

**FAETERJ-RIO – FACULDADE DE EDUCAÇÃO TECNOLÓGICA DO
ESTADO RIO DE JANEIRO**

GILMAR RIBEIRO SANTANA

**WORKBLOCK: SISTEMA DE CONTROLE DE PONTO
DE FUNCIONÁRIOS BASEADO
EM TECNOLOGIA BLOCKCHAIN**

Rio de Janeiro - RJ

2023

GILMAR RIBEIRO SANTANA

**WORKBLOCK: SISTEMA DE CONTROLE DE PONTO
DE FUNCIONÁRIOS BASEADO
EM TECNOLOGIA BLOCKCHAIN**

Trabalho de Conclusão de Curso
apresentado ao curso de Análise e
Desenvolvimento de Sistemas da
Faculdade de Educação Tecnológica do
Estado do Rio de Janeiro, sob orientação
do Prof. André Neves, como exigência
para conclusão do curso de graduação

Rio de Janeiro - RJ

2023

TERMO DE APROVAÇÃO

GILMAR RIBEIRO SANTANA

WORKBLOCK: SISTEMA DE CONTROLE DE PONTO DE FUNCIONÁRIOS BASEADO EM TECNOLOGIA BLOCKCHAIN

Trabalho de Conclusão de Curso apresentado ao curso de Análise e Desenvolvimento de Sistemas da Faculdade de Educação Tecnológica do Estado do Rio de Janeiro, sob orientação do Prof. André Neves, como exigência para conclusão do curso de graduação

Professor 1

Professor 2

Professor 3

Rio de Janeiro, 10 de julho de 2023

RESUMO

O presente trabalho visa propor uma nova metodologia no processo de gestão e acompanhamento de controle de ponto de funcionários de uma empresa utilizando a tecnologia Blockchain. Com a crescente mudança do estilo de trabalho, onde se permite atuar fora das dependências físicas da empresa, se faz necessário a criação de um sistema capaz de atender este cenário garantindo a idoneidade dos registros de marcação de ponto e gestão, e a fidedignidade de todas as ações realizadas para dirimir, de forma segura e assertiva, demandas judiciais de contestação de horas trabalhadas. Através da utilização desse sistema, será possível padronizar o procedimento de gestão e acompanhamento dos registros de ponto, tanto de colaboradores que atuam nas dependências da empresa, como aqueles que atuam de forma híbrida, remota e externa, e também manter um registro de todas as operações que foram realizadas ao longo do seu desenvolvimento. Por fim, com a criação desses registros, tanto gestores como colaboradores poderão gerar relatórios de registros de ponto para acompanhamento e integração com outros sistemas. Como entrega final, será possível fazer o *download* dos arquivos que compõem esta obra: Arquivo ‘PDF’ – Monografia e Arquivo ‘ZIP’ - códigos-fonte.

ABSTRACT

The present work aims to propose a new methodology in the process of management and monitoring of employee attendance controls of a company using the innovative Blockchain technology. With the growing change in the style of work, where it is possible to work outside the physical premises of the company, it is necessary to create a system capable of meeting this scenario, guaranteeing the suitability of the clocking and management records, guaranteeing the reliability of all actions carried out to resolve, in a safe and assertive manner, judicial demands for contestation of hours worked. Through the use of this system, it will be possible to standardize the procedure for managing and monitoring time attendance records, both for employees who work on the company's premises and those who work in a hybrid, remote and external way, and also keep a record of all operations that were carried out throughout its development. Finally, with the creation of these records, both managers and employees will be able to generate point records reports for monitoring and integration with other systems. As a final delivery, it will be possible to download the files that make up this work: 'PDF' file - Monograph and 'ZIP' file - source codes.

LISTA DE FIGURAS

Figura 1 - Framework de avaliação para uma estrutura blockchain	22
Figura 2 - Framework de avaliação para uma estrutura blockchain	23
Figura 3 - Diagrama de representação do projeto web para questionário	26
Figura 4 - Página de apresentação do projeto ..	27
Figura 5 - Página para questionário ..	27
Figura 6 - Resultado questão 1 do perfil Profissional	28
Figura 7 - Resultado questão 2 do perfil Profissional	29
Figura 8 - Resultado questão 3 do perfil Profissional	29
Figura 9 - Resultado questão 4 do perfil Profissional	30
Figura 10 - Resultado questão 5 do perfil Profissional	30
Figura 11 - Resultado questão 6 do perfil Profissional	31
Figura 12 - Resultado questão 7 do perfil Profissional	31
Figura 13 - Resultado questão 8 do perfil Profissional	32
Figura 14 - Resultado questão 9 do perfil Profissional.....	32
Figura 15 - Resultado questão 10 do perfil Profissional	33
Figura 16 - Resultado questão 1 do perfil Gestor	33
Figura 17 - Resultado questão 2 do perfil Gestor	34
Figura 18 - Resultado questão 3 do perfil Gestor	34
Figura 19 - Resultado questão 4 do perfil Gestor	35
Figura 20 - Resultado questão 5 do perfil Gestor	35
Figura 21 - Resultado questão 6 do perfil Gestor	36
Figura 22 - Resultado questão 7 do perfil Gestor	36
Figura 23 - Resultado questão 8 do perfil Gestor	37

Figura 24 - Resultado questão 9 do perfil Gestor	37
Figura 25 - Resultado questão 10 do perfil Gestor	38
Figura 26 - Cadeia de blocos	41
Figura 27 - Diferentes tipos de criptografia	43
Figura 28 - Criptografia.....	46
Figura 29 - Autenticação	47
Figura 30 - Criptografia e autenticação	48
Figura 31 - Transação na rede Bitcoin	49
Figura 32 - Mecanismo de Hash	51
Figura 33 - Função Hash 256.....	52
Figura 34 - Cadeia de blocos do primeiro nó	53
Figura 35 - Blocos 4 e 5 do segundo nó	54
Figura 36 - Blocos 4 e 5 do terceiro nó	55
Figura 37 - Blocos 3 e 4 originais do segundo nó	56
Figura 38 - Blocos 3 e 4 modificados do segundo nó	56
Figura 39 - Blocos 4 e 5 afetados pela modificação do bloco 3 do segundo nó	57
Figura 40 - Blocos 3 e 4 do primeiro nó.....	57
Figura 41 - Blocos 3 e 4 do terceiro nó	58
Figura 42 - Blocos 3 e 4 do segundo nó após a mineração do bloco 3.....	58
Figura 43 - Árvore de Merkle para 4 transações A, B, C e D	60
Figura 44 - Árvore de Merkle para 4 transações A, B, C e D	61
Figura 45 - Envio de 2 Hashs para 4 transações	62
Figura 46 - Envio de 3 Hashs para 8 transações	62
Figura 47 - Diferença entre rede descentralizada e rede centralizada	63
Figura 48 - Tipos de nó na rede blockchain	64
Figura 49 - Bloco da rede Bitcoin	68

Figura 50 - Bloco da rede Casper	70
Figura 51 - Estrutura de um bloco na rede blockchain	73
Figura 52 - Aplicação Web2 vs aplicação Web3	77
Figura 53 - Projetos utilizando Ethereum	82
Figura 54 - Smart Contracts utilizando a rede Ethereum	82
Figura 55 - Smart Contracts utilizando a rede Ethereum	83
Figura 56 - Cadeia de estados na EVM	84
Figura 57 - Cadeia de estados na EVM	85
Figura 58 - Chave privada, pública e endereço de uma conta	86
Figura 59 - Cadeia de estados na EVM	87
Figura 60 - Transação na EVM	87
Figura 61 - Exemplos de equipamentos físicos para registro de ponto	90
Figura 62 - Exemplo de software para registro de ponto	90
Figura 63 - BPMN - Processo de cadastro	95
Figura 64 - BPMN - Processo de marcação do ponto	95
Figura 65 - BPMN - Processo de emissão de relatório	96
Figura 66 - UC00 – Diagrama de Casos de Uso	100
Figura 67 - UC01 – Casos de Uso Manter Administrador	102
Figura 68 - UC02 – Casos de Uso Manter Empregador.	104
Figura 69 - UC03 – Casos de Uso Manter Funcionário.	106
Figura 70 - UC04 – Casos de Uso Emitir Relatório	108
Figura 71 - UC05 – Casos de Uso Marcar Ponto	110
Figura 72 - CD00 – Contracts	112
Figura 73 - CD01 – Util Contract	113
Figura 74 - CD02 – Administrator Contract	114
Figura 75 - CD03 – Employer Contract	117

Figura 76 - CD04 – Employee Contract	120
Figura 77 - CD05 – Ponto Block	124
Figura 78 - CD06 – Ponto Block Reports.	128
Figura 79 - SEQ – Diagrama de Sequência.	130
Figura 80 - SEQ – Verificação do funcionário	131
Figura 81 - SEQ – Verificação e retorno dos dados do funcionário	132
Figura 82 - SEQ – Validação dos dados do funcionário	132
Figura 83 - SEQ – Chamada da função startWork()	133
Figura 84 - SEQ – Verificação da existência do funcionário no sistema	133
Figura 85 - SEQ – Verificação se o chamador (smart contract Ponto Block) está no roll de administradores	134
Figura 86 - SEQ – Envio dos dados do funcionário	134
Figura 87 - SEQ – Tratamento do timestamp, execução da função startWork() e registro do evento através da função StartWorkRegistered()	135
Figura 88 - SEQ – Retorno da mensagem de confirmação do registro de ponto ...	135
Figura 89 - Tela de cadastro de funcionário	137
Figura 90 - Tela de marcação de ponto	138
Figura 91 - Tela de relatório	139
Figura 92 - Arquitetura do sistema	142
Figura 93 - Ambiente de desenvolvimento dos smart contracts	143
Figura 94 - Página principal do contrato no explorador de blocos na rede Mumbai Polygon	144
Figura 95 - Código do contrato no explorador de blocos na rede Mumbai Polygon	145
Figura 96 - Eventos do contrato no explorador de blocos na rede Mumbai Polygon	145
Figura 97 - Pasta criada pela ferramenta Coverage	146
Figura 98 - Cobertura de testes dos smart contracts	146
Figura 99 - Cobertura de testes em AdministratorContract	146

Figura 100 - Desenho de arquitetura da API	147
Figura 101 - Tabelas no banco de dados WorkBlock	148
Figura 102 - Diagrama de classe de AdministratorContractController	149
Figura 103 - Endpoints de AdministratorContract e EmployeeContract	150
Figura 104 - Endpoints de EmployerContract e PontoBlock	151
Figura 105 - Endpoints de PontoBlockReports e UtilContract.....	152
Figura 106 - Exemplo de chamada de um endpoint exibido pelo Swagger.	153
Figura 107 - Diagrama de classe da página Create do Administrador	156
Figura 108 - Diagrama de classe de AdministratorService	156
Figura 109 - Diagrama de classe de AdministratorRepository	157
Figura 110 - Listagem de colaboradores	157
Figura 111 - Página de cadastro de Colaborador	158
Figura 112 - Página de atualização de Colaborador	158
Figura 113 - Página de Exibição de relatórios de marcação de ponto	159
Figura 114 - Dados do colaborador	160
Figura 115 - Teste de validação frontend (inválido).....	160
Figura 116 - Teste de validação frontend (válido).....	161
Figura 117 - Teste de validação backend (inválido).....	161
Figura 118 - Teste de validação resposta API (inválido).....	162
Figura 119 - Resultado de atualização de registro	163
Figura 120 - Registro da atualização no banco de dados.....	163
Figura 121 - Registro da transação no explorador de blocos.	163
Figura 122 - Detalhe da transação no explorador de blocos	164
Figura 123 - Registro do evento da operação registrado no smart contract	164
Figura 124 - Representação da arquitetura do projeto PontoBlock	168
Figura 125 - Página principal da página Ponto Block.	169
Figura 126 - Mensagem de erro caso não possua o MetaMask.	170

Figura 127 - Conexão com sucesso na aplicação	171
Figura 128 - Conexão sem sucesso na aplicação	171
Figura 129 - Visão completa após a conexão com sucesso	172
Figura 130 - Toaster com mensagem de erro para marcações inválidas	173
Figura 131 - Janela de confirmação MetaMask	173
Figura 132 - Detalhes de conta na janela de confirmação MetaMask	174
Figura 133 - Detalhes de transação na janela de confirmação MetaMask	175
Figura 134 - Registro de início de jornada	176
Figura 135 - Registros de ponto de um dia de jornada.....	176
Figura 136 - Registro da transação no explorador de blocos da rede.	177
Figura 137 - Detalhe da transação de início de jornada	177
Figura 138 - Código da função de início de jornada	177
Figura 139 - Registro do evento na blockchain.....	177
Figura 140 - Histórico de marcação de ponto	178
Figura 141 - Ponto Block em dispositivo móvel.	179
Figura 142 - MetaMask Google Play Store	179

LISTA DE QUADROS

Quadro 1 – Requisitos Funcionais.....	97
Quadro 2 – Requisitos Não Funcionais	98
Quadro 3 – Casos de Uso.....	99
Quadro 4 – Diagrama de Classe.....	111
Quadro 5 – Endereço de publicação dos projetos.....	142
Quadro 6 – Endereço de publicação dos smart contracts	144
Quadro 7 – Custo de deploy por smart contract	180
Quadro 8 – Custo de utilização do sistema por operação de ponto	181
Quadro 9 – Evolução de custo por funcionário.	181
Quadro 10 – Previsão de faturamento anual	181

SUMÁRIO

1.0 INTRODUÇÃO	16
2.0 PROBLEMA.....	17
3.0 JUSTIFICATIVA.....	17
4.0 OBJETIVO GERAL.....	23
5.0 OBJETIVOS ESPECÍFICOS.....	24
6.0 METODOLOGIA CIENTÍFICA	25
6.1 QUESTIONÁRIO.....	25
7.0 FUNDAMENTAÇÃO TEÓRICA	38
7.1 BLOCKCHAIN	38
7.1.1 HISTÓRIA	38
7.1.2 FUNCIONAMENTO.....	39
7.1.3 CRIPTOGRAFIA E VALIDAÇÃO.....	42
7.1.4 MECANISMO DE HASH	50
7.1.5 ÁRVORE DE MERKLE	59
7.1.6 REDE DESCENTRALIZADA	63
7.1.7 PROTOCOLO DE CONSENSO	66
7.1.7.1 PROOF OF WORK (PoW) – PROVA DE TRABALHO	67
7.1.7.2 PROOF OF STAKE (Pos) – PROVA DE PARTICIPAÇÃO	69
7.1.7.3 PROOF OF AUTHORITY (PoA) – PROVA DE AUTORIDADE	72
7.1.8 ESTRUTURA DO BLOCO	72
7.2 EVOLUÇÃO DA WEB.....	73
7.3 SOLUÇÕES BLOCKCHAIN	77
7.4 SMART CONTRACTS	79
7.5 ETHEREUM	81
7.5.1 ETHEREUM VIRTUAL MACHINE (EVM)	83
7.5.2 CONTAS	85
7.5.3 TRANSAÇÕES	87

7.5.4 PATRICIA MERKLE TREE	89
8.0 SOLUÇÕES DE CONTROLE DE PONTO UTILIZADAS NO MERCADO.....	90
8.1 MODELO DE NEGÓCIO.....	91
9.0 REGRAS DE NEGÓCIO	91
10.0 MODELAGEM DE PROCESSOS.....	94
11.0 REQUISITOS	97
12.0 CASOS DE USO.....	99
12.1 UC00 - GERAL.....	100
12.2 UC01 – MANTER ADMINISTRADOR.....	102
12.3 UC02 – MANTER EMPREGADOR	104
12.4 UC03 – MANTER FUNCIONÁRIO.....	106
12.5 UC04 – EMITIR RELATÓRIO	108
12.6 UC05 – MARCAR PONTO.....	110
13.0 DIAGRAMA DE CLASSE	111
13.1 CD00 - CONTRACTS.....	112
13.2 CD01 – UTIL CONTRACT	113
13.3 CD02 – ADMINISTRATOR CONTRACT	114
13.4 CD03 – EMPLOYER CONTRACT	117
13.5 CD04 – EMPLOYEE CONTRACT	120
13.6 CD05 – PONTO BLOCK.....	124
13.7 CD06 – PONTO BLOCK REPORTS.....	128
14.0 DIAGRAMA DE SEQUÊNCIA	130
15.0 PROTOTIPAÇÃO DE TELAS	136
16.0 CONSIDERAÇÕES NO DESENVOLVIMENTO COM SOLIDITY.....	140
17.0 ARQUITETURA DO SISTEMA	141
17.1 DESENVOLVIMENTO DE CONTRATOS.....	143
17.2 API DE GESTÃO	147
17.3 PROJETO WEB DE GESTÃO	155
17.4 PROJETO WEB DE MARCAÇÃO DE PONTO	167

18.0 POTENCIALIDADES DE MERCADO	180
19.0 CONCLUSÃO	182
REFERÊNCIAS.....	184

1.0 INTRODUÇÃO

A tecnologia blockchain tem se apresentado como uma das soluções mais disruptivas e inovadoras da atualidade. Palavras como criptomoedas, Bitcoin, *NFT* (Tokens Não Fungíveis), e redes descentralizadas já começam a ser mais difundidas e despertam curiosidade sobre o que esses termos significam e qual sua utilidade.

Neste contexto de inovação, a tecnologia blockchain, também conhecida como Tecnologia de Livro razão Distribuído (*Decentralized Ledger Technology – DLT*), traz as bases para este novo mundo que é ferramenta eficaz para atingir os alvos da Web 3.0 – em momento oportuno, nesta obra, abordaremos com mais detalhes os alvos da Web 3.0.

Inicialmente, a tecnologia blockchain foi utilizada para atender um novo produto, um novo mercado, uma nova solução financeira: as criptomoedas, mais especificamente, o Bitcoin, sendo a primeira e até hoje a mais sólida e conhecida das criptomoedas. Hoje, essa tecnologia disruptiva já é utilizada em diversos cenários trazendo um grau a mais de segurança, sigilo e velocidade em muitos outros negócios.

