UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

AUGUSTO ARAUJO BORGES

Aplicações descentralizadas baseadas em blockchain: estudo e implementação

Goiânia
2023

Processo: 23070.047922/2023-57 Documento: 3988731

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO (TECA) PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES

E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo das Teses e Dissertações disponibilizado na BDTD/UFG é de responsabilidade exclusiva do autor. Ao encaminhar o produto final, o autor(a) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do material bibliográfico

[ ] Dissertação [ ] Tese [X] Outro*: TCCG

*No caso de mestrado/doutorado profissional, indique o formato do Trabalho de Conclusão de Curso, permitido no documento de área, correspondente ao programa de pós-graduação, orientado pela legislação vigente da CAPES.

Exemplos: Estudo de caso ou Revisão sistemática ou outros formatos.

2. Nome completo do autor

Augusto Araujo Borges

3. Título do trabalho

Aplicações descentralizadas baseadas em blockchain: estudo e implementação

4. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador)

Concorda com a liberação total do documento [X] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante:
a) consulta ao(à) autor(a) e ao(à) orientador(a);
b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo da tese ou dissertação.
O documento não será disponibilizado durante o período de embargo.
Casos de embargo:
- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro;
- Publicação da dissertação/tese em livro.

Obs. Este termo deverá ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Sérgio Teixeira De Carvalho**, **Professor do Magistério Superior**, em 24/08/2023, às 12:31, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Augusto Araújo Borges**, **Discente**, em 24/08/2023, às 23:34, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3988731** e o código CRC **6145F6BB**.

Referência: Processo nº 23070.047922/2023-57 SEI nº 3988731

AUGUSTO ARAUJO BORGES

Aplicações descentralizadas baseadas em blockchain: estudo e implementação

Trabalho de Conclusão apresentado à Coordenação do Curso de Sistemas de Informação do Instituto de Informática da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Área de concentração: Sistemas de Informação.

Orientador: Prof. Sérgio Teixeira de Carvalho

Goiânia
2023

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Borges, Augusto Araujo
Aplicações descentralizadas baseadas em blockchain [manuscrito] : estudo e implementação / Augusto Araujo Borges. - 2023.
XLV, 45 f.: il.

Orientador: Prof. Dr. Sérgio Teixeira de Carvalho.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Goiás, Instituto de Informática (INF), Sistemas de Informação, Goiânia, 2023.
Bibliografia. Apêndice.
Inclui lista de figuras.

1. Blockchain. 2. Ethereum. 3. dApps. 4. aplicações descentralizadas. I. Carvalho, Sérgio Teixeira de, orient. II. Título.

CDU 004

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

ATA DE DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO

Aos 24 dias do mês de agosto do ano de 2023 iniciou-se a sessão pública de defesa do Trabalho de Conclusão de Curso (TCC) intitulado “Aplicações descentralizadas baseadas em blockchain: estudo e implementação”, de autoria de Augusto Araujo Borges, do curso de Bacharelado em Sistemas de Informação do Instituto de Informática da UFG. Os trabalhos foram instalados pelo Orientador Professor Doutor Sergio Teixeira de Carvalho (INF/UFG) com a participação dos demais membros da Banca Examinadora: Professor Doutor Valdemar Vicente Graciano Neto e Mestra Gislainy Crisostomo Velasco. Após a apresentação, a banca examinadora realizou a arguição do estudante. Posteriormente, de forma reservada, a Banca Examinadora atribuiu a nota final de 9,0, tendo sido o TCC considerado aprovado.

Proclamados os resultados, os trabalhos foram encerrados e, para constar, lavrou-se a presente ata que segue assinada pelos Membros da Banca Examinadora.

Documento assinado eletronicamente por **Gislainy Crisostomo Velasco**, **Usuário Externo**, em 24/08/2023, às 12:22, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Sérgio Teixeira De Carvalho**, **Professor do Magistério Superior**, em 24/08/2023, às 12:22, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Valdemar Vicente Graciano Neto**, **Professor do Magistério Superior**, em 24/08/2023, às 12:22, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3988732** e o código CRC **2141A506**.

Resumo

Borges, Augusto Araujo. **Aplicações descentralizadas baseadas em blockchain: estudo e implementação**. Goiânia, 2023. 45p. Relatório de Graduação. Instituto de Informática, Universidade Federal de Goiás.

Desde o seu surgimento, a internet se espalhou por praticamente o mundo inteiro e impactou diretamente na dinâmica da sociedade contemporânea. À medida que a mesma evoluiu, observou-se o surgimento de grandes conglomerados corporativos que acumularam grande poder e influência. De forma contrária a este cenário, um novo paradigma emergiu através da descentralização e se consolidou como Web 3.

Baseando-se em blockchain, contratos inteligentes e redes descentralizadas, a Web 3 proporciona uma nova dinâmica na internet e neste contexto surgem os dApps, aplicações descentralizadas.

Este trabalho tem como objetivo explorar a base tecnológica, a arquitetura e outros elementos do universo dos dApps. Para isso, foi implementada uma aplicação web descentralizada utilizando uma plataforma pública de blockchain baseada na EVM (Ethereum), sistemas de armazenamento e de mensagens descentralizados.

A aplicação está inserida no contexto de um exame de pupilometria e diagnóstico de glaucoma, e tem como objetivos o armazenamento dos dados deste exame, controle da concessão de permissão de acesso sobre os dados por parte de um paciente, e o acesso permissionado por parte de um profissional de saúde. O desenvolvimento da aplicação foi realizado buscando atender aos princípios do ecossistema da Web 3 e entender suas vantagens, desvantagens, possibilidades e limitações.

Palavras–chave

Blockchain, Ethereum, dApps, aplicações descentralizadas

Abstract

Borges, Augusto Araujo. **Descentralized applications back-end**. Goiânia, 2023. 45p. Relatório de Graduação. Instituto de Informática, Universidade Federal de Goiás.

Since its inception, the internet has spread virtually all over the world and has had a direct impact on the dynamics of contemporary society. As it evolved, the emergence of large corporate conglomerates that accumulated significant power and influence was observed. Contrary to this scenario, a new paradigm emerged through decentralization and solidified itself as Web 3.

Based on blockchain, smart contracts, and decentralized networks, Web 3 introduces a new dynamic to the internet, giving rise to decentralized applications or dApps.

This work aims to explore the technological foundation, architecture, and other elements of the dApps universe. To achieve this, a decentralized web application was implemented using a public blockchain platform based on the EVM (Ethereum), decentralized storage and messaging systems

The application is situated within the context of pupillometry examination and glaucoma diagnosis. Its objectives comprehend the storage of data from these examinations, the control of access permissions to the data by patients, and authorized access by healthcare professionals. The development of the application was carried out with the intention of adhering to the principles of the Web 3 ecosystem and comprehending its advantages, disadvantages, possibilities, and limitations.

Keywords

Blockchain, Ethereum, dApps, Descentralized applications

Augusto Araujo Borges

Graduando em Sistemas de Informação na UFG - Universidade Federal de Goiás.

Dedico este trabalho a toda a minha família, que sempre esteve ao meu lado.

Agradecimentos

Agradeço à minha familia por todo o apoio e compreensão. Agradeço aos colegas de faculdade pelo companheirismo na jornada da graduação. Agradeço ao meu orientador Sérgio Teixeira de Carvalho pela orientação e persistência ao longo dessa etapa.

Tudo o que você precisa saber por enquanto é que o universo é muito mais complicado do que você pode imaginar.

Douglas Adams,
Praticamente inofensiva (1992).

Sumário

Lista de Figuras

CAPÍTULO 1

Introdução

1.1 Contexto

Alternativa ao estado atual da arquitetura predominante no desenvolvimento web, a descentralização da web já é uma realidade em aplicações e em alguns contextos na internet. Uma forma um pouco diferente de como a internet atualmente se encontra estabelecida, a web descentralizada tem se desenvolvido e está se consolidando cada vez mais no ecossistema conhecido por Web 3.0.

Até chegar no seu estado atual de desenvolvimento, a internet passou por diferentes momentos desde o seu surgimento. O princípio de tudo foi uma rede de poucas máquinas conectadas entre si na década de 1970 que levava o nome de ARPANET. Somente na década de 1990 que a internet como conhecemos hoje começou a se popularizar devido ao surgimento da World Wide Web. A Web foi criada no CERN (Centro Europeu para Física Nuclear), onde Tim Berners-Lee e um grupo de pessoas desenvolveram as primeiras versões do HTML, HTTP, um servidor Web e um navegador (*browser*) que são os principais componentes para um usuário realizar alguma interação na internet [10].

A expansão e a popularização da internet marcam o início da primeira fase da evolução da World Wide Web, a Web 1.0. Dadas as limitações da época, a Web 1.0 era marcada por uma experiência de usuário mais passiva onde o conteúdo era estático, as conexões eram mais limitadas e o acesso se dava apenas por computadores. Esse estágio ficou caracterizado por ser *read-only* (somente leitura) pelo fato dos usuários conseguirem buscar informações, contudo sem a possibilidade de se expressar [17].

O próximo estágio na evolução da web ficou marcado como a web *read-write* (leitura e escrita). A Web 2.0 pode ser entendida como uma evolução das páginas estáticas da Web 1.0, pois já era possível a criação de um conteúdo dinâmico e interativo. Como consequência do desenvolvimento tecnológico, as conexões já não são mais tão limitadas e o acesso não era mais feito somente por um computador do tipo *desktop* [18].

Diante de menos limitações e novos recursos, plataformas que proporcionam novas experiências foram surgindo e se tornaram gigantes como Facebook, YouTube, Twitter, etc. A Web 2.0 permitiu novas formas de conexão e comunicação, criação e

compartilhamento de conteúdo. Além das novas plataformas sociais, também emergiram novos serviços, como a computação em nuvem e modelos de software como serviço. Grandes corporações como Amazon, Google e Microsoft cresceram vertiginosamente proporcionando recursos computacionais para o desenvolvimento de novas plataformas e aplicações na internet. E, apesar das vantagens de custo e manutenção em relação a ter o próprio servidor, isso gerou uma grande centralização na internet. O impacto disso pode ser percebido quando um desses serviços falha e como consequência, inúmeras plataformas e serviços online param de funcionar temporariamente [16].

