferentes formas de funcionamento, tecnologias utilizadas e resultados gerados. Porém, para gerar e gerenciar identidades digitais são poucas no mundo. A seguir serão apresentadas essas aplicações com uma breve descrição.

3.1 Civic

A Civic é uma plataforma de identidade segura, verificada e descentralizada com tecnologia *blockchain* Ethereum.

Segundo Civic (2019), a Civic emprega uma identidade segura alimentada por *blockchain* para autenticar usuários em aplicativos da *web* e móveis sem a necessidade de nomes de usuário e senhas. A Civic lida com a verificação de números de celulares, *e-mails*, biometria no dispositivo móvel e verificação de documentos de identidade, incluindo passaportes e carteiras de habilitação, e, a realização de verificações de investidores credenciados. O mesmo pode ser encontrado por meio do *link* de acesso: https://www.civic.com/.

3.2 uPort

A uPort é uma *DApp* para identidade aberta, que permite que o usuário registre sua identidade em um *blockchain* Ethereum, enviem e solicitem credenciais, assinem transações e gerencie com segurança chaves e dados (UPORT, 2019).

A uPort utiliza verificação de dados e biometria de dispositivos móveis para uma carteira digital, de maneira similar à Civic, como uma identidade digital para que o usuário controle suas transações. uPort utiliza o *JWT* para validar acesso ao serviço e controle por tipos de requisições e

nível de usuários, utilizando a infraestrutura de uma *blockchain* pública e cobrando por cada transação, uma taxa proporcional à carga computacional que a execução irá impor ao *blockchain*. Este valor é conhecido como "gas". O mesmo pode ser acessado pelo *link*: https://www.uport.me/.

3.3 Universal Login

Universal Login é uma solução de integração do usuário de *DApps* que utilizam *Ethereum*. Com a Universal Login os usuários podem criar uma conta e começar um *DApp*, em qualquer navegador normalmente sem a necessidade de se instalar componentes adicionais. Não são necessárias extensões de navegador, bastando se criar um *wallet* [26]e se adquirir criptomoedas ETH[27].

A Universal Login permite que os usuários comprem criptomoedas instantaneamente com um cartão de crédito ou conta bancária, para que possam começar a usar o *DApp* imediatamente. A Universal Login, que tem capacidade de recuperação de conta de vários dispositivos (KIREJCZYK; BRONISZEWSKA, 2019). O mesmo pode ser acessado pelo *link*: https://universallogin.io/.

3.4 EduRoam

O EduRoam[28] é uma rede internacional de serviços de *roaming* disponível para estudantes e pesquisadores de instituições de ensino superior com conexão *wi-fi* em diversos países. Ele elimina a necessidade de fornecer contas temporárias para usuários visitantes que têm o EduRoam em sua instituição de origem. EduRoam fornece uma solução única para todos os dispositivos móveis de uma instituição com comunicação encriptada.

No Brasil, a Rede Nacional de Ensino e Pesquisa (RNP) é quem gerencia a estrutura e a disponibilização do serviço. Inicialmente utilizado na Alemanha, Croácia, Finlândia, Holanda, Portugal e no Reino Unido, existe no Brasil desde 2002. Fora da Europa começou na Austrália logo chegou às Américas, Canadá e Estados Unidos, hoje estando em uso em mais de 90 países do mundo todo. O EduRoam é um sistema centralizado administrador de uma entidade central, sendo responsável por gerenciar os *logins* de acessos de estudantes e pesquisadores em instituições de ensino superior, que fazem parte da rede de uso do sistema. O mesmo pode ser acessado pelo *link*: https://www.rnp.br/servicos/gestores-de-ti/colaboracao-a-distancia/eduroam.

3.5 Comparação entre as DApps existentes com o Blid

[26] Carteira digital
[27] Criptomoeda do Ethereum
[28] Education roaming

A aplicação Blid, comparada com os *DApps* correlatos mencionados, possui algumas diferenças e vantagens. Os *DApps* usam a plataforma *Ethereum* em uma rede pública, sendo necessário adquirir a moeda digital para utilizar os *DApps*.

O Blid utiliza a plataforma *NEO* para a implementação do *blockchain*, que está se tornando muito popular, principalmente por ter uma proposta diferente e ser mais simples para a implementação comparada ao *Ethereum*.

Já que o *Ethereum* necessita de ferramentas de terceiros como o *Metamask*[29] (carteira digital *Ethereum*, instalado no *browser* do usuário) e o *Ganache*[30] (para desenvolver uma rede privada e utilizar contratos inteligentes *Ethereum*). Além disso, para o *Ethereum* atingir uma melhor capacidade de uso e facilitar a forma com que os usuários consomem os *DApps*, é necessário se aprender a programar na linguagem *Solidity*[31], que é orientada a objetos e permite se implementar contratos inteligentes *Ethereum*. Esta aprendizagem da linguagem *Solidity* é um esforço adicional necessário por parte dos desenvolvedores de aplicações baseadas no *Ethereum*.

Para que não ocorra afunilamento no processamento, tornando-o mais rápido, o *NEO* propõe um sistema diferente de prioridade para seu *blockchain*, onde os nodos que criam os blocos operam como centros de processamento de dados, recebendo as informações dos demais, processando-as no *blockchain*. Os demais nodos operam como exploradores de blocos, que validam e garantem a sua veracidade (MACIEL, 2018).

Diferente dos exemplos citados, que utilizam uma rede pública com *blockchain Ethereum*, o Blid utiliza uma rede privada. Para isto, o uso do *NEO* é essencial, pois permite se editar os valores, que são cobrados para se executar as transações, mantendo esses valores em zero, e assim possibilita que uma rede de empresas utilize o serviço do *blockchain* sem custo.

A vantagem de se utilizar uma rede privada é que, em o anonimato do usuário não sendo prioridade, pode-se manter uma estrutura para base de dados externa, conhecida como *off-chain* que facilita e acelera pesquisas sobre transações, e utilizar *tokens JWT* e tipos de requisições que agilizam o direcionamento dos usuários na aplicação.

Outra diferença é que os *DApps* citados geram uma identidade para o usuário após ser realizado o cadastro na aplicação, sem validação por reputação, permitindo assim se realizar vários cadastros diferentes para o mesmo usuário, o que não é desejado para uma identidade digital. Já o Blid gera uma identidade apenas após a validação pela reputação do usuário na rede, impedindo de serem geradas duas identidades para um mesmo usuário.

[29] https://metamask.io/
[30] https://www.trufflesuite.com/ganache
[31] https://solidity.readthedocs.io/en/v0.5.12/

Em comparação com os *DApps*, o Blid tem a desvantagem de não utilizar fotos dos usuários, biometria e outros formatos de arquivos na validação. O Blid pode ser utilizado como uma alternativa para o EduRoam, que é uma aplicação centralizada. No Blid, a responsabilidade é igualmente dividida entre os participantes da rede distribuída para gerenciar as identidades de acesso e validação da reputação dos usuários.

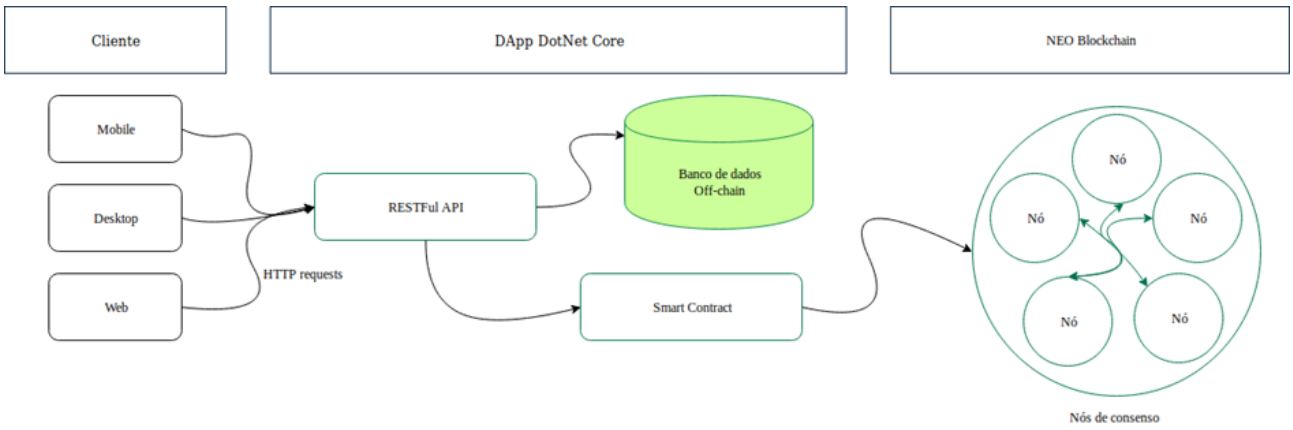
4 DESCRIÇÃO DA API PARA IDENTIDADE DIGITAL PRIVADA

4.1 Visão Geral do Sistema

A *DApp*[32], com arquitetura mostrada na Figura 3, é composta pela união de serviços separados, que em conjunto conectam o cliente a uma rede descentralizada independente da sua origem, seja *mobile*, *desktop* ou uma aplicação *web*.