Já não são isoladas as soluções na área de educação, logística, engenharia social, IoT, inteligência artificial, entre outras, além da área financeira, é claro, que utilizam esta inovação.

Assim sendo, o presente trabalho também faz uso desta tecnologia, apresentando um sistema e controle de ponto de funcionários armazenando os dados e as mudanças de estado em uma rede pública que utiliza a tecnologia de livro razão distribuído, ora chamada, blockchain.

2.0 PROBLEMA

Uma empresa que utiliza um sistema de controle de ponto que efetua os registros através de aparelho de leitura biométrica tipicamente, registra 4 marcações de ponto por dia de um funcionário, a saber: início de jornada, início de pausa, fim de pausa e fim de jornada. 4 comprovantes por dia aplicado a 22 dias úteis no mês, isso gera 88 comprovantes mensais e 1056 comprovantes anuais. Agora imagine um funcionário com 5 anos de empresa, que não é algo incomum, seriam 5280 comprovantes.

Diante deste cenário, percebemos o quanto inviável é para qualquer funcionário gerenciar tais registros, sem falar que os comprovantes são gerados utilizando papel térmico e que se não tiverem um armazenamento adequado, longe do calor e luz intensa, podem se apagar. Por outro lado, o empregador muitas vezes é contestado em juízo e dentro da própria rotina diária por funcionários que se sentem lesados pelos registros, havendo, então, desconfiança por ambas as partes.

Diante de um conflito de interesses tão grande haveria uma solução viável capaz de atender os preceitos técnicos de confiança, segurança e imutabilidade que fosse ponto pacífico entre as duas partes (empregado e empregador) e que fornecesse os meios necessários para que o juiz trabalhista também se sinta confortável em dirimir tais tipos de conflitos? Esta pergunta é a que o presente trabalho pretende responder.

3.0 JUSTIFICATIVA

Conforme estabelecido pela Lei da Liberdade Econômica (CONGRESSO NACIONAL, Lei 13.874/2019) as empresas com mais de 20 trabalhadores devem fazer uso de um mecanismo para controle de ponto, mecanismo no qual serão realizadas as anotações da

jornada de trabalho, com marcações de início e fim da jornada de trabalho bem como o início e fim da pausa.

Como requisitos mínimos e padronização destes dispositivos, foi introduzido no ordenamento jurídico a portaria 671/2021 (MTP, 671/2021) do Ministério do Trabalho e Previdência (MTP) que entre várias disposições, trata sobre a anotação da hora de entrada e de saída em registro manual, mecânico ou eletrônico. Estas disposições atualizam as portarias 1510/2009 (MTE, 1510/2010) e 373/2011 (MTE, 373/2011), ambas do Ministério do Trabalho e Emprego (MTE). A primeira trata sobre o uso de relógio eletrônico de ponto (REP), definindo configurações mínimas, informações obrigatórias e aspectos de segurança do equipamento, já a segunda, atendendo à modernização tecnológica, reconhece os programas e aplicativos utilizados para este fim.

A portaria 671/2021 utiliza a Seção IV Subseção I para definir os meios eletrônicos para o controle da jornada, reconhecendo, então, três categorias de dispositivos para este fim, a saber:

1. Registrador Eletrônico de Ponto Convencional (REP-C);
2. Registrador Eletrônico de Ponto Alternativo (REP-A);
3. Registrador Eletrônico de Ponto Via Programa (REP-P).

Em seu artigo 76, a portaria define o REP-C:

Art. 76. O REP-C é o equipamento de automação monolítico, identificado pelo seu número de fabricação e cujo modelo possui certificado de conformidade especificado no art. 90, utilizado exclusivamente para o registro de jornada de trabalho e com capacidade para emitir documentos decorrentes da relação do trabalho e realizar controles de natureza fiscal trabalhista, referentes à entrada e à saída de empregados nos locais de trabalho.

Na sequência, em seu artigo 77, é definido o REP-A:

Art. 77. O REP-A é o conjunto de equipamentos e programas de computador que tem sua utilização destinada ao registro da jornada de trabalho, autorizado por convenção ou acordo coletivo de trabalho.

E para terminar a classificação, em seu artigo 78 é definido o último tipo de registrador, o REP-P:

Art. 78. O REP-P é o programa (software) executado em servidor dedicado ou em ambiente de nuvem com certificado de registro nos termos do art. 91, utilizado exclusivamente para o registro de jornada e com capacidade para emitir documentos decorrentes da relação do trabalho e realizar controles de natureza fiscal trabalhista, referentes à entrada e à saída de empregados nos locais de trabalho.

O Anexo IX deste mesmo documento estabelece os requisitos técnicos aos quais o REP-P deve atender, entre as quais elenca em seu item 6: **armazenamento com redundância, alta disponibilidade e confiabilidade**, conceitos que estão intimamente ligados e até mais que isso, fazem parte da essência da tecnologia blockchain. As demais determinações são de ordem mais formal definindo as informações que devem ser registradas e o formato apresentado, como por exemplo, permitir a identificação do empregador e trabalhador, possuir relógio com sincronismo com a hora de Brasília, identificação da pessoa que faz o registro ou alterações no sistema, entre outras.

As regras estabelecidas acima são perfeitamente implementáveis na solução proposta, conforme será exposto adiante, além disso, as exigências de imutabilidade,

identificação e segurança estabelecidos na portaria são, por essência, implementados na tecnologia blockchain, conforme, também, será exposto em momento oportuno no decorrer dessa obra.

Diante da importância do tema, é fundamental apresentar uma solução capaz de atender a esta demanda de mercado com critérios de segurança e integridade que impossibilitem a adulteração de dados para proteger a sociedade; e a tecnologia blockchain tem se mostrado, em muitos cenários, como solução viável e segura para repositório de dados de forma descentralizada, com persistência de histórico de operações e identificação de transação permitindo sua rastreabilidade.

Sendo o blockchain uma tecnologia de Livro Razão Distribuído materializado através de uma cadeia de blocos onde o bloco da transação atual tem a referência do bloco que fez a transação anterior e assim sucessivamente, uma transação só é validada se houver um histórico e um referência a um bloco anterior válido, e para que esta transação seja válida é preciso aprovação de mais de 50% dos integrantes da rede blockchain (SCHULTZ, 2019), ou seja, ao descentralizarmos o repositório das informações e criarmos mecanismos de validação, dificultamos que uma pessoa sozinha faça modificações indevidas nos dados da cadeia (NAKAMOTO, 2008)¹. Desta maneira, um registro de fim de jornada de trabalho, por exemplo, seguindo a ordem de precedência de blocos da cadeia, só pode ser gerado para um trabalhador que, minimamente, tenha sido registrado dentro do sistema do empregador e iniciado sua jornada de trabalho.

Através da implantação de uma solução utilizando a tecnologia blockchain, onde todos os dados do empregador e empregado bem como seus registros de ponto com início, fim e pausa da jornada, é possível acompanhar toda a cadeia.

¹ Apesar de matematicamente possível, devido ao processamento computacional dispensado é praticamente impossível alteração de um dado da cadeia blockchain enquanto os nós honestos forem a maioria da rede.

Com esta abordagem, um agente fiscalizador pode consultar quantos empregados registrados há no sistema, os registros de um empregado específico, o histórico de todas as marcações de todos os empregados em uma data específica, em um período de datas ou até mesmo todo o ciclo de vida da empresa. Perceba que uma vez registrado na blockchain tais informações, elas ficam lá independentemente de mudança de fornecedor de software ou até mesmo a extinção da empresa, não dependendo de um banco de dados próprio da empresa ou funcionário para se ter o registro gravado.

Além do histórico idôneo de jornada de trabalho, outras ferramentas tecnológicas como os smart contracts (FACHINI, 2023)², que são contratos eletrônicos autoexecutáveis podem ser inseridos nesta cadeia, promovendo ações diversas como bloqueio do empregado em caso de abandono de emprego, notificações ao empregador em caso de falta etc.

Desde o início das operações com blockchain, em especial com a mais famosa das criptomoedas, bitcoin (BITCOIN, 2023)³, muitas soluções tem sido propostas para aprimorar a segurança em operações e persistência de dados, e o setor público não fica de fora deste cenário onde temos propostas como a gravação de operações policiais e registro em blockchain (DAVIDSON, 2017), combate à corrupção e lavagem de dinheiro (ENCCLA, 2020) além de diversas outras estratégias (SILVA e MARQUES, 2021) apresentadas por SILVA e MARQUES em seu trabalho de pesquisa sobre as propostas desta tecnologia para o setor público.

A proposta deste trabalho entra, então, em consonância com as mais recentes estratégias de uso desta tecnologia para solução de problemas conhecidos. O autor da obra reconhece o caractere ainda em evolução da tecnologia mencionada e que ela não é a

² Conforme define FACHINI, Tiago: “Smart Contract é um tipo de contrato digital, que usa tecnologia blockchain de ponta para garantir a auto execução das cláusulas, sempre que as condições contratuais previstas são atendidas.”

³ Bitcoin is a consensus network that enables a new payment system and a completely digital money. It is the first decentralized peer-to-peer payment network that is powered by its users with no central authority or middlemen. From a user perspective, Bitcoin is pretty much like cash for the Internet. Bitcoin can also be seen as the most prominent [triple entry bookkeeping system](#) in existence.

resposta para todos os problemas e muito menos há ausência de limitações e cenários para implantação desta solução, e realmente há cenários em que uma solução com uma autoridade central é mais vantajosa do que outra que usa autoridade descentralizada, como a blockchain, onde o artigo de Sing Kuang Lo, Xiwei Xu, Yin Kia Chiam e Qinghua Lu (2017) em conferência sobre a engenharia de sistemas complexos, da IEEE, em 2017 mostra esta análise. A figura abaixo, retirada do artigo supramencionado, mostra um fluxograma muito útil na tomada de decisão sobre a viabilidade de utilização de uma arquitetura de livro razão descentralizada, como a blockchain.

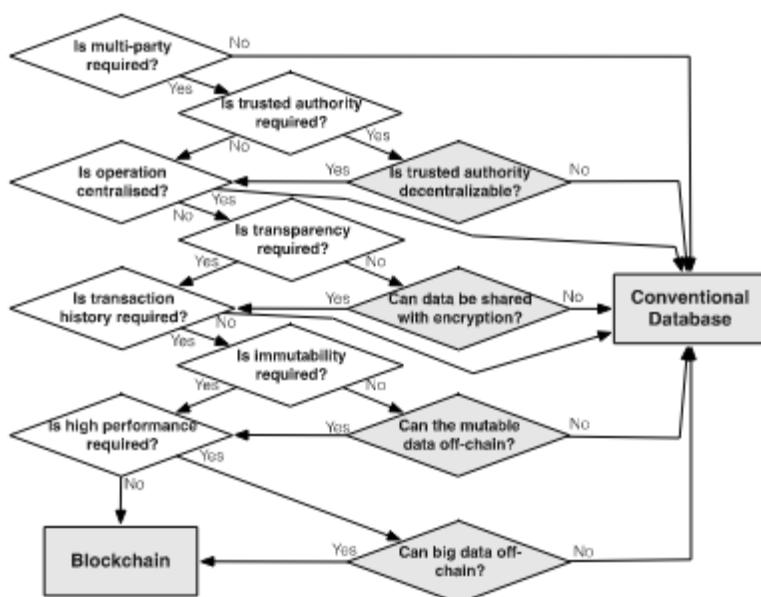


Figura 1 – Framework de avaliação para uma estrutura blockchain. [LO, XU, CHIAM, LU] 2017. P. 159

De igual forma, o trabalho apresentado por WÜST e GERVAIS (2018) mostra um fluxograma para tomada de decisão de um projeto envolvendo, ou não, a tecnologia blockchain. Em resumo, os autores propõem o uso apenas em situação em que não temos uma confiança em terceiros e estes são, também, os responsáveis pela governança de um sistema.

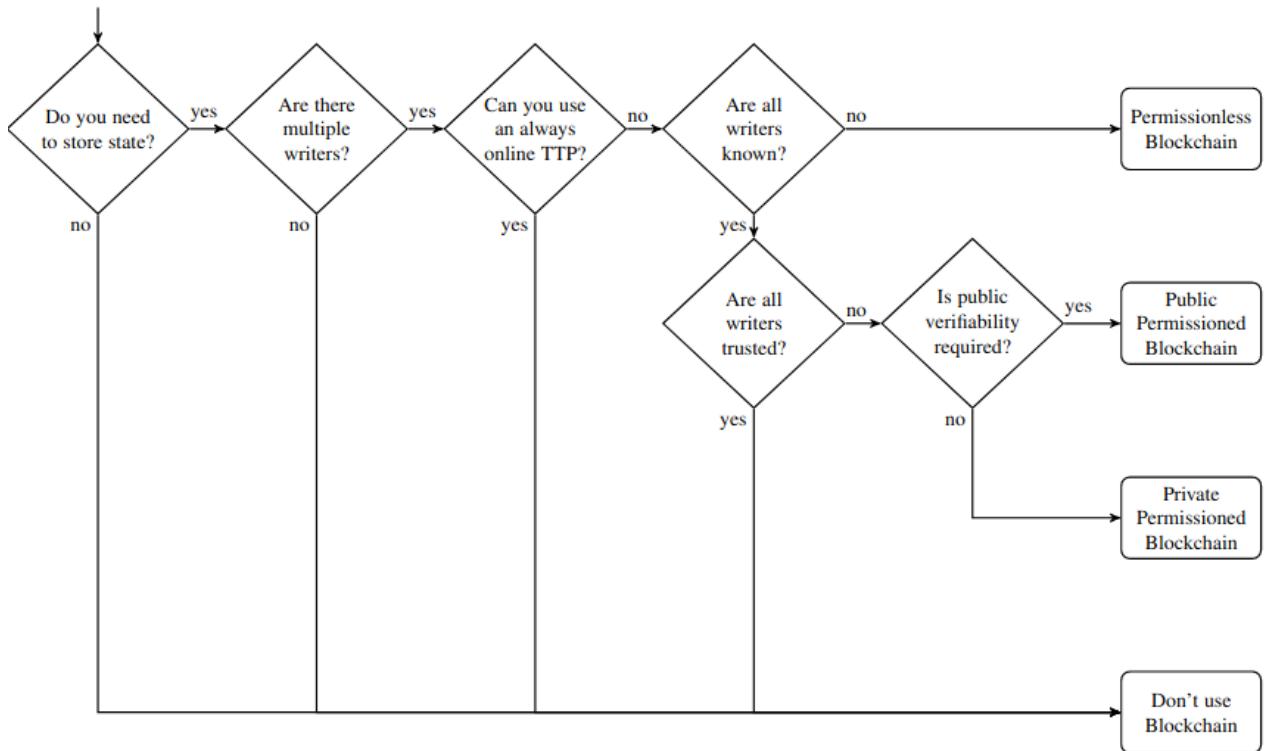


Figura 2 – Framework de avaliação para uma estrutura blockchain. [WÜST e GERVAIS] 2018. P. 3

4.0 OBJETIVO GERAL

Criar uma solução em blockchain para registrar os eventos de marcação de ponto para os empregados de uma empresa, desde o seu registro até a emissão de relatórios com uso de livro razão descentralizado utilizando a plataforma **Polygon**⁴, solução de segunda camada da **Ethereum**⁵ para desenvolver e hospedar esta solução com a implementação de smart contracts.

⁴ **Polygon** é uma solução de segunda camada da rede Ethereum utilizada para conectar e escalar projetos em blockchain, sobretudo aqueles relacionados e compatíveis com a rede Ethereum.

⁵ **Ethereum** é uma plataforma descentralizada que faz uso da tecnologia de TLD (Tecnologia de Livro Razão Distribuído) onde, além de possuir sua própria criptomoeda (ETH), é possível executar contratos inteligentes, criada em 2014 pelo russo-canadense Vitalik Buterin..

5.0 OBJETIVOS ESPECÍFICOS

Este projeto tem como objetivos de domínio:

1. Demonstrar as vantagens de um sistema utilizando uma estratégia de registro de livro razão descentralizado em face a uma aplicação tradicional centralizada.
2. Apresentar o funcionamento atual do sistema de controle de ponto tradicional e seus pontos vulneráveis.
3. Criar uma aplicação web para consultar e registrar informações referentes ao registro de ponto de funcionários de uma empresa.

E como objetivos técnicos:

1. Criar smart contracts utilizando a linguagem **Solidity**⁶, testá-los e publicá-los através da ferramenta **HardHat**⁷.
2. Criar uma Web API em **.NET**⁸ para interagir com a plataforma **Polygon**.
3. Criar um projeto Web para gerenciamento do sistema com a Web API utilizando a tecnologia **.NET**.
4. Criar um projeto Web para interagir com os smart contracts na **Polygon** e a Web API em **.NET** para registro dos pontos pelo funcionário.
5. Criar uma solução em blockchain para registrar as marcações de ponto dos funcionários de uma empresa, desde o registro da empresa e funcionários, bem como a emissão de relatórios.

⁶ Solidity é uma linguagem orientada a objetos de alto nível, onde é possível implementar smart contracts.

⁷ Hardhat is a development environment for Ethereum software. It consists of different components for editing, compiling, debugging and deploying your smart contracts and dApps, all of which work together to create a complete development environment. Hardhat Runner is the main component you interact with when using Hardhat. It's a flexible and extensible task runner that helps you manage and automate the recurring tasks inherent to developing smart contracts and dApps.

⁸ .NET is a free, cross-platform, open-source developer platform for building many different types of applications. With .NET, you can use multiple languages, editors, and libraries to build for web, mobile, desktop, games, IoT, and more.