Em uma nova fase de evolução da internet, a emergente Web 3.0 é um paradigma que busca mudanças no meio digital fomentando maior descentralização, privacidade de dados e maior controle para os seus usuários [18]. O termo Web 3.0 se refere ao uso da tecnologia de blockchain, contratos inteligentes e redes descentralizadas como base para a interação na web [17].

1.2 Motivação e Objetivos

Com base no ecossistema da Web 3.0, podem ser feitas aplicações descentralizadas (DApps) baseadas em blockchain. Assim, é possível desenvolver aplicações com menor dependência em relação a recursos de computação em nuvem de grandes empresas, reduzindo o risco de censura ou bloqueios, tempo de inatividade (*downtime*) de possíveis falhas desses serviços, e funcionando de maneira confiável e transparente.

O objeto de interesse deste trabalho é o contexto da Web 3.0 e a arquitetura de aplicações descentralizadas. Nesse sentido, o objetivo geral deste trabalho de conclusão de curso é realizar um estudo exploratório para se compreender os principais elementos arquiteturais de aplicações descentralizadas baseadas em blockchain. Para isso, uma aplicação web descentralizada é implementada a fim de verificar a viabilidade, possibilidades e limitações da tecnologia.

A aplicação está inserida no contexto de um exame de pupilometria e diagnóstico de glaucoma, e tem como objetivos o armazenamento dos dados deste exame, controle da concessão de permissão de acesso sobre os dados por parte de um paciente, e o acesso permissionado por parte de um profissional de saúde.

1.3 Organização

Este trabalho está organizado em quatro capítulos, além deste introdutório. O Capítulo 2 apresenta o referencial teórico sobre Web 3.0, blockchain e descentralização. O Capítulo 3 apresenta o cenário da aplicação desenvolvida, sua arquitetura, casos de uso

e contrato inteligente implementado. Por fim, o Capítulo 4 apresenta considerações sobre a arquitetura implementada, desafios, limitações e possibilidades de melhorias.

CAPÍTULO 2

Referencial teórico

2.1 Blockchain

O conceito de blockchain tem sua origem associada a um artigo publicado sob o pseudônimo de Satoshi Nakamoto em 2008 [15], que tinha como propósito um sistema de transações financeiras eletrônicas que não precisassem um intermediário como, por exemplo, bancos. No modelo proposto, posteriormente implementado e conhecido como Bitcoin, a própria forma de funcionamento do sistema baseadas em provas criptográficas seria responsável pela confiabilidade das transações.

Transações realizadas entre duas partes são geralmente conduzidos de forma centralizada, sendo necessária uma terceira parte de confiança, como instituições financeiras ou um banco. A blockchain é uma tecnologia que emergiu como uma alternativa a isso, trazendo a possibilidade de uma troca de recursos entre duas partes de maneira confiável e sem essa necessidade dessa terceira parte regulatória [12].

A blockchain é um livro-razão compartilhado *(distributed ledger)*, distribuído em uma rede *peer-to-peer* onde as transações propostas por um cliente são recebidas por servidores (definidos como nós ou *nodes*), validadas pelos mesmos através de um protocolo de consenso e armazenadas de maneira imutável em uma cadeia de blocos e replicada em cada nó da rede [8].

Para compreender melhor esse conceito de blocos em cadeia, primeiramente, é preciso entender como são formados os blocos que a compõem e, principalmente, o conceito de criptografia. Hashes e funções hash são elementos criptográficos incorporados na blockchain que são imprescindíveis para trazer a confiabilidade e imutabilidade dos dados inseridos na blockchain.

Funções *hash* são algoritmos matemáticos que transformam um tipo de dado em um número de tamanho fixo, independentemente de qual seja o dado de entrada (*input*). Hash, por sua vez, é o próprio resultado (*output*). Algumas propriedades das funções hash são indispensáveis para o tecnologia que envolve a blockchain. Primeiramente, essas funções são determinísticas, ou seja, o seu resultado dado uma mesma entrada sempre será idêntico. Concomitantemente, as funções hash também são pseudoaleatórias, o que

significa que o valor de um hash gerado muda de forma imprevisível. Ainda são também unidirecionais, de forma que não é possível fazer o processo inverso, buscando obter uma entrada (input) apenas com a saída (output). Além disso, são resistentes à colisão, que se refere à dificuldade de se obter uma mesma saída com diferentes entradas de dados [11].

Valendo-se de todas essas características, a cadeia de blocos ou *ledger* é formada com cada novo bloco ligado criptograficamente com o bloco anterior, formando uma estrutura de dados similar à de uma lista encadeada (Figura 2.1). Esta estrutura tem como resultado a confiabilidade na imutabiliade da informação, dado que qualquer alteração em uma transação de um bloco invalidará o valor hash do bloco seguinte e por consequência todo o restante da estrutura.

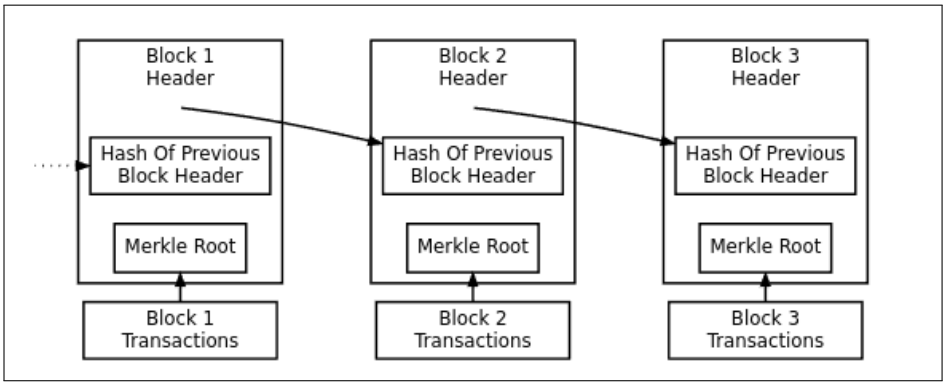

Figura 2.1: *Simplificação visual da estrutura geral de uma blockchain [3]*

A estrutura é distribuída e replicada em cada nó de uma rede peer-to-peer, fazendo com que cada nó mantenha a sua própria cópia dos blocos com as transações, impedindo que qualquer fraude possa acontecer sem a necessidade de uma autoridade central. Para que qualquer alteração maliciosa possa ser feita, seria preciso adulterar, não apenas um ledger de um nó da rede, mas sim 51% de todos os nós da rede [11].

2.1.1 Evolução da tecnologia de blockchain

Como mencionado anteriormente, a primeira blockchain implementada com sucesso foi a Bitcoin, que se encontra em funcionamento até os dias atuais. Essa primeira geração da tecnologia, denominada de *Blockchain 1.0*, foi a origem das criptomoedas [19]. Apesar de ser um sistemas de transações seguro, sem uma autoridade central e ao mesmo tempo anônima em relação aos seus usuários, a tecnologia também possui problemas. Alguns deles se referem à baixa capacidade de processamento das transações e ao protocolo de consenso utilizado *(Proof of Work)*, que precisa de poder computacional para resolver problemas matemáticos complexos, gerando um alto consumo de energia.

Os problemas citados da primeira geração e possibilidades limitadas a apenas moedas digitais foram elementos que estimularam o desenvolvimento da Ethereum, uma

blockchain baseada em contratos inteligentes *(smart contracts)*. A ideia dos contratos inteligentes é o conceito chave que diferencia a geração da *Blockchain 2.0* da anterior, pois foi através dela que se permitiu expandir a tecnologia para além de transações com criptomoedas [14].

Apesar da expansão da tecnologia e o seu desenvolvimento, a escalabilidade ainda é um problema, bem como a sustentabilidade. Nesse sentido novas formas de consenso foram desenvolvidas, como, por exemplo, o *Proof of Stake*, que proporcionou um processamento de transações em maior quantidade sem a necessidade do alto poder computacional exigido pelo *Proof of Work*. A consequência disso é o leque de possibilidades que se abre para o desenvolvimento de dApps em diversas áreas em que as características da blockchain agregam valor [14]. A Blockchain 3.0 é a geração que combina esse cenário com diversos tipos de aplicações descentralizadas e os aspectos de melhor escalabilidade e sustentabilidade [19].

2.1.2 Consenso

A blockchain é composta por blocos encadeados e nestes blocos estão registradas as transações enviadas por clientes. No entanto, todas as transações, antes de serem armazenadas, passam por uma validação para garantir a manutenção da consistência dos dados, sendo que um nó da rede é escolhido para publicar o novo bloco. Quando um novo bloco é publicado, o nó escolhido é recompensado com um valor na criptomoeda da rede. Isso cria uma competição que tem como consequência a maioria dos nós trabalhando em favor da rede, impedindo que a rede seja manipulada [17, 14].

Este processo de decisão pode variar e existem diferentes mecanismos de consenso como o *Proof of Work (PoW)*, *Proof of Stake (PoS)*, *Proof of Elapsed Time (PoET)*, etc. A rede Bitcoin utiliza o método do *Proof of Work*, assim como a rede Ethereum também utilizou até 15 de Setembro de 2022, quando realizou uma atualização para a utilização do *Proof of Stake*. Essa mudança reduziu o consumo de energia da rede em aproximandamente 99.95%, seguindo um caminho direcionado para maior sustentabilidade condizente com a geração 3.0 das blockchains [7].