A *Dapp* Blid oferece funcionalidades de cadastro, gerenciamento de níveis de acesso via *tokens* temporários, resposta sobre o andamento do processo por *e-mail*, armazenamento e validação das informações pessoais dos usuários em um ambiente seguro utilizando *blockchain* privado. Ela permite também o cadastro de nodos definidos para realizar a validação dos dados pessoais por meio da reputação perante a rede, com o objetivo de gerar uma identidade digital soberana para o usuário na rede distribuída e com total controle pelo usuário.

Figura 3 - Arquitetura da Aplicação Blid.



Fonte: Adptado de Lewis (2018)

Na figura 3, a **RESTFul API** é o componete básico da arquitetura e consiste em uma aplicação construída em C#, usando como plataforma de desenvolvimento o .NET Core. Para realizar o controle de acesso, a RESTful API utiliza o sistema *JWT*[33], que se trata de um *token*, contendo um tempo pré-determinado para se conceder direito ao usuário para consumir a API de identidade digital, através da qual o usuário tem acesso a rede *blockchain*.

[32] Aplicação distribuída
[33] JSON Web Token

Ainda na Figura 3, o componete **Off-chain** se utiliza de um banco de dados MongoDB onde o sistema mantém os dados de *login* do usuário, que solicita a identidade, dados sobre os documentos que o usuário submeteu para validação e dados sobre transações realizadas pelos nodos validadores da rede.

Outro componete monstrdo na Figura 3, é o **Smart Contract** que é voltado para a criação dos contratos inteligentes e se utiliza da plataforma *NEO*, para o desenvolvimento de um contrato específico para o sistema. Este contrato específico, realiza a ligação da API com o *blockchain*, transformando a aplicação em uma *DApp*, e é utilizado para armazenar nos blocos os dados específicos requeridos sobre as transações do usuário no *blockchain.*

O **NEO Blockchain**, mostrado na Figura 3, consite em uma rede privada de nodos, onde cada nodo validador mantém uma plataforma *NEO CLI*, que faz o gerenciamento da conexão com as outras máquinas *NEO* nos demais nós. O *NEO blockchain*, mantém os dados dos servidores, chaves públicas dos nodos e realizam o consenso das transações.

4.2 Funcionalidades do Sistema

A API para identidade digital fornece acesso às funcionalidades através de requisições *HTTP* convencionais, utilizando rotas específicas para que sejam realizados *POST* de dados dos usuários, através de formulários diferentes para cadastro, *login*, solicitar a identidade digital, realizar votação sobre a validação dos documentos do usuário como demonstrado no Anexo I, gerenciar especificações do sistema e *GET* de dados em pesquisas específicas sobre estado da solicitação da identidade digital e pesquisa sobre transações realizadas.

Para ter acesso às rotas, a API exige um *token* de acesso *JWT*. As únicas rotas abertas são para criar perfil e realizar o *login* na API. Após validar os dados informados de usuário e senha na rota de *login*, é gerado um *token JWT* único para cada usuário que contém um tempo de duração pré-determinado e com nível do usuário logado, especificando a quais funcionalidade da API esse usuário tem acesso como demonstrado no Anexo I.

As demais rotas tem liberação de acesso apenas com *tokens* específicos para cada nível do usuário, seja ele um solicitante da identidade digital, um usuário validador estabelecido pela rede privada ou um administrador do sistema.

É utilizado o sistema de *off-chain* com o banco de dados MongoDB, o qual armazena os dados de cadastros do usuário, que solicita a identidade digital, dados de *login* dos usuário validadores, dados de *login* do usuário administrador, dados sobre transações realizadas e dados sobre especificações do sistema como quantidade de votos positivos para gerar a identidade do usuário.

O MongoDB gera um *objectId* único para cada cadastro no documento de usuário. Esse Id é o que implementa a ligação com os dados de documentos informados para validação de identidade do usuário, que será armazenada no *blockchain*.

A aplicação solicita os dados dos documentos do usuário para validar a identidade digital, utilizando o número de identidade para realizar pesquisa de dados do usuário no *off-chain*. Os dados dos documentos informados pelo usuário, após realizada a validação quanto ao tipo dos dados enviados e não sendo nulos, são armazenados no *off-chain*, conforme no Anexo II, ligados ao Id gerado pelo MongoDB.

A API invoca o contrato inteligente encaminhando os dados, como demonstrado no Anexo III, que são validados conforme especificações do contrato, verificando a existência do usuário na rede através das chaves informadas.

Após validação do contrato, o bloco é montado contendo os dados do usuário solicitante e a identidade do nodo validador, formada pela chave e endereço da carteira digital do usuário, assim formando uma assinatura única para o bloco, que é então distribuído na rede para mineração e validação da *hash* do bloco.

A rede seleciona um nó participante que será o primeiro a minerar o bloco, após a validação da *hash* o bloco é encaminhado para o próximo até que todos os nós de consenso terminem a sua validação, e, por fim, é armazenado no *blockchain*.

Para realizar a mineração, a plataforma *NEO* utiliza o dBFT2.0[34], que implementa como cálculo de aceitação de erro $F = \lfloor (N - 1) / 3 \rfloor$. No *NEO*, $N = |R|$ sugere o número total de nós, enquanto R representa o conjunto de nós de consenso, F representa o número máximo de nós com falha permitidos no sistema (NEO, 2019).

Assim, a RESTful API envolve o contrato inteligente, o *off-chain* e o *NEO CLI*. Uma máquina *NEO* gera uma carteira digital para a API, que será utilizada para realizar todas as transações referentes à requisição de identidade digital. Cada nodo validador também contém uma máquina *NEO* e uma carteira digital própria para realizar as transações.

Cada máquina *NEO* contém os dados dos servidores das outras máquinas, os IP's e as portas. Os servidores mantêm um intervalo de portas reservados para comunicação entre os nodos. Cada nodo *NEO* contém os *token* públicos das carteiras dos outros nodos para realizarem as transações entre si e são configuradas com o *GAS* em zero para minimizar gastos com transações na rede privada como demonstrado no Anexo IV.

O contrato inteligente utilizado na aplicação tem como base o *template do Smart Contract ICO* da *NEO* disponível no GitHub[35], com as alterações necessárias para tornar o contrato inteligente

[34] Tolerância a falhas bizantina delegada
[35] https://github.com/neo-project/examples-csharp/blob/master/ICO_Template/ICO_Template.cs

único para o sistema. Foram ralizadas alterações de tipos de variáveis e funções que a aplicação realiza, validando todos os dados informados e *tokens* das carteiras dos nodos, confirmando transferências e validando os valores cobrados para cada transação, caso necessário como demonstrado no Anexo V.

No desenvolvimento do sistema, o contrato inteligente para identidade utiliza uma *string* com nome, *int* com número de CPF, *int* com número de identidade e uma *string* com endereço.

Para gerar a identidade digital para o usuário, os nodos de consenso, participantes da rede, realizam a conferência dos dados dos documentos do usuário pela reputação perante a rede, após a API realizar a transferência dos dados para um nodo, esse adiciona um voto de validação ou não dos dados transferidos e realiza a transferência para o próximo nodo validador e a cada transação realizada a API invoca o *smart contract* e armazena essa transação no *blockchain*, como demonstrado no Anexo III, ligando transação entre usuários e realiza a validação garantindo confidencialidade, integridade, autenticação e não repúdio.

Com o resultado da votação favorável, o sistema gera uma identidade digital para o usuário, criando uma carteira digital *NEO* na rede privada da empresa ou rede de empresas, com chaves públicas e privadas, nome da carteira e senha para a mesma, garantindo assim total certeza da identidade dos participantes da rede. Este processo possibilita a união da API de identidade digital com uma rede privada de *blockchain NEO*, direcionada para qualquer tipo de finalidade, podendo realizar transações com o menor custo possível.