6.0 METODOLOGIA CIENTÍFICA

O presente trabalho trata de uma pesquisa de natureza aplicada. O produto apresentado nesta obra destina-se a solucionar o problema pertinente ao armazenamento de informações centralizadas na gestão de controle de ponto. Quanto à abordagem, será de caráter qualitativo porque envolve menos quantidade de dados e aprofunda melhor as questões envolvidas.

Sendo assim, a forma como será desenvolvida esta pesquisa será de natureza descritiva, uma vez que serão apresentados diversos diagramas como os de caso de uso, classe e sequência, entre outros, definidos pela UML⁹. Além disso, com relação aos procedimentos de pesquisa, será realizado um estudo de caso do tema problematizado acima.

6.1 QUESTIONÁRIO

Na busca do entendimento do mercado e do cenário buscado no atendimento à demanda de criação da solução, foi adotada a estratégia de criar um questionário com 20 perguntas, sendo 10 delas voltadas para os funcionários e 10 delas voltadas para os gestores que cuidam da parte de RH (Recursos Humanos).

Neste cenário, foi criado um pequeno sistema web para registro das questões e armazenamento das respostas utilizando arquitetura MVC conforme abaixo:

⁹ **UML – UNIFIED MODELING LANGUAGE.** Linguagem de Modelagem Unificada ajuda a especificar, visualizar e documentar modelos de sistema de software, incluindo sua estrutura e arquitetura, como forma de organizar todos esses requisitos. UML é marca registrada de **OMG**. Adaptado.

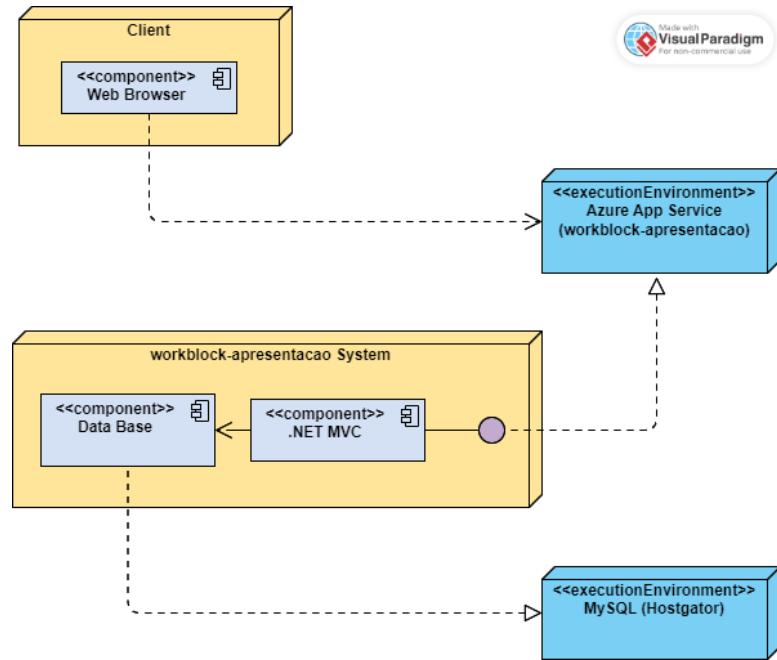


Figura 3 – Diagrama de representação do projeto web para questionário. Fonte: Autor.

Esta aplicação apresenta duas partes distintas, sendo elas: uma página web estática com uma apresentação sobre o projeto, e um questionário, onde de acordo com o perfil desejado (profissional ou gestor de RH) são apresentadas as perguntas do questionário. A URL do projeto pode ser acessada no seguinte endereço: <https://workblock-apresentacao.azurewebsites.net/>.



Figura 4 – Página de apresentação do projeto. Fonte: Autor

Figura 5 – Página para questionário. Fonte: Autor

Após a divulgação da proposta e questionário em rede social, entre os profissionais foram obtidos os resultados conforme abaixo:

Respostas referente a profissionais.

Analizando os dados notamos que o uso do registro de ponto é algo plenamente natural e de ampla utilização, sendo a opção mais adotada o registro de ponto utilizando meio eletrônico para tal. Dentre os entrevistados foi notado que mais da metade já enfrentaram

problemas com seu registro de ponto, mesmo que de forma não frequente, aliado a isso, há o fato de cerca de apenas 1/4 dos entrevistados terem acesso aos seus registros diretamente, fazendo então com que ampla maioria ache relevante um sistema com mais recursos como uso por aplicativo ou reconhecimento facial e cerca de um pouco mais da metade não confie plenamente no sistema adotado para registro de ponto. Já quanto à adoção de uma tecnologia blockchain, aproximadamente metade dos entrevistados apoia a ideia e nenhum deles reconhece essa proposta como contrária à segurança dos registros e otimização na operação do sistema.

De modo geral, percebemos com os resultados que a utilização de um meio eletrônico e a adoção da tecnologia blockchain para este fim são apoiadas. Logicamente, a pesquisa poderia ser ampliada para um universo maior de entrevistados para um resultado mais conclusivo, mas esse levantamento inicial já nos mostra sinais interessantes no apoio de nosso projeto.

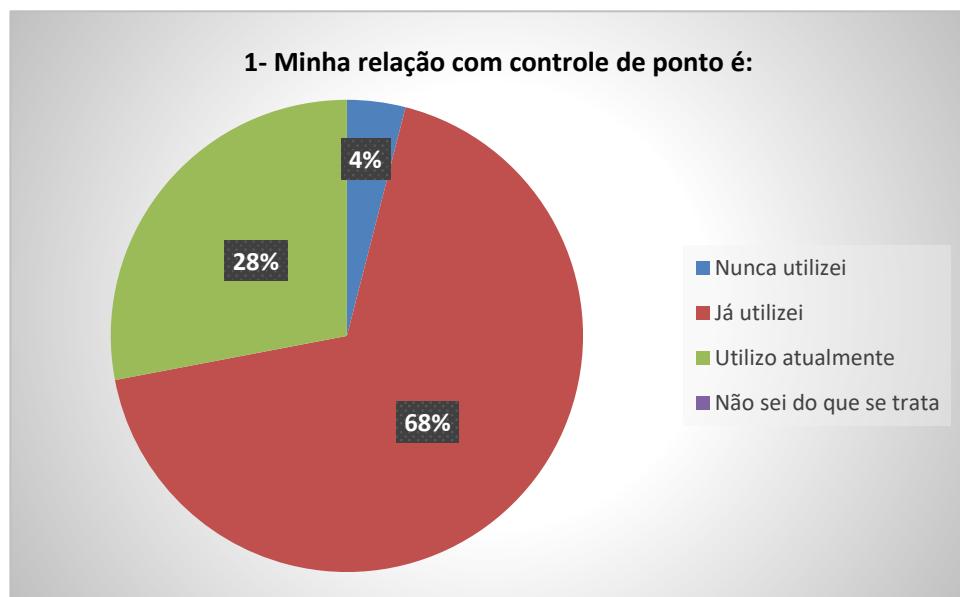


Figura 6 – Resultado questão 1 do perfil Profissional. Fonte: Autor

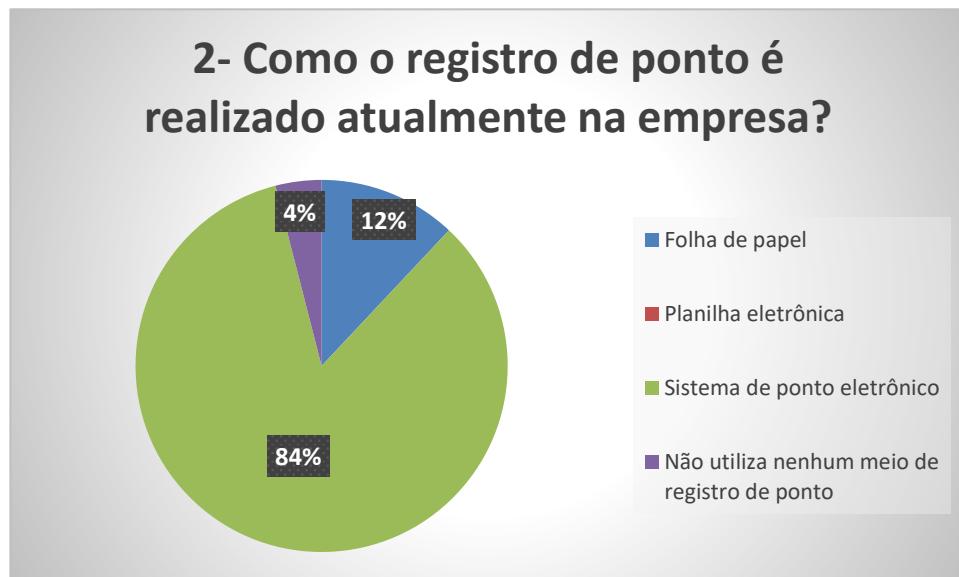


Figura 7 – Resultado questão 2 do perfil Profissional. Fonte: Autor

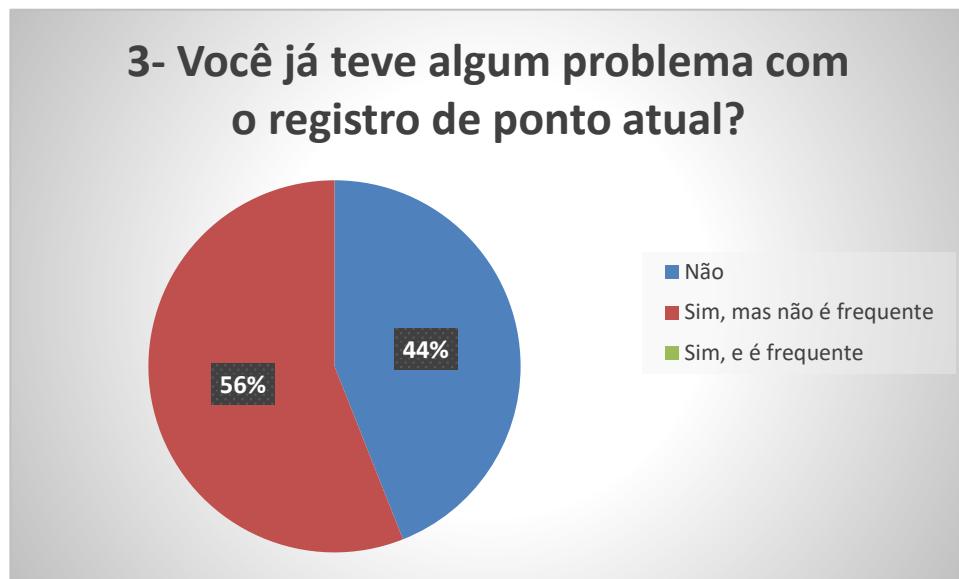


Figura 8 – Resultado questão 3 do perfil Profissional. Fonte: Autor



Figura 9 – Resultado questão 4 do perfil Profissional. Fonte: Autor



Figura 10 – Resultado questão 5 do perfil Profissional. Fonte: Autor



Figura 11 – Resultado questão 6 do perfil Profissional. Fonte: Autor

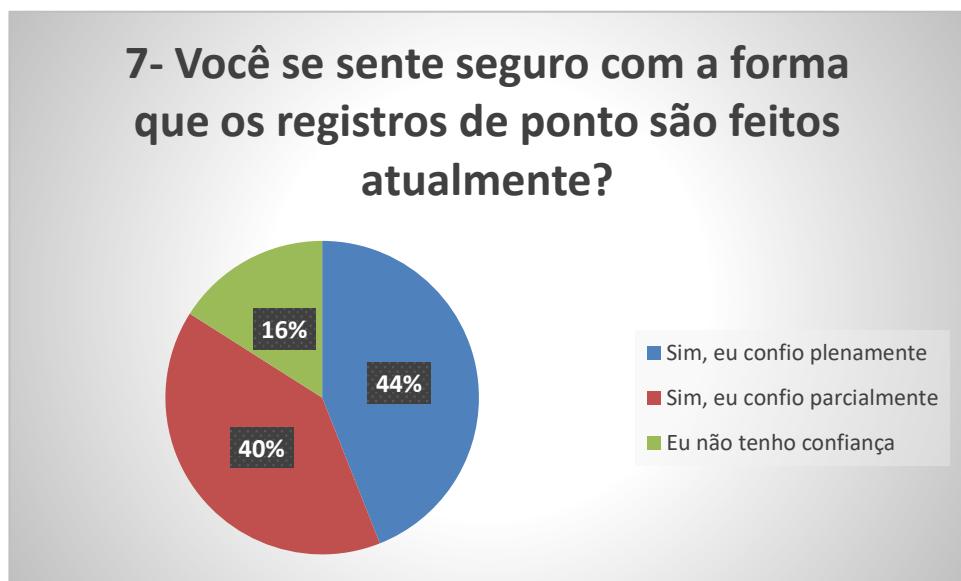


Figura 12 – Resultado questão 7 do perfil Profissional. Fonte: Autor



Figura 13 – Resultado questão 8 do perfil Profissional. Fonte: Autor

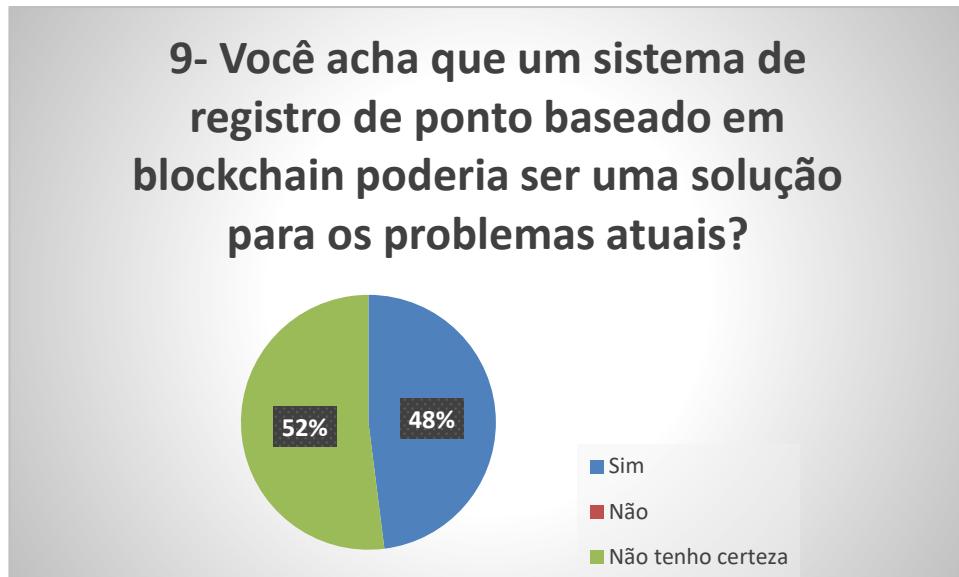


Figura 14 – Resultado questão 9 do perfil Profissional. Fonte: Autor

10- Você acha que um sistema de registro de ponto baseado em blockchain poderia ser mais seguro que o atual?

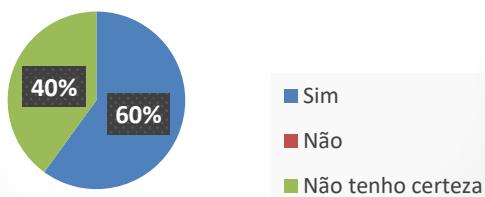


Figura 15 – Resultado questão 10 do perfil Profissional. Fonte: Autor

Respostas referente a gestores.

Com a pesquisa realizada com o grupo de gestores percebemos que é amplamente utilizado um sistema de controle de ponto eletrônico, além disso, a lentidão do sistema é algo que incomoda, então, um sistema que permitisse processos automáticos seria interessante para este público. Além dos comportamentos de necessidade de estabilidade, segurança e disponibilidade on line dos serviços serem características necessárias para o sistema, o público não rejeita a possibilidade de um sistema utilizando blockchain como solução para o segmento.

1- Como o registro de ponto é realizado atualmente na empresa?



Figura 16 – Resultado questão 1 do perfil Gestor. Fonte: Autor



Figura 17 – Resultado questão 2 do perfil Gestor. Fonte: Autor



Figura 18 – Resultado questão 3 do perfil Gestor. Fonte: Autor

4- Existe algum problema de confiabilidade com o processo atual de registro de ponto?

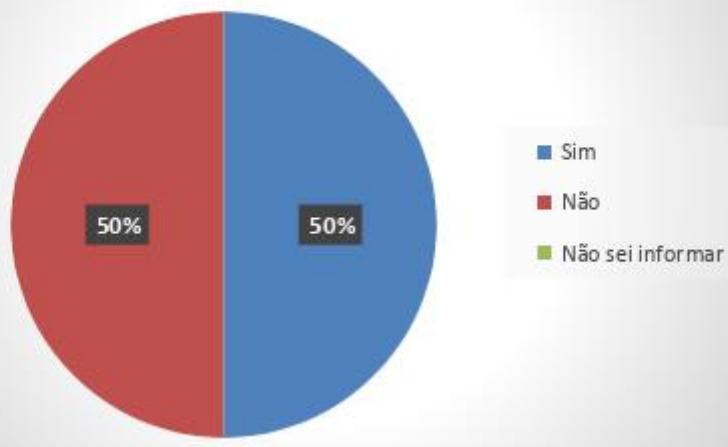


Figura 19 – Resultado questão 4 do perfil Gestor. Fonte: Autor

5- É desejável a possibilidade de consultar o registro de ponto de forma online?