2.1.3 Blockchains permissionadas e não permissionadas

As redes de blockchain podem ser categorizadas com base no seu modelo de permissionamento, que determina quem publicará os blocos na cadeia. Se um bloco pode ser publicado por qualquer um, a rede é considerada não permissionada ou pública *(permissionless)*. Se existe um controle sobre os nós que podem participar da rede e publicar novos blocos, a rede é considerada privada ou permissionada *(permissioned)* [27].

As blockchains públicas são redes completamente descentralizadas, em que não é necessária permissão alguma para a participação na rede, de forma que qualquer entidade pode se tornar um usuário, minerador ou desenvolvedor. Em geral essas redes são *open source*, disponíveis gratuitamente e não há controle ou conhecimento sobre os participantes da rede. Diante disso, a rede poderia ter problemas com algum usuário que poderia tentar publicar blocos de maneira maliciosa. Para prevenir isso, as redes públicas utilizam de um sistema de consenso que busca impedir tentativas de subversão do sistema. Um exemplo de blockchain pública é a rede do Bitcoin ou a própria Ethereum.

As blockchain privadas são redes que requerem a permissão de alguma autoridade para a participação. Dessa forma, pode haver controle e conhecimento de todos os participantes, incluindo a possibilidade de acesso de leitura ou de envio de transações. Essas redes podem ou não ser *open source*, e geralmente utilizam sistemas de consenso diferentes das redes públicas, já que não há a mesma procupação com usuários mal-intencionados. Exemplos de blockchain privadas são as plataformas Hyperledger e R3 Corda [27, 14].

2.1.4 Contratos inteligentes

Smart contract ou contrato inteligente é um termo de 1994 definido por Nick Szabo como "um protocolo de transação informatizado que executa os termos de um contrato". Segundo Szabo, o objetivo geral de um contrato inteligente é satisfazer condições contratuais comuns, minimizar exceções maliciosas ou acidentais e minimizar a necessidade de confiança entre os intermediários [22].

Os contratos inteligentes podem ser entendidos como uma coleção de código e dados (ou funções e estado) que são implantados na blockchain através de transações, e uma vez que isso ocorre eles podem ser executados dentro da rede da blockchain de maneira a facilitar, executar e impor os termos de um acordo entre as partes [27]. Dada a natureza da blockchain, após a implantação do contrato não há mais possibilidade de alteração, sendo que para executá-lo basta que uma transação seja feita no endereço deste contrato. Essa transação é executada pelos nós validadores da rede e, após consenso, o estado será atualizado de acordo com o estabelecido pelo contrato [1].

Para um contrato implantado em uma blockchain pública, o usuário que solicita uma transação precisa pagar pelo custo da execução do código. No caso da rede Ethereum, por exemplo, é requerido do usuário uma taxa de *gás*. O *gás* é uma unidade de medida utilizada na rede Ethereum referente ao esforço computacional requerido para executar operações, o qual é pago utilizando a própria moeda da rede, o *ether*. A existência de uma taxa em uma rede pública é uma segurança para previnir *spamming* de usuários, loops acidentais ou maliciosos, e até mesmo má utilização de recursos [5].

2.2 Aplicações descentralizadas *(dApps)*

No contexto da Web 3.0, a descentralização se refere ao movimento de mudança de servidores centralizados, data centers e intermediários para redes distribuídas, protocolos peer-to-peer e enfoque maior no usuário. A descentralização e as bases da tecnologia da blockchain reduzem o risco de censura, tornam extremamente difícil causar interrupções de serviço, proporcionando alta disponibilidade e redundância por definição [18].

Com o desenvolvimento das blockchains e o surgimento dos smart contracts emergiu um cenário com diversas aplicações descentralizadas. Essas aplicações podem ter *clients*, aplicativos e até mesmo servidores próprios, assim como as aplicações tradicionais, no entanto, seus dados e operações principais podem ser armazenados em contratos em uma blockchain.

A adoção de dApps ocorre gradualmente em vários segmentos da indústria, como jogos, finanças, saúde, gerenciamento da cadeia de suprimentos, entre outros [25]. A exemplo, o Walmart passou a fazer o rastreio da origem de mais de 25 produtos utilizando o Hyperledger Fabric, que trouxe mais facilidade, agilidade e confiança no processo de busca pela procedência dos alimentos [23]. No segmento dos jogos, o Axie Infinity é um jogo que iniciou o seu desenvolvimento em 2018 com base na Ethereum e ficou mundialmente conhecido pelo seu formato *play to earn* [2].

2.2.1 Arquitetura das *dApps*

De maneira geral, uma aplicação web cliente-servidor normalmente possui em sua arquitetura três elementos principais: o *front-end* (que geralmente utiliza HTML, CSS e Javascript), o *back-end* (uma API Web que pode ser feita em diversas linguagens como PHP, Python, Java, NodeJs, etc) e o banco de dados (MySQL, PostgreSQL, Mongo, etc) (Figura 2.2). Nessa estrutura, o front-end é o responsável pela visualização da informação e lógica de interação com o usuário, o back-end mantém as regras de negócio, e o banco de dados é o responsável pelo armazenamento de dados.

Normalmente, estes elementos são hospedados em algum servidor centralizado como a AWS da Amazon ou GCP da Google, por exemplo. Observando esse processo de maneira mais simplificada, o usuário pode acessar a aplicação front-end por meio do browser, em seguida o front-end se comunica com o back-end, que por sua vez se comunica com o banco de dados.

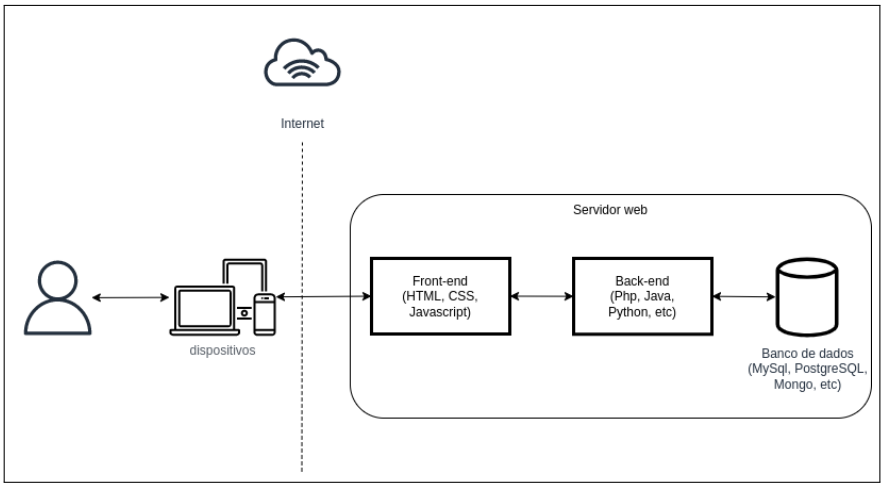

Figura 2.2: *Arquitetura básica de uma aplicação web cliente-servidor (adaptado de The evolution of web, 2023[17])*

Uma aplicação descentralizada, assim como uma aplicação web cliente-servidor, também possui um front-end em sua estrutura para a interação com o usuário. Porém, a aplicação descentralizada não possui necessariamente uma API e nem mesmo um banco de dados centralizado, mas sim uma comunicação direta com os contratos inteligentes sem a necessidade de um intermediário (Figura 2.3).

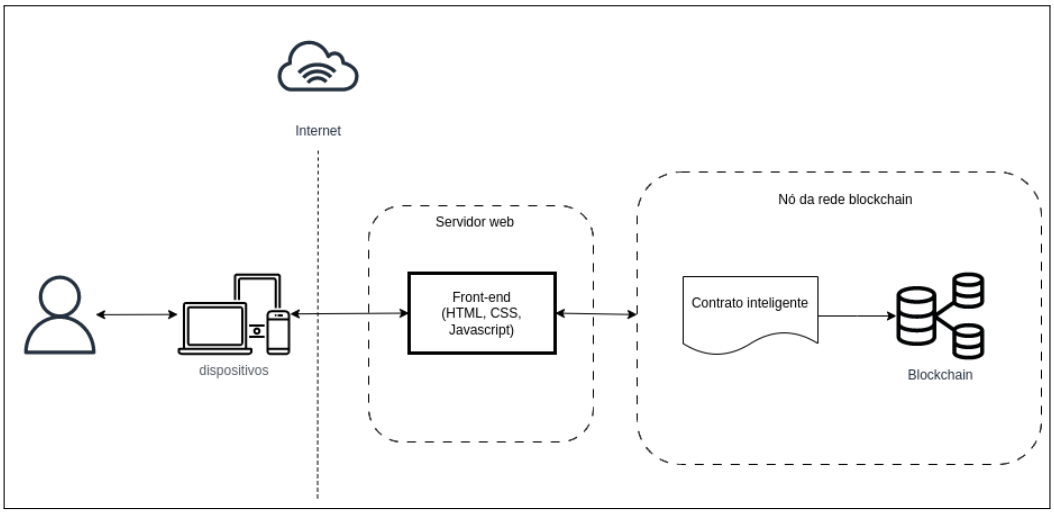

Figura 2.3: *Arquitetura básica de uma aplicação web descentralizada (adaptado de The evolution of web, 2023[17])*

Quando um usuário solicita uma transação pelo front-end, o contrato inteligente recebe essa transação, que é processada e acrescentada em conjunto com outras transações a um bloco, o qual, por sua vez, é armazenado na blockchain. Ao fim deste processo o front-end recebe a resposta de confirmação e atualiza a interface [17]. Essa comunicação com os contratos inteligentes é feita por meio do protocolo JSON-RPC, um protocolo de chamada de procedimento remoto codificado em JSON agnóstico de transporte, possibilitando que sejam utilizados *sockets* ou HTTP [6].

