UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Diego Batistuta Ribeiro de Andrade

Desenvolvimento de Aplicativos descentralizados para o contrato VaccSC

Uberlândia, Brasil

2024

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Diego Batistuta Ribeiro de Andrade

Desenvolvimento de Aplicativos descentralizados para o contrato VaccSC

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Sistemas de Informação.

Orientador: Ivan da Silva Sendin

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2024

Resumo

O trabalho aborda o desenvolvimento de aplicativos descentralizados utilizando o contrato VaccSC. O principal intuito é criar uma solução segura, transparente e confiável para a fase III dos testes em humanos no desenvolvimento de vacinas, empregando tecnologias como blockchain, contratos inteligentes, Dart e Flutter. A metodologia envolve a definição das tecnologias de front-end (Dart e Flutter), análise do contrato inteligente VaccSC, implementação dos clientes, além de testes e análises. Os resultados demonstram que a solução proposta é viável, proporcionando melhorias significativas em termos de segurança, transparência e confiança. Conclui-se que o uso de aplicativos descentralizados pode transformar a fase III dos testes em humanos no desenvolvimento de vacinas, tornando o processo mais eficiente e confiável.

Palavras-chave: Aplicativos Descentralizados, Contratos Inteligentes, Desenvolvimento de Vacinas.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ABCDE	*Agile Block Chain DApp Engineering*
API	*Application Programming Interface*
AWS	*Amazon Web Service*
DAPP	*Decentralized Applications*
ERC	*Ethereum Request for Comments*
EVM	*Ethereum Virtual Machine*
IBM	*International Business Machines Corporation*
iOS	Sistema operacional da Apple
JSON	*JavaScript Object Notation*
MFA	*Multi-factor Authentication*
NFT	*Non-Fungible Token*
NPM	*Node Package Manager*
OMS	Organização Mundial da Saúde
OPAS	Organização Pan-Americana da Saúde
P2P	*Peer-to-Peer*
RPC	*Remote Procedure Call*
SC	*Smart Contracts*
SDK	*Software Development Kit*
SHA256	Algoritmo de hash seguro de 256 bits
UI	*User Interface*
UML	*Unified Modeling Language*
VE	*Vaccine efficiency*
WHO	*World Health Organization*
XOR	*Exclusive OR*

Sumário

1 Introdução

De acordo com a Organização Pan-Americana da Saúde (2024), o surto do novo coronavírus, identificado inicialmente em dezembro de 2019 na cidade de Wuhan, China, rapidamente se espalhou pelo mundo, levando a World Health Organization (2020b) a declarar uma emergência de saúde pública de importância internacional em janeiro de 2020. O vírus, denominado SARS-CoV-2, causa a doença COVID-19, que pode variar de sintomas leves a graves, incluindo pneumonia e insuficiência respiratória. A pandemia resultou em milhões de casos e mortes globalmente, impactando severamente sistemas de saúde, economias e a vida cotidiana das pessoas. Medidas como distanciamento social, uso de máscaras e campanhas de vacinação em massa foram implementadas para controlar a disseminação do vírus. Um dos principais impactos foi no desenvolvimento de vacinas, que anteriormente se esperava levar anos ou até décadas para serem analisadas, desenvolvidas, testadas, aprovadas e disponibilizadas ao mercado. No entanto, grande parte desse processo foi concluída em questão de meses, contrariando as expectativas anteriores na visão de Gallagher (2020).

Segundo a World Health Organization (2020a), depois que as vacinas são submetidas a exames, avaliações e testes para verificar sua segurança e potencial para prevenir doenças, elas passam por testes em humanos em três fases: Fase I, Fase II e Fase III. A Fase I envolve um pequeno grupo de voluntários, a Fase II inclui centenas de voluntários, e a Fase III é administrada a milhares de voluntários. Na Fase III, há voluntários que recebem a vacina e outros que recebem um placebo para determinar a eficácia da vacina. Um placebo é uma substância ou tratamento que não tem nenhum efeito real no corpo e é utilizado em pesquisas clínicas para avaliar a eficácia de novos medicamentos.

Uma vacina precisa comprovar que é segura e eficaz em uma vasta população antes de ser aprovada e introduzida em um programa nacional de vacinação. No entanto, muitas vezes, para ver os resultados desses processos, é muito trabalhoso para a sociedade como um todo.

Além da transparência, World Health Organization (2020a) também comenta que, durante os ensaios das fases de testes, os voluntários e os cientistas que participam no estudo são impedidos de saber quais voluntários receberam a vacina do ensaio ou o produto de comparação. Esse método, conhecido como duplo-cego, é necessário para garantir que nem os voluntários nem os cientistas sejam influenciados em sua avaliação sobre a segurança e eficácia, desconhecendo qual produto cada um recebeu.

Na visão da International Business Machines Corporation (2024a), a *Blockchain* é uma tecnologia que funciona como um livro-razão digital descentralizado, registrando

transações de forma segura e transparente. Cada transação é armazenada em blocos de dados, que são conectados em uma cadeia imutável. Isso significa que, uma vez registrada, a informação não pode ser alterada, garantindo a integridade dos dados. A descentralização elimina a necessidade de intermediários, pois a rede é mantida por vários computadores que verificam e validam as transações.

Além disso, a *Blockchain* possibilita a execução de Contratos Inteligentes, que são programas autoexecutáveis que garantem o cumprimento automático de acordos. Conforme Antonopoulos e Wood (2018), os "Contratos Inteligentes" são softwares imutáveis, executados deterministicamente no contexto de uma máquina virtual Ethereum como parte do protocolo de rede Ethereum, ou seja, uma vez implantado o Contrato Inteligente, ele não pode ser alterado e o resultado da sua execução é o mesmo para todos que o executam.

Sendin e Miani (2021) apresentam um protocolo inovador baseado em Contratos Inteligentes e esquemas de compromisso, visando garantir transparência, contabilidade e confidencialidade na Fase III de experimentos de vacinas. A principal contribuição do trabalho deles é a utilização de esquemas de compromisso para assegurar as propriedades de duplo-cego, randomização e auditabilidade dos dados clínicos, mesmo na presença de participantes desonestos. Além disso, os Contratos Inteligentes podem determinar se o experimento atingiu a eficiência desejada e decidir sobre a aprovação da vacina.

Neste trabalho, foi proposto o desenvolvimento de aplicativos para os participantes da Fase III dos testes de vacinas, utilizando o contrato desenvolvido por Sendin e Miani (2021). Dessa forma, os participantes não precisarão ter conhecimento sobre Contratos Inteligentes nem saber como interagir diretamente com eles. Isso facilitará e agilizará o processo de testes, além de garantir a transparência e a confiança nos resultados.

1.1 Objetivo Geral

O objetivo deste trabalho é desenvolver aplicativos descentralizados que utilizarão o contrato de Sendin e Miani (2021), de complexidade razoável e já existente, para dar aplicabilidade a tal contrato. Dessa forma, serão desenvolvidos clientes para utilizar esse contrato, garantindo resiliência, transparência, confiança e resistência à censura, que são características importantes dos aplicativos descentralizados.

1.2 Objetivos Específicos

Para atingir o Objetivo Geral proposto, os seguintes Objetivos específicos foram detectados:

Escolher uma plataforma e uma tecnologia/linguagem: A escolha da plataforma e da tecnologia ou linguagem de programação é crucial para o desenvolvimento de um DApp. É importante considerar fatores como a escalabilidade, a segurança e a compatibilidade com a *Blockchain* escolhida. A decisão deve ser baseada nas necessidades específicas do projeto e na experiência da equipe de desenvolvimento.

Desenvolver um sistema de comunicação com o contrato: A comunicação eficiente com o Contrato Inteligente é vital para o funcionamento de um DApp. Isso envolve a criação ou utilização de APIs ou outras interfaces que permitam ao DApp enviar e receber dados do Contrato Inteligente na *Blockchain*. A implementação deve garantir que as transações sejam realizadas de forma segura e que os dados sejam transmitidos de maneira confiável e eficiente, respeitando as regras e condições definidas no contrato.

Desenvolver uma interface: A interface do usuário (UI) é a parte do DApp com a qual os usuários interagem diretamente. É importante que a UI seja intuitiva, responsiva e fácil de usar, proporcionando uma experiência agradável ao usuário. O design deve considerar a usabilidade e a acessibilidade, garantindo que todas as funcionalidades do DApp sejam facilmente acessíveis. Ferramentas e *frameworks* modernos de desenvolvimento *front-end* podem ser utilizados para criar interfaces dinâmicas e interativas.

1.3 Justificativa

Os aplicativos descentralizados ou *Decentralized applications (DApps)* são programas que operam em redes *Blockchain*, eliminando a necessidade de uma autoridade central. Eles oferecem transparência, segurança robusta e autonomia, pois seu código-fonte é aberto e suas regras são executadas automaticamente. Além disso, muitos DApps utilizam tokens nativos para facilitar transações e incentivar a participação dos usuários, revolucionando setores como finanças, jogos e redes sociais.

Utilizar DApps é uma maneira eficaz de se beneficiar da segurança, transparência e imutabilidade proporcionadas pelos Contratos Inteligentes e pela *Blockchain*. Dessa forma, os DApps são essenciais para a aplicabilidade prática dos Contratos Inteligentes, permitindo a execução automática de transações e acordos sem a necessidade de intermediários.

No entanto, para que os DApps funcionem de maneira eficiente e segura, é necessário atender a vários requisitos. Além da resiliência, transparência e resistência à censura, que são características intrínsecas dos DApps, é crucial implementar medidas robustas de segurança e controle de acesso. Isso inclui a proteção contra vulnerabilidades, a garantia de que apenas usuários autorizados possam interagir com o aplicativo e a manutenção da

integridade dos dados. Esses requisitos são fundamentais para assegurar que os DApps operem de forma confiável e segura, proporcionando uma experiência de usuário robusta e confiável.

2 Referencial Teórico

Neste capítulo, exploraremos a criptografia, os conceitos fundamentais da *Blockchain* e os Contratos Inteligentes, além de discutir questões técnicas pertinentes ao desenvolvimento do DApp. O capítulo conclui com os trabalhos correlatos.

2.1 Criptografia

De acordo com a *International Business Machines Corporation (2024b)*, a criptografia é a prática de desenvolver e usar algoritmos codificados para proteger e obscurecer informações transmitidas, garantindo que apenas aqueles com permissão possam acessá-las. Na era digital moderna, a criptografia se tornou uma ferramenta essencial de cibersegurança, protegendo dados sensíveis contra hackers e outros cibercriminosos. Ela é usada para transformar mensagens em um formato ilegível (cifrado) que só pode ser decifrado pelo destinatário autorizado, utilizando uma chave secreta específica.

2.1.1 Funções Hash

Como aponta Menezes, Oorschot e Vanstone (1996), as funções hash são algoritmos que transformam uma entrada de dados (ou mensagem) de comprimento arbitrário em uma saída de comprimento fixo, conhecida como hash ou valor hash. Essas funções são amplamente utilizadas em criptografia para garantir a integridade dos dados e a autenticação de mensagens. Por exemplo, ao enviar um arquivo pela internet, podemos calcular o hash do arquivo antes de enviá-lo e, em seguida, calcular o hash do arquivo novamente quando ele chega ao destino para verificar se o hash é o mesmo, o que indica que o arquivo não foi alterado durante a transferência. Isso permite uma verificação eficiente da integridade e autenticidade da mensagem original, pois as funções hash não são reversíveis. Além disso, funções hash são desenvolvidas para serem resistentes a colisões, ou seja, é extremamente difícil encontrar duas entradas diferentes que produzam o mesmo valor hash.