Durante o processo para gerar a identidade digital, a API envia uma mensagem para o *e-mail* do usuário sobre o *status* da sua identidade. Caso sua identidade seja negada por causa da reputação perante a rede, o usuário receberá um *e-mail* sobre o status de negado e se a avaliação de sua reputação for positiva, o usuário receberá um email com o status de conformação e os dados de sua identidade na rede contendo sua chave pública, seu endereço de sua carteira com nome e senha da carteira na rede distribuída privada como demonstrado no Anexo V. A reputação na rede é realizada por nodos pré-selecionados para avaliar os dados submetidos do usuário. Cada transação da avaliação é armazenada no *blockchain* privado para garantir a imutabilidade dos dados.

4.3 Desenvolvimento do Sistema

4.3.1 Processo de Software

Como processo de *software* foi adotado o processo ágil Scrum para o desenvolvimento, utilizando o método Kanban para gerenciar o projeto. O Scrum divide o projeto em ciclos chamados *sprints* e as funcionalidades a serem desenvolvidas são mantidas em um *backlog*, e, no início de cada *sprint* algumas funcionalidades são alocadas para desenvolvimento. Elas deverão ser todas entregues

e concluídas no final do *sprint* (SUTHERLAND, 2016). No desenvolvimento do sistema foi assumido um sprint com tempo de duração de 1 mês.

Para gerenciar o andamento do projeto, destacando em que etapa as funcionalidades estavam, dentro de cada *sprint*, foi utilizado o Kanban para manter um controle visual e acompanhar o trabalho conforme o andamento das etapas do desenvolvimento. O Kanban permite acompanhar o tempo de execução da tarefa, possibilitando a otimização do processo (KNIBERG; SKARIN, 2009). No projeto foi utilizado o *Trello*[36] para gerenciar o desenvolvimento, com colunas divididas em *Backlog*[37], *ToDo*[38], *Doing*[39], *Testing*[40] e *Done*[41]. As colunas servem para mapear e ilustrar o fluxo de trabalho e onde cada item está no fluxo de andamento do trabalho e colocado no quadro Kanban na coluna correspondente a etapa do item desenvolvido.

4.3.2 Análise

Durante o processo de análise, foram identificados os elementos necessários e as maiores dificuldades na criação da *RESTful* API, nas construções do *blockchain* privado, da rede distribuída, do *smart contract*[42] específico para a utilização da API de identidade digital e do *off-chain*, que utiliza o banco de dados NoSQL[43] MongoDB. Com base nas informações encontrada, foi desenvolvida a documentação do sistema, realizado o levantamento de requisitos funcionais e não funcionais, e também, a montagem dos diagramas de caso de uso (LARMAN, 2007). O Anexo VI mostra o diagrama de casos de uso do sistema.

Com base no processo de análise de requisitos e definição dos casos de uso, UC-001 (Cadastrar a documentação na API de identidade), UC-002 (Validar a documentação na rede distribuída privada), UC-003 (Registrar no *off-chain*), UC-004 (Visualizar o andamento das transação), UC-005 (Escrever um *feedback* sobre a validação), UC-006 (Gerar uma identidade digital), UC-007 (Autenticar-se na API), foi então desenvolvido o planejamento do desenvolvimento do sistema. Este planejamento consistiu na definição de quatro iterações sequenciais, nas quais foram alocados o desenvolvimento dos casos de uso, dando prioridade para aqueles com mais relevância para o funcionamento e para a proposta central da API.

Finalizado o processo de planejamento, foram então desenvolvidos os casos de uso expandidos, o diagrama de classe, de domínio e os diagramas de sequência monstrados no Anexo

[36] www.trello.com/
[37] Lista de pendências
[38] Lista de fazer
[39] Lista de fazendo
[40] Lista de testando
[41] Lista de feito
[42] Contrato inteligente
[43] Não relacional

VII. Estes diagramas foram sendo incrementados conforme as iterações foram sendo desenvolvidas e finalizadas. O diagrama de classe completo das quatro iterações é mostrado no Anexo VIII.

4.3.3 Projeto

Na fase de projeto, foi planejada a estrutura da API e da rede distribuída privada, com base na análise realizada no desenvolvimento do diagrama de classes e no conjunto de diagramas de sequência das interações com o usuário (LARMAN, 2007). O Anexo IX apresenta o diagrama de componentes desenvolvido.

4.3.4 Codificação

Na fase de construção, foi realizada a codificação da API. Para tal, a linguagem de programação utilizada foi o C#, linguagem orientada a objeto criada pela *Microsoft*. A plataforma de desenvolvimento foi .NET Core[44], que é uma plataforma aberta de uso geral, criada e mantida pela *Microsoft*. O .NET Core é uma multiplataforma, que pode ser executada nos sistemas operacionais *Windows*, *MacOS* e *Linux*. Se trata de uma ferramenta de linha de comando, utilizada para desenvolvimento local e integração contínua. O *blockchain* privado e o *smart contract* foram construídos utilizando a tecnologia *NEO* Smart Economy[45] que pode ser desenvolvido utilizando várias linguagens, mas para o projeto foi utilizada a linguagem C# na plataforma .NET Core para ser executada em multiplataforma.

Para o usuário consumir a *RESTful* API, construída em .NET Core, que é encarregada do *blockchain* e da identidade digital, foi desenvolvido um sistema para realizar o *login* do usuário, armazenando o cadastro com seus dados em um banco de dados *MongoDB*. Este é um banco NoSQL de código aberto escrito em C++, orientado a documentos, que armazena dados em formato JSON com esquema dinâmico e também multiplataforma.

Após confirmação dos dados de *login* do usuário, a *RESTful* API cria um *token JWT* válido, que contêm um tempo pré-determinado para expirar o acesso do usuário e uma chave pública ligada a uma chave privada para gerenciar o nível de acesso em rotas específicas para cada função, tendo os dados sobre suas ações na aplicação armazenadas na rede *blockchain* privada.

4.3.5 Testes

Os testes foram executados no método exploratório de caixa preta, onde as funcionalidades são testadas simulando a interação com o usuário (SOMMERVILLE, 2011). Os testes foram

[44] https://docs.microsoft.com/pt-br/dotnet/core/
[45] https://docs.neo.org/docs/en-us/index.html

realizados ao final de cada iteração do desenvolvimento e ao final do projeto. Durante a validação ao final de cada iteração, utilizou-se o *software* Postman[46] que é uma ferramenta para testar *RESTful* API por meio de requisições *HTTP* (POSTMAN, 2019). Postman foi utilizado para verificar o resultado de cada comando executado na função, rotas de acesso dos usuários, níveis de acessos, e validação dos tempos dos *tokens* dos níveis de acesso. Sendo utilizado o Postman para automaizar e realizar os testas de testes de integração HTTP, testes de sistema e testes de aceitação.

Tambem foram realizados testes das transações e armazenamento de registros no *blockchain*, utilizando o *smart contract* da API de identidade digital. Para realizar os testes iniciais do contrato, foi feito o uso de uma rede privada *NEO* para testes privados de *DApps* antes de publicações em redes públicas, oferecida pela *COZ*[47], que é um grupo internacional e independente de desenvolvedores de código aberto, que trabalha com economia inteligente.

A utilização do Docker[48] oferecido pela *COZ*, contendo uma rede privada se dá pela necessidade de que para desenvolver um contrato inteligente vinculado a uma rede *blockchain* privada ou pública são necessários um valor de 100 a 1000 *GAS* para o *deploy* do contrato e vínculo a rede privada, e uma quantidade não especificada de *GAS* para os testes de registros de transações no *blockchain*. O *docker* é disponibilizado com uma quantidade de 100 milhões de *NEO* e *GAS*, que são as moedas utilizadas pela rede *blockchain NEO smart economy* para que desenvolvedores possam realizar todos os testes necessários de contratos inteligentes e transações nas redes privadas das *DApps*.

4.3.6 Operacionalização

Para operacionalizar a aplicação de identidade digital e construção da rede distribuída, é necessário contratar um serviço de hospedagem, que ofereça servidores com suporte ao sistema operacional desejado pelo cliente. No caso da API um sistema *Linux* 18.04 ou superior, de banco de dados com suporte a *MongoDB* 4.0.12 ou superior, *nginx* para gerenciar o *firewall*, acesso externos, *NEO CLI* v2.9.1 ou superior, e as bibliotecas *libleveldb-dev*, *sqlite3*, *libsqlite3-dev* e *libunwind8-dev* necessárias para executar o sistema *NEO Smart Economy*.

Inicialmente, o servidor de hospedagem contratado deve ter um mínimo de processador de 2 *cores* e 4GB de memória *RAM*, porém com a contínua atualização da aplicação, esta demanda deve aumentar e o serviço de hospedagem deve permitir a escalabilidade necessária para esta situação. O Anexo X apresenta o diagrama de *deploy*.