Assim como em uma aplicação web cliente-servidor, as tecnologias utilizadas no lado do cliente para o desenvolvimento de uma dApp podem ser as mesmas, utilizando HTML, CSS, Javascript, bibliotecas e frameworks, como, por exemplo, React, Vue, Angular, Svelte, entre outros. Porém, as diferenças começam a surgir ao realizar uma interação com o back-end. Em uma aplicação web cliente-servidor os usuários normalmente são identificados por meio de credenciais como e-mail, nome de usuário e senha. Já em uma dApp a interação é feita utilizando-se de uma carteira *(wallet)* [17].

2.2.2 Carteiras (*wallets*)

Uma carteira é a chave para utilizar uma blockchain e existem diversos tipos de carteiras para acessar dados e operar em uma blockchain. Qualquer usuário que queira interagir com uma plataforma de blockchain precisa de uma. As carteiras são aplicações de software que podem ser usadas para verificar o saldo de criptomoedas e interagir com a blockchain realizando transações [21].

Os elementos essenciais são as chaves publicas e privadas que são utilizados para fazer assinaturas digitais. De maneira similar a uma assinatura manual em um papel, a assinatura digital serve para confirmar a identidade do signatário. Quando uma transação é enviada, a chave privada é utilizada para assinar a transação enviada para a blockchain, e a chave pública é utilizada para a validação e verificação dessa assinatura junto aos nós validadores da rede [17].

Existem diversos tipos de carteiras, sendo elas aplicações desktop, móveis, online, ou até mesmo baseadas em hardware. Uma carteira muito popular e comumente utilizada é a Metamask, pois ela funciona tanto como *signer*, para assinar as transações, quanto como *provider*, para realizar a ponte de comunicação com um nó da rede [13].

2.2.3 Armazenamento de dados

Apesar da blockchain ser a plataforma ideal para salvar informações de maneira transparente, segura e confiável, a tecnologia não é adequada para salvar grandes quantidades de dados. Salvar imagens ou vídeos em uma blockchain não é viável, considerando o alto custo da transação e a baixa velocidade. No caso de dados assim, é comum utilizar outros serviços fora da blockchain *(off-chain)* para o armazenamento.

Serviços de armazenamento descentralizado, como o IPFS *(Interplanetary File System)* e o Swarm, são protocolos de sistemas de arquivos descentralizado que trazem a possibilidade de armazenar e acessar arquivos em uma rede *peer-to-peer*, suprindo a necessidade de armazenamento das dApps e mantendo os princípios da descentralização [17].

O conteúdo armazenado através do IPFS é imutável, sendo que um *hash* é criado a partir das informações de um arquivo quando adicionado. A partir desse hash, é construído o *CID*, endereço utilizado para referenciar o arquivo armazenado. Consequentemente, uma mudança no arquivo mudaria o *hash* que, por sua vez, mudaria o CID.

Essa forma de funcionamento pode trazer problemas para conteúdos que precisam ser atualizados regularmente. Para isso, existe um sistema chamado IPNS, que permite criar ponteiros mutáveis para CIDs, denominados *IPNS names*. Os *IPNS names* são como um link que pode ser atualizado.

CAPÍTULO 3

Aplicação

3.1 Cenário da aplicação

A proposta de implementação é apresentar uma aplicação descentralizada no contexto de exames de pupilometria para o diagnóstico de glaucoma. A pupilometria é uma avaliação do comportamento da pupila em que se mede o diâmetro das mesmas ao longo de um período que são gerados estímulos visuais ou luminosos.

Este exame é realizado com um pupilômetro que realiza a gravação da pupila por meio de câmeras infravermelho, gerando um *output* com vídeos e imagens que são posteriormente utilizados para o diagnóstico [20].

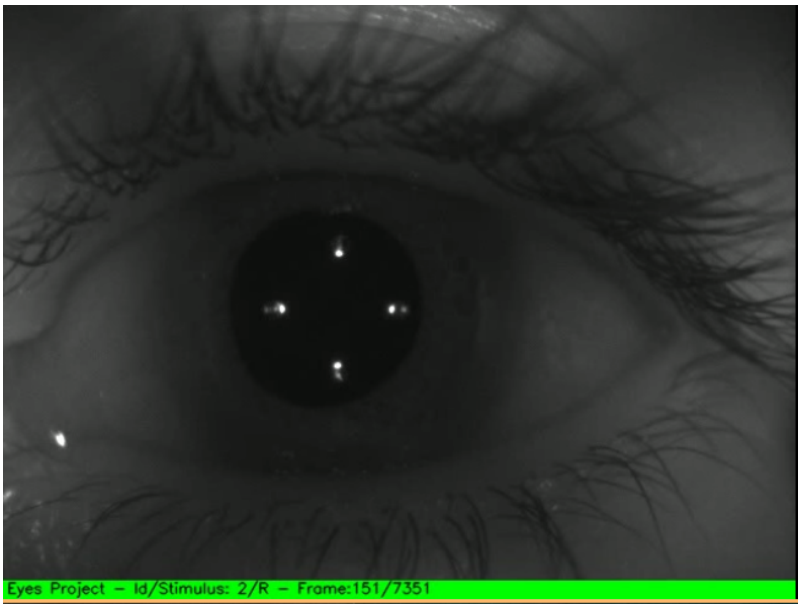

Figura 3.1: *Frame de um vídeo do exame de pupilometria*

O cenário da aplicação começa pela captura do exame com o paciente utilizando o pupilômetro. A implementação busca fazer o armazenamento, gerenciar o acesso aos dados e a concessão de acesso aos dados através de uma aplicação web e contratos inteligentes.

3.2 Arquitetura

A aplicação é baseada na rede Ethereum, que utiliza a EVM *(Ethereum Virtual Machine)* como o ambiente para executar os contratos inteligentes. Importante ressaltar que, embora a implementação utilize como base a rede Ethereum, os contratos podem ser utilizados em outras redes de blockchain que se baseiam na EVM.

A arquitetura desenvolvida se assemelha à arquitetura básica descrita no Capítulo 2, em que um front-end se comunica com a rede de blockchain tendo a Metamask como carteira e como *provider*. Além disso, novos elementos foram incorporados para suprir as necessidades de desenvolvimento da aplicação. Como a blockchain não é adequada para salvar dados em maiores quantidades como imagens ou vídeos, foi preciso utilizar a tecnologia descentralizada de armazenamento de dados IPFS.

Os dados de *output* gerados pelo exame de pupilometria precisam ser armazenados no IPFS e as suas referências posteriormente gravadas na blockchain. Uma vez que o armazenamento no IPFS é público, não seria viável armazenar dados particulares de um paciente em um sistema no qual qualquer pessoa poderia acessar. Assim, foi utilizada a API de criptografia nativa do *browser* para que os arquivos sejam criptografados antes do armazenamento.

A criptografia dos dados a serem salvos gera uma nova necessidade que é o compartilhamento das chaves de maneira segura para o acesso aos arquivos. O XMTP *(Extensible Message Transport Protocol)* é um protocolo aberto de mensagens privadas baseado em uma rede descentralizada [26]. As suas mensagens possuem criptografia de ponta-a-ponta *(end-to-end)* e a autenticação é feita com base na carteira de uma blockchain, que no caso da implementação, é a Metamask.

A Figura 3.2 apresenta os elementos que compõem a arquitetura da DApp, e como elas interagem entre si. Os elementos são os seguintes:

- **Browser**: aplicação que interage com o front-end e com a Metamask, e fornece as APIs utilizadas para lidar com os arquivos (FileReader, Blob e Crypto);
- **Front-end**: aplicação responsável pelas interfaces e interações entre o usuário e a blockchain;
- **Metamask** (*crypto wallet* e *gateway*): provê a identidade de um usuário da blockchain, assina as transações e também exerce o papel de *provider*, fazendo a comunicação entre o front-end e o contrato inteligente baseada em JSON-RPC;
- **IPFS**: sistema de arquivos no qual o exame de pupilometria (dados de *output*) é armazenado;
- **XMTP**: responsável por realizar a comunicação de mensagens entre as entidades envolvidas;

- **Contrato inteligente**: artefato que executa as regras de negócio e atualiza o estado da blockchain.

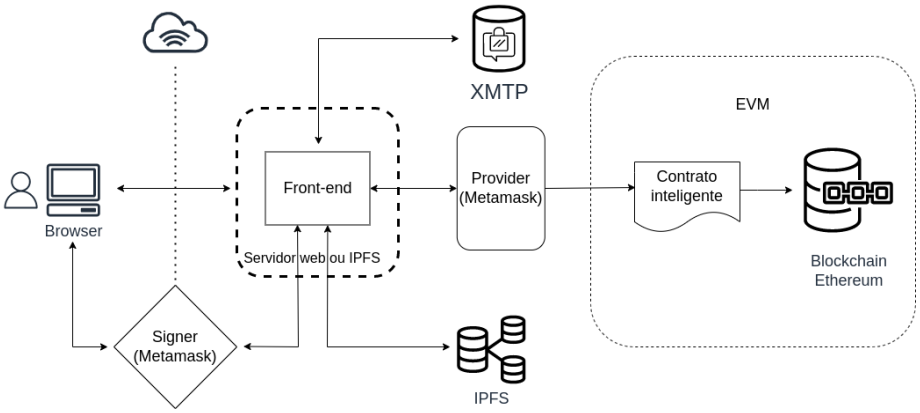

Figura 3.2: *Arquitetura implementada para o dApp*

3.2.1 Ferramentas para o desenvolvimento

A aplicação envolve diversas tecnologias para que todos os elementos dessa arquitetura funcionem de maneira integrada em um ambiente local de desenvolvimento. No front-end foi utilizado o *framework* Vue.js para a construção das interfaces web, em conjunto com algumas bibliotecas para facilitar as interações entre os outros elementos arquiteturais. Entre elas estão a *kubo-rpc-client*[1], *xmtp-js*[2] *e ethers*[3]. A biblioteca *kubo-rpc-client* é uma coleção de códigos que implementa um cliente para fazer a comunicação entre o front-end e o nó do IPFS. O *xmtp-js* é o *SDK* (kit de desenvolvimento de software) utilizado para fazer o envio e o recebimento de mensagens pelo XMTP. *Ethers*, por sua vez, é uma biblioteca que facilita as interações com o ecossistema da blockchain.