2.1.2 Esquema de Compromisso

Segundo Menezes, Oorschot e Vanstone (1996), um esquema de compromisso permite que uma parte se comprometa com um valor específico enquanto mantém esse valor oculto até um momento posterior, quando poderá ser revelado e verificado por outras partes. Assim, a utilização de esquemas de compromisso contribui significativamente para a robustez e a confiança em sistemas criptográficos avançados. Esses esquemas são funda-

mentais para garantir a integridade e a autenticidade em diversas aplicações criptográficas, como leilões eletrônicos e votações seguras. O processo geralmente envolve duas fases: a fase de compromisso, onde o valor é escondido, e a fase de revelação, onde o valor é revelado e verificado.

2.1.2.1 *Cara ou coroa por telefone*

O cara ou coroa por telefone é um protocolo criado por Blum (1981), que permite a realização de uma decisão justa entre duas partes que não confiam completamente uma na outra, sem a necessidade de um árbitro. Esse protocolo pode ser descrito da seguinte forma:

1. **Escolha de um número:** Uma das partes, chamada Alice, escolhe um número aleatório (x) e aplica uma função unidirecional (f(x)) para gerar um valor (y).
2. **Envio do valor:** Alice envia (y) para a outra parte, chamada Bob, sem revelar o número original (x).
3. **Adivinhação:** Bob tenta adivinhar se (x) é par ou ímpar.
4. **Revelação:** Alice revela o número original (x).
5. **Verificação:** Bob verifica se sua adivinhação estava correta, usando (x) e (f(x)).

De acordo com Menezes, Oorschot e Vanstone (1996), este protocolo é relevante na criptografia moderna, pois demonstra como funções unidirecionais podem ser usadas para resolver problemas de confiança em ambientes distribuídos. Ele serve como base para o desenvolvimento de outros protocolos criptográficos que exigem decisões justas e verificáveis, como Contratos Inteligentes e sistemas de votação eletrônica.

2.1.3 Assinaturas digitais

Na visão de Stallings (2017), as assinaturas digitais são um mecanismo criptográfico que assegura a autenticidade e a integridade de documentos eletrônicos. Utilizando algoritmos de chave pública e privada, uma assinatura digital é gerada a partir de um hash criptográfico do documento, que é então criptografado com a chave privada do signatário. Esse processo garante que qualquer alteração no documento após a assinatura será detectada, pois o hash resultante não corresponderá ao original. Além disso, a verificação da assinatura pode ser feita por qualquer pessoa com acesso à chave pública do signatário, confirmando a identidade do autor e a integridade do documento .

Segundo Rivest, Shamir e Adleman (1978), a autenticidade refere-se à garantia de que a identidade do remetente ou signatário é verdadeira, ou seja, que a pessoa que assinou

o documento é realmente quem diz ser. Já a integridade diz respeito à certeza de que o conteúdo do documento não foi alterado desde a sua assinatura. As assinaturas digitais garantem esses dois aspectos ao utilizar certificados digitais emitidos por autoridades certificadoras confiáveis, que validam a identidade do signatário, e ao gerar um hash criptográfico do documento, que assegura que qualquer modificação será detectada.

2.2 Web3

Como apresentado por *Amazon Web Services (2024b)*, a Web3 refere-se a um conjunto de tecnologias que visam descentralizar a propriedade e o controle de dados na internet, utilizando principalmente *Blockchain*. Diferente das aplicações tradicionais da internet, que são controladas por entidades centralizadas, a Web3 permite que os usuários finais tenham maior controle sobre seus dados e contribuições para o desenvolvimento técnico das plataformas. Essa nova fase da internet promove a ausência de confiança em autoridades centrais, a interoperabilidade entre diferentes tecnologias e a utilização da web semântica para uma compreensão mais profunda dos dados.

2.3 Blockchain

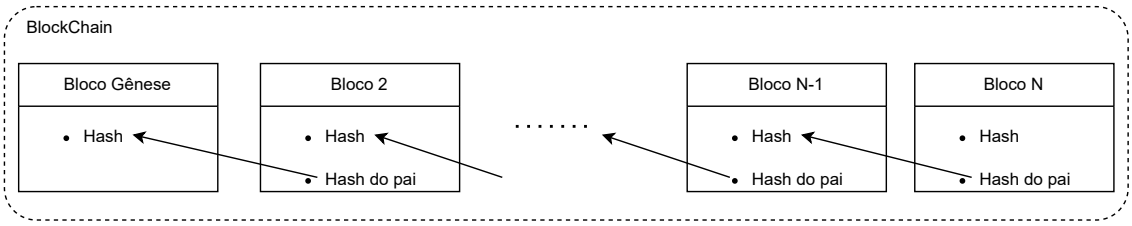
Segundo a *Amazon Web Services (2024a)*, a tecnologia *Blockchain* é um mecanismo de banco de dados sofisticado que possibilita o compartilhamento transparente de informações na rede de uma empresa. A *Blockchain* armazena dados em blocos interconectados em uma cadeia, garantindo consistência cronológica, pois a cadeia não pode ser excluída ou modificada sem o consenso da rede. Isso permite o uso da tecnologia *Blockchain* para criar um registro inalterável e imutável para monitorar pedidos, pagamentos, contas e outras transações. Além disso, o sistema possui mecanismos integrados que impedem entradas de transações não autorizadas e garantem consistência na visualização compartilhada dessas transações.

Na visão da *International Business Machines Corporation (2024a)*, a *Blockchain* é uma tecnologia que funciona como um livro-razão digital descentralizado, registrando transações de forma segura e transparente. Cada transação é armazenada em blocos de dados, que são conectados em uma cadeia imutável. Isso significa que, uma vez registrada, a informação não pode ser alterada, garantindo a integridade dos dados. A descentralização elimina a necessidade de intermediários, pois a rede é mantida por vários computadores (nós) que verificam e validam as transações. Além disso, a *Blockchain* permite a execução de Contratos Inteligentes, que são programas autoexecutáveis que garantem o cumprimento automático de acordos.

Cada bloco na *Blockchain* é distinguido por um hash em seu cabeçalho, gerado pelo

algoritmo criptográfico SHA256. Além disso, cada bloco faz referência ao seu predecessor, o 'bloco pai', por meio do campo 'hash do bloco anterior' em seu próprio cabeçalho. Em termos simples, o cabeçalho de cada bloco incorpora o hash do bloco pai. Esta cadeia de hash, que conecta cada bloco ao seu antecessor, forma uma trilha rastreável até o primeiro bloco já criado, conhecido como 'bloco gênese'.

Figura 1 – Exemplo Simplificado de como funciona o encadeamento de hash na *Blockchain*.



Fonte: Elaborado pelo autor (2024)

O campo 'hash do bloco anterior', presente no cabeçalho do bloco, influencia o hash do bloco atual e qualquer alteração em um bloco altera seu hash, afetando assim todos os blocos subsequentes. Portanto, um bloco com muitos sucessores não pode ser modificado sem recalcular todos os blocos seguintes e este recálculo exigiria um processamento computacional significativo. Logo, a presença de uma longa cadeia de blocos torna a *Blockchain* imutável, uma característica fundamental para sua segurança.

2.4 Contrato Inteligente

Para Szabo (1994), um Contrato Inteligente é um protocolo de transação computadorizado que executa os termos de um contrato. Os objetivos gerais do design de Contratos Inteligentes são satisfazer condições contratuais comuns (tais como condições de pagamento, garantias, confidencialidade e até mesmo execução), minimizar exceções tanto maliciosas quanto acidentais, e minimizar a necessidade de intermediários confiáveis. Os objetivos econômicos relacionados incluem a redução de perdas por fraude, custos de arbitragem e execução e outros custos de transação.

De acordo com Ethereum (2022), em um nível muito básico, você pode pensar em um Contrato Inteligente como uma espécie de máquina de venda automática: um script que, quando chamado com certos parâmetros, executa algumas ações ou cálculos se determinadas condições forem satisfeitas. Além disso, qualquer desenvolvedor pode criar um Contrato Inteligente e torná-lo público na rede, usando o *Blockchain* como sua camada de dados, porém deve-se pagar uma taxa à rede e qualquer usuário pode chamar o Contrato Inteligente para executar seu código, novamente pagando uma taxa à rede.

Contratos Inteligentes são geralmente escritos em uma linguagem de alto nível, como Solidity, porém são compilados no *bytecode* de baixo nível executado na Máquina

Virtual Ethereum(EVM) e, após compilados, eles são implantados na plataforma Ethereum usando uma transação especial de criação de contrato, ou seja, uma transação para o endereço 0x0. Cada contrato é identificado por um endereço Ethereum, derivado da transação de criação do contrato em função da conta de origem e do *nonce*, assim esse endereço pode ser usado em uma transação como destinatário, enviando fundos para o contrato ou chamando uma das funções do contrato. Além disso, o criador do contrato não possui privilégios especiais ao nível de protocolo; no entanto, eles podem codificar privilégios diretamente no Contrato Inteligente (ANTONOPOULOS; WOOD, 2018).

2.4.1 Ethereum

Conforme mencionado por Antonopoulos e Wood (2018), de forma prática o Ethereum é uma infraestrutura de computação de código aberto e globalmente descentralizada que executa Contratos Inteligentes e utiliza uma *Blockchain* para sincronizar e armazenar as mudanças de estado do sistema, com uma criptomoeda chamada ether para medir e restringir os custos dos recursos de execução. Dessa forma, os desenvolvedores podem criar aplicativos descentralizados com funções econômicas integradas, além de oferecer alta disponibilidade, auditabilidade, transparência e neutralidade; também reduz ou elimina a censura e reduz certos riscos de contraparte.

Segundo Ethereum (2022), no universo Ethereum, existe um único computador canônico (chamado de Ethereum Virtual Machine, ou EVM) cujo estado todos na rede Ethereum concordam. Todos que participam da rede Ethereum mantêm uma cópia do estado deste computador. Além disso, qualquer participante pode transmitir uma solicitação para que este computador execute cálculos arbitrários. Sempre que uma solicitação desse tipo é transmitida, outros participantes da rede verificam, validam e executam a computação. Essa execução causa uma mudança de estado no EVM, que é confirmada e propagada por toda a rede.

De acordo com Antonopoulos e Wood (2018), do ponto de vista da ciência da computação, o Ethereum é uma máquina de estado determinística, mas praticamente ilimitada, consistindo em um estado *singleton* globalmente acessível e uma máquina virtual que aplica alterações nesse estado. Segundo Gamma (2009), o padrão *Singleton* é um padrão de design de *software* onde uma classe tem apenas uma instância durante a execução do programa, fornecendo um ponto de acesso global a essa instância. Sendo particularmente útil em situações onde é necessário que apenas um objeto gerencie ações específicas, como uma configuração centralizada que deve ser acessível a partir de vários pontos do código. Este padrão é implementado utilizando um método estático que controla a criação da instância única, evitando a instanciação múltipla da classe.

2.5 Aplicativo Descentralizado(DApp)

Segundo a BinanceAcademy (2023), aplicativos descentralizados (DApps) são aplicativos ou programas digitais baseados em Contratos Inteligentes, que são executados em *Blockchains* em vez de servidores centralizados. Eles são semelhantes aos aplicativos comuns de dispositivos móveis e oferecem uma ampla variedade de serviços e funções para jogos, finanças, redes sociais e muito mais.

Na visão da Ethereum (2022), as vantagens dos DApps são:

Tempo de inatividade zero: Uma vez que o Contrato Inteligente é implantado na *Blockchain*, a rede como um todo sempre será capaz de atender clientes que procuram interagir com o contrato. Atores mal-intencionados, portanto, não podem lançar ataques de negação de serviço direcionados a DApps individuais.

Privacidade: Você não precisa fornecer identidade do mundo real para implantar ou interagir com um DApp.