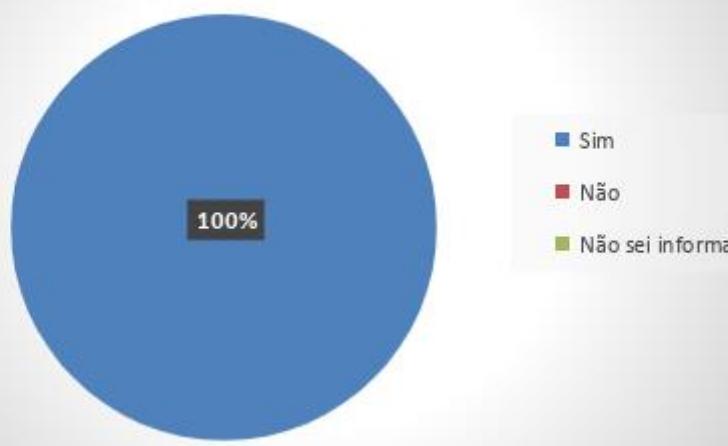


Figura 20 – Resultado questão 5 do perfil Gestor. Fonte: Autor

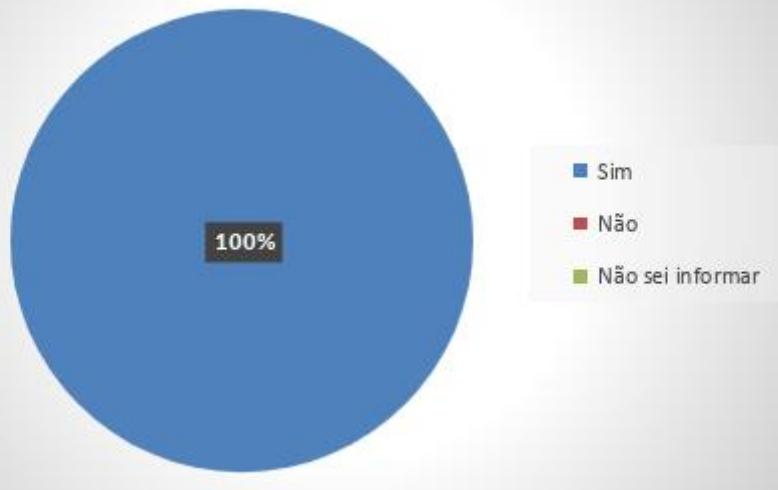
6- É importante que o registro de ponto seja imutável e seguro?

Figura 21 – Resultado questão 6 do perfil Gestor. Fonte: Autor

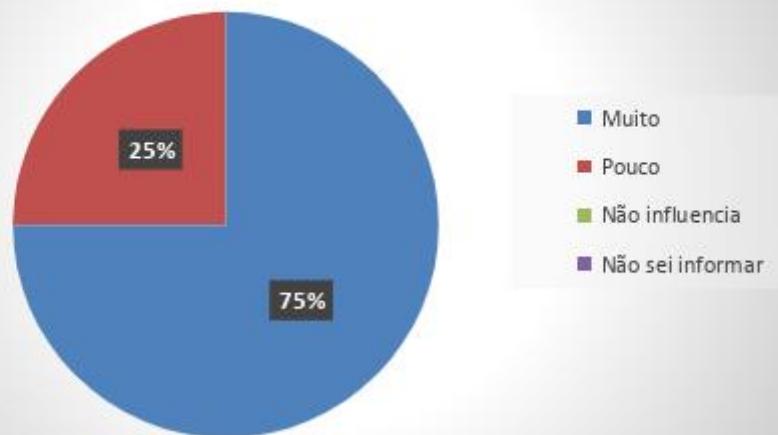
7- Como o registro de ponto atualmente influencia nas decisões gerenciais?

Figura 22 – Resultado questão 7 do perfil Gestor. Fonte: Autor

8- Os colaboradores tem acesso ao seu histórico de registros?

Figura 23 – Resultado questão 8 do perfil Gestor. Fonte: Autor

9- O sistema atual já apresentou falhas que ocasionaram a perda de dados?

Figura 24 – Resultado questão 9 do perfil Gestor. Fonte: Autor

10- Você acredita que um sistema de controle de registro de ponto baseado em blockchain possa ser mais seguro que o atual?

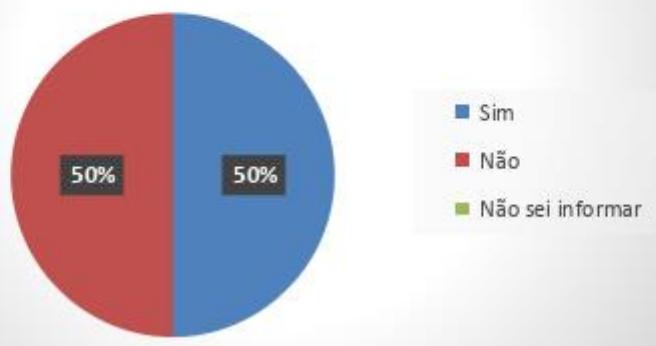


Figura 25 – Resultado questão 10 do perfil Gestor. Fonte: Autor

7.0 FUNDAMENTAÇÃO TEÓRICA

7.1 BLOCKCHAIN

O que é blockchain? Como surgiu? Por que é tão importante? Hoje milhares de projetos usam blockchain, alguns revolucionários e outros nem tanto, mas é essa estrutura de registros que permite a descentralização das informações e que vai continuar mudando a nossa vida nos próximos anos.

7.1.1 HISTÓRIA

Conforme menciona ESCOBAR (2021), em 1991, quase vinte anos antes do Bitcoin, STUART HABER e SCOTT STORNETTA, que trabalhavam na Xerox, criaram uma forma de trazer segurança às informações. Seu objetivo era criar um sistema de registros digitais que fosse imutável, inspirados pelo trabalho de CHAUM DAVID de 1979 que propunha um

sistema de cofres criptográficos. Sua motivação vinha de um escândalo com um artigo de biologia que tinha sido fraudado na época. Nesta fraude, o artigo foi alterado utilizando uma tinta especial e a dupla pensou que se foi possível falsificar um artigo em papel, em mídia digital isso seria muito mais fácil, bastava apertar a tecla delete. Para resolver isso, eles imaginaram blocos de informação atrelados uns aos outros de forma imutável, onde nada pode ser editado ou excluído.

Em 1992 eles incluíram criptografia nesse mecanismo de registros e até 2008 essa invenção não tinha nome e nenhum caso de uso real. Mas a tecnologia ganhou visibilidade quando foi publicado o *White Paper* de NAKAMOTO, SATOSHI, o grande teórico que influenciou a criação do projeto Bitcoin. Foi aí que apareceu a palavra blockchain como o termo é reconhecido hoje. Ele se referia blocos de informação – Blocks - e dados em cadeia – Chain - usando funções de hash. De tanto que as palavras Block e Chain apareceram no Paper, foi natural unir essas duas palavras e chamar esse sistema de Blockchain.

7.1.2 FUNCIONAMENTO

Em tradução literal, **blockchain** significa corrente de blocos. São blocos de informações atrelados uns aos outros. É como tecido digital, você não consegue puxar o bloco do meio da cadeia sem afetar os blocos seguintes, assim como quando você puxa o fio de um tecido de uma roupa você altera toda a costura. Blockchain é como uma costura digital e se um bloco for alterado, todo mundo vai perceber que tem algo errado naquele ponto da cadeia.

Depois de registrado na blockchain, essas informações ficam gravadas nessa cadeia de blocos e é por isso que o blockchain também serve como uma linha do tempo em que os fatos não podem ser modificados, por isso é uma rede imutável e irreversível.

Para modificar os blocos de uma cadeia, tomando como base o exemplo do Bitcoin, a única chance que você tem é naqueles primeiros momentos em que os blocos recentes foram criados. É possível tentar reverter esses blocos de informação, mas isso precisaria de poder computacional absurdo para invadir pelo menos 51% da rede e uma janela de tempo muito pequeno, em torno de 10 a 30 minutos, isso gastaria também Bilhões de Dólares e ainda com a chance de não conseguir fazer um ataque furtivo, no máximo reverter, talvez, uma transação da *main pool*, e não o bloco inteiro.

Esse mecanismo de proteção contra modificações é implantado com o uso de um protocolo de consenso fazendo com que a cada transação realizada, os outros nós da rede avaliem o bloco e pelo consenso da maioria, 50% + 1, reconheçam-no como válido. Caso haja divergência, esse “desempate” é resolvido na transação seguinte, fazendo com que a cadeia mais longa prevaleça sobre a cadeia de blocos menor, por isso que depois de 6 blocos criados, ou em média uma hora, se diz que as informações se tornam imutáveis, porque a cada bloco criado é maior a dificuldade e o custo para reverter informações e nesse sentido a blockchain do Bitcoin é a mais longa e a que exige maior poder computacional para ser invadida para reverter blocos, sendo considerada por muitos, a rede mais segura até hoje.

Essas informações são inseridas na rede por computadores superpotentes chamados de mineradores. Eles fazem cálculos matemáticos para resolver uma espécie de quebra-cabeça criptográfico e quando encontram a resposta certa a rede valida resposta, registra o bloco na blockchain e os mineradores recebem Bitcoin como recompensa por terem emprestado seu poder computacional para rodar a rede. Esse mecanismo se chama prova de trabalho ou **Proof of Work** (PoW) na rede Bitcoin e destrava as novas moedas do protocolo, conforme os blocos de informação são criados e funciona como um jogo, onde o computador que resolver o problema criptográfico primeiro recebe os bitcoins da rodada e os mineradores ficam competindo entre si o tempo todo para ver quem chega primeiro na

resposta e recebe os bitcoins do próximo bloco. Só vão ser criados 21 milhões de bitcoins até o ano de 2140, e isso gera escassez e valor à moeda.

Cada bloco contém informações sobre as transações financeiras feitas na rede tipo, endereço A enviou 2 bitcoins para endereço B, e tem uma marcação de tempo, um carimbo de data e hora, chamado **Time Stamp**. Todos esses dados formam o conteúdo de cada bloco e são misturados de forma aleatória e transformados em um hash. Hash é um código criptografado que identifica tudo que está dentro daquele bloco de informação e é a partir do Hash que a mágica funciona.

Depois de criado o Hash 1 do bloco 1 ele vai ser inserido junto com conteúdo do próximo bloco, o bloco 2, que também será misturado de forma aleatória e vai formar justamente o Hash 2, por isso o Hash 2 resume todo o conteúdo do seu Bloco e do bloco anterior porque o Hash 1 foi inserido dentro do bloco 2 e assim sucessivamente o Hash 3 vai ser o resumo criptográfico do bloco 3 que tem também o seu conteúdo o Hash do bloco 2 anterior.

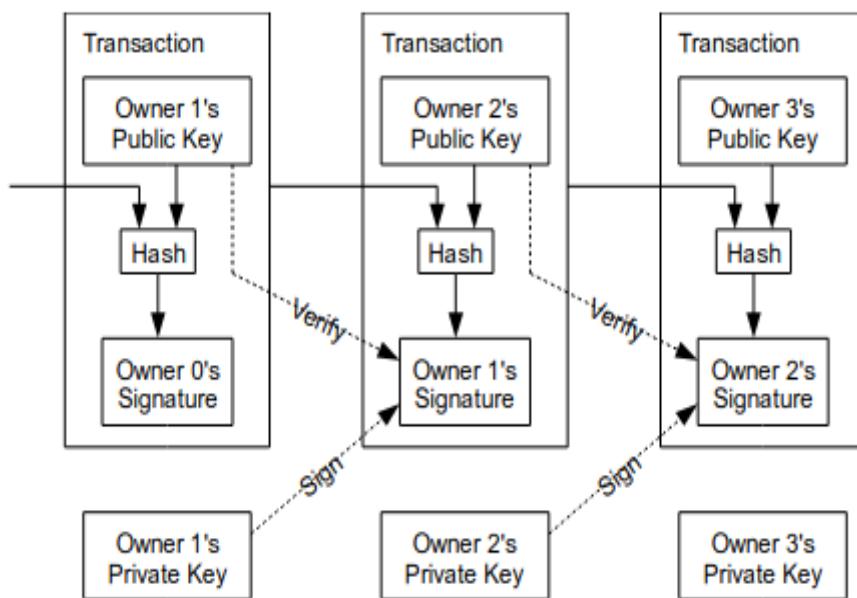


Figura 26 – Cadeia de blocos. [NAKAMOTO] 2008 P. 2

A grande percepção trazida pelo artigo de NAKAMOTO foi a resolução do problema do gasto duplo, impedindo que um mesmo bem pudesse ser transacionado mais de uma vez, gerando escassez e valor e a mudança do centro gravitacional da confiança, onde antes ficava na mão de um terceiro confiável que validasse as operações de troca/compra e passando agora para o indivíduo, que através da criptografia, protocolo de consenso e descentralização da rede tem a autonomia.

7.1.3 CRIPTOGRAFIA E VALIDAÇÃO

Foi mencionado anteriormente que o blockchain utiliza a criptografia como elemento de segurança em sua estrutura. Criptografia nada mais é que uma forma de se ocultar a informação de pessoas não autorizadas. Mas além de ocultar estas informações, no caso, o conteúdo do bloco, é necessário que a rede do blockchain identifique o proprietário, o responsável por aquela transação, e através do uso da criptografia assimétrica conseguimos atingir esses dois objetivos: ocultar os dados de pessoas não autorizadas e garantir a autenticidade.

Podemos dividir as técnicas de criptografia em dois grandes grupos, os que utilizam **chaves simétricas** e os que utilizam **chaves assimétricas**. Resumidamente, os métodos que fazem uso de chaves simétricas utilizam a mesma chave para ocultar (criptografar) e revelar (descriptografar) a mensagem original, fazendo uso de recursos de substituição e permutação. Nesta abordagem, é preciso enviar a senha para o destinatário de forma segura, para que somente a pessoa alvo tenha acesso à chave e consiga revelar a mensagem oculta.

Inicialmente, as técnicas de chaves simétricas faziam uso de algoritmos que podiam ser calculados a mão, passando então por uma grande mudança ao fazer uso do rotor eletromecânico, produzindo, então, algoritmos extremamente complexos, e com o advento

dos computadores esta complexidade aumentou ainda mais, mas ainda permaneciam os problemas de privacidade e autenticação que as chaves simétricas não conseguiam atender eficazmente.

Para atender estas lacunas, as técnicas de **chaves assimétricas** foram introduzidas. Basicamente, esta técnica faz uso de duas chaves (diferente da chave simétrica, que usa apenas uma), uma **chave pública** e a outra **chave privada**, em que a pública é divulgada através de um meio público e a privada fica de posse de seu proprietário. Somente o conjunto desse par de chaves é capaz de ocultar e revelar adequadamente a mensagem original.

Importante ressaltar que não se pode declarar que as técnicas de chaves assimétricas são superiores às de chaves simétricas. O verdadeiro diferencial entre uma técnica e outra é o tamanho e complexidade da chave utilizada. Também não se pode falar que uma técnica substitui a outra, pois para cada cenário há uma técnica que melhor se adapta e ainda há a possibilidade de criar soluções que fazem uso de técnicas de chaves simétricas e chaves assimétricas trabalhando juntas.

O trabalho de STALLINGS (2008) mostra o funcionamento da criptografia de chave pública. Nesta abordagem, se pretende atacar os dois problemas mais difíceis associados à criptografia simétrica, a situação de dois participantes terem que compartilhar a mesma chave entre eles e o problema da autenticação de mensagens.

O compartilhamento da mesma chave entre dois sujeitos compromete a essência da criptografia pois abre a possibilidade da quebra de sigilo da comunicação por um terceiro que por ações como suborno, ameaça ou uma invasão bem sucedida da comunicação

conseguiria acessar um conteúdo de forma indevida originariamente destinada apenas aos integrantes daquele canal de comunicação. No mesmo sentido dos problemas envolvendo a comunicação com chaves simétricas, se na troca de documentos físicos de papel há situações onde é importante ter a certeza de quem enviou o documento como com a presença de uma assinatura manuscrita, este mesmo comportamento pode ser transferido para os cenários de troca de documentos de forma digital, onde a presença de uma assinatura identificando de forma única um participante do canal de comunicação é capaz de ser o diferencial entre uma transação válida ou não.

Exatamente neste sentido é que as chaves assimétricas conseguem trazer robustez a uma comunicação criptografada, além de prover mecanismos de segurança diferenciados e a autenticidade. O quadro abaixo mostra, em linhas gerais, as diferenças entre o sistema de criptografia convencional e criptografia de chave pública.

Criptografia convencional	Criptografia de chave pública
<p>Necessário para funcionar:</p> <ol style="list-style-type: none"> 1. O mesmo algoritmo com a mesma chave é usado para criptografia e decriptografia. 2. O emissor e o receptor precisam compartilhar o algoritmo e a chave. <p>Necessário para a segurança:</p> <ol style="list-style-type: none"> 1. A chave precisa permanecer secreta. 2. Deverá ser impossível ou pelo menos impraticável decifrar uma mensagem se nenhuma outra informação estiver disponível. 3. O conhecimento do algoritmo mais amostras do texto cifrado precisam ser insuficientes para determinar a chave. 	<p>Necessário para funcionar:</p> <ol style="list-style-type: none"> 1. Um algoritmo é usado para criptografia e decriptografia com um par de chaves, uma para criptografia e outra para decriptografia. 2. O emissor e o receptor precisam ter uma das chaves do par casado de chaves (não a mesma chave). <p>Necessário para a segurança:</p> <ol style="list-style-type: none"> 1. Uma das duas chaves precisa permanecer secreta. 2. Deverá ser impossível ou pelo menos impraticável decifrar uma mensagem se nenhuma outra informação estiver disponível. 3. O conhecimento do algoritmo mais uma das chaves mais amostras do texto cifrado precisam ser insuficientes para determinar a outra chave.