[46] https://www.getpostman.com/
[47] CityOfZion
[48] https://hub.docker.com/r/cityofzion/neo-privatenet

5 EXPERIMENTOS

Foi desenvolvido um roteiro de atividades a serem realizadas no sistema utilizando três tipos de usuários: pelo usuário que solicita a identidade digital; pelos usuários que validam a reputação; e pelo usuário administrador do sistema, para se validar a usabilidade do sistema, no ponto de vista de cada usuário e no final a criação da identidade digital.

5.1 Resultados Obtidos

Os usuários desenvolvedores de aplicações que podem consumir a API entenderam e aprovaram a forma simplificada de consumo da funções abertas para *login* e cadastro. No Anexo XI, é mostrado o código de uso da API em C#, que o cliente utiliza para realizar uma requisição para a API.

As funções com limites de acesso, como a de solicitação de identidade digital, realizar a votação e configurar parâmetros básicos do sistema, incluindo como mudar quantidade de avaliações positivas para gerar a identidade digital, sofrem uma pequena alteração, sendo adicionado o *JWT* de validação, conforme mostrado nos Anexos XI e XII.

A aplicação que consome a API seja uma aplicação *web* ou *mobile*, não tem a necessidade de tratar uma quantidade alta de dados para mostrar aos usuários. A API faz o tratamento da maior quantidade de dados possíveis, retornando apenas o necessário em um formato simplificado. Uma vez que o usuário realize o cadastro e insere seus dados, adicionando seu *e-mail*, o sistema responde então enviando uma mensagem diretamente para cada usuário conforme mostrado no Anexo XII.

6. CONCLUSÃO

Desenvolver uma aplicação para gerar uma identidade digital utilizando um sistema distribuído com *blockchain* privado se mostrou um grande desafio, principalmente por se tratar de uma tecnologia recente, em uma fase inicial no ambiente corporativo e educacional, contendo pouco conteúdo *on-line*, como tutoriais e suporte.

Uma vez identificado os profissionais que utilizam o sistema e em que ambiente a aplicação se encaixaria, foram selecionadas as ferramentas, as linguagens e as tecnologias utilizadas. Desenvolveu-se um sistema que explorasse o máximo do *blockchain* para inserir e apoiar seu aprendizado no meio acadêmico, tendo em vista a crescente demanda por profissionais nessa tecnologia no mercado internacional e nacional.

O desenvolvimento da tecnologia envolveu um grau elevado de complexidade, por utilizar um contrato inteligente, uma rede *blockchain* para armazenar registros e um *off-chain*, sob demanda de uma API de ligação com o usuário.

Outro desafio que colaborou para o aumento no grau de dificuldade do desenvolvimento da aplicação, foi a escolha da tecnologia utilizada para desenvolver a rede *blockchain*, onde parte da documentação está escrita na língua do país de criação da tecnologia (China), além da ocorrências de muitos problemas, durante o desenvolvimento da aplicação não apresentarem suporte ou problemas similares *online*, pois se trata de uma tecnologia com apenas 2 anos de mercado.

Em comparação com tecnologias, como o *Ethereum*, que devido a quantidade de aplicações existentes, cursos em várias plataformas de educação e ao grau de familiaridades dos usuários onde existe suporte para quase todos os problemas que um desenvolvedor pode sofrer durante a criação de uma aplicação, o grau de dificuldade de desenvolvimento e curva de aprendizado são expressivamente menores, facilitando o desenvolvimento de uma primeira aplicação e em um ambiente acadêmico.

Ao final do projeto, com os experimentos concluídos, foram identificados, como trabalho futuro, a repetição dos testes, mas com a aplicação em produção, sendo utilizada junto com uma rede principal distribuída com *blockchain* e mais usuários. Poderá ser acrescida a função para aceitar diferentes formatos de arquivos no *off-chain*, adicionando no *MongoDB* o *GridFS*. Com este intuito, poderá ser buscado parcerias para incentivos financeiros para alimentar a rede, realizar testes e utilizar o sistema para fins educacionais.

REFERÊNCIAS

BATISTA, A. O. A., DIAS, E. R. B.,; SILVA, M. B. Identificação digital baseada em blockchain: Um conceito disruptivo no ciberespaço. *Anais do V Simpósio Internacional de Inovação em Mídias Interativas*. Goiânia: Media Lab / UFG. Maio, 2018

BERNERS-LEE, T. *"i was devastated": Tim berners-lee, the man who created the world wide web, has some regrets*. [Entrevista concedida a] Katrina Brooker. VANITY FAIR Technology, 2018.

BIGCHAINDB. *BigchainDB Documentation*. Disponível em: <https://docs.bigchaindb.com/en/latest/index.html>. Acesso em: 04 out. 2019.

BROOKER, K. "i was devastated": Tim berners-lee, the man who created the world wide web, has some regrets. *VANITY FAIR Technology*, 2018.

CAI, W. *et al*. Decentralized applications: The blockchain-empowered software system. *IEEE Xplore Digital Library*. Hong Kong. Setembro. 2018.

CHEN, P., JIANG, B., WANG, C. Blockchain-based Payment Collection Supervision System using Pervasive Bitcoin Digital Wallet. *IEEE Xplore Digital Library*. *13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob).* Beijing. Outubro. 2017.

CIVIC. *Secure identity platform*. Disponível em:<https://www.civic.com/products/secure-identity-platform/>. Acesso em: 28 set. 2019.

CONG, L. W.; HE, Z. Blockchain Disruption and Smart Contracts. *Oxford academic: The Review of Financial Studies.* Volume 32, paginas 1754–1797, Fevereiro, 2019.

COSTA, B. *et al*. Evaluating a Representational State Transfer (REST) Architecture. What is the impact of REST in my architecture?. *IEEE Xplore Digital Library, IEEE/IFIP Conference on Software Architecture*. Rio de Janeiro, 2014.

DANG, D.Q. Macroeconomics and Blockchain. *Metropolia University of Applied Sciences*. Tese Administração de Empresas. Maio, 2019.

DANNEN, C. *Introducing Ethereum and Solidity*. Foundations of Cryptocurrency and Blockchain Programming for Beginners. Appress. Berkeley. 2017.

DOMINGO, A. I. S.; ENRíQUEZ Álvaro M. Digital identity: the current state of affairs. *Research, BBVA Bank, Economic Research Department.* Janeiro, 2018.

ETHELBERT, O. *et al*. A json token-based authentication and access management schema for cloud saas applications.. *IEEE Xplore Digital Library, 5th International Conference on Future Internet of Things and Cloud (FiCloud)*. Agosto, 2017.

HALLER, M.; MULLER, B. Characteristics of personality and identity in population surveys. *Bulletin of sociological methodology*. Julho, 2008.

HERBERT, J.; LITCHFIELD, A. D. A Novel Method for Decentralised Peer-to-peer Software License Validation Using Cryptocurrency Blockchain Technology. *Australian Computer Society.* Sydney, Janeiro, 2015.

HOPE, R. M.; SCHOELLES, M. J.; GRAY, W. D. Simplifying the interaction between cognitive models and task environments with the json network interface. *Behavior Research Methods*. Volume 46, paginas 1007–1012. Dezembro, 2014.

JOHNSON, S.; ROBINSON, P.; BRAINARD, J. Sidechains and interoperability. *Cornell University, Computer Science: Cryptography and Security.* Brisbane, 2019.

JONES, M. B. The Emerging JSON-Based Identity Protocol Suite. *W3C Workshop on Identity in the Browser.* Abril. 2011.

KLEINAKI, A. S. *et al*. A blockchain-based notarization service for biomedical knowledge retrieval. *Computational and Structural Biotechnology Journal*. Abril, 2018.

KIREJCZYK, M.; BRONISZEWSKA, J. *Universal Login documentation.* Disponível em:<https://universalloginsdk.readthedocs.io/en/latest/index.html>. Acesso em: 04 out. 2019.

KNIBERG, H.; SKARIN, M. *Kanban e Scrum - obtendo o melhor de ambos.* C4Media Inc., 2009.

LARMAN, C. *Utilizando UML e Padrões.* São Paulo: Artmed editora S.A. 3th edition, 2007.

LEWIS, J. *Designing and developing a dapp on neo.* GitHub. Disponível em: <https://github.com/jnlewis/neo-dapp-tutorial>. Acesso em: 15 jan. 2019.

MACIEL, F. A. *Introdução as criptomoedas: Uma análise de possíveis impactos na economia, investimentos e contabilidade.* Trabalho de conclusão de curso em contabilidade. Universidade de Caxias do Sul – UCS. Fevereiro. 2018.