Foram utilizados também, o IPFS Desktop, uma aplicação que permite executar um nó IPFS completo, utilizado em ambiente local de desenvolvimento, e o XMTP, que provê o seu próprio ambiente de desenvolvimento online, dispensando a implementação do seu próprio nó.

Para estabelecer a comunicação entre o front-end e a blockchain, fez-se uso da Metamask, que pode ser configurada para rodar em ambiente local de desenvolvimento e gerenciar diferentes carteiras.

Por fim, para o desenvolvimento do contrato inteligente foi utilizado o *Hardhat*, um ambiente de desenvolvimento com ferramentas para simular uma rede Ethereum em

[1] https://github.com/ipfs/js-kubo-rpc-client
[2] https://github.com/xmtp/xmtp-js
[3] https://github.com/ethers-io/ethers.js

ambiente local. Com ele pode-se executar scripts, testes unitários para os contratos, etc. Como a plataforma de blockchain é a Ethereum, foi desenvolvido o contrato em *Solidity*.

3.2.2 Casos de uso e fluxos de funcionamento

No cenário da realização de um exame de pupilometria foram implementados quatro casos de uso com três atores distintos (Figura 3.3):

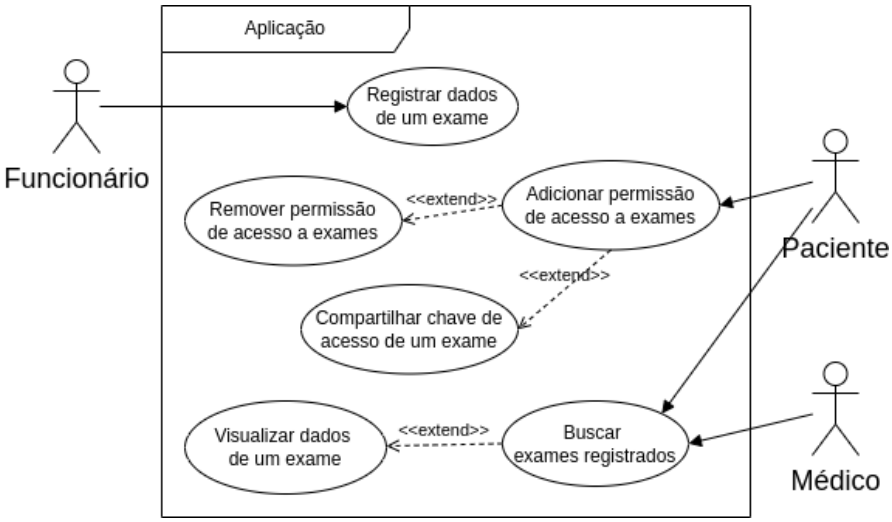

Figura 3.3: *Casos de uso para o dApp*

- Funcionário;
- Paciente;
- Médico.

Entrada de dados

O primeiro caso de uso (Registrar dados de um exame) trata da alimentação dos dados do exame no sistema e da associação deles a um paciente. Esse caso de uso pode ser executado por algum ator que está acompanhando a realização do exame com o paciente ou pode ser integrado de forma automatizada com o pupilômetro. No contexto da implementação, o ator fica representado por um funcionário da clínica que alimentaria o sistema com os dados.

Ao acessar a plataforma representada na Figura 3.4, o ator deve apenas preencher o campo de “Endereço”, acrescentar os arquivos gerados pelo exame e clicar em salvar. O evento do clique gera uma sequência de ações (representada na Figura 3.5) que começa pela geração de uma chave aleatória para a criptografia dos arquivos com a chave. Após a criptografia, os arquivos são enviados para o sistema de armazenamento (IPFS) e são obtidos os CIDs (endereços dos arquivos) como resposta.

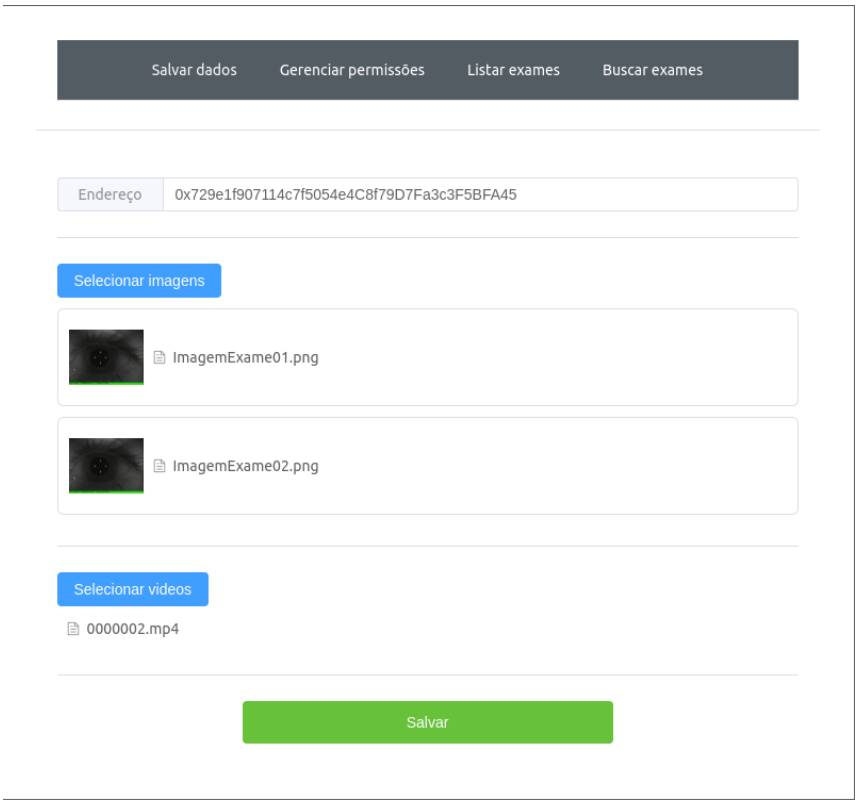

Figura 3.4: *Interface para a funcionalidade de entrada de dados*

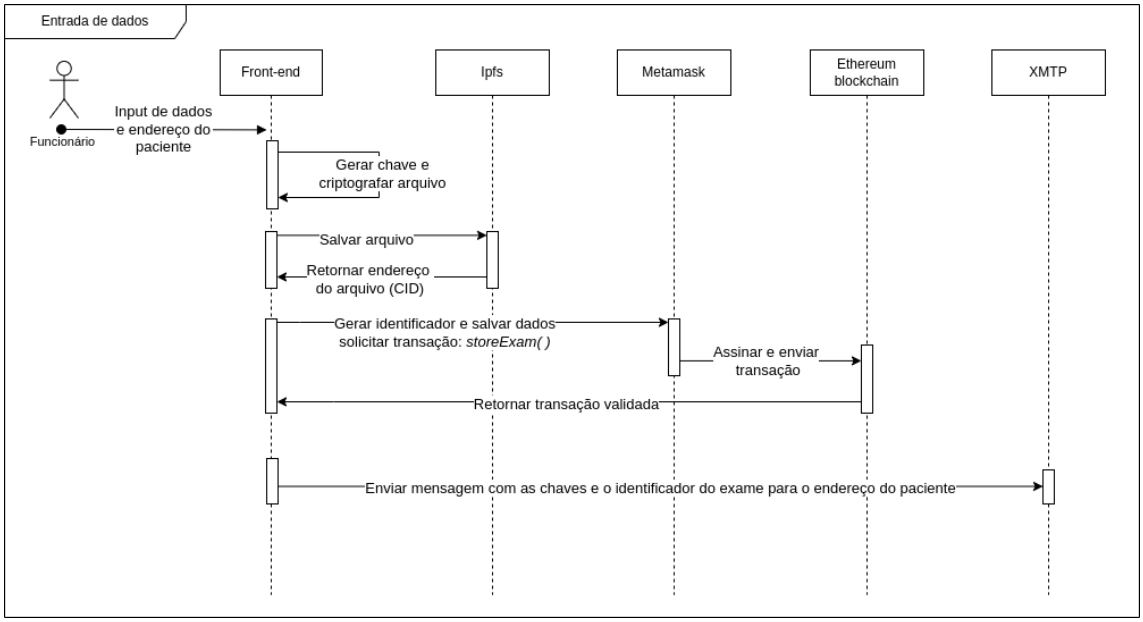

Figura 3.5: *Diagrama de sequência para o caso de uso de entrada de dados.*

Em seguida, é gerado um identificador *(timestamp)* e junto com os CIDs é enviada uma solicitação de transação para a execução da função *storeExam* do contrato inteligente por meio da Metamask. O usuário confirma a transação pela Metamask, que a assina e a envia para a blockchain para ser executada. Após a confirmação da transação validada no front-end, é enviada uma mensagem com a chave e o identificador gerados por meio do XMTP, finalizando o processo de entrada de dados.

Concessão de acesso

O segundo caso é a concessão de acesso que o paciente realiza em relação ao acesso a seus exames. Nesse sentido, o paciente pode adicionar e remover um endereço de uma lista que controla quem pode buscar os seus exames na blockchain. Para um controle mais granular e em consequência da criptografia dos dados, é preciso que o paciente compartilhe as chaves de cada exame para o acesso completo.