Figura 27 – Diferentes tipos de criptografia. [STALLINGS] 2008 P. 184

Um sistema criptográfico de chave pública possui 5 componentes básicos:

- **Texto claro:** mensagem ou dados que são alimentados no algoritmo como entrada
- **Algoritmo de Criptografia:** processamento matemático e computacional que

transforma o texto claro

- **Chaves pública e privada:** par de chaves que são usadas para criptografia e descriptografia.
- **Texto cifrado:** mensagem codificada produzida como saída do resultado do processamento entre texto claro, algoritmo e chave (pública ou privada)
- **Algoritmo de descriptografia:** utilizando o texto cifrado e a chave, entrega como resultado o texto claro

As etapas essenciais deste processo são:

- Cada usuário gera um par de chaves a ser utilizado para criptografia e descriptografia das mensagens
- Cada usuário coloca uma das duas chaves em um registro ou arquivo público acessível
- Se Carlos deseja enviar mensagem confidencial para Tales, Carlos criptografa usando a chave pública de Tales
- Quando Tales recebe a mensagem, ele a descriptografa usando sua chave privada.

As duas imagens abaixo mostram a dinâmica do uso da criptografia de chave pública para fins de sigilo e autenticação. Na primeira imagem, há uma mensagem que é criptografada utilizando a chave pública de Alice, assim, se a chave privada de Alice for mantida em sigilo, apenas Alice conseguirá fazer a descriptografia da mensagem e ver o texto claro através do uso de sua chave privada. Com esta abordagem se atinge o sigilo da mensagem, garantindo que apenas o autor e o receptor devido tenham conhecimento do

conteúdo

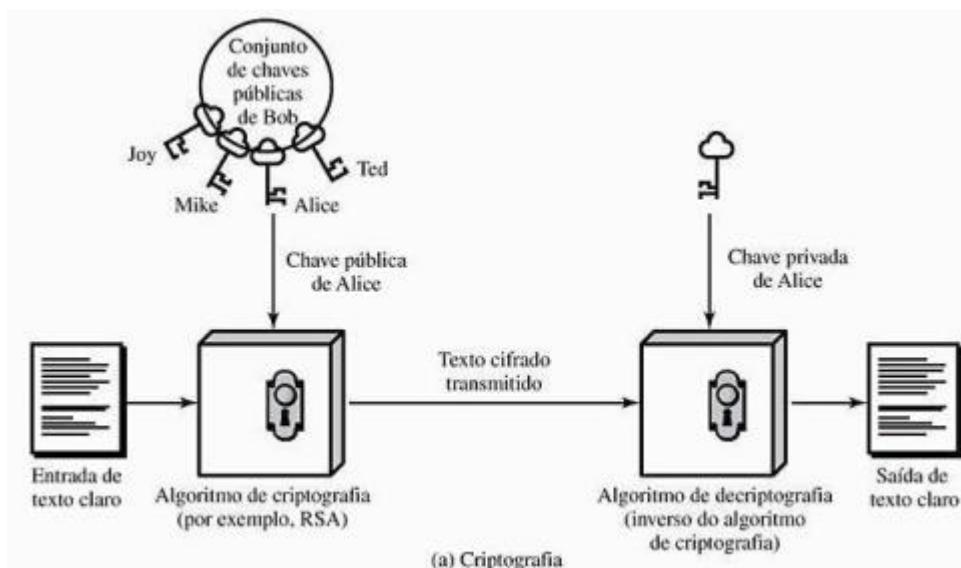


Figura 28 - Criptografia. [STALLINGS] 2008 P. 184

Na imagem seguinte vemos o processo inverso, agora utilizando a chave privada de Bob. A partir de uma mensagem de texto claro é feita a criptografia utilizando a chave privada de Bob. Após o envio da mensagem, aqueles que tiverem a chave pública de Bob poderão ver o conteúdo da mensagem. Com esta abordagem se atinge o objetivo da autenticidade, pois independente de quem acesse a mensagem, como tem a chave pública de Bob, automaticamente, tem a certeza de que a mensagem teve Bob como remetente.

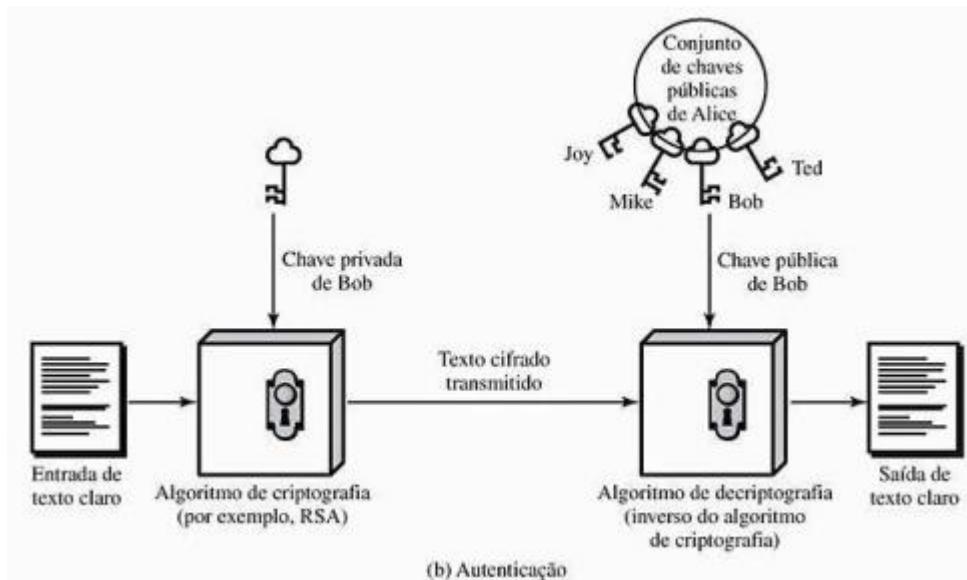


Figura 29 - Autenticação. [STALLINGS] 2008 P. 184

Vale ressaltar que várias combinações podem ser utilizadas dentro deste cenário, inclusive com chaves pública e privada para criptografar e descriptografar, atingindo, então, os alvos de sigilo e autenticação. A imagem abaixo traz exatamente este cenário onde uma mensagem original é criptografada com a chave privada de **A**, em seguida passa por uma nova criptografia com a chave pública de **B** e é transmitida até seu destino. No destino, **B** faz a primeira descriptografia com sua chave privada em seguida uma nova descriptografia com a chave pública de **A** entrega novamente o texto claro.

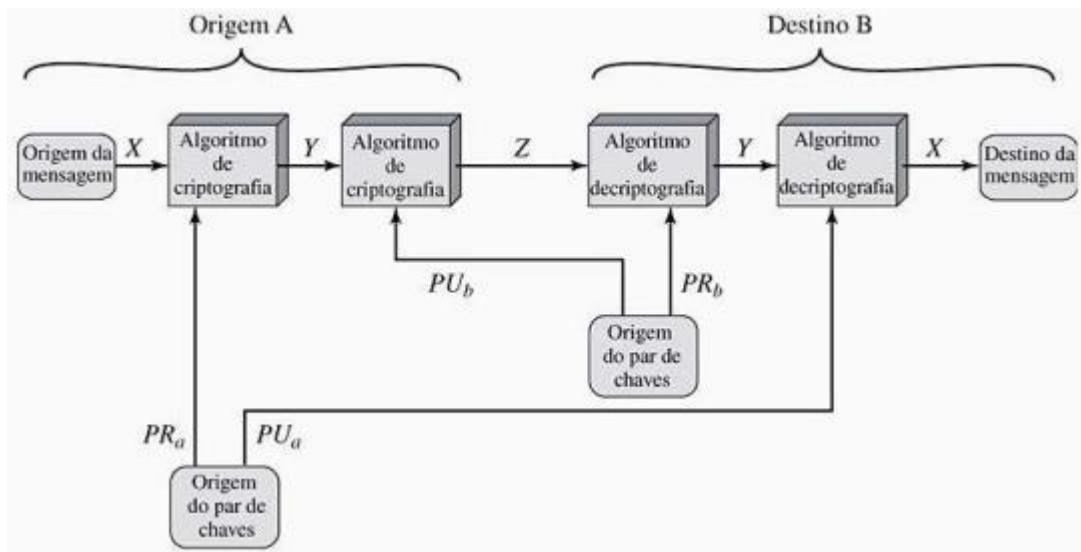


Figura 30 – Criptografia e autenticação. [STALLINGS] 2008 P. 186

Um outro cenário possível, ainda, é a utilização de uma combinação de sistemas de criptografia de chave simétrica em conjunto com outro de chave assimétrica. Enfim, as combinações entre estes elementos são inúmeras e a melhor estratégia deve estar adequada a cada projeto em particular.

Isto posto, tipicamente, uma rede blockchain registra a autenticidade das transações através dos endereços delas e estes endereços nada mais são do que a chave pública de cada uma das pessoas que interagiram na rede, deixando ali sua identidade, sua impressão digital. O site **Blockchain.com** mostra as transações realizadas na rede do blockchain. Nele é possível verificar os detalhes de uma transação na rede do Bitcoin, com seus registros de endereço de origem e destino, Hash, hora e valor. A imagem abaixo mostra esse detalhe.



Figura 31 – Transação na rede Bitcoin.¹⁰

Na figura acima temos algumas informações relevantes:

- Em verde – o Hash da transação ocorrida;
- Em laranja – a data da transação;
- Em roxo – o valor da transação;
- Em vermelho – o endereço de origem do bitcoin;
- Em azul – o endereço de destino do bitcoin.

Os endereços de origem e destino da transação não identificam propriamente um proprietário, mas sim uma carteira, e esta carteira possui um proprietário. Desta maneira é que são rastreadas todas as transações do blockchain, identificando seus endereços de origem e destino.

Importante ressaltar que os registros na rede blockchain podem ser verificados por qualquer pessoa, garantindo a publicidade e verificação das transações, mas a identidade, nome, CPF, RG, sexo, idade, CNPJ ou qualquer outro identificador civil e pessoal de quem

¹⁰ <https://www.blockchain.com/btc/tx/88dcb0188df671f0ae5154780382c89fb062582186a6038a585880945f3cbbc0>

fez a transação ficam resguardados, garantindo a privacidade dos utilizadores da rede. É possível identificar, sim, a carteira que movimentou a transação, mas não é possível identificar pessoalmente o proprietário da carteira. Qualquer pessoa com o endereço dessa transação

(<https://www.blockchain.com/btc/tx/88dcb0188df671f0ae5154780382c89fb062582186a6038a585880945f3cb0>) ocorrida na rede poderá confirmar tais dados assim como foi apresentado na figura acima.

7.1.4 MECANISMO DE HASH

Conforme menciona STALLINGS, um mecanismo de autenticação de mensagens atua em dois níveis, no primeiro, ele funciona como um autenticador, ou seja, um valor que identifica tal mensagem como autêntica; já no segundo nível, atua como um mecanismo de não repúdio, fazendo conhecer o verdadeiro autor da mensagem. Neste sentido, uma **Função de Hash** relaciona uma mensagem de qualquer tamanho a um valor de tamanho fixo que serve como autenticador.

Conceitualmente, uma função de hash pode ser matematicamente representada pela expressão:

$$h = H(M)$$

Sendo:

h → valor de hash resultante

M → mensagem de comprimento variável

H(M) → função de hash de comprimento fixo

Nestes termos, um código Hash não usa uma chave, sendo uma função apenas de mensagem de entrada. Aliada à criptografia, uma função hash pode atuar como solução para critérios de sigilo, autenticidade da mensagem, não repúdio e controle de erros. A figura abaixo ilustra este exemplo de aplicação, onde **A** compartilha sua chave pública com **B**, que ao receber a mensagem precisa confirmar se ela não foi alterada durante o envio e se **A** é realmente o emissor de tal mensagem.

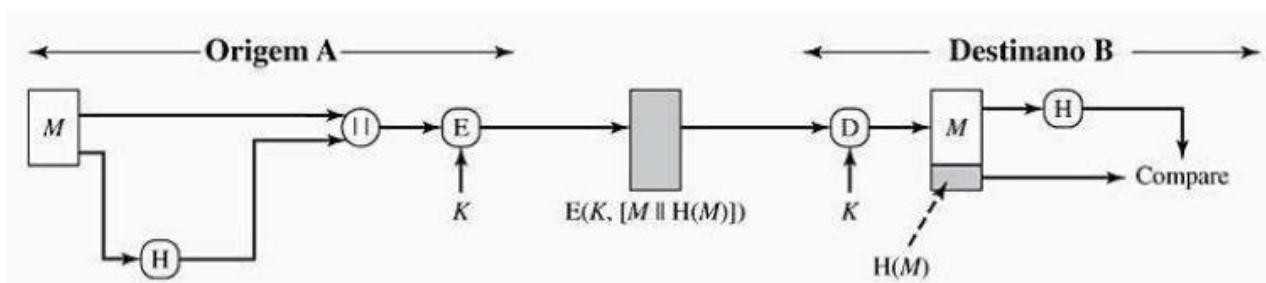


Figura 32 – Mecanismo de Hash. [STALLINGS] 2008 P. 234.

Ainda de acordo com STALLINGS, uma função de Hash **H** precisa ter as seguintes propriedades:

1. **H** pode ser aplicado a uma entrada de qualquer tamanho;
2. **H** produz uma saída de tamanho fixo;
3. **H(M)** é relativamente fácil de calcular para qualquer **M**;
4. Para um valor de **h** dado, deve ser computacionalmente inviável encontrar **M**, atendendo o princípio da **resistência à primeira inversão** ou **propriedade unidirecional**;
5. Para um bloco **M**, deve ser computacionalmente inviável encontrar **M' ≠ M** tal que **H(M') = H(M)**, atendendo à **resistência à segunda inversão** ou **resistência fraca a colisões**;

6. Deve ser computacionalmente inviável encontrar o par ordenado (M, M') tal que $H(M) = H(M')$, atendendo à **resistência a colisões** ou **resistência forte a colisões**.

No caso específico do Bitcoin, a função Hash utilizada é a **SHA-256**, que independentemente do tamanho da entrada, entrega como resultado **256 bits** que contém **64 símbolos** alfanuméricos. O código abaixo representa um exemplo de aplicação de uma função Hash que para a mensagem **Blockchain is innovative** encontra como resultado o seguinte

Hash:

0a94aa3f0c84ccc061bef0eb59903903732800eed9828c2f5c04c19ff3f329e9.

SHA256 Hash



Figura 33 – Função Hash 256. <https://andersbrownworth.com/blockchain/hash>

Para ilustrar esse mecanismo de funcionamento do Hash, o site <https://andersbrownworth.com/blockchain/coinbase> possui um simulador onde mostra uma cadeia com 5 blocos gerenciada por uma rede com 3 nós. Em cada um desses nós há uma cópia da cadeia de blocos, de forma que se algum fraudador tentar mudar a cadeia de blocos administrada por seu nó, será desmascarado pela simples comparação de sua cadeia com as demais cadeias de blocos da rede. Vamos considerar a cadeia de 5 blocos representada pelos blocos abaixo:

Block:	#	1
Nonce:	16651	
Coinbase:	\$ 100.00	-> Anders
Tx:		
Prev:	00	
Hash:	0000438d7625b86a6f366545b1929975a0d3ff1f88- <div style="border: 2px solid red; padding: 2px;"></div>	
	<button>Mine</button>	

Block:	#	2
Nonce:	215458	
Coinbase:	\$ 100.00	-> Anders
Tx:	\$ 10.00	From: Anders -> Sophia
	\$ 20.00	From: Anders -> Lucas
	\$ 15.00	From: Anders -> Emily
	\$ 15.00	From: Anders -> Madison
Prev:	0000438d7625b86a6f366545b1929975a0d3ff1f88-	
Hash:	0000baeab68c2a60f9a6fa56355438d97c672a1549-	
Mine		

Block:	#	3
Nonce:	146	
Coinbase:	\$ 100.00	-> Anders
Tx:	\$ 10.00	From: Emily -> Jackson
	\$ 5.00	From: Madison -> Jackson
	\$ 20.00	From: Lucas -> Grace
Prev:	0000baeab68c2a60f9a6fa56355438d97c672a1549.	
Hash:	0000df1d632b734f5a5fc126a0f0e8894fb4c8314b.	
Mine		

Block:	#	4
Nonce:	18292	
Coinbase:	\$ 100.00	-> Anders
Tx:	\$ 15.00	From: Jackson -> Ryan
	\$ 5.00	From: Emily -> Madisor
	\$ 8.00	From: Sophia -> Jackson
Prev:	0000df1d632b734f5a5fc126a0f0e8894fb4c8314b	
Hash:	0000c694336f88129f3685bd3ba5d67c445dfd8d18	
	Mine	

Block:	#	5
Nonce:	108899	
Coinbase:	\$ 100.00	-> Sophia
Tx:	\$ 2.00	From: Jacksor -> Alexander
	\$ 6.00	From: Ryan -> Carter
	\$ 4.00	From: Ryan -> Riley
	\$ 9.95	From: Grace -> Katherine
Prev:	0000c694336f88129f3685bd3ba5d67c445fdf8d181	
Hash:	0000fc766a6850dbb98d0cd354c302249ec291df36	
Mine		

Figura 34 – Cadeia de blocos do primeiro nó. <https://andersbrownworth.com/blockchain/coinbase>

A cada bloco notamos a informação do hash do bloco atual e a informação do hash do bloco anterior, fazendo a referência direta entre blocos, sendo o primeiro bloco um bloco especial, reconhecido como bloco gênese, que no campo de hash anterior tem o valor preenchido com zeros, e para cada bloco, para que seu hash seja considerado válido é preciso realizar as operações matemáticas necessárias para que seu hash seja iniciado com 4 zeros (**0000**).

Cada um desses blocos tem um conjunto de transações, e o hash é o resultado matemático do fruto dessas operações e do bloco anterior. Como resultado, este mesmo conjunto de blocos é registrado nos outros três nós da cadeia. Note que os hash dos blocos são iguais em qualquer um dos três nós, evidenciando o registro descentralizado dos dados em qualquer um dos nós da rede.