Resistência à censura: Nenhuma entidade na rede pode impedir que os usuários enviem transações, implantem DApps ou leiam dados do *Blockchain*.

Integridade completa dos dados: Os dados armazenados na *Blockchain* são imutáveis e indiscutíveis, graças as primitivas criptográficas. Atores mal-intencionados não podem forjar transações ou outros dados que já foram tornados públicos.

Computação confiável/comportamento verificável: Contratos Inteligentes podem ser analisados e são garantidos para execução de formas previsíveis, sem a necessidade de confiar em uma autoridade central. Isso não é verdade nos modelos tradicionais; por exemplo, quando usamos sistemas bancários online, devemos confiar que as instituições financeiras não usarão indevidamente nossos dados financeiros, adulterarão registros ou serão hackeadas.

Enquanto isso, as desvantagens das DApps são:

Manutenção: DApps pode ser mais difícil de manter porque o código e os dados publicados no *Blockchain* são mais difíceis de modificar. É difícil para os desenvolvedores fazer atualizações para seus DApps (ou os dados subjacentes armazenados por um DApp) uma vez que eles são implantados, mesmo que *bugs* ou riscos de segurança sejam identificados em uma versão antiga.

Sobrecarga enorme/Dimensionamento difícil: Para alcançar o nível de segurança, integridade, transparência e confiabilidade que o Ethereum aspira, cada nó é executado e armazena cada transação, além disso, o consenso de prova de participação

também leva tempo, podendo gera uma sobrecarga enorme e um difícil dimensionamento.

Congestionamento de rede: Quando um DApp usa muitos recursos computacionais, toda a rede recebe backup. Atualmente, a rede só pode processar cerca de 10-15 transações por segundo; se as transações estiverem sendo enviadas mais rápido do que isso, o pool de transações não confirmadas pode rapidamente encher.

Experiência do usuário: Pode ser mais difícil projetar experiências fáceis de usar, pois o usuário final médio pode achar muito difícil configurar uma pilha de ferramentas necessária para interagir com o *Blockchain* de forma verdadeiramente segura.

Centralização: Soluções fáceis de usar e amigáveis para desenvolvedores construídas em cima da camada base do Ethereum podem acabar parecendo serviços centralizados de qualquer maneira. Por exemplo, esses serviços podem armazenar chaves ou outros dados confidenciais do lado do servidor, servir a um *front-end* usando um servidor centralizado ou executar uma lógica de negócios importante em um servidor centralizado antes de escrever para o *Blockchain*. A centralização elimina muitas (se não todas) das vantagens do *Blockchain* em relação ao modelo tradicional

2.6 Aplicações

Os DApps estão sendo adotados em muitas áreas, por exemplo, jogos, câmbio, saúde, identidade, etc. Além disso, esses DApps normalmente são parcialmente descentralizados; por exemplo, alguns utilizam tokens não fungíveis (NFT) para representar itens ou personagens nos jogos, ou utilizam a *Blockchain* para validar as identidades de usuários.

2.6.1 Jogos

No contexto dos jogos, os DApps oferecem uma nova dimensão de interatividade e propriedade digital, permitindo que os jogadores possuam, troquem e monetizem ativos virtuais dentro do jogo. Exemplos notáveis incluem o Axie-Infinity (S.I), que utiliza a *Blockchain* da Ethereum para permitir que jogadores coletem, treinem e batalhem com criaturas digitais chamadas Axies. Segundo o Herrera (2022), a popularidade desses jogos baseados em *Blockchain* cresceu exponencialmente, evidenciada pelo aumento de 2.000% na atividade de jogos *Blockchain* no primeiro trimestre de 2022 em comparação com o ano anterior. Esses DApps não apenas mudaram as experiências dos jogos, mas também introduziram novos modelos econômicos, como o *"play-to-earn"*, onde os jogadores são recompensados com criptomoedas por seu tempo e esforço investidos no jogo.

2.6.2 Finanças

Segundo BinanceAcademy (2023), através dos DApps, todos podem usar serviços financeiros sem a necessidade de uma autoridade central, além de manter o controle total sobre seus ativos. O setor de finanças descentralizadas também oferece benefícios a pessoas de baixa renda, oferecendo acesso a uma ampla gama de serviços financeiros a custos significativamente mais baixos.

2.6.3 Entretenimento

De acordo com Coinext (S.I.), a Audius (S.I), é um protocolo de streaming de música descentralizado, que oferece uma plataforma na qual os artistas conseguem hospedar suas faixas de forma independente e se conectar com seus ouvintes. Isso acontece sem a necessidade de intermediários, como grandes corporações e gravadoras, mas sim por meio da tecnologia *Blockchain*.

2.6.4 Governança

Para a BinanceAcademy (2023), os DApps também capacitam seus usuários para desempenhar um papel importante na governança de organizações online, introduzindo um mecanismo de tomada de decisão mais focado na comunidade. Com o auxílio de Contratos Inteligentes, os usuários que possuem tokens de governança de um projeto *Blockchain* específico podem criar propostas para a comunidade e votar anonimamente nas propostas de outras pessoas.

2.7 Trabalho Correlatos

Nesta seção, serão apresentados trabalhos correlatos que tratam sobre a criação de DApps.

No artigo de Johnson et al. (2019), o objetivo é servir como um tutorial para o desenvolvimento de um protótipo funcional completo de um DApp e destacar os benefícios específicos do uso de um DApp como um método para os participantes da pesquisa compartilharem características de seus dados brutos, preservando ao mesmo tempo a privacidade do participante, não revelando os dados brutos em si. Este estudo foi motivado por um caso de uso real em pesquisa biomédica que exige privacidade de dados. Neste trabalho, foi descrita a arquitetura de um DApp, os detalhes de implementação de um Contrato Inteligente, um exemplo de DApp do sistema operacional do iPhone (iOS) que interage com o Contrato Inteligente e as ferramentas e bibliotecas de desenvolvimento necessárias para começar.

No trabalho de Marchesi, Marchesi e Tonelli (2020) foi descrita a arquitetura de um DApp e apresentados os problemas e práticas específicas necessárias para o desenvolvimento de DApp. Além disso, também foi descrito o processo *Agile Block Chain DApp Engineering*(ABCDE) proposto em todos os detalhes, incluindo as modificações de alguns diagramas UML para lidar com conceitos do Solidity, avaliação de segurança e o que é necessário para estender a notação a outras linguagens para desenvolvimento de SC. Ademais, um exemplo simplificado foi demonstrado, extraído de um caso real, juntamente com relatórios sobre os usos reais do ABCDE.

De acordo com Marchesi, Marchesi e Tonelli (2020), o ABCDE aproveita práticas ágeis, pois o desenvolvimento de DApps geralmente trata da implementação rápida de sistemas cujos requisitos não são totalmente compreendidos no início e tendem a mudar com o tempo. Porém, dada a especificidade dos *Blockchains*, o ABCDE complementa o desenvolvimento incremental e iterativo por meio de iterações em caixa, típicas da agilidade, com ferramentas mais formais. Essas ferramentas incluem uma modelagem completa de interações entre software tradicional e ambiente *Blockchain*, usando diagramas de classes UML, diagramas de casos de uso UML, diagramas de sequência UML, todos especializados para desenvolvimento de aplicativos baseados em *Blockchain* usando estereótipos.

O artigo de Renu e Banik (2021) tem como objetivo relacionar o conceito de cidade inteligente através da introdução do sistema de transporte inteligente e explorar as oportunidades de adoção da tecnologia *Blockchain* em serviços de compartilhamento de viagens. Este artigo propõe uma estrutura baseada em *Blockchain* a partir da estrutura centralizada existente para um serviço de compartilhamento de viagens. Este DApp usa o algoritmo de correspondência mínima para combinar os passageiros que solicitam carona compartilhada para economizar a distância total da viagem.

No trabalho de Cai et al. (2018), são apresentadas cinco características desejáveis em uma *Blockchain* para os DApps. Primeiramente, melhor desempenho, com baixa latência, alto rendimento e desempenho sequencial rápido. Em segundo lugar, a habilitação de transações offline, pois, se um subconjunto de dispositivos se desconectar da internet e realizar transações entre si, não há garantia de que o gasto duplo não ocorrerá. Em terceiro lugar, um custo monetário razoável, com uma baixa taxa de transação, uma vez que atualmente as taxas são altas tanto para implantar quanto para executar Contratos Inteligentes, além de um modelo de negócio com a flexibilidade de oferecer serviços gratuitos aos usuários. Em quarto lugar, manutenção flexível, com atualizações do sistema e flexibilidade no suporte a *bugs* e erros. E, por último, uma gestão de identidade mais simples, que possibilita uma gestão de identidade mais direta e eficiente.

No trabalho de Udokwu, Anyanka e Norta (2020) são apresentados 5 métodos para desenvolver DApps:

Utilizando o Ancile: O Ancile criado por Foster et al. (2019) é uma ferramenta projetada para abordar questões críticas de privacidade e segurança de dados no contexto do gerenciamento de prontuários eletrônicos de pacientes em sistemas de saúde. Ele oferece um conjunto de técnicas que garantem que os dados dos pacientes sejam protegidos durante o armazenamento e transferência, prevenindo acessos não autorizados e mantendo a integridade dos registros.O Ancile utiliza criptografia avançada e outros mecanismos de segurança para assegurar que apenas indivíduos autorizados possam acessar e manipular os dados, proporcionando assim uma camada adicional de confiança e proteção em sistemas de saúde digitalizados.

Orientada por ontologia: O estudo propõe técnicas de modelagem de dados orientadas por ontologia para projetar o sistema *Blockchain* para permitir o rastreamento de itens em uma cadeia de suprimentos

Orientada ao modelo UML: O método propõe o uso de modelos UML como diagramas de casos de uso, diagramas de classes e diagramas de atividades na descrição do requisito e da arquitetura de um DApp.

Práticas atuais: No desenvolvimento de aplicativos descentralizados em uma *Blockchain*: metodologia *Agile* no desenvolvimento de DApps devido à flexibilidade dos processos e ferramentas comuns de gerenciamento de tarefas, como Scrum e Kanban.

Técnicas existentes: No desenvolvimento de software DApps: apresentação de ferramenta de modelagem existentes para DApps

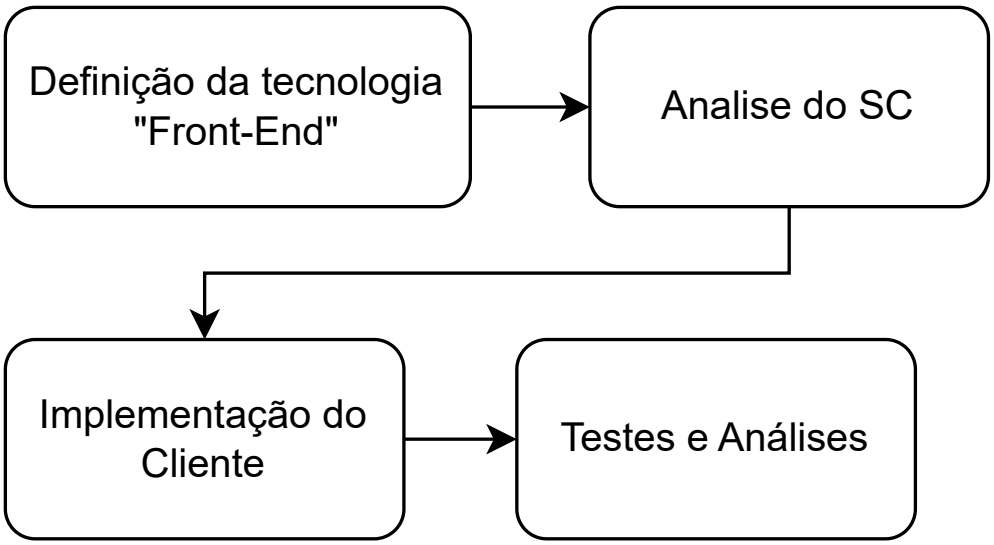
3 Desenvolvimento

Neste capítulo, serão apresentadas as etapas e detalhes envolvidos na criação do aplicativo descentralizado. A sequência deste processo é ilustrada na Figura 2 e será detalhada nas seções subsequentes.