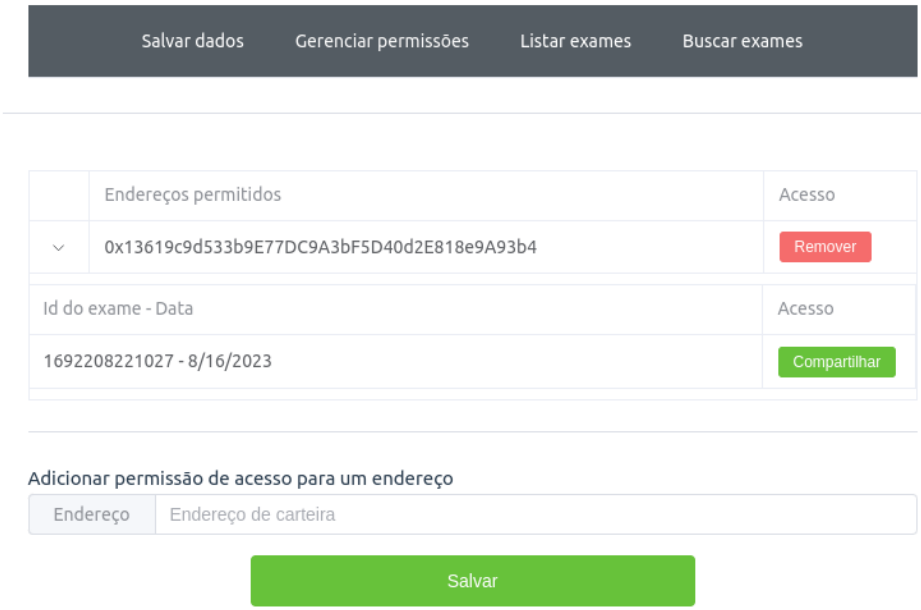

Figura 3.6: *Interface para a concessão de acesso do paciente.*

Na Figura 3.6 o paciente possui um endereço já com permissão, e um exame para o qual ainda não foi compartilhada a chave para descriptografar os arquivos. Para adicionar mais um endereço que pode acessar os seus registros é preciso preencher o campo de endereço e salvar, desencadeando o fluxo descrito na Figura 3.7. Nessa primeira etapa, uma nova solicitação de transação é enviada para a blockchain para a execução da função *addPermission* do contrato inteligente. Quando a transação é validada, o novo endereço é acrescentado à tabela e o usuário precisa, então, em uma segunda ação, compartilhar as chaves de cada exame com este novo endereço. Esse compartilhamento

é feito clicando no botão de compartilhar para cada exame, o qual envia uma mensagem com a chave de acesso pelo XMTP.

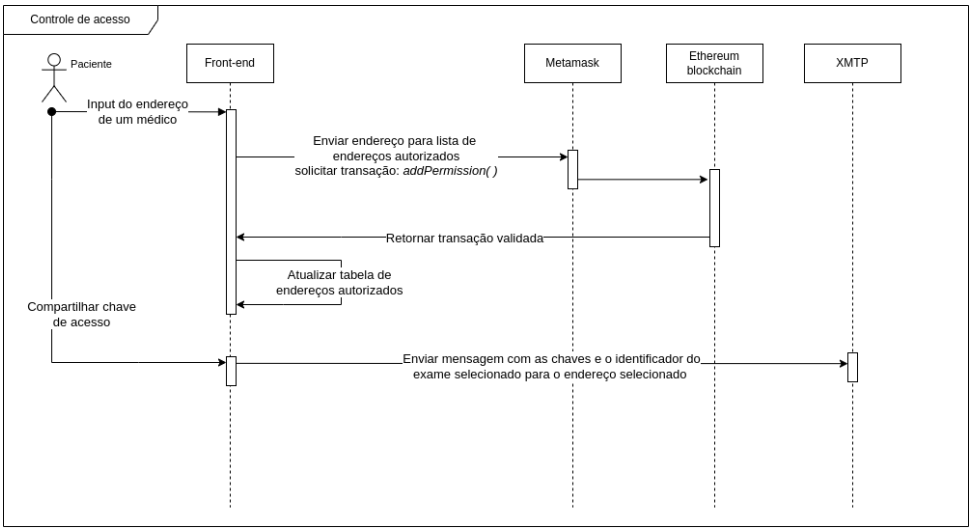

Figura 3.7: *Diagrama de sequência para uma nova permissão concedida.*

Ao remover o acesso, um fluxo semelhante é executado (Figura 3.8), dessa vez removendo o endereço da lista de endereços autorizados através da função *removePermission* do contrato inteligente. Após a remoção, o endereço removido não consegue mais recuperar a lista de exames do paciente.

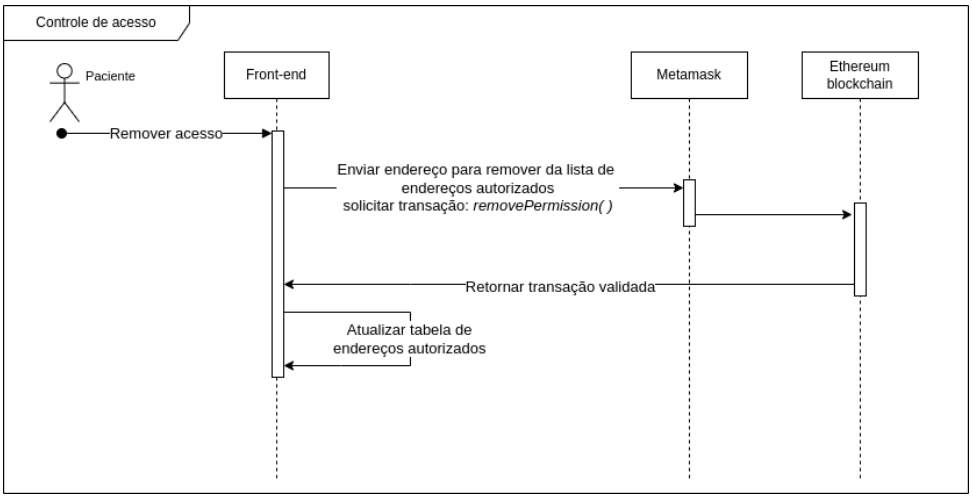

Figura 3.8: *Diagrama de sequência para remoção de autorização.*

Buscar exames

O terceiro caso de uso pode ser exercido tanto por um paciente quanto por um médico que recebeu concessão de acesso (permissão) de um paciente. As diferenças para cada um dos atores está na função do contrato utilizada e a interface. O paciente não

precisa especificar um endereço para fazer a busca, diferentemente do médico, que precisa especificar uma carteira (*wallet*) para só então realizar a busca. Ambos visualizam uma listagem semelhante à apresentada na Figura 3.9.

Figura 3.9: *Interface de listagem de exames registrados para um usuário*

No caso do paciente, a função executada no contrato inteligente para listar os exames é a *getOwnExams*, que retorna os registros de exames da carteira (*wallet*) que realizou a chamada à função. No caso do médico, a função do contrato chamada é a *searchPatientExam*, que possui uma verificação de autorização e retorna os exames do endereço fornecido, se autorizado.

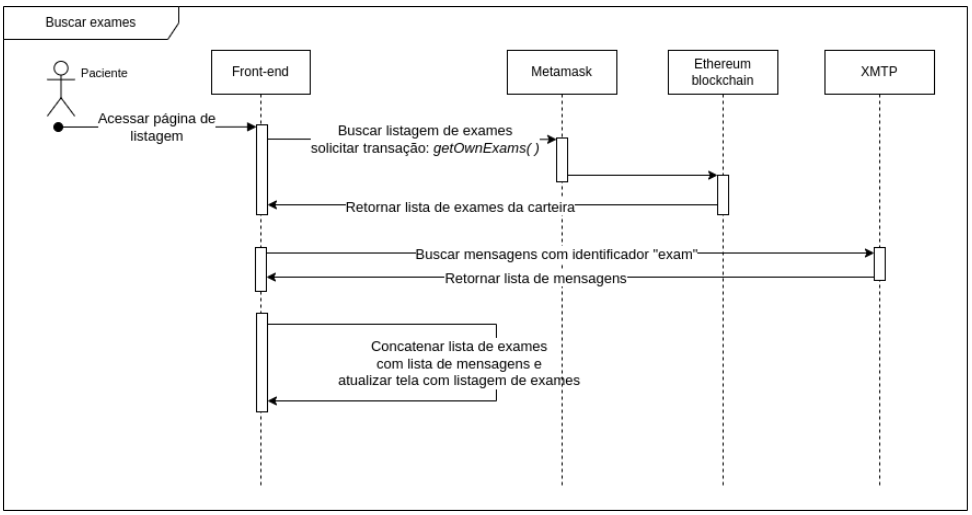

Figura 3.10: *Diagrama de sequência da listagem de exames.*

Visualizar dados do exame

A visualização do exame também é a mesma para pacientes e médicos. Para acessar a visualização, é preciso selecionar um exame a partir da listagem. A listagem já possui as informações buscadas da blockchain (CIDs) e a chave para descriptografar as mensagens (XMTP). Nesse caso, como representado no diagrama da Figura 3.12, é necessário apenas buscar os arquivos do sistema de armazenamento IPFS, descriptografá-los e apresentá-los na interface para download, como apresentado na Figura 3.11.