Block:	#	4
Nonce:	18292	
Coinbase:	\$ 100.00	-> Anders
Tx:	\$ 15.00	From: Jacksor -> Ryan
	\$ 5.00	From: Emily -> Madisor
	\$ 8.00	From: Sophia -> Jacksor
Prev:	0000df1d632b734f5a5fc126a0f0e8894fb4c8314b	
Hash:	0000c694336f88129f3685bd3ba5d67c445dfd8d18	
Mine		

Block:	#	5
Nonce:	108899	
Coinbase:	\$ 100.00	-> Sophia
Tx:	\$ 2.00	From: Jacksor -> Alexand
	\$ 6.00	From: Ryan -> Carter
	\$ 4.00	From: Ryan -> Riley
	\$ 9.95	From: Grace -> Katheri
Prev:	0000c694336f88129f3685bd3ba5d67c445dfd8d18	
Hash:	0000fc766a6850dbb98d0cd354c302249ec291df36	
Mine		

Figura 35 – Blocos 4 e 5 do segundo nó. <https://andersbrownworth.com/blockchain/coinbase>

Block:	#	4						
Nonce:	18292							
Coinbase:	\$ 100.00		->	Anders				
Tx:	\$ 15.00	From:	Jackson	->	Ryan			
	\$ 5.00	From:	Emily	->	Madisor			
	\$ 8.00	From:	Sophia	->	Jackson			
Prev:	0000df1d632b734f5a5fc126a0f0e8894fb4c8314b							
Hash:	0000c694336f88129f3685bd3ba5d67c445dfdf8d18							
	Mine							

Block:	#	5						
Nonce:	108899							
Coinbase:	\$ 100.00		->	Sophia				
Tx:	\$ 2.00	From:	Jackson	->	Alexand			
	\$ 6.00	From:	Ryan	->	Carter			
	\$ 4.00	From:	Ryan	->	Riley			
	\$ 9.95	From:	Grace	->	Katheri			
Prev:	0000c694336f88129f3685bd3ba5d67c445dfdf8d18							
Hash:	0000fc766a6850dbb98d0cd354c302249ec291df36							
	Mine							

Figura 36 – Blocos 4 e 5 do terceiro nó. <https://andersbrownworth.com/blockchain/coinbase>

Se em nossa simulação de rede blockchain modificarmos qualquer dado de um bloco, isso comprometerá toda a cadeia de blocos na sequência, evidenciando a manipulação. Para exemplificar, vamos alterar uma transação do bloco3 do segundo nó. Vamos modificar a transação realizada entre Madison e Jackson de **5,00** para **5,01**. Note que o seu hash será modificado, devido à mudança de um dado do bloco. Uma mudança pequena, é verdade, mas suficiente para denunciar que algo foi modificado.

Block:	# 3												
Nonce:	146												
Coinbase:	\$ 100.00 -> Anders												
Tx:	<table border="1"> <tr> <td>\$ 10.00</td> <td>From: Emily</td> <td>-></td> <td>Jackson</td> </tr> <tr> <td>\$ 5.00</td> <td>From: Madison</td> <td>-></td> <td>Jackson</td> </tr> <tr> <td>\$ 20.00</td> <td>From: Lucas</td> <td>-></td> <td>Grace</td> </tr> </table>	\$ 10.00	From: Emily	->	Jackson	\$ 5.00	From: Madison	->	Jackson	\$ 20.00	From: Lucas	->	Grace
\$ 10.00	From: Emily	->	Jackson										
\$ 5.00	From: Madison	->	Jackson										
\$ 20.00	From: Lucas	->	Grace										
Prev:	0000baeab68c2a60f9a6fa56355438d97c672a1549-												
Hash:	0000df1d632b734f5a5fc126a0f0e8894fb4c8314b-												
	<button>Mine</button>												

Block:	# 4												
Nonce:	18292												
Coinbase:	\$ 100.00 -> Anders												
Tx:	<table border="1"> <tr> <td>\$ 15.00</td> <td>From: Jackson</td> <td>-></td> <td>Ryan</td> </tr> <tr> <td>\$ 5.00</td> <td>From: Emily</td> <td>-></td> <td>Madison</td> </tr> <tr> <td>\$ 8.00</td> <td>From: Sophia</td> <td>-></td> <td>Jackson</td> </tr> </table>	\$ 15.00	From: Jackson	->	Ryan	\$ 5.00	From: Emily	->	Madison	\$ 8.00	From: Sophia	->	Jackson
\$ 15.00	From: Jackson	->	Ryan										
\$ 5.00	From: Emily	->	Madison										
\$ 8.00	From: Sophia	->	Jackson										
Prev:	0000df1d632b734f5a5fc126a0f0e8894fb4c8314b-												
Hash:	0000c694336f88129f3685bd3ba5d67c445dfd8d18-												
	<button>Mine</button>												

Figura 37 – Blocos 3 e 4 originais do segundo nó. <https://andersbrownworth.com/blockchain/coinbase>

Block:	# 3												
Nonce:	146												
Coinbase:	\$ 100.00 -> Anders												
Tx:	<table border="1"> <tr> <td>\$ 10.00</td> <td>From: Emily</td> <td>-></td> <td>Jackson</td> </tr> <tr> <td>\$ 5.01</td> <td>From: Madison</td> <td>-></td> <td>Jackson</td> </tr> <tr> <td>\$ 20.00</td> <td>From: Lucas</td> <td>-></td> <td>Grace</td> </tr> </table>	\$ 10.00	From: Emily	->	Jackson	\$ 5.01	From: Madison	->	Jackson	\$ 20.00	From: Lucas	->	Grace
\$ 10.00	From: Emily	->	Jackson										
\$ 5.01	From: Madison	->	Jackson										
\$ 20.00	From: Lucas	->	Grace										
Prev:	0000baeab68c2a60f9a6fa56355438d97c672a1549-												
Hash:	8fbe07a5d117b586229957e00295731f263ad8b480-												
	<button>Mine</button>												

Block:	# 4												
Nonce:	18292												
Coinbase:	\$ 100.00 -> Anders												
Tx:	<table border="1"> <tr> <td>\$ 15.00</td> <td>From: Jackson</td> <td>-></td> <td>Ryan</td> </tr> <tr> <td>\$ 5.00</td> <td>From: Emily</td> <td>-></td> <td>Madison</td> </tr> <tr> <td>\$ 8.00</td> <td>From: Sophia</td> <td>-></td> <td>Jackson</td> </tr> </table>	\$ 15.00	From: Jackson	->	Ryan	\$ 5.00	From: Emily	->	Madison	\$ 8.00	From: Sophia	->	Jackson
\$ 15.00	From: Jackson	->	Ryan										
\$ 5.00	From: Emily	->	Madison										
\$ 8.00	From: Sophia	->	Jackson										
Prev:	8fbe07a5d117b586229957e00295731f263ad8b480-												
Hash:	b9adccf4f7a7250aa6e5aea0d4f94e9563e3fa7b96-												
	<button>Mine</button>												

Figura 38 – Blocos 3 e 4 modificados do segundo nó. <https://andersbrownworth.com/blockchain/coinbase>

Perceba que mesmo com a pequena alteração aplicada de **0,01**, o bloco já teve o seu hash modificado, onde antes da alteração era **0000df1d...**, passando para **8fbe....**. Essa modificação altera todos os blocos seguintes da cadeia, fazendo com que seja identificada, de imediato, uma mudança.

<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>Block:</td><td>#</td><td>4</td></tr> <tr><td>Nonce:</td><td colspan="2">18292</td></tr> <tr><td>Coinbase:</td><td>\$ 100.00</td><td>-></td><td>Anders</td></tr> <tr><td>Tx:</td><td>\$ 15.00</td><td>From:</td><td>Jackson</td><td>-></td><td>Ryan</td></tr> <tr><td></td><td>\$ 5.00</td><td>From:</td><td>Emily</td><td>-></td><td>Madison</td></tr> <tr><td></td><td>\$ 8.00</td><td>From:</td><td>Sophia</td><td>-></td><td>Jackson</td></tr> <tr><td>Prev:</td><td colspan="5">8fbe07a5d117b586229957e00295731f263ad8b480</td></tr> <tr><td>Hash:</td><td colspan="5" rowspan="2">b9adccf4f7a7250aa6e5aea0d4f94e9563e3fa7b96</td></tr> <tr><td colspan="6" style="text-align: center;"><button>Mine</button></td></tr> </table>	Block:	#	4	Nonce:	18292		Coinbase:	\$ 100.00	->	Anders	Tx:	\$ 15.00	From:	Jackson	->	Ryan		\$ 5.00	From:	Emily	->	Madison		\$ 8.00	From:	Sophia	->	Jackson	Prev:	8fbe07a5d117b586229957e00295731f263ad8b480					Hash:	b9adccf4f7a7250aa6e5aea0d4f94e9563e3fa7b96					<button>Mine</button>						<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>Block:</td><td>#</td><td>5</td></tr> <tr><td>Nonce:</td><td colspan="2">108899</td></tr> <tr><td>Coinbase:</td><td>\$ 100.00</td><td>-></td><td>Sophia</td></tr> <tr><td>Tx:</td><td>\$ 2.00</td><td>From:</td><td>Jackson</td><td>-></td><td>Alexander</td></tr> <tr><td></td><td>\$ 6.00</td><td>From:</td><td>Ryan</td><td>-></td><td>Carter</td></tr> <tr><td></td><td>\$ 4.00</td><td>From:</td><td>Ryan</td><td>-></td><td>Riley</td></tr> <tr><td></td><td>\$ 9.95</td><td>From:</td><td>Grace</td><td>-></td><td>Katheri</td></tr> <tr><td>Prev:</td><td colspan="5">b9adccf4f7a7250aa6e5aea0d4f94e9563e3fa7b96</td></tr> <tr><td>Hash:</td><td colspan="5" rowspan="2">0d0ce9aeab89dc07a2da5675170b7b6c979204764c1</td></tr> <tr><td colspan="6" style="text-align: center;"><button>Mine</button></td></tr> </table>	Block:	#	5	Nonce:	108899		Coinbase:	\$ 100.00	->	Sophia	Tx:	\$ 2.00	From:	Jackson	->	Alexander		\$ 6.00	From:	Ryan	->	Carter		\$ 4.00	From:	Ryan	->	Riley		\$ 9.95	From:	Grace	->	Katheri	Prev:	b9adccf4f7a7250aa6e5aea0d4f94e9563e3fa7b96					Hash:	0d0ce9aeab89dc07a2da5675170b7b6c979204764c1					<button>Mine</button>					
Block:	#	4																																																																																																	
Nonce:	18292																																																																																																		
Coinbase:	\$ 100.00	->	Anders																																																																																																
Tx:	\$ 15.00	From:	Jackson	->	Ryan																																																																																														
	\$ 5.00	From:	Emily	->	Madison																																																																																														
	\$ 8.00	From:	Sophia	->	Jackson																																																																																														
Prev:	8fbe07a5d117b586229957e00295731f263ad8b480																																																																																																		
Hash:	b9adccf4f7a7250aa6e5aea0d4f94e9563e3fa7b96																																																																																																		
<button>Mine</button>																																																																																																			
Block:	#	5																																																																																																	
Nonce:	108899																																																																																																		
Coinbase:	\$ 100.00	->	Sophia																																																																																																
Tx:	\$ 2.00	From:	Jackson	->	Alexander																																																																																														
	\$ 6.00	From:	Ryan	->	Carter																																																																																														
	\$ 4.00	From:	Ryan	->	Riley																																																																																														
	\$ 9.95	From:	Grace	->	Katheri																																																																																														
Prev:	b9adccf4f7a7250aa6e5aea0d4f94e9563e3fa7b96																																																																																																		
Hash:	0d0ce9aeab89dc07a2da5675170b7b6c979204764c1																																																																																																		
<button>Mine</button>																																																																																																			

Figura 39 – Blocos 4 e 5 afetados pela modificação do bloco 3 do segundo nó.
<https://andersbrownworth.com/blockchain/coinbase>

Ao compararmos estes mesmos blocos nos outros nós da rede percebemos que eles não foram alterados.

<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>Block:</td><td>#</td><td>3</td></tr> <tr><td>Nonce:</td><td colspan="2">146</td></tr> <tr><td>Coinbase:</td><td>\$ 100.00</td><td>-></td><td>Anders</td></tr> <tr><td>Tx:</td><td>\$ 10.00</td><td>From:</td><td>Emily</td><td>-></td><td>Jackson</td></tr> <tr><td></td><td>\$ 5.00</td><td>From:</td><td>Madison</td><td>-></td><td>Jackson</td></tr> <tr><td></td><td>\$ 20.00</td><td>From:</td><td>Lucas</td><td>-></td><td>Grace</td></tr> <tr><td>Prev:</td><td colspan="5">0000baeab68c2a60f9a6fa56355438d97c672a1549</td></tr> <tr><td>Hash:</td><td colspan="5" rowspan="2">0000df1d632b734f5a5fc126a0f0e8894fb4c8314b</td></tr> <tr><td colspan="6" style="text-align: center;"><button>Mine</button></td></tr> </table>	Block:	#	3	Nonce:	146		Coinbase:	\$ 100.00	->	Anders	Tx:	\$ 10.00	From:	Emily	->	Jackson		\$ 5.00	From:	Madison	->	Jackson		\$ 20.00	From:	Lucas	->	Grace	Prev:	0000baeab68c2a60f9a6fa56355438d97c672a1549					Hash:	0000df1d632b734f5a5fc126a0f0e8894fb4c8314b					<button>Mine</button>						<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>Block:</td><td>#</td><td>4</td></tr> <tr><td>Nonce:</td><td colspan="2">18292</td></tr> <tr><td>Coinbase:</td><td>\$ 100.00</td><td>-></td><td>Anders</td></tr> <tr><td>Tx:</td><td>\$ 15.00</td><td>From:</td><td>Jackson</td><td>-></td><td>Ryan</td></tr> <tr><td></td><td>\$ 5.00</td><td>From:</td><td>Emily</td><td>-></td><td>Madison</td></tr> <tr><td></td><td>\$ 8.00</td><td>From:</td><td>Sophia</td><td>-></td><td>Jackson</td></tr> <tr><td>Prev:</td><td colspan="5">0000df1d632b734f5a5fc126a0f0e8894fb4c8314b</td></tr> <tr><td>Hash:</td><td colspan="5" rowspan="2">0000c694336f88129f3685bd3ba5d67c445dfd8d18</td></tr> <tr><td colspan="6" style="text-align: center;"><button>Mine</button></td></tr> </table>	Block:	#	4	Nonce:	18292		Coinbase:	\$ 100.00	->	Anders	Tx:	\$ 15.00	From:	Jackson	->	Ryan		\$ 5.00	From:	Emily	->	Madison		\$ 8.00	From:	Sophia	->	Jackson	Prev:	0000df1d632b734f5a5fc126a0f0e8894fb4c8314b					Hash:	0000c694336f88129f3685bd3ba5d67c445dfd8d18					<button>Mine</button>					
Block:	#	3																																																																																											
Nonce:	146																																																																																												
Coinbase:	\$ 100.00	->	Anders																																																																																										
Tx:	\$ 10.00	From:	Emily	->	Jackson																																																																																								
	\$ 5.00	From:	Madison	->	Jackson																																																																																								
	\$ 20.00	From:	Lucas	->	Grace																																																																																								
Prev:	0000baeab68c2a60f9a6fa56355438d97c672a1549																																																																																												
Hash:	0000df1d632b734f5a5fc126a0f0e8894fb4c8314b																																																																																												
<button>Mine</button>																																																																																													
Block:	#	4																																																																																											
Nonce:	18292																																																																																												
Coinbase:	\$ 100.00	->	Anders																																																																																										
Tx:	\$ 15.00	From:	Jackson	->	Ryan																																																																																								
	\$ 5.00	From:	Emily	->	Madison																																																																																								
	\$ 8.00	From:	Sophia	->	Jackson																																																																																								
Prev:	0000df1d632b734f5a5fc126a0f0e8894fb4c8314b																																																																																												
Hash:	0000c694336f88129f3685bd3ba5d67c445dfd8d18																																																																																												
<button>Mine</button>																																																																																													

Figura 40 – Blocos 3 e 4 do primeiro nó. <https://andersbrownworth.com/blockchain/coinbase>

Block:	#	3
Nonce:	146	
Coinbase:	\$ 100.00	-> Anders
Tx:	\$ 10.00	From: Emily -> Jacksor
	\$ 5.00	From: Madison -> Jacksor
	\$ 20.00	From: Lucas -> Grace
Prev:	0000baeab68c2a60f9a6fa56355438d97c672a1549	
Hash:	0000df1d632b734f5a5fc126a0f0e8894fb4c8314b	
	Mine	

Block:	#	4
Nonce:	18292	
Coinbase:	\$ 100.00	-> Anders
Tx:	\$ 15.00	From: Jackson -> Ryan
	\$ 5.00	From: Emily -> Madisor
	\$ 8.00	From: Sophia -> Jacksor
Prev:	0000df1d632b734f5a5fc126a0f0e8894fb4c8314b	
Hash:	0000c694336f88129f3685bd3ba5d67c445dfd8d18	
	Mine	

Figura 41 – Blocos 3 e 4 do terceiro nó. <https://andersbrownworth.com/blockchain/coinbase>

Ainda quanto ao segundo nó, mais especificamente o bloco alterado, mesmo após minerado, com o Hash de seu bloco corrigido, os blocos 4 e 5 desta cadeia continuarão denunciando a modificação.