MAKSIMOVIC, Z. *MongoDB 3 Succinctly*. Sebastopol: O'REILLY Media Inc. First Edition, 2017.

MASSE, M. *REST API Design Rulebook*. Sebastopol: O'REILLY Media Inc. First Edition, 2012.

MICROSOFT. *X.509 Public Key Certificates***.** Disponível em: <https://docs.microsoft.com/en-us/windows/desktop/seccertenroll/about-x-509-public-key-certificates>. Acesso em: 28 set. 2019.

MONGODB. *A MongoDB White Paper - MongoDB Architecture Guide*. MongoDB Inc., 2018.

NEO. *Neo white paper - a distributed network for the smart economy*. Disponível em: <https://docs.neo.org/en-us/whitepaper.html>. Acesso em: 03 set. 2019.

NEO-PROJECT. *The Neo Project*; Disponível em: <https://github.com/neo-project>. Acesso em: 18 set. 2019.

PIAZENTIN, G. A. *Fake news: Jornalismo, internet e fact checking.* Trabalho de conclusão de curso em jornalismo. Universidade metodista de piracicaba, faculdade de comunicação e informática, graduação em jornalismo. Piracicaba. 2018.

POSTMAN. *API Documentation*. Disponível em: <https://learning.getpostman.com/docs/postman/api_documentation/intro_to_api_documentation/>. Acesso em: 18 set. 2019.

RICHARDSON, L.; AMUNDSEN, M. *RESTful Web APIs.* Sebastopol: O'REILLY Media Inc, 2013.

SHRIER, D., WU, W., PENTLAND, A. Blockchain infrastructure(identity, data security) part 3. *Massachusetts Institute of Technology*. Maio. 2016

SINGHAL, B.; DHAMEJA, G.; PANDA, P. S. *Beginning Blockchain - A Beginner's Guide to Building Blockchain* Solutions. Berkeley: Apress, 2018.

SOMMERVILLE, I. *Software Engineering*. London: Pearson Education, 6th edition, 2011.

SUTHERLAND, J. *SCRUM: A arte de fazer o dobro do trabalho na metade do tempo.* São Paulo: LEYA, 2016.

SWAN, M. *Blockchain blueprint for a new economy*. Sebastopol: O'REILLY Media Inc., 1th edition, 2015.

TAPSCOTT, D.; KAPLAN, A. Blockchain revolution in education and lifelong learning. Preparing for disruption, leading the transformation. *Blockchain Research Institute and IBM Institute for Business Value.* Abril, 2019.

uPORT. *Products*. Disponível em: <https://www.uport.me/>. Acesso em: 29 set. 2019.

WINDLEY, P. J. *Digital Identity.* Sebastopol: O'REILLY Media Inc., 1th edition , 2005.

ANEXOS

Anexo I - Rota *Login* do usuário e criar *JWT*.

```
[HttpPost("LoginMember")]
public IActionResult Create([FromBody]Users user)
{
  try
  {
    //pegar parâmetros do front-end
    var login = user.Login;
    var password = user.Password;
    //criptografar senha
    var EncryptedPassword = LoginValidation.NewEncrypt(password, 3);
    if (string.IsNullOrEmpty(login))
      return StatusCode(401, _MessagesResource["Username"].Value);
    if (string.IsNullOrEmpty(password))
      return StatusCode(401, _MessagesResource["Password"].Value);
    //buscar usuário no banco de dados
    var builder = Builders<Users>.Filter;
    var filter = builder.Where(
      x => x.Login == login &&
      x.Password == EncryptedPassword);
    var usuario = dbContext.Users.Find(filter).ToList();
    if (usuario == null)
      return StatusCode(401, _MessagesResource["Login"].Value);
    if (usuario != null)
    {
      foreach (var item in usuario)
      {
        //criar token de validação JWT após confirmação de login
        var email = item.Email;
        var token = new JwtTokenBuilder()
            .AddSecurityKey(JwtSecurityKey
            .Create("i#4#Mcs}237=RGKPen&*ZXf$Boja&$74PA%#53MczkU55ANY]x8y^}
-hq;WE)-_zU$;Y+L]IPp+7=qP"))
      .AddSubject(login)//nome do usuário
      .AddIssuer("DigitalIdentity")
      .AddAudience("Digital Identity bearer")
      .AddClaim("MembershipId", GenerateId())//identificador único
           .AddExpiry(240)//tempo de duração do token
           .Build();
  //retorno para front-end com token válido
  return StatusCode(200,
          new{
             token.Value,
             email
          }
       );
     }
     return StatusCode(401, _MessagesResource["Login"].Value);
```

Anexo II - Especificação de tipos e armazenar no off-chain

```
// modelagen dos tipo
public class Identity
{
  [BsonId]
  public ObjectId _id { get; set; }
  public string AddressWallet { get; set; }
  public string email { get; set; }
  public string Name { get; set; }
  public string ID { get; set; }
  public string CPF { get; set; }
  public string Address { get; set; }
  public uint Vote { get; set; }
}

//dados recebidos do front-end
string address_wallet =
"0369ddab69d9f428521fc2fb310227ee3d977fb17a73044e0bb0d54075fd6196e3";
string email_user = Identity.email;
string name = Identity.Name;
string id = Identity.ID;
string cpf = Identity.CPF;
string address = Identity.Address;
uint vote = 0;

// armazenar no off-chain
Identity identity = new Identity();
identity._id = new ObjectId();
identity.AddressWallet = address_wallet;
identity.email = email_user;
identity.Name = name;
identity.ID = id;
identity.CPF = cpf;
identity.Address = address;
identity.Vote = vote;
dbContext.Identity.InsertOne(identity);
```

Anexo III - Invocar contrato inteligente

```
// invocar contrato inteligente passando valores
var blockchain = Blockchain.InvokeScript("AddIdentity",
  new object[] {
      address_wallet,
      name,
      id,
      cpf,
      address,
      vote
  });
  if (blockchain == false)
    return StatusCode(406, _MessagesResource["InvokeScript"].Value);

// conexão com blockchain
private static string contractScriptHash =
"0x9527ae75e24139282a1f0649f5b9fddddf38c7f4";
public static bool InvokeScript(string method, object[] values)
{
  var scriptHash = new UInt160(contractScriptHash.HexToBytes());
  var api = NeoRPC.ForPrivateNet();
  var response = api.InvokeScript(scriptHash, method, values);
  if (response != null && response.result != null)
  {
    return response.result.GetBoolean();
  }
  else
  {
    Console.WriteLine("Null response received on InvokeScript");
    return false;
  }
}
```