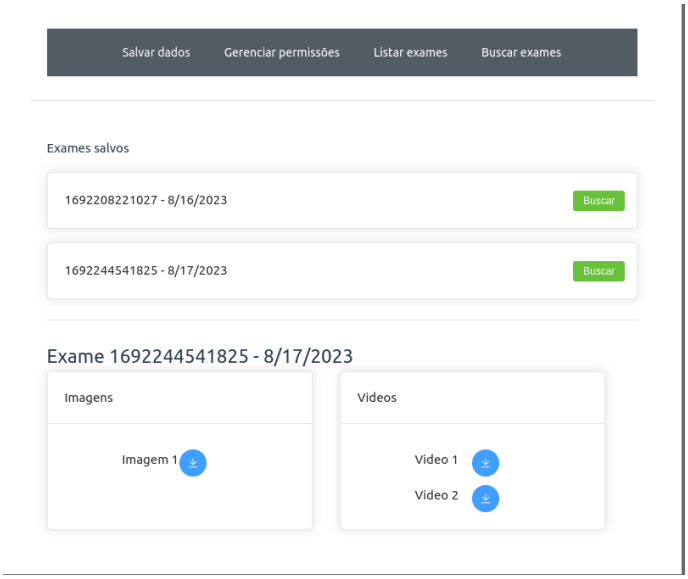

Figura 3.11: *Interface para a visualização do exame*

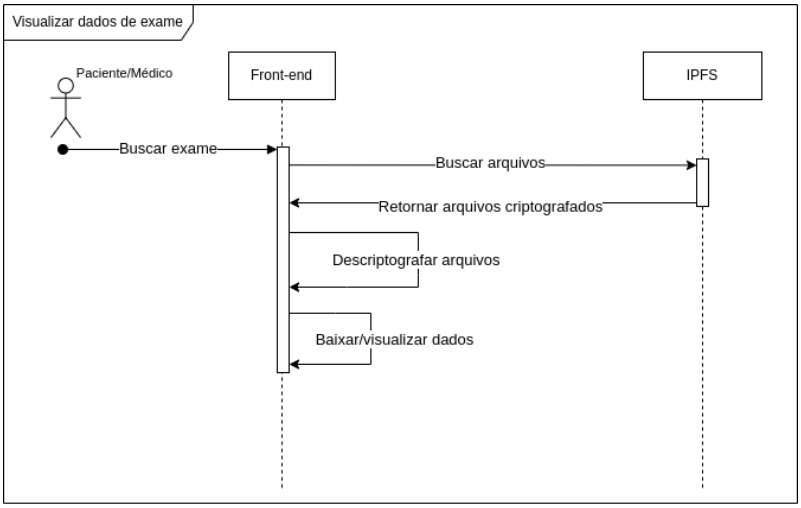

Figura 3.12: *Interface para a visualização do exame*

3.3 Implementação

Os contratos inteligentes contêm a lógica de negócio, constituindo-se no conjunto de instruções que são executados pela EVM (*Ethereum Virtual Machine*). A execução do contrato altera o estado da blockchain. Para um contrato ser implantado na rede é preciso que ele seja enviado por meio de uma transação de criação de contratos para a blockchain. O contrato criado possui um endereço, um saldo, um código predefinido executável e um estado. Uma vez criado, já é possível interagir com o contrato através da realização de novas transações [24].

Para atender os casos de uso explicados na Subseção 3.2.2, foi desenvolvido um contrato com seis funções e duas variáveis para armazenamento de dados no estado da blockchain. Primeiramente, foi declarada uma estrutura que define as informações de exames chamada *ExamResult*, que consiste de *arrays* de *strings* para os CIDs do IPFS e uma variável chamada *timestamp*, utilizada como identificador do exame. Em seguida são declarados dois *mappings* para armazenar os dados no estado da blockchain (Figura 3.13).

```
18 contract Exams {
17
16   struct ExamResult {
15     string[] videos;
14     string[] images;
13     string[] texts;
12     uint timestamp;
11   }
10
 9   mapping(address => ExamResult[]) exams;
 8   mapping(address => address[]) permissions;
```

Figura 3.13: *Variáveis de estado do contrato inteligente*

Mappings, na linguagem Solidity, funciona como uma *hash table*, e são usados para armazenar dados em forma de pares chave-valor. O primeiro, chamado *exams*, é um par chave-valor de endereço que aponta para um *array* da estrutura declarada inicialmente, ou seja, local onde são armazenados os exames de cada paciente. O segundo, chamado *permissions*, é um par chave-valor de endereço que aponta para um *array* de endereços autorizados de cada paciente.

Para salvar os exames, entrada de dados representada na Figura 3.4, a função utilizada no contrato é a *storeExam()*. Quando esta função é chamada com todos os parâmetros enviados, é criada a estrutura do tipo *ExamResult*, que é acrescentada ao *array* que pertence ao endereço do paciente.

```
function storeExam(address _pacientAddress, string[] memory _videos, string[] memory _images, string[] memory _texts, uint _timestamp) public {
  exams[_pacientAddress].push(ExamResult(_videos, _images, _texts, _timestamp));
}
```

Figura 3.14: *Função para armazenamento de dados do exame.*

O controle de acesso apresentado na Figura 3.6 utiliza três funções diferentes do contrato inteligente. Para a listagem dos endereços autorizados é executada uma transação que chama a função *getPermissions()*, que retorna o *array* de endereços pertencente ao endereço que executou a transação. A variável *msg* em Solidity é uma palavra reservada que permite acessar algumas propriedades da transação chamada; a propriedade *sender* de *msg* é o endereço que executou a transação que está chamando esta função, que neste caso é o endereço do paciente. As funções *addPermission()* e *removePermission()* permitem, respectivamente, acrescentar e remover endereços da lista de endereços autorizados por um paciente.

```
function addPermission(address _addressToPermit) public {
  for(uint i = 0; i < permissions[msg.sender].length; i++) {
    if(permissions[msg.sender][i] == _addressToPermit) {
      revert("Permission already granted");
    }
  }

  permissions[msg.sender].push(_addressToPermit);
}

function getPermissions() public view returns (address[] memory) {
  return permissions[msg.sender];
}

function removePermission(address _addressToRemove) public {
  for(uint i = 0; i < permissions[msg.sender].length; i++) {
    if(permissions[msg.sender][i] == _addressToRemove) {
      return delete permissions[msg.sender][i];
    }
  }

  revert("Address not found");
}
```

Figura 3.15: *Funções para o controle de acesso do paciente sobre seus exames.*

Para a listagem de exames de um paciente há duas funções diferentes (Figura 3.16). A função *getOwnExams()* retorna os exames do endereço que executou a transação, não sendo necessária nenhuma verificação, pois se trata dos dados do endereço que executa aquela transação. Já a função *searchPatientExam()* retorna a lista de exames de um endereço escolhido (paciente) por quem solicitou a transação. Esta lista, porém, só é retornada caso o endereço do solicitante esteja entre os endereços autorizados do paciente.

```
function getOwnExams() public view returns (ExamResult[] memory) {
  return exams[msg.sender];
}

function searchPatientExam(address _pacientAddress) public view returns (ExamResult[] memory) {
  for(uint i = 0; i < permissions[_pacientAddress].length; i++) {
    if(msg.sender == permissions[_pacientAddress][i]) {
      return exams[_pacientAddress];
    }
  }

  revert("Permission not found");
}
```

Figura 3.16: *Funções para listagem de exames*

CAPÍTULO 4

Conclusão

Este trabalho explorou tecnologia de blockchain e a arquitetura de aplicações descentralizadas, apresentando uma implementação prática dessa arquitetura no contexto do exame de pupilometria. Em contraste com uma aplicação web tradicional, a grande diferença da Web 2.0 para a Web 3.0 é em relação a como a informação é processada e armazenada.

Buscou-se implementar a aplicação seguindo os aspectos caracteríticos da Web 3.0, armazenando dados em sistemas descentralizados com uma camada de criptografia para garantir a privacidade, enviando mensagens através de um sistema descentralizado com criptografia de ponta-a-ponta e dando controle ao usuário sobre quem pode acessar seus dados. Foi implementado um front-end que interage com esses sistemas e com a blockchain, e desenvolvido um contrato inteligente que implementa a lógica para os casos de uso explicados no capítulo 3.

Embora a aplicação desenvolvida tenha efetividade, existem considerações importantes em relação à sua eficiência. A disponibilidade de conhecimento sobre a linguagem não é tão amplamente difundida quando comparado ao conhecimento disponível para a Web 2.0. Portanto, para um desenvolvedor sem experiência com a linguagem Solidity, é possível contratos funcionais, porém existem diversas técnicas mais avançadas que requerem mais experiência do desenvolvedor para evitar desperdícios computacionais e melhorar a performance da execução do contrato em relação ao consumo de *gás* na EVM.

É importante ressaltar também que operações que alteram o estado como o registro do armazenamento e o gerenciamento de permissões tem um custo a cada operação realizada, uma vez que a rede em que a aplicação se baseia é pública. Mesmo com possíveis melhorias no consumo de gás, essas operações podem ter um custo alto em razão do Ether possuir um preço de mercado muito elevado.

Olhando para a aplicação desenvolvida, os custos estão incidindo sobre os atores funcionário e paciente. Isso pode tornar a aplicação inviável, visto que o custo pode ser alto e não deveria incidir sobre os mesmos. Sendo assim, um próximo passo a ser tomado para aprimoramento deste trabalho seria levar em consideração outras plataformas de blockchain compatíveis com a EVM, e principalmente levar em consideração plataformas

privadas. Assim, o custo das alterações realizadas no estado da blockchain seriam menores ou, no caso da rede privada, não cairiam sobre a identidade daqueles que estão realizando as atividades.

Uma outra melhoria futura que pode ser implementada é no cenário do médico, onde é necessário buscar um endereço para visualizar os exames compartilhados na aplicação desenvolvida. Um cenário em que o médico pode visualizar os endereços que compartilharam seus exames é mais interessante pensando na usabilidade da aplicação.

Um problema a ser considerado é com relação ao compartilhamento da chave criptográfica através do XMTP, que ainda não possui uma forma de exclusão da mensagem enviada. Portanto, mesmo que o paciente remova o acesso de uma identidade, ainda seria possível recuperar os dados do exame se o compartilhamento da chave tiver sido realizado. Isso ocorre por dois motivos: o fato de ainda não haver a possibilidade de exclusão da mensagem através do XMTP, e o fato de que os dados da blockchain são públicos. Apesar de não ser tão simples, em posse da chave criptográfica, ainda seria possível buscar os endereços dos arquivos no IPFS armazenados na blockchain e recuperá-los.