Block:	#	3
Nonce:	5490	
Coinbase:	\$ 100.00	-> Anders
Tx:	\$ 10.00	From: Emily -> Jacksor
	\$ 5.01	From: Madisor -> Jacksor
	\$ 20.00	From: Lucas -> Grace
Prev:	0000baeab68c2a60f9a6fa56355438d97c672a1549	
Hash:	0000feab7bea9f68940c57d0c0a045fc07377c34bd	
	Mine	

Block:	#	4
Nonce:	18292	
Coinbase:	\$ 100.00	-> Anders
Tx:	\$ 15.00	From: Jackson -> Ryan
	\$ 5.00	From: Emily -> Madisor
	\$ 8.00	From: Sophia -> Jacksor
Prev:	0000feab7bea9f68940c57d0c0a045fc07377c34bd	
Hash:	9635e848346e6b3c4054ac1767043552712b63d2f5	
	Mine	

Figura 42 – Blocos 3 e 4 do segundo nó após a mineração do bloco 3.
<https://andersbrownworth.com/blockchain/coinbase>

Note que agora, o bloco se inicia com **0000** sendo esta sequência de 4 zeros o conceito validador para os blocos desta cadeia, diferente do valor de hash que tinha antes (**8fbe...**), mas como o bloco 4 tem em seu registro o hash do bloco 3 anterior à manipulação realizada, apesar de o bloco 3 estar validado nesta cadeia os blocos 4 e 5 denunciam que houve alguma alteração em um bloco (no bloco 3).

Se o fraudador da rede em nosso exemplo quisesse ter sucesso efetivo em seu golpe ele teria que gastar processamento para minerar os blocos 4 e 5 da rede do segundo nó e ainda os blocos 3, 4 e 5 da cadeia do primeiro ou terceiro nó para que sua manipulação de dados atinja mais de 50% da rede, e tudo isso num curto espaço de tempo, pois se os nós 1 e 2 realizassem uma nova transação, todo o seu trabalho seria em vão, pois precisaria modificar o novo bloco inserido na sua rede e mais os outros blocos dos nós adjacentes. Esse mecanismo faz com que os mineradores sejam muito mais estimulados a manter a rede íntegra do que a corrompê-la, pois é perfeitamente rastreável o ponto onde a fraude, ou pelo menos, tentativa dela ocorreu.

7.1.5 ÁRVORE DE MERKLE

Até aqui creio que tenha ficado claro que o bloco guarda um conjunto de transações e estas transações vão se acumulando no tempo e os blocos se relacionam entre si tendo um único bloco pai, exceto no caso do primeiro bloco da cadeia que é conhecido como bloco gênese, que por ser o primeiro, não possui seu bloco pai, tendo no campo da informação do Hash do bloco anterior um conjunto com 64 zeros (ZHENG, XIE, DAI, CHEN, WANG, 2017).

A função Hash é capaz de grandes trunfos e na rede blockchain ela é muito bem utilizada para fazer a checagem de grande quantidade de dados organizados. A estratégia

utilizada faz uso de uma estrutura chamada Árvore de Merkle (*Merkle Tree*) relacionando seus identificadores em uma estrutura única em forma de dados (KUNTZ).

Proposta por Ralph Merkle em 1979, essa estrutura agrupa nós hierarquicamente, de forma que o identificador de cada nó é formado a partir de seus nós filhos recursivamente até chegar ao identificador único dos nós, identificado como Raiz de Merkle (*Merkle Root*).

A Árvore de Merkle acaba trazendo algumas vantagens, entre elas, num cenário onde há grande número de transações, não se faz necessário enviar o Hash de cada uma das transações para que uma transação seja validada, economizando volume de dados trafegados entre os nós da rede.

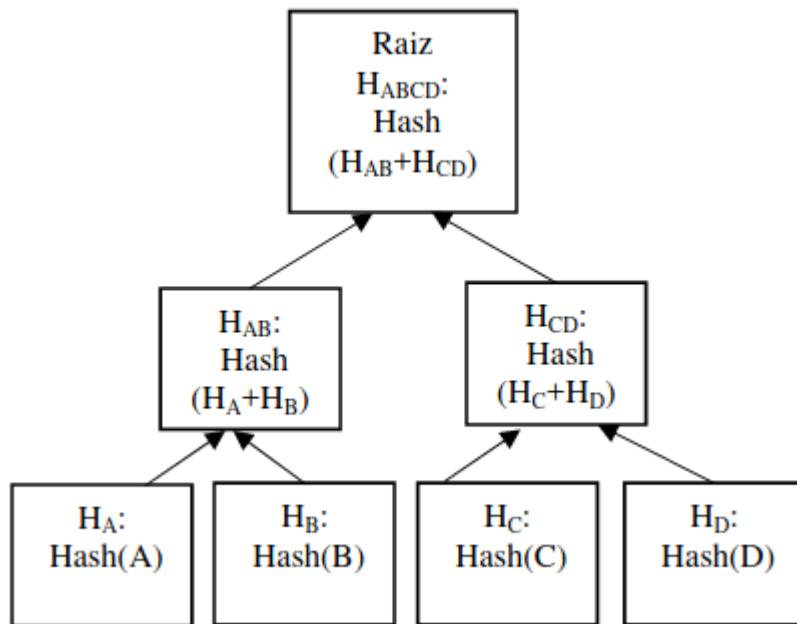


Figura 43 – Árvore de Merkle para 4 transações A, B, C e D. [RODRIGUES] 2017 P. 152

Na figura acima é mostrada a árvore de Merkle para um conjunto de 4 transações, **A**, **B**, **C** e **D**. Vamos adiantar aqui um conceito da rede descentralizada que é de nós fracos e nós fortes. Em linhas gerais, podemos entender que os nós fracos guardam apenas a raiz de Merkle enquanto os nós fortes guardam todas as transações. Isto posto, se um nó fraco precisa confirmar com o nó forte se a transação **A** está naquele nó, o nó forte não precisa

transmitir os Hashs das 4 transações, bastando apenas transmitir 2 transações, sendo o resultado de

$$n \log_2$$

com $n \rightarrow$ número de transações do bloco

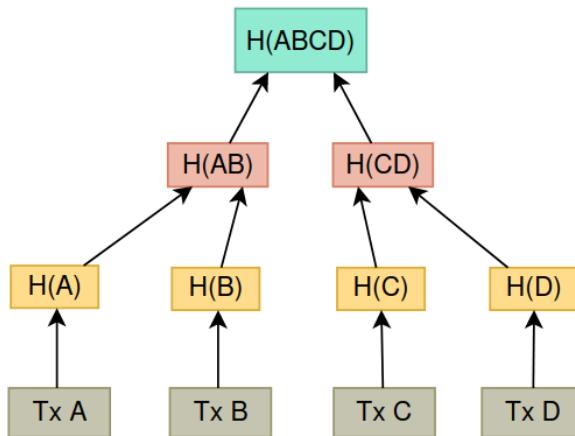


Figura 44 – Árvore de Merkle para 4 transações A, B, C e D. Fonte: Autor

Da mesma forma que a figura 20, a figura 21 também traz a Árvore de Merkle para 4 transações, mas agora com os detalhes de cor para poder entender melhor a dinâmica de funcionamento. Os quadrados em cinza são as transações, em laranja o Hash de cada uma delas, em vermelho o agrupamento e em verde a Raiz de Merkle, sendo esta raiz o resultado da operação de Hashs sucessivos para cada um dos elementos antecessores.

Continuando nossa busca de confirmação da presença da transação A em um nó forte, se o nó forte enviar o Hash de **B** e o Hash **H(CD)**, como o nó fraco tem em seu escopo a transação **A**, logo, ele conseguirá chegar ao Hash **H(AB)** que combinado ao Hash **H(CD)** enviado, conseguirá montar sua raiz e confirmar a presença da transação **A**.

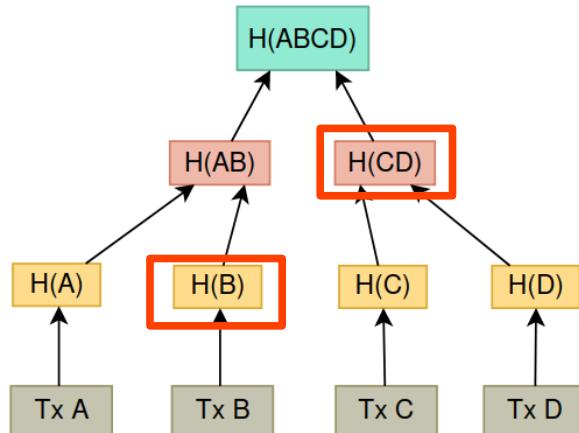


Figura 45 – Envio de 2 Hashs para 4 transações. Fonte: Autor

Se fossem 8 transações, não seria necessário enviar 7 Hashs para essa confirmação, bastaria enviar 3 Hashs ($8 \log_2$) para esta confirmação.

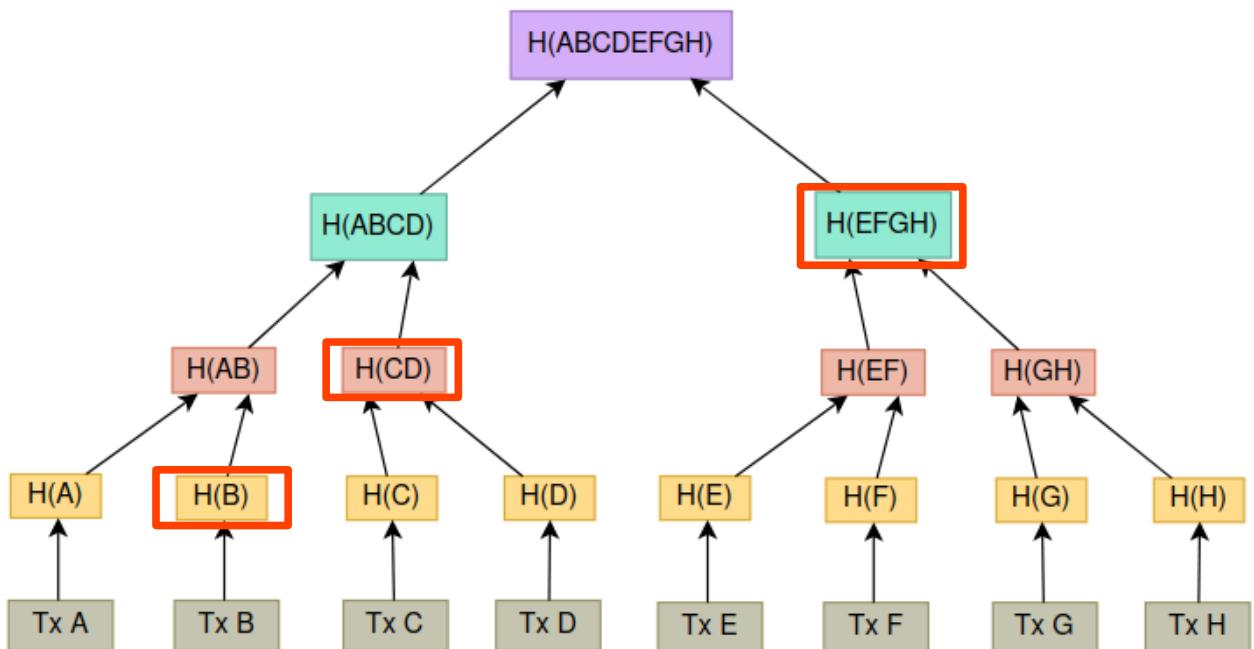


Figura 46 – Envio de 3 Hashs para 8 transações. Fonte: Autor

Notamos aqui, que de forma bem inteligente, com o envio dos Hashs **H(B)**, **H(CD)** e **H(EFGH)** é perfeitamente possível remontar a Raiz de Merkle para validar um conjunto de transações, economizando tráfego na rede. Se fossem 100 transações, da mesma forma,

seguindo a fórmula matemática de **100 log₂**, chegaríamos ao resultado de 7 Hashs enviados, num universo de 100, para remontar a Raiz de Merkle. Muito inteligente, não?!!!

7.1.6 REDE DESCENTRALIZADA

O blockchain também pode ser considerado um sistema de registros distribuídos funcionando em uma rede ponto-a-ponto (*Peer-to-Peer Network*). Todos os computadores que ajudam a processar a rede possuem cópias de todos os blocos de informação que já foram criados e qualquer informação alterada, seja uma vírgula, um espaço, um acento, um valor por menor que seja, qualquer coisa, vai mudar o hash do bloco e não vai combinar com toda a cadeia de Hashs já consolidada. A rede não vai aceitar uma mudança porque não fecha com a cópia do blockchain que todos os nós têm salvo no seus registros. É por isso que o bloco é um mecanismo extremamente resistente e inteligente.

A rede descentralizada tem como premissa a retirada da dependência de um centralizador para armazenar informação. Os dados ficam distribuídos de forma descentralizada com várias cópias. Para uma blockchain ser destruída seria necessário destruir todas as cópias que existem pois, se uma cópia ainda estiver disponível, a rede vai sobreviver a partir dessa única cópia.

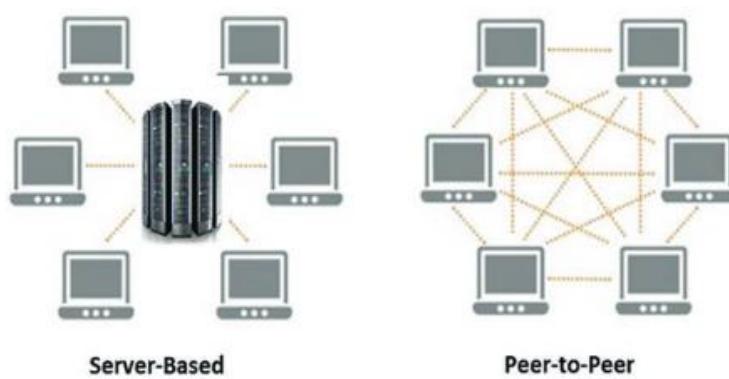


Figura 47 – Diferença entre rede descentralizada e centralizada. [ATTARAN e GUNASEKARAN] 2019 P.14

Esse sistema garante que nenhum dado seja perdido. Todos podem verificar a veracidade dos registros. Tudo é feito de forma transparente, permitindo que qualquer pessoa possa auditar a rede, o que aumenta também a segurança, eficiência, confiança nos registros e elimina a necessidade de terceiros ou intermediários em uma única fonte de informação e confiança.

Uma rede ponto-a-ponto é formada por NÓS (*Nodes*) e esta é uma peça fundamental neste grande quebra cabeça. O nó na rede blockchain é o ponto onde há uma cópia dos dados da rede blockchain. Muito mais que um repositório de dados, o nó exerce papéis podendo acumular funções ou se especializar em uma delas.

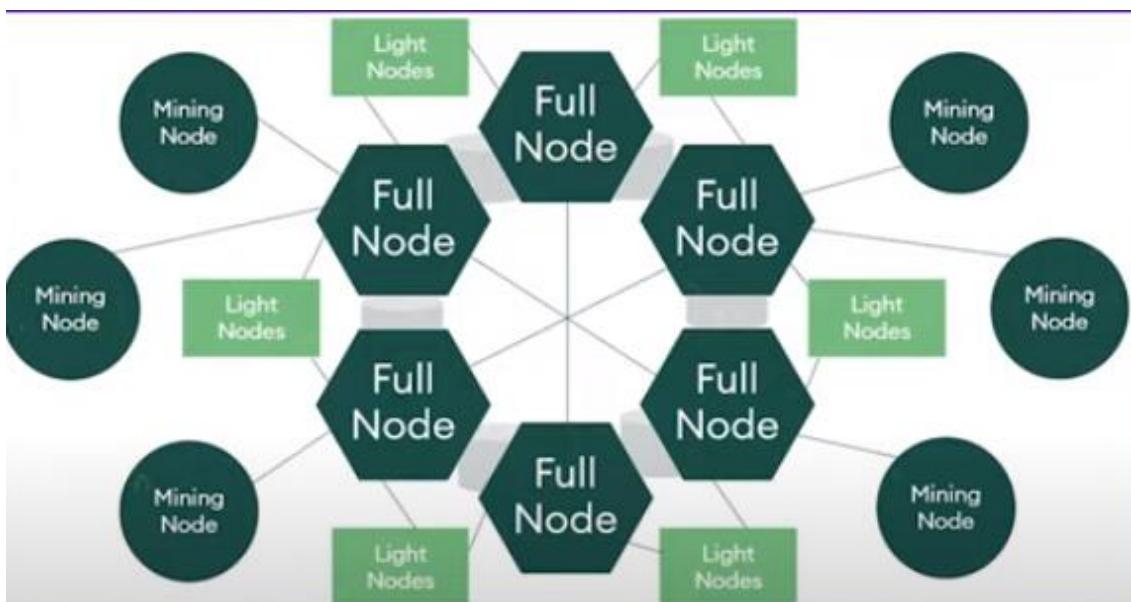


Figura 48 – Tipos de nó na rede blockchain. [BSC Army] 2022

Entre os diferentes tipos de nós, alguns se destacam, entre eles:

1. Nós Completos (*Full Nodes*) – Contém todo o histórico e informações relacionadas a cada bloco, desde a primeira transação na rede. Uma de suas características mais distintivas é a verificação da autenticidade da assinatura em cada transação do bloco.

2. Nós Fracos (*Lightweight Nodes*) – Conhecido também como verificação de pagamento simples, ao invés de armazenar informações completas, os nós fracos contêm informações relacionadas a um bloco anterior específico ao qual está conectado.

Muitas vezes eles utilizam os nós completos para acessar a rede, verificar transações e recuperar informações. Eles se ligam a nós completos funcionando como pontes.

3. Nós de Mineração (*Mining Nodes*) - Estes nós são responsáveis pela tarefa de minerar. Como resultado de seu trabalho, são gerados novos blocos.

Como já dito, o grande trunfo do blockchain é eliminar a necessidade de intermediários, e uma rede que descentraliza confiança, só poderia ter o seu primeiro caso de uso relacionado a dinheiro. O sistema financeiro é onde a gente mais depende de intermediários, tanto para criar moedas e as políticas monetárias com os bancos centrais, quanto para transmitir a moeda para a população, através dos bancos comerciais, só que como bem disse NAKAMOTO, a nossa história está repleta de violações essa confiança, seja pelos bancos centrais manipulando as políticas monetárias inflacionando as próprias moedas, seja via bancos comerciais criando mecanismos que dificultam o acesso das pessoas ao seu próprio dinheiro. Você já tentou sacar valores altos da sua conta bancária e precisou passar pelo gerente? Isso mostra o quanto a gente não é livre para movimentar o próprio dinheiro, precisa sempre pedir permissão e fica dependente desses intermediários liberarem ou não. O valor do Bitcoin, talvez o case de maior sucesso envolvendo a tecnologia blockchain, está na descentralização e na gama de tecnologias utilizadas como blockchain, Proof of Work e criptografia que tornam a rede segura.