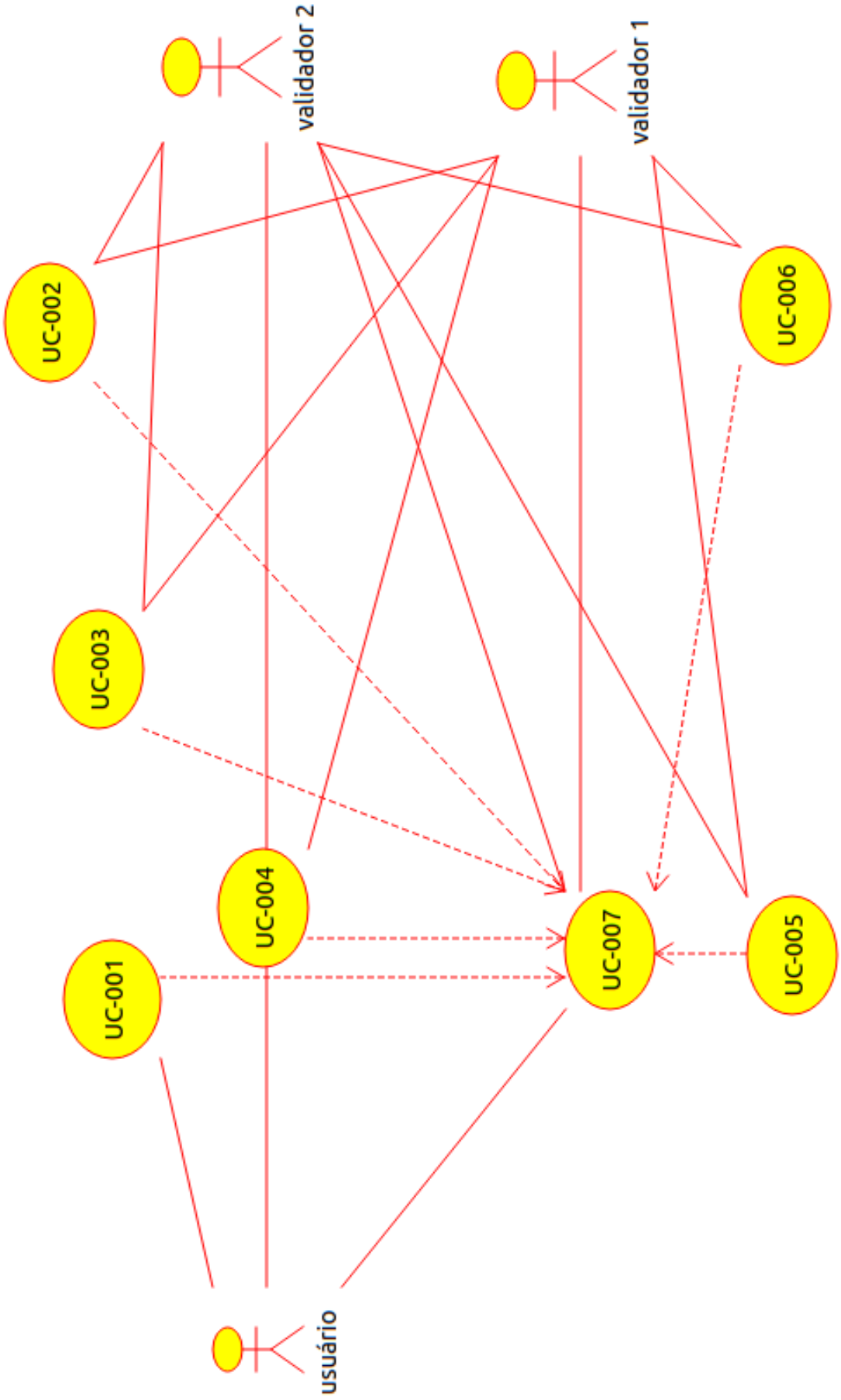
Anexo IV - Conexão entre nodos

```
"ProtocolConfiguration": {
// endereço publico de mineradores e api
"StandbyValidators": [
  "027b7263a00aaa6ee0f19c389eb1bd313c335d9bab8958661e87e5d5b5680c9817",
  "02042996bd7bdc5c15eeaf3ea42e581a8305b0b49645d32b2e0ed024582445befd",
  "0250ffb67e07bd42419237194550bc65edd7ccc86dab2225bec8e0d6c867f92”,
  ”0371f7bc9cccf11ec5715268b4130a35ac6b27ba2a67059574c03026709f1dcc26",
  "0369ddab69d9f428521fc2fb310227ee3d977fb17a73044e0bb0d54075fd6196e3"
],
//endereços do servidor de nodos participantes
"SeedList": [
  “157.230.3.158:10333",
  "68.183.108.114:10333",
  "68.183.21.243:10333",
  "159.65.239.5:10333",
  "159.65.236.70:10333"
],
//zerar gastos de transações
"SystemFee": {
  "EnrollmentTransaction": 0,
  "IssueTransaction": 0,
  "PublishTransaction": 0,
  "RegisterTransaction": 0
}
//especificar porta do nodo
"P2P": {
 "Port": 10333,
 "WsPort": 10334
}
```

Anexo V - Contrato inteligente adicionar transação no bloco

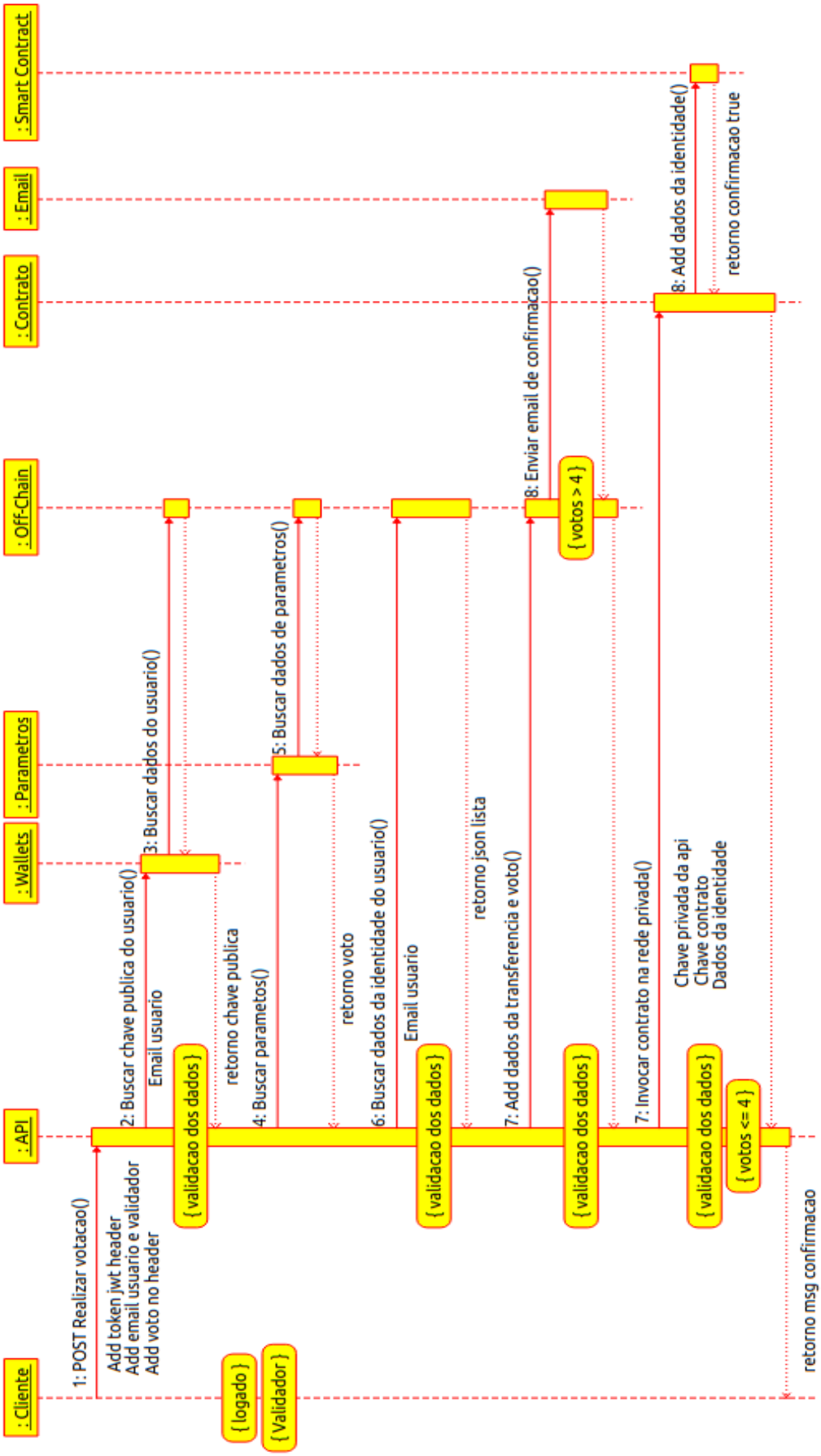
```
//validação dos dados da API
if (operation == "AddIdentity")
{
  if (args.Length != 6) return false;
    byte[] AddressWallet = (byte[])args[0];
  string Name = (string)args[1];
  string ID = (string)args[2];
  string CPF = (string)args[3];
  string Address = (string)args[4];
  uint Vote = (uint)args[5];
  return AddIdentity(AddressWallet, Name, ID, CPF, Address, Vote);
}
//criar bloco adicionando dados da API
private static bool AddIdentity(byte[] AddressWallet, string Name, string ID,
string CPF, string Address, uint Vote)
{
  if (AddressWallet == null || Name == null || ID == null || CPF == null ||
Address == null || Vote < 0)
  {
    Runtime.Log("AddIdentity: One or more required parameter is not specified/Um
ou mais parâmetros obrigatórios não são especificados.");
    return false;
  }
  Storage.Put(Storage.CurrentContext, "Identity_AddressWallet", AddressWallet);
  Storage.Put(Storage.CurrentContext, "Identity_Name", Name);
  Storage.Put(Storage.CurrentContext, "Identity_ID", ID);
  Storage.Put(Storage.CurrentContext, "Identity_CPF", CPF);
  Storage.Put(Storage.CurrentContext, "Identity_Address", Address);
  Storage.Put(Storage.CurrentContext, "Identity_Vote", Vote);
  Runtime.Log("AddIdentity: Successfully added/Adicionado com sucesso.");
  return true;

//chamada para enviar email
var msg = "Request for Digital ID in progress please wait.";
var subject = "Digital Identity.";

var resultSend = Email.SendEmail(subject, msg, email_user);

if (resultSend == false)
    return StatusCode(406, _MessagesResource["Erro406"].Value);
```