3.1 Metodologia

O processo de desenvolvimento deste trabalho foi dividido nas seguintes etapas:

Figura 2 – Etapas da metodologia.



Fonte: Elaborado pelo autor (2024)

Definição das tecnologias *front-end*: Estudo e definição das tecnologias utilizadas no desenvolvimento da interface do usuário, incluindo *frameworks*, bibliotecas e ferramentas.

Análise do SC: Avaliação do Contrato Inteligente (SC) fornecido.

Implementação do cliente: Desenvolvimento da interface do usuário, implementação de sistemas de autenticação e integração com o Contrato Inteligente para comunicação eficiente.

Testes e Análises: Realização de testes no cliente seguido de uma análise dos resultados obtidos.

3.2 Tecnologias

Para a execução deste trabalho, foram sondadas algumas opções de tecnologias para serem utilizadas no desenvolvimento do *front-end*. A seguir, apresentaremos uma breve descrição sobre as tecnologias escolhidas.

3.2.1 Dart

Segundo o Dart (2023), ele é uma linguagem otimizada para o cliente para o desenvolvimento de aplicativos rápidos em qualquer plataforma. Seu objetivo é oferecer a linguagem de programação mais produtiva para desenvolvimento multiplataforma, combinada com uma plataforma de tempo de execução flexível para estruturas de aplicativos. O Dart foi projetado priorizando o desenvolvimento e as experiências de produção de alta qualidade em uma ampla variedade de destinos de compilação.

3.2.2 Flutter

Na visão do Alberto (2023), o Flutter é um *framework* de desenvolvimento com foco em multiplataforma para dispositivos móveis. Desenvolvido pelo Google, é amplamente utilizado no mercado e passou a permitir a criação de aplicações para *desktop* (Linux, Windows e macOS). Na prática, o Flutter permite o desenvolvimento de aplicativos para diferentes plataformas - Android e iOS - simultaneamente com um único código. Isso traz benefícios como versatilidade, menor curva de aprendizado e agilidade. É particularmente vantajoso para as empresas, pois um único desenvolvedor pode criar aplicativos para diferentes sistemas operacionais, ao contrário do modelo nativo, que exige um desenvolvedor para cada plataforma.

O Flutter utiliza a linguagem de programação Dart, que é orientada a objetos e oferece vários recursos avançados. Ele também inclui uma ampla biblioteca de *Widgets* personalizáveis, que permite criar interfaces de usuário atraentes e responsivas. Além disso, o Flutter inclui um conjunto de ferramentas integradas, como o Flutter SDK, o Flutter Engine e o Flutter *Framework*, que permitem criar, compilar, testar e depurar aplicativos de maneira fácil e rápida.

Por fim, o Flutter é uma opção popular para o desenvolvimento de aplicativos móveis e web, oferecendo recursos avançados, ferramentas integradas e a possibilidade de criar aplicativos para múltiplas plataformas ao mesmo tempo, tudo isso enquanto reduz o tempo e o custo do desenvolvimento. Portanto, analisando as vantagens do Flutter junto ao meu interesse em aprender mais sobre, ele foi escolhido como o *framework* que será utilizado para a criação do DApp.

3.2.2.1 Web3Dart

O Web3dart é uma biblioteca para o Flutter que permite a interação com redes baseadas em Ethereum, como a mainnet Ethereum, testnets e redes privadas. Essa biblioteca foi especialmente projetada para facilitar o desenvolvimento de aplicativos que interagem com Contratos Inteligentes, permitindo aos desenvolvedores acessar informações na *Blockchain* e realizar transações.

Como aponta o pwa.ir (2023), o publicador da biblioteca, O Web3dart fornece funcionalidades como criação de transações, leitura de informações de Contratos Inteligentes, assinatura de transações, e várias outras operações comuns em aplicativos que interagem com redes *Blockchain* baseadas na Ethereum.

3.2.3 *Truffle Suite*

O *Truffle Suite* é uma estrutura de desenvolvimento que possui um conjunto de ferramentas para a construção de DApps no *Blockchain* Ethereum. Ele oferece uma gama de ferramentas que agilizam todo o processo de desenvolvimento, desde a criação e teste de Contratos Inteligentes até a implantação e gerenciamento de ativos. Neste projeto, foram utilizados o Truffle e o Ganache, que são explicados a seguir.

3.2.3.1 Truffle

O Truffle (2023) é uma ferramenta de desenvolvimento de Contratos Inteligentes que, de acordo com eles, é um ambiente de desenvolvimento de classe mundial, com estruturas de testes e pipelines de ativos para *Blockchains* usando a *Ethereum Virtual Machine (EVM)*, com o objetivo de facilitar a vida dos desenvolvedores. Com as seguintes possibilidades:

- Compilação de Contrato Inteligente integrada, vinculação, implantação e gerenciamento binário.
- Depuração avançada com pontos de interrupção, análise de variáveis e funcionalidade de etapas.
- Usar o console.log em seus Contratos Inteligentes
- Implantações e transações através do MetaMask com Truffle *Dashboard* para proteger seu mnemônico.
- Executor de script externo que executa scripts dentro de um ambiente Truffle.
- Console interativo para comunicação direta por contrato.
- Testes automatizados de contrato para rápido desenvolvimento.

- Estrutura de implantação e migrações extensível e programável.
- Gerenciamento de rede para implantação em qualquer número de redes públicas e privadas.
- Gerenciamento de pacotes com NPM, utilizando o padrão ERC190.
- *Pipeline* de construção configurável com suporte para integração total.

3.2.3.2 Ganache

O Ganache (2023) é uma *Blockchain* de desenvolvimento local que, de acordo com eles, é um *Blockchain* pessoal para o rápido desenvolvimento de aplicativos distribuídos Ethereum e Filecoin, pois é possível utilizar o Ganache em todo o ciclo de desenvolvimento; permitindo o desenvolvimento, implantação e testes dos DApps em um ambiente seguro e determinístico. Além disso, o Ganache também oferece:

- Console.log em Solidity.
- *Forking* da *Mainnet* e *Testnet* sem configuração.
- *Fork* qualquer rede Ethereum sem esperar para sincronizar.
- Suporte a Ethereum JSON-RPC.
- *Snapshot*/reverter estado.
- Minere blocos instantaneamente, sob demanda ou em um intervalo.
- Tempo de avanço rápido.
- Personifique qualquer conta (não são necessárias chaves privadas).
- Escuta solicitações JSON-RPC 2.0 por HTTP/*WebSockets*.
- Uso programático no Node.js.
- Transações pendentes.

3.3 Aplicação

Segundo Sendin e Miani (2021) transformar um conceito de vacina em um produto de vacina real é um processo complexo que envolve a identificação de antígenos apropriados e a superação de obstáculos regulatórios, técnicos e de fabricação. Uma questão crítica nesse processo é a condução dos ensaios clínicos. A monitoração e garantia da integridade dos dados dos ensaios, utilizando os sistemas tradicionais, nem sempre são viáveis. Isso se

torna evidente na busca por uma vacina contra o coronavírus SARS-CoV-2, ilustrando a dificuldade em assegurar a credibilidade científica dos resultados dos ensaios clínicos de várias vacinas e como isso afeta as percepções sobre os benefícios e riscos do medicamento. Neste contexto, a aplicação de tecnologias como *Blockchain* e Contratos Inteligentes no âmbito da saúde torna-se uma opção altamente promissora.

3.3.1 VaccSC

Sendin e Miani (2021) apresentam o VaccSC, um protocolo fundamentado em Contrato Inteligente, com o propósito de fomentar a transparência, a integridade e a confidencialidade nos experimentos de Fase III voltados para vacinas. Desenvolvido na linguagem Solidity, os resultados obtidos evidenciam a capacidade do VaccSC em permitir a condução de estudos duplo-cegos e randomizados, assim como a auditabilidade dos dados clínicos, mesmo em cenários nos quais participantes desonestos estão presentes.

3.3.2 Implantação do contrato

No momento da implantação do Contrato Inteligente, o Administrador deve informar:

Número de participantes: Correspondente ao número de doses disponíveis.

Limiar de Infectados: Número de pacientes que precisam ser infectados antes que os valores comprometidos sejam revelados e a eficiência da vacina seja calculada.

Eficiência alvo: VE mínimo necessário para a aprovação da vacina.

Clínicas de Vacinação: Endereço Ethereum das clínicas que aplicarão as vacinas e atualizarão o VaccSC.

Tabela 1 – Estrutura de dados utilizada pelo VaccSC para manter informações dos testes das vacinas.

Informação	**Tipo**	**Acesso e Controle**
Commit	hash	Imutável, criado pelo Administrador
Clinic	Address	Cada clínica é adicionada pelo Administrador
Patient	Address	Clínica e paciente
gotSick	Boolean	Paciente
Vaccine Type	Vaccine/Placebo	VaccSC preenche o campo com a revelação fornecida pelo Administrador

Fonte: adaptado de Sendin e Miani (2021)

A estrutura de dados que armazena as informações necessárias para calcular a eficiência vacinal é apresentada na Tabela 1. Os campos são apresentados em ordem cronológica em que são alterados durante a execução do contrato.

3.3.3 Participantes

O protocolo modela a participação de três tipos de entidades, sendo elas:

Administrador: Responsável pela distribuição das vacinas as clinicas e proteção das informações sobre seu conteúdo. Ele também é responsável pela implantação do contrato, sendo o único participante que consegue distinguir a vacina real da controle, criar os *commits* das vacinas, que serão utilizadas como identificadores no contrato.

Clinica: Responsável por aplicar as injeções e atualizar o VaccSC, realizando a ligação da injeção com um paciente específico.

Paciente: Recebe a injeção e avisa o contrato quando ficar doente.

3.3.4 Atribuição aleatória de paciente/injeção

A associação aleatória entre a injeção e o paciente evita preconceitos. A geração de números aleatórios no ambiente SC é complicada e deve ser feita com cuidado. Neste protocolo, eles optaram por utilizar a abordagem em que cada participante gera um número aleatório (R1 e R2). O SC combina esses números usando uma operação *XOR* bit a bit. Com essa abordagem, basta uma parte ser honesta para que o número seja aleatório.

Figura 3 – Etapas da atribuição aleatória de paciente/injeção.

Fonte: Adaptado de Sendin e Miani (2021).

Na figura 3 cada participante gera de forma privada um número aleatório e confirma esse número. Após ambos os números estarem comprometidos com o contrato, cada participante revela o valor (etapas 3 e 4). O SC combina esses números – R1 *XOR* R2 – para associar uma das injeções disponíveis ao paciente.