Por fim, seria interessante considerar a descentralização completa da aplicação. Normalmente as aplicações front-end são desenvolvidas e implementadas em servidores centralizados, mas é possível usar o próprio IPFS para fazer a hospedagem da aplicação e utilizar serviços de DNS *(Domain Name Systems)* descentralizados como o ENS *(Ethereum Name Service)*.

Referências Bibliográficas

[1] ALHARBY, M.; VAN MOORSEL, A. **Blockchain-based smart contracts: A systematic mapping study**. *arXiv preprint arXiv:1710.06372*, 2017.

[2] https://whitepaper.axieinfinity.com/technology. Acesso em: 06/08/2023.

[3] https://developer.bitcoin.org/devguide/block_chain.html. Acesso em: 31/07/2023.

[4] https://github.com/gtoborges/pupilometro-ethereum-dapp. Acesso em: 22/08/2023.

[5] https://ethereum.org/en/developers/docs/gas/. Acesso em: 02/08/2023.

[6] https://ethereum.org/pt-br/developers/docs/apis/json-rpc/. Acesso em: 08/08/2023.

[7] https://ethereum.org/en/roadmap/merge/. Acesso em: 03/08/2023.

[8] GREVE, F. G.; SAMPAIO, L. S.; ABIJAUDE, J. A.; COUTINHO, A. C.; VALCY, Í. V.; QUEIROZ, S. Q. **Blockchain e a revolução do consenso sob demanda**. *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)-Minicursos*, 2018.

[9] https://github.com/ipfs/ipfs-desktop. Acesso em: 22/08/2023.

[10] KUROSE, J. F.; ROSS, K. W.; ZUCCHI, W. L. **Redes de Computadores ea Internet: uma abordagem top-down**. Pearson Addison Wesley, 2014.

[11] LEÃO, L. C. D. S. **Uma introduçao ao estudo de bitcoins e blockchains**. Master's thesis, 2019.

[12] LUZ, M. A. D.; FARIAS, K. **The use of blockchain in financial area: A systematic mapping study**. In: *XVI Brazilian Symposium on Information Systems*, p. 1–8, 2020.

[13] https://docs.metamask.io/wallet/concepts/architecture/. Acesso em: 14/08/2023.

[14] MUKHERJEE, P.; PRADHAN, C. **Blockchain 1.0 to blockchain 4.0—the evolutionary transformation of blockchain technology**. In: *Blockchain technology: applications and challenges*, p. 29–49. Springer, 2021.

[15] NAKAMOTO, S. **Bitcoin: A peer-to-peer electronic cash system**. *Decentralized business review*, 2008.

[16] PETERS, J. **Prolonged aws outage takes down a big chunk of the internet**. https://www.theverge.com/2020/11/25/21719396/amazon-web-services-aws-outage-down-internet. Acesso em: 20/07/2023.

[17] QUYEN, H. **The evolution of web**. 2023.

[18] RAY, P. P. **Web3: A comprehensive review on background, technologies, applications, zero-trust architectures, challenges and future directions**. *Internet of Things and Cyber-Physical Systems*, 2023.

[19] SARMAH, S. S. **Understanding blockchain technology**. *Computer Science and Engineering*, 8(2):23–29, 2018.

[20] SILVA, C. R. G.; OTHERS. **Pupilometria na investigação de diabetes mellitus tipo ii**. 2018.

[21] SURATKAR, S.; SHIROLE, M.; BHIRUD, S. **Cryptocurrency wallet: A review**. In: *2020 4th international conference on computer, communication and signal processing (ICCCSP)*, p. 1–7. IEEE, 2020.

[22] SZABO, N. **Smart contracts**. https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html. Acesso em: 01/08/2023.

[23] https://www.hyperledger.org/case-studies/walmart-case-study. Acesso em: 06/08/2023.

[24] WOHRER, M.; ZDUN, U. **Smart contracts: security patterns in the ethereum ecosystem and solidity**. In: *2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, p. 2–8. IEEE, 2018.

[25] WU, K. **An empirical study of blockchain-based decentralized applications**. *arXiv preprint arXiv:1902.04969*, 2019.

[26] https://xmtp.org/blog/journey-to-decentralization. Acesso em: 15/08/2023.

[27] YAGA, D.; MELL, P.; ROBY, N.; SCARFONE, K. **Blockchain technology overview**. *arXiv preprint arXiv:1906.11078*, 2019.

APÊNDICE A

Instalação da aplicação

Durante este trabalho foi construído um repositório [4] na plataforma Github para a disponibilização dos códigos-fonte da aplicação descentralizada gerada através deste trabalho. Esta parte tem como objetivo trazer um guia para o processo de instalação de todos os componentes para a replicação da aplicação por outros desenvolvedores.

A.1 Pré-requisitos

O projeto foi desenvolvido utilizando um sistema operacional Linux, mais especificamente uma distribuição baseada no Ubuntu 22.04 LTS. Embora a escolha do sistema operacional não faça diferença à nível das aplicações desenvolvidas, é recomendável utilizar a distribuição mencionada.

Para rodar as aplicações é preciso ter instalado o ambiente de desenvolvimento da linguagem NodeJS e o gerenciador de pacotes NPM. Para esta aplicação foram utilizada as versões 18.16.0 e 9.5.1, respectivamente. É preciso ter instalado o sistema de controle de versões distribuído *Git*, para clonar os repositórios das aplicações.

Também foi utilizada uma aplicação para rodar um nó do IPFS, que se encontra também em um repositório[9] no Github.

A.2 Clonando os repositórios e executando as aplicações

Afim de facilitar os diretórios aqui explicados, crie um novo diretório chamado *dapp* e acesse-o para clonar as aplicações a partir dele com o comando no terminal:

```
mkdir dapp
cd dapp
```

A.2.1 Repositório do IPFS-Desktop

Para clonar o repositório do IPFS-Desktop, digite o comando:

```
git clone https://github.com/ipfs/ipfs-desktop.git
```

Após o download do repositório da aplicação será possível acessar o diretório da mesma com o comando e instalar os pacotes necessários para rodá-la:

```
cd ipfs-desktop
npm install
```

Após a instalação dos pacotes necessários, inicie a aplicação com o comando:

```
npm run start
```

O terminal irá inicializar a execução da aplicação e será possível ver uma interface que dá alguns controles sobre a mesma. Nesta tela, vá em *Settings*, e em seguida procure o campo *IPFS Config*, que possui um *JSON* com alguns parâmetros de configuração. Nesses parâmetros, procure o campo *"Access-Control-Allow-Origin"* e acrescente o domínio da aplicação front-end *("http://localhost:5173")* para que tenha o acesso livre e salve essa configuração.

Figura A.1: *Configuração de CORS do ipfs-desktop*

A.2.2 Repositório da aplicação desenvolvida

Em um outro terminal, acesse novamente o diretório *dapp* e clone o repositório da aplicação desenvolvida com o comando:

```
git clone https://github.com/gtoborges/pupilometro-ethereum-dapp
```

Este repositório contém mais duas aplicações que serão executadas, o *front-end* que contém as interfaces, e o *back-end* que simula o ambiente da blockchain e contém os contratos inteligentes.

Executando o front-end

Para executar o *front-end*, acesse o diretório da aplicação a partir do diretório dapp, instale as dependências do projeto e execute a aplicação com os respectivos comandos:

```
cd pupilometro-ethereum-dapp/frontend
npm install
npm run dev
```

Quando a aplicação estiver sendo executada, será possível acessar as interfaces através do navegador acessando o endereço *"http://localhost:5173"*. Para acessar a aplicação é preciso também ter instalado a extensão da Metamask no navegador, caso não esteja instalada a aplicação irá redirecionar automaticamente para o site de instalação da mesma. É preciso instalá-la e alterar sua configuração para conectar-se ao ambiente local que será executado.

Executando o back-end

Em dois novos terminais acesse novamente o diretório do back-end a partir do diretório *dapp* e instale as dependencias com o comando:

```
cd pupilometro-ethereum-dapp/backend
npm install
```

Em um dos terminais, será executada a simulação do ambiente de blockchain através do *Hardhat* com o comando:

```
npx hardhat node
```

Este comando iniciará a execução de um ambiente de simulação da blockchain ao qual deverá ser feito o *deploy* do contrato inteligente desenvolvido para a aplicação. No segundo terminal, compile o contrato desenvolvido e em seguida faça o deploy do mesmo para a blockchain com os comandos:

```
npx hardhat compile
npx hardhat run scripts/deploy.ts --network localhost
```

Ao executar o *deploy*, será possível ver no terminal o endereço do contrato, como na figura A.2

```
❯ npx hardhat run scripts/deploy.ts --network localhost
Compiled 1 Solidity file successfully
0x5FbDB2315678afecb367f032d93F642f64180aa3
```

Figura A.2: *Endereço do contrato ao realizar o deploy*

No terminal que está executando o ambiente da blockchain, também será possível ver o endereço do contrato após o *deploy* nos *logs* como mostra a figura A.3

```
eth_sendTransaction
  Contract deployment: Exams
  Contract address:    0x5fbdb2315678afecb367f032d93f642f64180aa3
  Transaction:         0x33fc93373a85e10cef8366daa114908314a478d5e3c33a77fb327a987452982a
  From:                0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
  Value:               0 ETH
  Gas used:            1574328 of 1574328
  Block #1:            0xba696c3a26982472607100a17efc57b475aff29db3063b34d3bfe9f32eaa2ec2
```

Figura A.3: *Endereço do contrato nos logs do ambiente simulado da blockchain*

Este endereço deverá ser atualizado no arquivo *.env* na variável de ambiente *"VITE_CONTRACT_ADDRESS"*, que se encontra no diretório do *front-end*.

Por fim, para a execução das transações que alteram o estado da aplicação é preciso que haja algum saldo nas carteiras que estejam interagindo com a blockchain. Para isso, existe um script no mesmo diretório do script de *deploy* que irá transferir Ether para uma carteira que pode ser definida no código do script.