Anexo VI - Diagrama geral de casos de uso



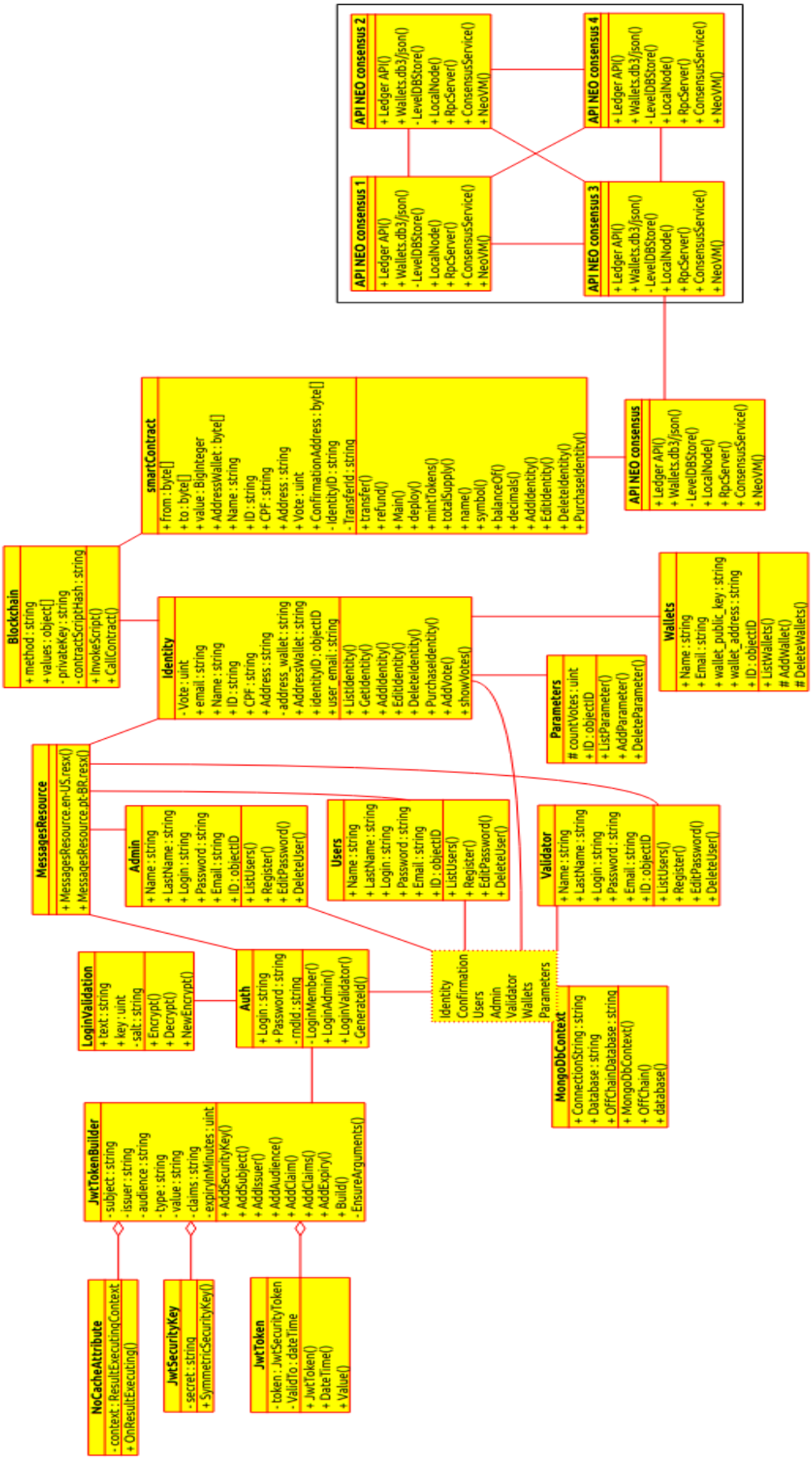
Fonte: Autor (2019)

Anexo VII - Diagrama de sequência da votação.



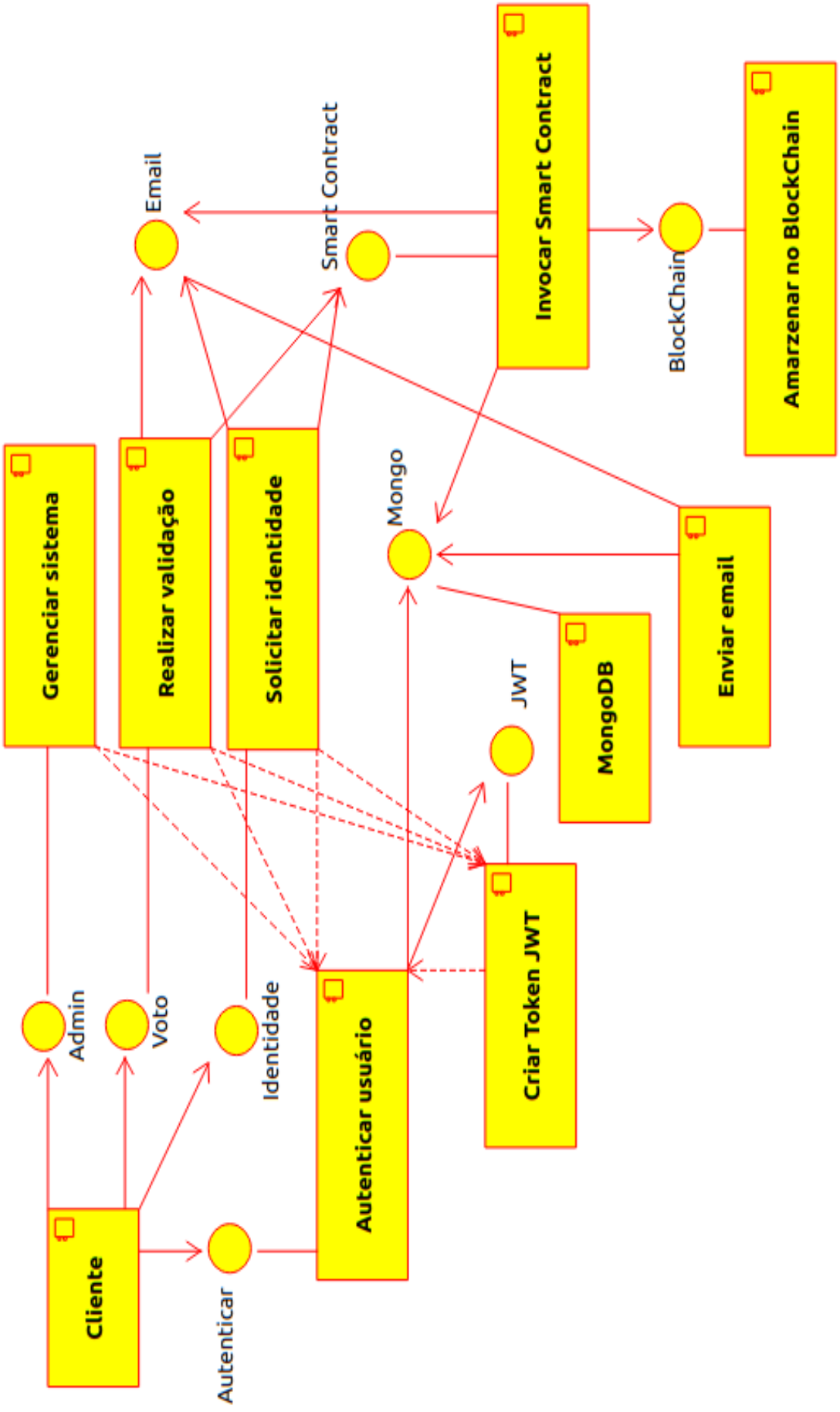
Fonte: Autor (2019)

Anexo VIII - Diagrama de classe completo com as iterações.