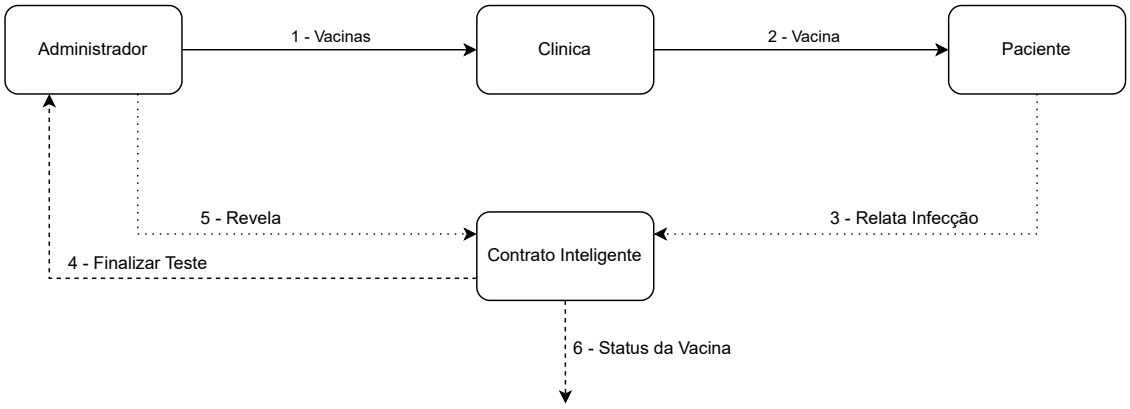
3.3.5 Duplo-cego

Como apontado por Sendin e Miani (2021), a Clínica de Vacinas não sabe o que está aplicando, assim como o paciente não sabe se receberá a vacina ou a injeção de controle. Inicialmente, o Administrador da vacina protege as informações sobre o conteúdo de cada injeção que fará parte da Fase III e posteriormente ficará protegido utilizando o compromisso 2.1.2 e a atribuição aleatória 3.3.4.

3.3.6 Execução

Depois que o Administrador implantou o VaccSC, o protocolo segue as seguintes etapas:

Figura 4 – Eventos de protocolo associados a uma injeção. A etapa quatro ocorre apenas uma vez. As linhas sólidas indicam eventos físicos com atualizações de contrato e as linhas tracejadas indicam interações VaccSC.



Fonte: Adaptado de Sendin e Miani (2021)

1. O Administrador entrega as vacinas em cada Clínica de Vacinação e faz a associação entre as vacinas e as clínicas do contrato.

2. A clínica aplica a injeção e associa o identificador da injeção a um paciente específico. O paciente confirma isso para o contrato.

3. Se o paciente adoecer, ele comunica o fato ao contrato.

4. Quando o limite de infecção é atingido, o contrato cria um evento informando o Administrador da Vacina.

5. Usando os endereços Ethereum dos pacientes infectados, o Administrador seleciona os *commits*. correspondentes. Depois disso, o Administrador envia para o contrato as informações dos pacientes que receberam o controle. O contrato calcula o número de pacientes que receberam a vacina.

6. A eficiência da vacina pode ser determinada, e seu status de aprovação pode ser obtido diretamente no contrato por qualquer pessoa

3.3.7 Arquitetura

O DApp possui 2 camadas, uma camada sendo o aplicativo *front-end* do lado do cliente e a outra sendo a camada *back-end* do Contrato Inteligente que é implantado na

rede *Blockchain*. A figura 5 mostra a arquitetura geral do DApp e a interação entre um cliente e o servidor.

Neste projeto, o protótipo foi feito utilizando o *Truffle Suite* 3.2.3, para implantar o contrato e servir como o lado do servidor, o framework Flutter 3.2.2, para fazer o aplicativo *front-end* para o cliente, e o pacote Web3Dart 3.2.2.1 para fazer a conexão entre o *front-end* e o *back-end*.

3.3.8 Aplicativos

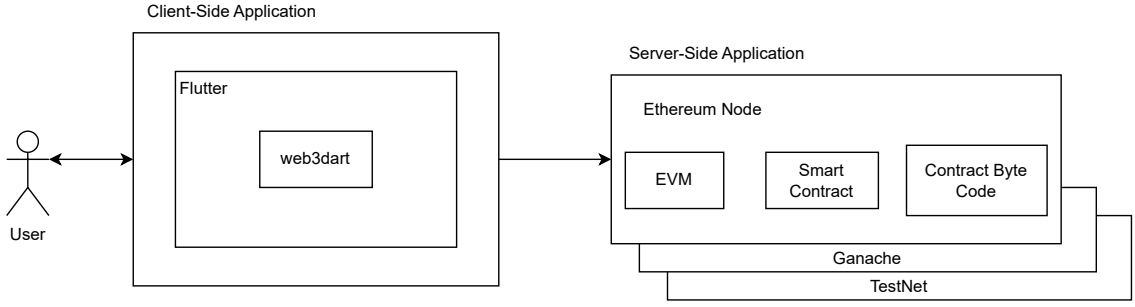
Este projeto foi dividido em três aplicativos distintos: o aplicativo do Administrador, cujas telas estão descritas na Tabela 2; o aplicativo da Clínica, com as telas listadas na Tabela 3; e o aplicativo do Paciente, com as telas apresentadas na Tabela 4. Além disso, os diagramas de sequência podem ser observados nas seguintes figuras: o diagrama do Administrador na Figura 6, o diagrama da Clínica na Figura 7 e o diagrama do Paciente na Figura 8.

Tabela 2 – Tabela que informa sobre as telas e funções do aplicativo do Administrador

Administrador	
Tela	**Função**
Revelar Aprovação	Finalizar os testes e verificar se a vacina foi aprovada ou não.
Revelar Controle	Revelar que injeção não é vacina, mas controle/placebo.
Adicionar Vacina	Adicionar injeções
Adicionar Clinica	Adicionar clinicas.

Fonte: Elaborado pelo autor (2024)

Figura 5 – Arquitetura do DApp.



Fonte: Elaborado pelo autor (2024)

Tabela 3 – Tabela que informa sobre as telas e funções do aplicativo da clinica

Clinica	
Tela	**Função**
Vacinar	Finalizar o processo de vacinação atribuindo um paciente a uma injeção.
Revelar Moeda	Revelar o valor escolhido pela clinica para acabar com o duplo-cego.
Iniciar Vacinação	Iniciar a vacinação de um paciente escolhendo um valor para utilizar no duplo-cego.

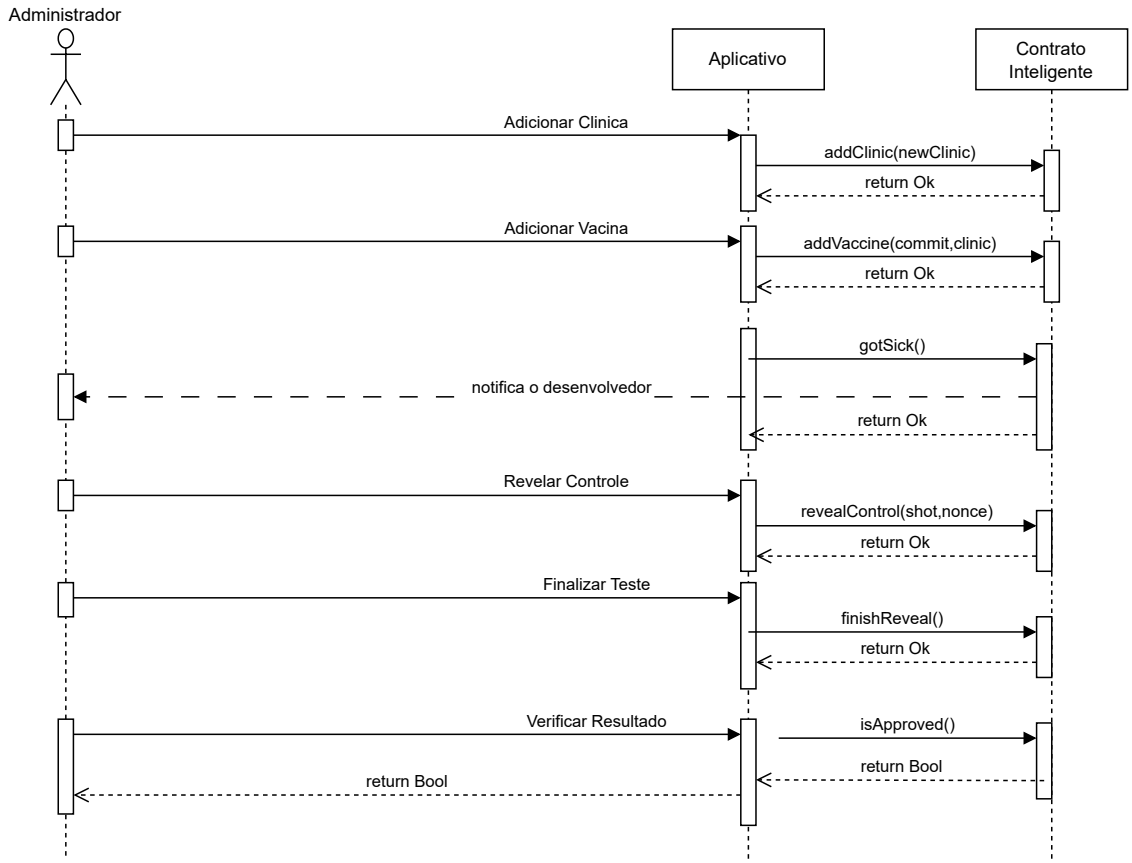
Fonte: Elaborado pelo autor (2024)

Tabela 4 – Tabela que informa sobre as telas e funções do aplicativo do Paciente

Paciente	
Tela	**Função**
Iniciar Vacinação	Iniciar a vacinação, onde paciente escolhe um valor para utilizar no duplo-cego.
Revelar Moeda	Revelar o valor escolhido pelo paciente para acabar com o duplo-cego.
Relatar Infecção	Paciente relatar infecção.
Relatar Vacinação	Paciente confirmar que recebeu a injeção.

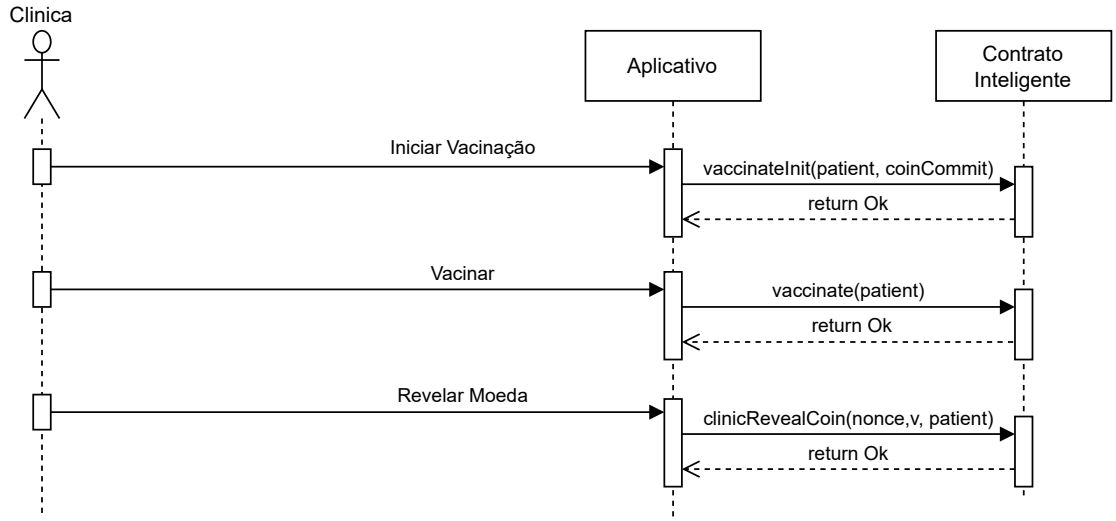
Fonte: Elaborado pelo autor (2024)

Figura 6 – Diagrama de sequencia do Administrador.



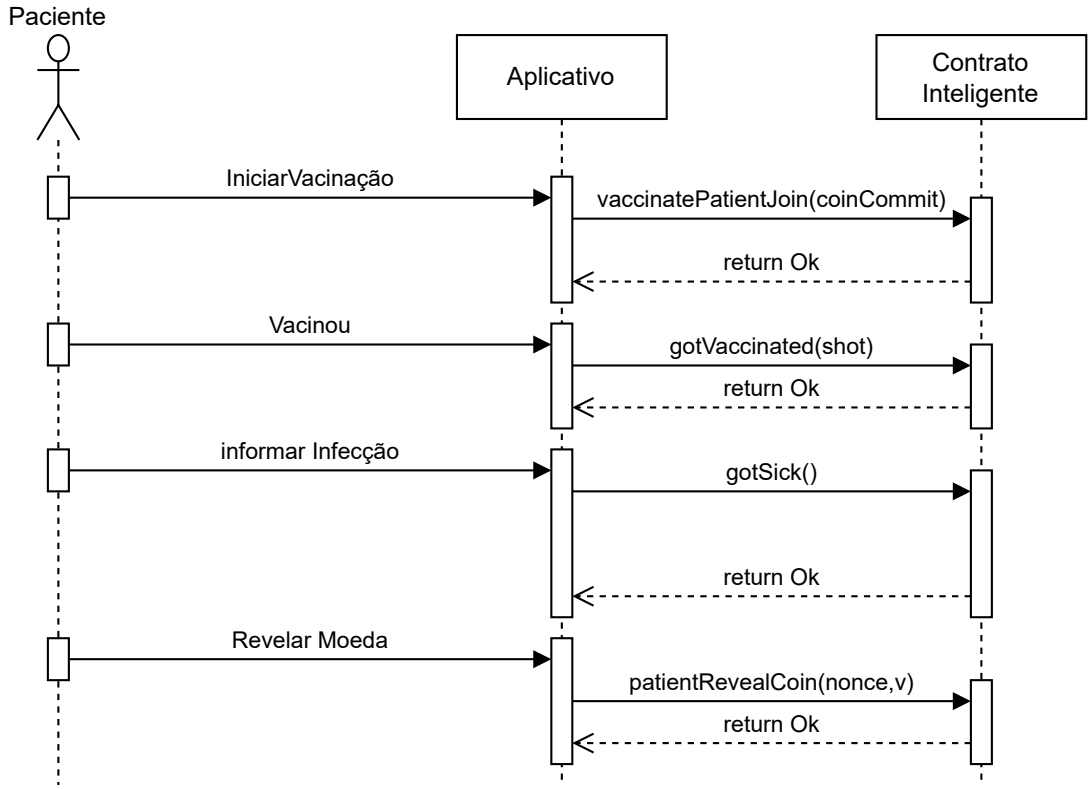
Fonte: Elaborado pelo autor (2024)

Figura 7 – Diagrama de sequencia da Clinica.



Fonte: Elaborado pelo autor (2024)

Figura 8 – Diagrama de sequencia do Paciente.



Fonte: Elaborado pelo autor (2024)

4 Resultados

Neste capítulo, detalhamos os resultados obtidos durante o desenvolvimento das Dapps utilizando o VaccSC. Apresentando as informações sobre o *front-end* que foi desenvolvido e os testes realizados, incluindo a criação de um *workspace*, *deploy* do contrato e um ciclo de utilização dos DApps.

4.1 *front-end*

No desenvolvimento do *front-end*, a escolha do Dart e do Flutter mostra-se vantajosa devido à eficiência e produtividade que esses *frameworks* oferecem. O Dart é uma linguagem de programação moderna e orientada a objetos que possui uma sintaxe clara e concisa, facilitando a escrita e manutenção do código. O Flutter, por sua vez, é um *framework* de UI desenvolvido pelo Google, que permite a criação de interfaces de usuário responsivas para Android, iOS e páginas da web a partir de um único código-fonte. A combinação de Dart e Flutter agiliza o processo de desenvolvimento, reduzindo custos e tempos de entrega.

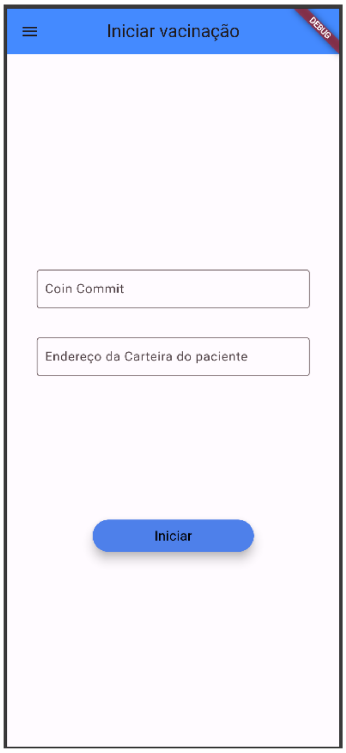
Adicionalmente, a utilização do pacote web3dart para a interação com Contratos Inteligentes na *Blockchain* Ethereum é essencial para a integração do *front-end* com a camada de *Blockchain*. O web3dart é uma biblioteca que permite a comunicação entre o aplicativo Flutter e a *Blockchain* Ethereum, facilitando a execução de transações e a chamada de métodos de Contratos Inteligentes diretamente do aplicativo. Isso é crucial para garantir que o DApp possa interagir de forma eficiente e segura com a *Blockchain*, proporcionando uma experiência de usuário fluida e confiável.

A escolha dessas tecnologias também é justificada pela comunidade de desenvolvedores e a documentação disponível, que oferecem suporte contínuo e recursos para solucionar problemas e otimizar o desenvolvimento. Essas vantagens combinadas demonstram a eficácia e a pertinência do uso de Dart, Flutter e web3dart no desenvolvimento do *front-end* do DApp.

4.1.1 Temas e modelos

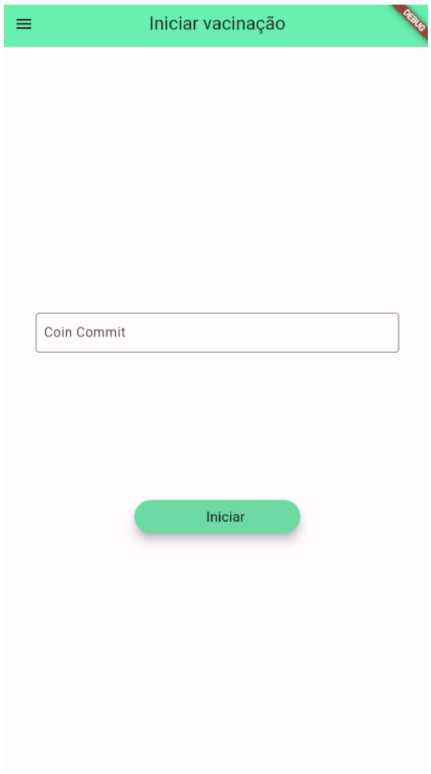
Como resultado do desenvolvimento do *front-end*, foram criados 3 DApps que utilizam o VaccSC. Na versão do administrador, foi adotado o tema roxo, conforme ilustrado na Figura 12. Para a versão da clínica, o tema escolhido foi azul, sendo uma de suas telas exibida na Figura 9. Por fim, a versão destinada aos pacientes utiliza a cor verde-água como tema, com uma de suas telas apresentada na Figura 10. As telas das aplicações seguiram os seguintes modelos: uma tela com um único campo, como ilustrado na Figura 10; uma tela com múltiplos campos, conforme mostrado na Figura 9; uma tela sem campos, contendo apenas um botão, representada na Figura 11; e uma tela sem campos, mas com texto informativo e um botão, como apresentado na Figura 12.

Figura 9 – vários campo



Fonte: Elaborado pelo Autor (2024)

Figura 10 – um campo

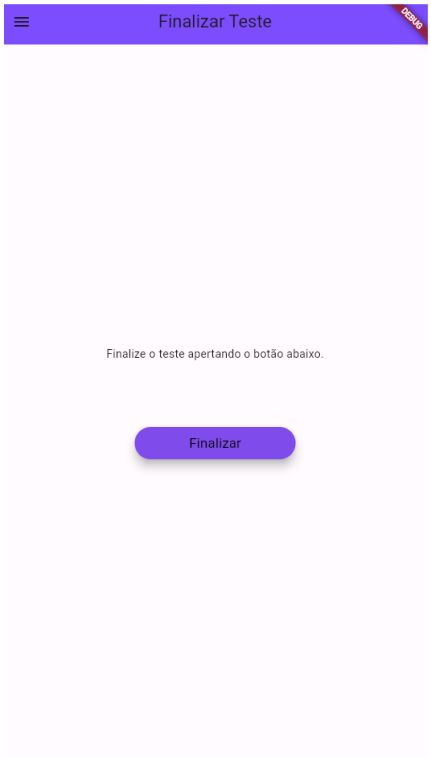


Fonte: Elaborado pelo Autor (2024)

Figura 11 – sem campos

Fonte: Elaborado pelo Autor (2024)

Figura 12 – sem campos com texto



Fonte: Elaborado pelo Autor (2024)

4.2 Testes

Para o desenvolvimento e teste, a escolha do Truffle e do Ganache, ambos componentes do *Truffle Suite*, foi fundamental devido a várias razões. Primeiramente, o Ganache permite criar um ambiente de teste local, oferecendo uma *Blockchain* pessoal que possibilita o desenvolvimento e teste de maneira rápida e segura. Essa característica elimina a necessidade de utilizar redes públicas para testes iniciais, reduzindo custos e riscos associados. Além disso, a capacidade de depuração oferecida pelo Ganache, com uma interface que facilita a visualização e correção de transações e Contratos Inteligentes, é essencial para identificar e resolver erros durante o desenvolvimento.

O Truffle, por sua vez, inclui uma série de ferramentas que agilizam o desenvolvimento, como migrações de contratos, *deploy* automatizado e gerenciamento de dependências, garantindo um fluxo mais eficiente. A robustez dessas ferramentas permite uma integração contínua e entrega mais suave e eficiente. Adicionalmente, o *Truffle Suite* possui uma comunidade ativa e uma documentação extensa, proporcionando um suporte contínuo e facilitando a solução de problemas ao longo do desenvolvimento. Esses fatores combinados tornam o Truffle e o Ganache escolhas ideais para o desenvolvimento e teste de DApps, demonstrando a eficácia e a praticidade dessas ferramentas.

4.2.1 *Workspace*

Utilizando o Truffle e o Ganache do *Truffle Suite* 3.2.3, foi criado um *workspace* para a realização dos testes do protótipo do aplicativo, conforme ilustrado na Figura 14. Após a criação do *workspace*, foi feito o *deploy* do Contrato Inteligente para os testes, baseando-se na seguinte premissa: uma vacina precisava atingir uma eficácia mínima de 50% em uma clínica com 20 participantes. Desses, 10 receberam a vacina e 10 receberam placebos. O critério para calcular a eficácia era que, no máximo, 13 participantes poderiam ser infectados e que 5 ou mais participantes que receberam a vacina não deveriam estar infectados. O armazenamento do contrato criado com os valores abordados pode ser observado na Figura 15, e a transação pode ser vista na Figura 13.