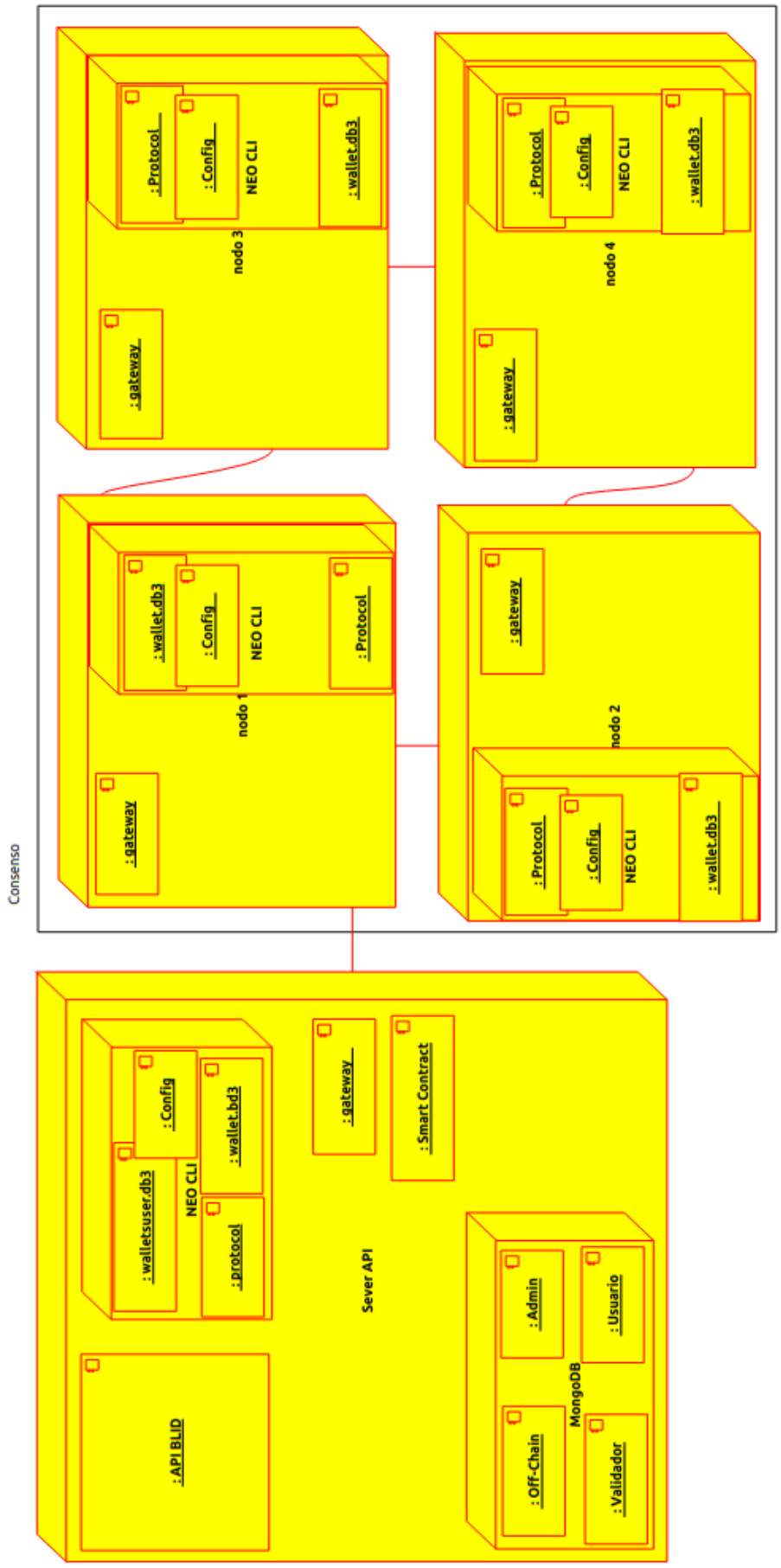
Fonte: Autor (2019)

Anexo IX - Diagrama de componentes



Fonte: Autor (2019)

Anexo X - Diagrama de Deploy



Fonte: Autor (2019)

Anexo XI - Códigos utilizados pelo front-end para consumir a API

```
//código utilizado pelo Front-end para realizar o cadastro na api
var client = new RestClient("https://159.65.236.70/Users/Register");
client.Timeout = -1;
var request = new RestRequest(Method.POST);
request.AddParameter("Name", "Marcelo");
request.AddParameter("LastName", "Cardoso");
request.AddParameter("Login", "MarceloCardoso");
request.AddParameter("Password", "12345");
request.AddParameter("Email", "marcelosilva@sou.faccat.br");
IRestResponse response = client.Execute(request);
Console.WriteLine(response.Content);

//código utilizado pelo Front-end para realizar login na api
var client = new RestClient("https://159.65.236.70/Auth/LoginMember");
client.Timeout = -1;
var request = new RestRequest(Method.POST);
request.AddHeader("Content-Type", "application/json");
request.AddParameter("application/json", "{\n  \"Login\": \"MarceloCardoso\",\n
      \"Password\": \"12345\"\n}",  ParameterType.RequestBody);
IRestResponse response = client.Execute(request);
Console.WriteLine(response.Content);

//código utilizado pelo Front-end para realizar a solicitação de uma identidade
digital, submetendo dados para a validação
var client = new RestClient("https://159.65.236.70/Identity/AddIdentity");
client.Timeout = -1;
var request = new RestRequest(Method.POST);
request.AddHeader("Authorization", "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJNYXJjZWxvQ2FyZG9zbyIsImp0aSI6Ijh
lYjE3YjQ2LTEwZWEtNDE1Ny04NDU3LTZlMTY0M2ZmMmMxMyIsIk1lbWJlcnNoaXBJZCI6IjQ0MDAzMyI
sImV4cCI6MTU2MjM5NjUyNCwiaXNzIjoiRGlnaXRhbElkZW50aXR5IiwiYXVkIjoiRGlnaXRhbCBJZGV
udGl0eSBiZWFyZXIifQ.-DPOv4_7xhuSTtBog5k0w-hbme4dZPFF_I0dHauFTpY  ");
request.AddHeader("Content-Type", "multipart/form-data; boundary=----------------
-----------541420750257050009794553");
request.AddParameter("email", "marcelosilva@sou.faccat.br");
request.AddParameter("Name", "Marcelo Cardoso");
request.AddParameter("ID", "3097228336");
request.AddParameter("CPF", "01460425065");
request.AddParameter("Address", "Joao Pessoa 338 Centro");
IRestResponse response = client.Execute(request);
Console.WriteLine(response.Content);
```

Anexo XII - Códigos utilizados pelo front-end para consumir a API

```
//código utilizado pelo Front-end para realizar a votação sobre os dados
submetidos pelo usuário.
var client = new RestClient("https://159.65.236.70/Identity/AddVote");
client.Timeout = -1;
var request = new RestRequest(Method.POST);
request.AddHeader("Authorization", "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJNYXJjZWxvQ2FyZG9zbyIsImp0aSI6ImI
4YjgwMWZhLTE0OGMtNGJlZS04ZmNlLTVjZTFiMGU0YjRkZiIsIk1lbWJlclZhbGlkYXRvcklkIjoiMjY
4NTAxIiwiZXhwIjoxNTYyMzkzOTU1LCJpc3MiOiJEaWdpdGFsSWRlbnRpdHkiLCJhdWQiOiJEaWdpdGF
sIElkZW50aXR5IGJlYXJlciJ9.9ZQ92WXC1kwQv_vV668sbudVOu6bkU8SHb43TtpKfOE  ");
request.AddHeader("Content-Type", "multipart/form-data; boundary=----------
----------9088828898306990483600 63");
request.AddParameter("email", "desenv8@idata.com.br");
request.AddParameter("user_email", "marcelosilva@sou.faccat.br");
request.AddParameter("vote", "true");
IRestResponse response = client.Execute(request);
Console.WriteLine(response.Content);
//o cliente front-end informa apenas os emails e a api sabe quais as chaves de
uso da rede privada dos usuários, pois não se consegue vincular os dados dos
documentos do usuário com mais de um email.

//código utilizado para editar parâmetro base do sistema, quantidade de
validações para gerar uma identidade digital para um usuário
var client = new RestClient("https://159.65.236.70/Parameters/AddParameter");
client.Timeout = -1;
var request = new RestRequest(Method.POST);
request.AddHeader("Authorization", "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJNYXJjZWxvQ2FyZG9zbyIsImp0aSI6ImU
1OTFiMGZlLTg4NmMtNGVlMC1iNjI5LWRiYzFhYWVjMWRhYyIsIk1lbWJlckFkbWluSWQiOiIzMzQ1OTA
iLCJleHAiOjE1NjIwNDQ0MDMsImlzcyI6IkRpZ2l0YWxJZGVudGl0eSIsImF1ZCI6IkRpZ2l0YWwgSWR
lbnRpdHkgYmVhcmVyIn0.P5V_ix83gDZ6L4usLaQEm-_bSZxQIWYCko_LPbu_plw  ");
request.AddHeader("Content-Type", "multipart/form-data; boundary=----------
----------5799939128714542283 13888");
request.AddParameter("countVotes", "4");
IRestResponse response = client.Execute(request);
Console.WriteLine(response.Content);
```