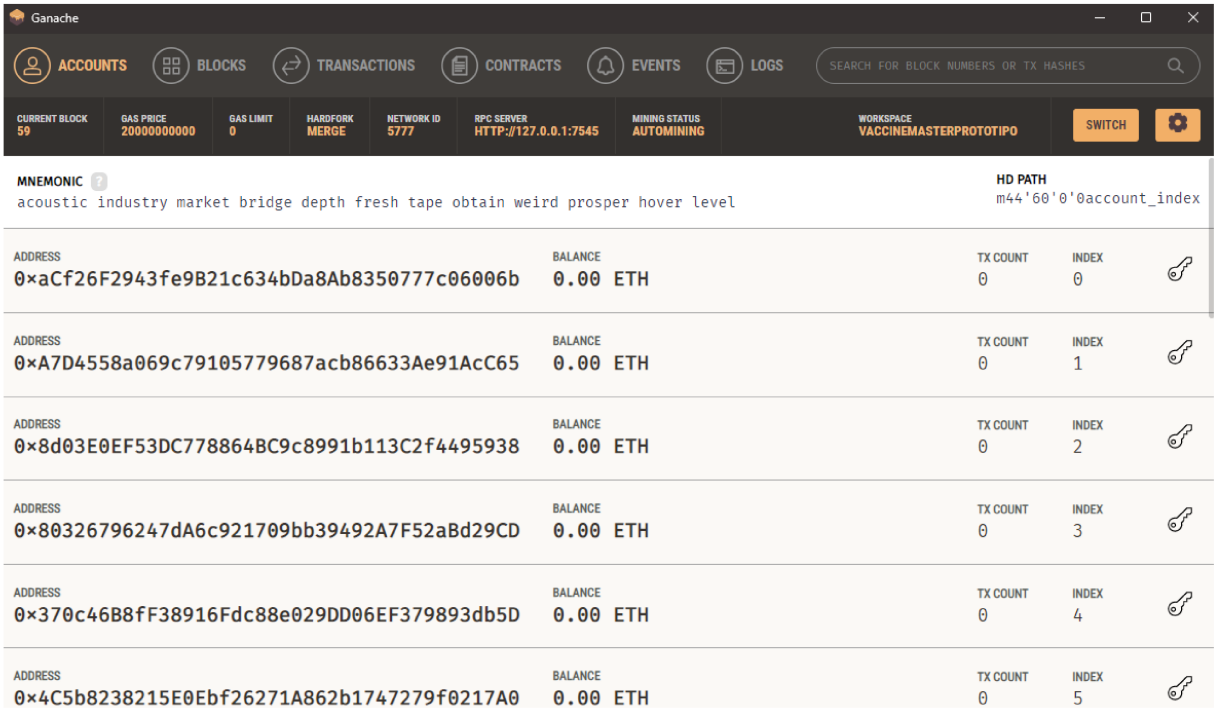
No *Workspace* criado utilizando o Ganache, na aba de *accounts*, podemos observar as carteiras criadas na rede. Nesta aba, o campo *'address'* exibe o endereço da carteira; o campo *'balance'* mostra a quantidade de Ethereum na carteira; o campo *'tx count'* indica o número de transações realizadas pela carteira; e o campo *'index'* apresenta o índice da carteira. Além disso, há um ícone de chave que pode ser clicado para obter a chave privada da carteira.

As transações que ocorreram na rede podem ser observadas na aba *transactions* do *workspace*. Nessa seção, podemos visualizar o hash da transação em negrito, o endereço da carteira que fez a transação no campo *'Sender Address'*, o endereço do contrato que recebeu a transação no campo *'to contract address'*, o valor da transação em *'value'*, o gás utilizado para a transação em *'gas used'*, o preço do gás em *'gas price'*, o limite de gás no campo *'gas limit'*, o número do bloco no qual a transação foi minerada em *'mined in block'*, e os dados enviados como parte da transação no campo *'tx data'*. Além disso, se houver dados enviados na transação, eles aparecerão abaixo da parte da transação com o termo *'Contract'* no início. Na seção de dados, teremos o nome do contrato no campo *'Contract'*, o endereço do contrato no campo *'address'*, a função utilizada no campo *'function'*, e os parâmetros enviados para essa função no campo *'inputs'*.

Figura 13 – Transação utilizada para a criação do contrato

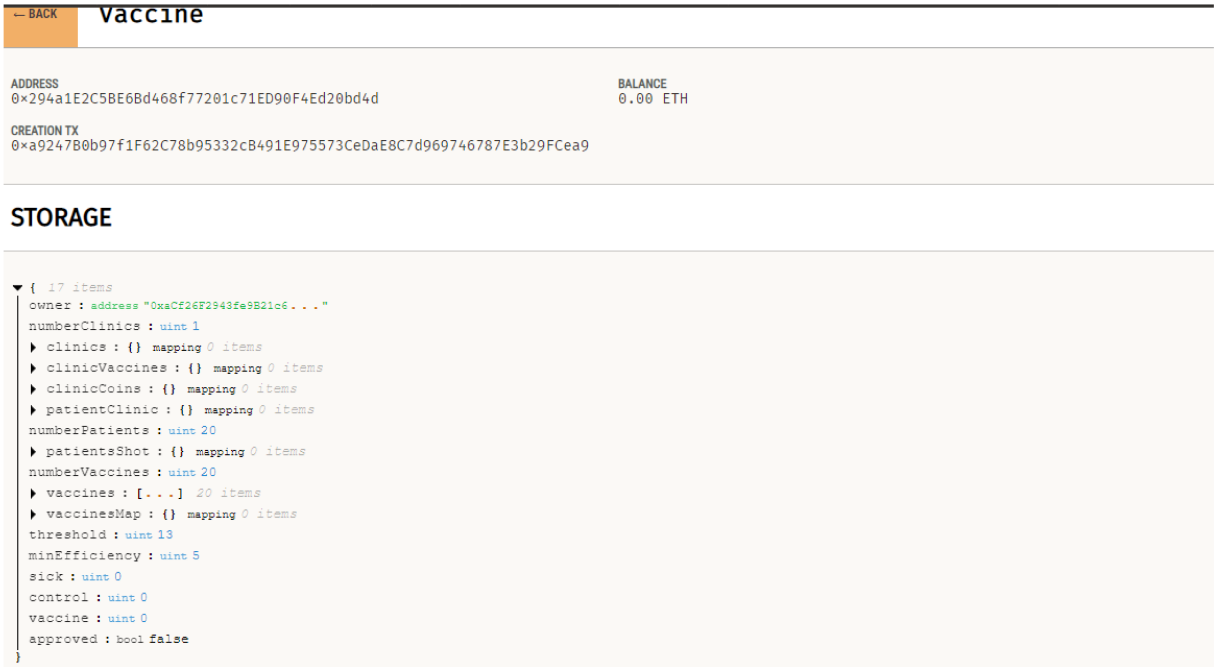
Fonte: Elaborado pelo Autor (2024)

Figura 14 – *Workspace* criado no Ganache para realização dos testes



Fonte: Elaborado pelo Autor (2024)

Figura 15 – *Storage* no Ganache do contrato



Fonte: Elaborado pelo Autor (2024)

4.2.2 Testes

Após a criação do *workspace* e o *deploy* do contrato, utilizando o aplicativo do administrador, a clínica foi adicionada utilizando o endereço da carteira dela. A transação utilizada para isso pode ser vista na Figura 16. Além da adição da clínica, também foram adicionadas as 20 injeções que serão utilizadas nos pacientes. Onde é necessário informar o endereço da clínica para a qual as injeções serão enviadas e um identificador para cada injeção. Uma transação do teste para adicionar uma injeção pode ser observada na Figura 17.

Figura 16 – Transação adicionar clinica

Fonte: Elaborado pelo Autor (2024)

Figura 17 – Transação da adicionar injeção

Fonte: Elaborado pelo Autor (2024)

Após a adição da clínica e das injeções, a vacinação dos participantes pode ser iniciada utilizando o aplicativo da clínica. Na tela de iniciar vacinação da clínica, para iniciar a vacinação é necessário inserir um número para o duplo-cego, denominado "moeda", e o endereço da carteira do paciente (um exemplo de transação na Figura 18). Simultaneamente, o participante deve iniciar o procedimento no aplicativo correspondente, na tela de iniciar vacinação onde o participante precisa apenas informar o número para o duplo-cego, também denominado "moeda" (um exemplo de transação na Figura 19).

Figura 18 – Transação iniciar vacinação(Clinica)

Fonte: Elaborado pelo Autor (2024)

Figura 19 – Transação iniciar vacinação(Paciente)

Fonte: Elaborado pelo Autor (2024)

Após iniciar o processo de vacinação nos dois aplicativos, no aplicativo da clínica, na tela de vacinação, é necessário confirmar a vacinação do paciente informando o endereço da carteira dele (uma transação do teste na Figura 20). Simultaneamente, no aplicativo do paciente, é preciso registrar qual injeção foi recebida na tela de relato de vacinação (uma transação do teste na Figura 21).

Figura 20 – Transação vacinar

Fonte: Elaborado pelo Autor (2024)

Figura 21 – Transação relatar vacinação

Fonte: Elaborado pelo Autor (2024)

Após iniciar e concluir o processo de vacinação com todos os pacientes, é necessário que todos informem se forem infectados durante o período de teste por meio da tela de relato de infecção (uma transação do teste na Figura 22), bastando apenas pressionar o botão na tela. No teste realizado, 13 participantes foram escolhidos para serem infectados.

Figura 22 – Transação relatar infeção

Fonte: Elaborado pelo Autor (2024)

Na etapa de revelação, o paciente deve revelar sua moeda do duplo-cego no aplicativo, na tela de revelação de moeda (uma transação do teste na Figura 23). Simultaneamente, a clínica deve informar sua moeda na tela de revelação de moeda do seu aplicativo (uma transação do teste na Figura 24). Enquanto isso, o administrador deve utilizar a tela de revelação de controle para revelar quais injeções são de controle, informando o *nonce* e a injeção que são de controle (uma transação do teste na Figura 25). No teste realizado, dos 13 participantes infectados, 9 foram com injeções de controle e 4 com a vacina.

Figura 23 – Transação Revelar Moeda(Paciente)

Fonte: Elaborado pelo Autor (2024)

Figura 24 – Transação Revelar Moeda(Clinica)

Fonte: Elaborado pelo Autor (2024)

Figura 25 – Transação revelar controle

Fonte: Elaborado pelo Autor (2024)

Após a revelação das informações, o administrador deve finalizar o teste para verificar o resultado. Primeiro, ele deve concluir o teste na tela "Finalizar Teste" pressionando o botão na tela (um exemplo de transação do teste na figura 26) e, em seguida, verificar o resultado na tela "Resultado" pressionando o botão e depois esperando a tela mostrar o resultado. Dependendo do resultado, a tela pode informar se a vacina foi aprovada ou reprovada. No teste realizado, como o *threshold* é de 13 infectados e são 9 com injeções de controle, ou seja, apenas 4 participantes com a vacina foram infectados dos 10 participantes com vacina e precisava de uma eficácia mínima de 50%, a vacina foi aprovada.

Figura 26 – Transação Finalizar Teste

Fonte: Elaborado pelo Autor (2024)

5 Conclusões

Os Contratos Inteligentes surgem como uma solução inovadora para mediar cenários onde a honestidade é essencial, mas onde os participantes podem estar inclinados a agir de forma desonesta. Esses contratos, que são programas autoexecutáveis registrados em uma *Blockchain*, garantem que os termos acordados entre as partes sejam automaticamente aplicados sem a necessidade de intermediários. Ao codificar regras e condições específicas que são executadas automaticamente quando os critérios são atendidos, os Contratos Inteligentes eliminam a necessidade de confiança direta entre as partes, pois a execução é garantida pela tecnologia e não pela boa vontade dos participantes. Isso reduz o risco de manipulação e fraude, criando um ambiente mais seguro e previsível para todas as partes envolvidas.

Além disso, a transparência e a imutabilidade dos Contratos Inteligentes fortalecem ainda mais a confiança no sistema. Cada transação e alteração é registrada de forma permanente na *Blockchain*, tornando qualquer tentativa de fraude facilmente rastreável e praticamente impossível de alterar sem o consenso da rede. Isso assegura que todos os participantes cumpram suas obrigações conforme o acordado, já que qualquer tentativa de manipulação seria prontamente detectada. Portanto, em cenários onde a honestidade é crucial e onde há potencial para comportamentos desonestos, os Contratos Inteligentes oferecem uma camada adicional de segurança e controle, garantindo que os compromissos sejam cumpridos e protegendo os interesses de todos os envolvidos.

O VaccSC é um protocolo baseado em Contratos Inteligentes que traz transparência e confiabilidade à Fase III dos experimentos de desenvolvimento de vacinas. Este protocolo mantém a propriedade duplo-cega, além de impedir que informações relevantes sejam ocultadas pelo desenvolvedor das vacinas. Utilizando Esquemas de Compromisso, o VaccSC permite auditoria pública dos dados, garantindo que qualquer pessoa possa verificar a integridade dos testes. Além disso, o protocolo pode ser adaptado para incluir observadores independentes, reforçando ainda mais a transparência. Capturando a essência dos Contratos Inteligentes - “Código é lei” - e se posicionando como um elemento decisivo na aprovação de vacinas.

Para facilitar o uso dos Contratos Inteligentes pelos participantes da Fase III de testes de vacinas, foram utilizadas a linguagem Dart, o *framework* Flutter e o pacote Web3Dart. Com essas ferramentas, foram desenvolvidos três programas *front-end* com interfaces intuitivas, cada um destinado a um grupo específico de usuários: um para os desenvolvedores de vacinas, outro para as clínicas e um terceiro para os pacientes. Esses programas, em conjunto com o Contrato Inteligente desenvolvido por Sendin e Miani,

visam simplificar e agilizar a Fase III dos testes de vacinas, garantindo maior eficiência e acessibilidade para o público em geral. Além disso, a utilização dessas tecnologias permite uma integração mais fluida e segura dos dados, promovendo uma experiência mais transparente e confiável para todos os envolvidos no processo.

Testes foram realizados utilizando o Truffle e o Ganache do *Truffle Suite* para criar um *workspace* e fazer o *deploy* do Contrato Inteligente, permitindo que os programas desenvolvidos pudessem utilizá-lo. No teste de um ciclo completo, foi adotada a seguinte premissa: uma vacina precisava atingir uma eficácia mínima de 50% em uma clínica com 20 participantes. Desses, 10 receberam a vacina e 10 receberam placebos.

Como resultado, foi possível perceber a facilidade e rapidez de se utilizar os aplicativos. Em todas as versões, os participantes puderam exercer suas funções com poucos toques, muitas vezes apenas preenchendo campos de texto simples. Isso eliminou a necessidade de entender e interagir diretamente com o Contrato Inteligente, tornando o processo mais intuitivo e acessível para todos os usuários.

Diante de tudo que foi discutido durante este trabalho, conclui-se que os aplicativos, em conjunto com o Contrato Inteligente, simplificam e aceleram a fase III dos testes de vacinas. Além de trazer transparência e confiabilidade ao processo, esses recursos tecnológicos asseguram que os participantes cumpram suas obrigações de maneira eficiente. A automação proporcionada pelos Contratos Inteligentes reduz a necessidade de intervenção manual, minimizando erros e aumentando a precisão dos dados coletados. Isso não só acelera o desenvolvimento das vacinas, mas também fortalece a confiança das partes interessadas no processo.

5.1 Trabalhos futuros

5.1.1 Generalização

Atualmente, o Contrato Inteligente e os aplicativos são projetados para auxiliar especificamente na Fase III dos testes de vacinas. No entanto, é possível realizar modificações tanto no Contrato Inteligente quanto nos aplicativos para expandir seu uso e oferecer suporte tanto a outras fases quanto a testes de outros tipos de medicamentos. Essas adaptações podem incluir ajustes nos parâmetros de teste, integração de novos protocolos de segurança e a inclusão de funcionalidades adicionais para monitoramento e auditoria, garantindo assim a transparência e a confiabilidade em uma variedade de contextos clínicos.

5.1.2 Tela de Login

Atualmente, todos os protótipos do projeto utilizam a chave privada da carteira para autenticação. Embora apenas a utilizem para o login e não a armazenem no aplicativo, essa abordagem ainda apresenta riscos significativos de segurança. Para melhorar a proteção dos usuários, o futuro desenvolvimento deve eliminar o uso direto da chave privada para autenticação. Em vez disso, deve-se implementar um sistema de login que integre com aplicativos externos, como a MetaMask, para validar as transações dos usuários. Esse método assegura que a segurança da carteira do usuário não seja comprometida, mantendo a integridade das informações e reduzindo o risco de possíveis ataques.

Referências

ALBERTO, M. Alura, 2023. Acessado em 17/04/2024. Disponível em: <Flutter: oqueÃĺetudosobreoframework>. Citado na página 23.

Amazon Web Services. **O que é a tecnologia blockchain?** 2024. Acessado em 17/04/2024. Disponível em: <https://aws.amazon.com/pt/what-is/blockchain/?aws-products-all.sort-by=item.additionalFields.productNameLowercase&aws-products-all.sort-order=asc>. Citado na página 14.

_____. **O que é Web3?** 2024. Acesso em: 16 set. 2024. Disponível em: <https://aws.amazon.com/pt/what-is/web3/>. Citado na página 14.

ANTONOPOULOS, A. M.; WOOD, G. **Mastering Ethereum: Building Smart Contracts and DApps**. O'Reilly Media, Incorporated, 2018. ISBN 9781491971949. Disponível em: <https://github.com/ethereumbook/ethereumbook>. Citado 2 vezes nas páginas 9 e 16.

AUDIUS. **Audius**. S.I. Disponível em: <https://audius.co/>. Citado na página 19.

AXIE-INFINITY. **Axie Infinity**. S.I. Disponível em: <https://axieinfinity.com/>. Citado na página 18.

BINANCEACADEMY. **O que são os Aplicativos Descentralizados (DApps)?** 2023. Acessado em 16/05/2024. Disponível em: <https://academy.binance.com/pt/articles/what-are-decentralized-applications-dapps>. Citado 2 vezes nas páginas 17 e 19.

BLUM, M. Coin flipping by telephone. In: **Advances in Cryptology: A Report on CRYPTO 81**. [s.n.], 1981. p. 11–15. Acesso em: 16 set. 2024. Disponível em: <https://www.cs.cmu.edu/~mblum/research/pdf/coin/>. Citado na página 13.

CAI, W.; WANG, Z.; ERNST, J. B.; HONG, Z.; FENG, C.; LEUNG, V. C. M. Decentralized applications: The blockchain-empowered software system. **IEEE Access**, v. 6, p. 53019–53033, 2018. Citado na página 20.

COINEXT. **O que é Audius (AUDIO) e como funciona essa criptomoeda?** S.I. Acessado em 16/05/2024. Disponível em: <https://coinext.com.br/criptomoedas/audius>. Citado na página 19.

DART. **Dart**. 2023. Acessado em 24/08/2023. Disponível em: <https://dart.dev/overview>. Citado na página 23.

ETHEREUM. **Ethereum Development Documentation**. 2022. Disponível em: <https://ethereum.org/pt/developers/docs/>. Citado 3 vezes nas páginas 15, 16 e 17.

FOSTER, N.; BERLSTEIN, G.; WATERMAN, J.; BIRRELL, E.; BAGDASARYAN, E.; SCHNEIDER, F. B.; ESTRIN, D. **Ancile: Enhancing Privacy for Ubiquitous Computing with Use-Based Privacy**. 2019. WPES. Acesso em: 09 out. 2024. Disponível em: <https://ancile-project.github.io>. Citado na página 21.

GALLAGHER, J. **10 anos em 10 meses: como cientistas de Oxford criaram em tempo recorde um novo modelo de vacina contra o coronavírus**. 2020. Accessed: 2024-08-18. Disponível em: <https://www.bbc.com/portuguese/internacional-55049893>. Citado na página 8.

GAMMA, E. **Padrões de Projetos: Soluções Reutilizáveis**. Bookman, 2009. ISBN 9788577800469. Disponível em: <https://books.google.com.br/books?id=U91CYCqTCgkC>. Citado na página 16.

GANACHE. **Ganache**. 2023. Acessado em 24/08/2023. Disponível em: <https://trufflesuite.com/docs/ganache/>. Citado na página 25.

HERRERA, P. Dappradar x bga games report – q1 2022. 2022. Acesso em: 16 set. 2024. Disponível em: <https://dappradar.com/blog/dappradar-x-bga-games-report-q1-2022>. Citado na página 18.

International Business Machines Corporation. **O que é Blockchain?** 2024. Acessado em: 18 de agosto de 2024. Disponível em: <https://www.ibm.com/br-pt/topics/blockchain>. Citado 2 vezes nas páginas 8 e 14.

_____. **O que é criptografia?** 2024. Acessado em: 05 de setembro de 2024. Disponível em: <https://www.ibm.com/br-pt/topics/cryptography>. Citado na página 12.

JOHNSON, M.; JONES, M.; SHERVEY, M.; DUDLEY, J. T.; ZIMMERMAN, N. Building a secure biomedical data sharing decentralized app (dapp): Tutorial. **J Med Internet Res**, v. 21, n. 10, p. e13601, Oct 2019. ISSN 1438-8871. Disponível em: <https://www.jmir.org/2019/10/e13601>. Citado na página 19.

MARCHESI, L.; MARCHESI, M.; TONELLI, R. Abcde—agile block chain dapp engineering. **Blockchain: Research and Applications**, v. 1, n. 1, p. 100002, 2020. ISSN 2096-7209. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2096720920300026>. Citado na página 20.

MENEZES, A. J.; OORSCHOT, P. C. V.; VANSTONE, S. A. **Handbook of Applied Cryptography**. [S.l.]: CRC Press, 1996. Citado 2 vezes nas páginas 12 e 13.

Organização Pan-Americana da Saúde. **Histórico da pandemia de COVID-19**. 2024. Acesso em: 5 set. 2024. Disponível em: <https://www.paho.org/pt/covid19/historico-da-pandemia-covid-19>. Citado na página 8.

PWA.IR. **web3dart**. 2023. Acessado em 01/11/2023. Disponível em: <https://pub.dev/packages/web3dart>. Citado na página 24.

RENU, S. A.; BANIK, B. G. Implementation of a secure ridesharing dapp using smart contracts on ethereum blockchain. **International Journal of Safety and Security Engineering**, International Information and Engineering Technology Association, v. 11, n. 2, p. 167–173, 2021. Citado na página 20.

RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. **Communications of the ACM**, ACM, 1978. Citado na página 13.

SENDIN, I.; MIANI, R. **Towards reliable and transparent vaccine phase III trials with smart contracts**. 2021. Acessado em 01/11/2023. Citado 5 vezes nas páginas 9, 25, 26, 27 e 28.

STALLINGS, W. **Cryptography and Network Security: Principles and Practice**. [S.l.]: Pearson, 2017. Citado na página 13.

SZABO, N. **Smart contracts**. 1994. Acessado em 15/04/2024. Disponível em: <https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html>. Citado na página 15.

TRUFFLE. **Truffle**. 2023. Acessado em 24/08/2023. Disponível em: <https://trufflesuite.com/docs/truffle/>. Citado na página 24.

UDOKWU, C.; ANYANKA, H.; NORTA, A. Evaluation of approaches for designing and developing decentralized applications. 06 2020. Citado na página 20.

World Health Organization. Como são as vacinas desenvolvidas? **World Health Organization**, 2020. Acessado em: 18 de agosto de 2024. Disponível em: <https://www.who.int/pt/news-room/feature-stories/detail/how-are-vaccines-developed>. Citado na página 8.

_____. **WHO Declares Public Health Emergency on Novel Coronavirus**. 2020. Accessed: 2024-08-18. Disponível em: <https://www.paho.org/pt/news/30-1-2020-who-declares-public-health-emergency-novel-coronavirus>. Citado na página 8.