Se não houver uma moeda, se resolver usar só blockchain, e utilizar uma autoridade Central decidindo quais são as ações ou quais blocos são válidos ou não, lá se vão todas as propriedades boas da descentralização.

De forma geral, uma rede blockchain pública precisa respeitar 5 princípios, todos alinhados com a Web 3.0:

1- neutralidade - não importa quem você é, onde você mora, qual a sua aparência, qual o seu partido político, nada sobre você vai te impedir de usar rede, a sua transação vai ser propagada independentemente de qualquer característica sua, o que não acontece no sistema financeiro tradicional, se você não fornecer seu endereço, um CPF válido, você não consegue ter uma conta bancária, por exemplo, você acaba sendo excluído do sistema.

2- não territorialidade - assim como a internet não é de nenhum país, está em todos os lugares, a rede não fica presa a nenhum território, a nenhum país, não pertence a nenhuma nação, é de todos os países e ao mesmo tempo não possui um órgão centralizador específico.

3- abertura - qualquer pessoa pode acessar, é só baixar um aplicativo ou qualquer outro meio tecnológico adequado e se conectar na rede.

4 – resistência à censura - não tem como parar a rede porquê alguém ficou incomodado com seu conteúdo ou porquê tem algo ali que a incomoda, é uma forma de expressão, um direito humano.

5- publicidade – os dados estão disponíveis para todo mundo verificar, isso significa que ninguém pode trapacear ou falsificar estas informações.

7.1.7 PROTOCOLO DE CONSENSO

Sendo uma rede onde os dados são armazenados em diversos nós, garantir que a rede blockchain mantenha a consistência dos dados sem a utilização de uma autoridade central, o que comprometeria um de seus pilares, é um desafio (ZHENG, XIE, DAI, CHEN, WANG, 2017).

Neste contexto, os protocolos de consenso surgem como uma ferramenta de controle da blockchain, definindo qual o comportamento da cadeia no processo de adição de um novo

bloco, seja validando e distribuindo este bloco entre os nós da cadeia, seja rejeitando inserções de blocos indevidas, protegendo a rede contra ataques (KUNTZ, 2022).

As propostas para atender a esta necessidade são várias, dependendo da proposta da rede, alvos técnicos e filosofia adotada. Seguindo referências de KUNTZ e ZHENG e autores, apresento abaixo os principais protocolos de consenso utilizados na atualidade.

7.1.7.1 PROOF OF WORK (PoW) – PROVA DE TRABALHO

Utilizando o mecanismo de mineração, os nós da rede competem entre si para resolver um desafio matemático para decidir qual vai ser o nó vencedor para incluir um novo bloco na rede. Este é o mecanismo utilizado na blockchain do Bitcoin.

O mecanismo chave no bloco é o seu Hash, e o Hash de um bloco precisa atender certas características para ser reconhecido como válido e por consequência, um bloco válido. Duas características são principais no trabalho de mineração: um processo matemático factível, mas de difícil realização e de fácil verificação. Na busca de um Hash criptográfico válido, o minerador deve encontrar um número único, um valor numérico de 4 bytes unsigned, reconhecido como *nonce* (number once) que aliado às transações do bloco (**raiz de Merkle**) e o *TimeStamp* produza um Hash que atende as condições especificadas, denominada *target*, alvo.

Este trabalho torna-se mais difícil ainda visto que o *TimeStamp* do bloco se modifica a cada segundo fazendo com que os testes de Hash tenham validade apenas naquele segundo específico. Uma vez encontrado o *nonce*, qualquer nó da rede poderá validar o bloco com aquele *nonce* dado. É como aquele famoso jogo “Onde Está Wally”, do autor inglês Martin Handford, onde num cenário com muitas, muitas, muitas figuras, os participantes deveriam encontrar o rapaz com camisa listrada e óculos redondo, Wally. Uma vez que um

dos participantes do jogo encontra Wally e mostra para os outros, fica fácil deles comprovarem se o primeiro está certo ou não.

A figura abaixo retirada do site <https://www.blockchain.com> mostra os detalhes de um bloco da rede Bitcoin.

Hash	00000000000000000000000000000000682d7fbb875f4fc504954deeb8ef2c11d09d51b536920
Confirmations	1
Timestamp	2022-08-05 04:23
Height	748048
Miner	Unknown
Number of Transactions	3,149
Difficulty	28,174,668,481,289.41
Merkle root	5c77b5f066326ff1c9e0b5b6c5037787fde95f5620481c3d578a19c185eb03b0
Version	0x24344000
Bits	386,530,686
Weight	3,993,370 WU
Size	1,579,330 bytes
Nonce	2,455,368,199
Transaction Volume	102599.51052575 BTC

Figura 49 – Bloco da rede Bitcoin. <https://www.blockchain.com/btc/block/748048>

Podemos destacar, entre várias, algumas informações relevantes:

- Em vermelho – o Hash do bloco composto por 19 zeros à esquerda.
- Em amarelo – o *TimeStamp*, a data do bloco.
- Em verde – a raiz de Merkle, o Hash de todas as transações do bloco.
- Em laranja – o nonce do bloco

A quantidade de zeros à esquerda no Hash representa a dificuldade do bloco. Se considerarmos o SHA-256, que gera um Hash de 64 caracteres hexadecimais (4 bits x 64 caracteres = 256) temos um cenário de 16^{64} Hashs, aproximadamente 10^{77} valores diferentes.

Se criarmos uma blockchain que tem como regra, um target de 10 zeros à esquerda isso reduziria a nossa possibilidade de encontrar um Hash válido de $16^{64} - 10 = 16^{54}$, algo em torno de 10^{65} . Ainda parece um número grande, mas se calcularmos a probabilidade de acerto ($10^{65} / 10^{77}$) encontramos o resultado de 10^{-12} , ou seja, um número bem pequeno. Na figura 26, de exemplo da rede Bitcoin, o target com 19 zeros indica um cenário de dificuldade de $10^{54} / 10^{77} = 10^{-23}$.

Após encontrar o nonce válido para a dificuldade estabelecida, o minerador é recompensado financeiramente e o bloco é distribuído para os outros nós da rede, que em posse do nonce válido validam o bloco e o incluem em sua cadeia. Na situação em que há um empate, ou seja, dois nós distintos encontram o nonce válido no mesmo *TimeStamp*, o protocolo de consenso espera a próxima inserção de blocos na cadeia e aquela cadeia que inserir o próximo bloco válido é reconhecida como a cadeia mais longa, e, portanto, a cadeia válida para continuar a crescer na rede blockchain, desempatando o cenário anterior.

7.1.7.2 PROOF OF STAKE (Pos) – PROVA DE PARTICIPAÇÃO

A principal diferença entre PoS e PoW é que no PoS não há prova de mineração através daquele cálculo matemático que explicamos anteriormente, mas sim prova de validação. Umas das críticas ao PoW é a carga energética necessária para validação dos blocos pelos mineradores e a baixa escalabilidade da rede, e no protocolo *Proof of Stake* a validação é feita com uma espécie de “sorteio” daquele que vai validar o próximo bloco na rede.

Para que o participante, usuário validador, nó, participe deste processo de consenso é preciso que ele aloque moedas na rede numa operação especial indicando que deseja validar blocos na rede. Aqueles que desejam participar deste processo e alocam moedas na transação de validação formam um comitê que entre várias premissas para a decisão do

vencedor, tem a variável da quantidade de moeda alocada pelo participante, fazendo com que quanto maior a quantidade de moeda alocada, maior seja o seu peso na decisão.

Todo este processo de decisão deve acontecer dentro de uma janela de tempo chamada de *slot* (fenda) e havendo consenso entre os validadores o bloco é aprovado e inserido na cadeia. Um único bloco pode ser inserido em um slot e um conjunto de slots é conhecido como *epoch* (época). Um comitê é válido apenas dentro de uma época.

Em seu mecanismo de aprovação, o PoS recompensa os validadores que votaram a favor da validação do bloco com a volta de suas moedas aportadas inicialmente na transação e mais um valor financeiro. Caso o bloco seja rejeitado, ou seja, não seja considerado como válido para inserção na rede, aqueles que votaram positivamente para a sua inserção não recebem de volta suas moedas aportadas e acabam tendo perda financeira.

Timestamp	Aug 7, 2022, 6:43:19 AM
Block Hash	282ba7e683b60f7dbbee4912f774669e73a35caceeac206fa61d56b109c57ce1 Copy
Block Height	1,002,369
Era	5913
Parent Hash	11ae9a8d555eae3eecdaf09b21e6f4baf66d5998ff5fc87dbc17f03331c9704 Copy
State Root Hash	636502459bb316ede980cfa1a92b738fe79f92e9af7f9d1bab68ef339832a619
Validator	 017b9a85b657e0a8c2e01bf2d80b6b2e6f8d8b4bc6d7c479f21e59dceea761710b Copy

Figura 50 – Bloco da rede Casper.

<https://cspr.live/block/282ba7e683b60f7dbbee4912f774669e73a35caceeac206fa61d56b109c57ce1>

Na figura 27 vemos um bloco da rede Casper, que utiliza Proof of Stake. Nele, encontramos informações relevantes referentes ao bloco inserido:

- TimeStamp – é a data do bloco, data de inserção na rede;

- Block Hash – é o Hash identificador do bloco, que o qualifica unicamente na rede;
- Era – é o **epoch** do bloco;
- Parent Hash – é o seu bloco anterior da cadeia de blocos;
- Validator – é a identificação, endereço, do nó validador.

Como vantagens sobre o protocolo Proof of Work, o Proof of Stake apresenta:

1. Menores custos associados a equipamentos e estrutura física.
2. Maior velocidade, permitindo mais transações em menor tempo.
3. Menor gasto energético, pois não utiliza mineração.
4. É mais descentralizado que o PoW, visto que vários atores participam do processo de validação.

Se por um lado, a fragilidade do PoW está em um ataque à maioria dos nós da rede, a fragilidade do PoS estaria em um participante que tivesse em seu poder uma grande monta de moeda corrente da rede para fazer seus aportes de participação em comitê de validação. As discussões são muitas entre as diferenças entre as redes que utilizam PoW e PoS, não existindo uma melhor ou pior, mas cenários onde uma proposta se adéqua mais que outra. Se uma proposta prioriza segurança e imutabilidade, o PoW, a princípio, se apresenta como mais adequado, se a busca for por maior escalabilidade e velocidade, sem deixar de lado, é claro, a segurança, o PoS, a princípio, se apresenta como melhor proposta.

Não é incomum uma rede iniciar seu trabalho utilizando um determinado protocolo de consenso e depois mudar para outro protocolo. Este é o caso da rede Ethereum. Apesar de iniciar a operar oficialmente utilizando o protocolo Proof of Work, ela está migrando suas operações para o protocolo Proof of Stake (**The merge**¹¹).

¹¹ **The merge**. Disponível em: <<https://ethereum.org/en/upgrades/merge/>>. Acessado em: 10 de março de 2023.

7.1.7.3 PROOF OF AUTHORITY (PoA) – PROVA DE AUTORIDADE

O *Proof of Authority* (PoA) é um protocolo de consenso com funcionamento muito parecido com o PoS, com a diferença que no PoA os validadores são agentes previamente selecionados e muitas vezes necessitando algum tipo de validação externa documental ou governamental, mas seguindo a mesma ideia de consenso estabelecida no PoS.

Essa característica quebra um dos pilares da blockchain pública que é o seu caráter descentralizado, na medida em que há um agente central que define quem pode e quem não pode participar da rede, fazendo uma solução típica para blockchains privadas. A critério de curiosidade, o Real Digital, que é uma proposta de **Carteira Digital de Banco Central** (CDBC) está apontando para uso de um protocolo de PoA.

7.1.8 ESTRUTURA DO BLOCO

À esta altura, mencionar que o bloco é o elemento central de toda a estrutura deveria ser redundante, e a sua formatação depende das especificações da rede em uso, mas tipicamente, o bloco é composto por duas estruturas fundamentais, o cabeçalho e o corpo. No cabeçalho ficam as metainformações e no corpo o histórico de transações (KUNTZ, 2022).

O cabeçalho é uma estrutura de 80 bytes distribuídos da seguinte forma:

1. 4 bytes para sua identificação e regra para validação do bloco (Block Version);
2. 32 bytes para armazenar o Hash do bloco atual (Merkle Tree Root Hash);
3. 4 bytes para o TimeStamp;
4. bytes para representar a dificuldade de mineração (nBits);
5. 4 bytes para representar o Nonce;
6. 32 bytes para armazenar o Hash do bloco anterior (Parent Block).

Já o corpo do bloco contém um contador de transações (Transaction Counter) e as transações do bloco (TX), propriamente ditas. O número máximo de transações e a quantidade delas depende do tamanho do bloco e do tamanho das transações.

A imagem abaixo mostra a organização de todos esses elementos do bloco

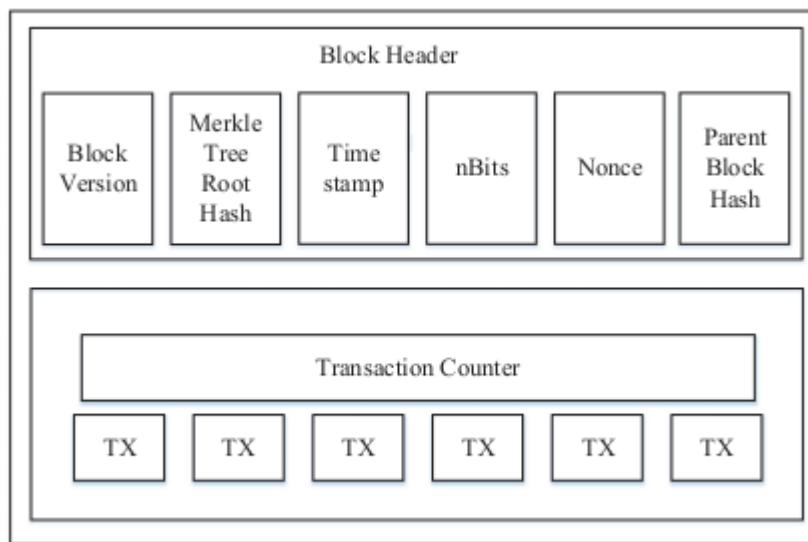


Figura 51 – Estrutura de um bloco na rede blockchain. [ZHENG] 2017 P. 558

7.2 EVOLUÇÃO DA WEB

Importante trabalho apresentado por OLIVEIRA, MAZIERO e ARAÚJO (2018) mostra a evolução da Web até a nova Web 3.0. Neste trabalho os autores apresentam a internet como um recurso provido pela informática que conecta pessoas, sendo a Web 3.0 uma evolução deste recurso, onde os elementos de inteligência e organização são ampliados.

É difícil precisar um marco zero, um dia D, se assim podemos falar, sobre o início da internet, mas, sem dúvida, os trabalhos desenvolvidos nos idos de 1991 e protagonizados por Tim Berners-Lee nos laboratórios do CERN (*Conseil Européen pour la Recherche Nucléaire*) na Suíça, são grandes marcos desta tecnologia.

Na esteira da evolução, a Web 1.0 (década de 1990) se caracterizava com recursos

de conectividade entre os ambientes e pessoas, mas com baixa interação, onde elementos como links e páginas estáticas eram a essência. Em seguida, a Web 2.0 (início dos anos 2000) trouxe elementos de maior colaboração e mídias sociais, onde a relação entre as pessoas e comunidades ganhou destaque e a interação entre seus participantes é o foco. Com a Web 3.0 - iniciada em 2001 por artigo de Tim Berners-Lee, ainda em desenvolvimento - se propõe uma internet mais inteligente e voltada para as máquinas, criando um conteúdo mais propício à leitura feita por estas, Web Semântica.

Os autores continuam sua obra e descrevem as características da Web Semântica, sendo então uma técnica para construção da Web voltada para a leitura do conteúdo por não humanos, assim, utilizando a linguagem XML2 ao invés de HTML, é possível ampliar o poder de indexação dos elementos. Com o poder de indexação ampliado, é possível filtrar o conteúdo da rede de forma cada vez mais personalizado, ampliando a experiência de uso e a especialização das informações e comunidades participantes.

Esta nova estrutura conta com a introdução, ao lado dos dados propriamente ditos, de metadados e ontologia, marcando a semântica, significado, daquele dado apresentado e sua ontologia, seu fim, objetivo.

Além disso, a Web 3.0 se destaca por uma nova proposta, onde o foco não são mais os grandes *players* como Google®, Meta®, Apple®, entre outros, mas o indivíduo, assim sendo, novos negócios e filosofias que fazem o compartilhamento de dados (sejam eles um vídeo, um documento ou até mesmo valor monetário) de forma independente da participação de um terceiro para validação ganha espaço e coloca o poder novamente na mão das pessoas e não das grandes companhias.

Outro famoso artigo escrito por ZAGO (2018) faz, também, essa comparação entre as diferentes fases da Web e cita algumas vantagens esperadas com a descentralização proposta pela Web 3.0:

1. Ausência de um ponto de controle central – governos ou entidades perdem a capacidade de bloquear sites e serviços e não há um indivíduo central podendo controlar a identidade das pessoas.
2. Propriedade de dados – usuários finais recuperaram o controle dos dados e fazem uso de criptografia.
3. Redução de ataques *hackers* e vazamento de dados – como os dados são descentralizados e distribuídos, para que os dados fossem comprometidos, um ataque teria que afetar todos os nós da rede e aliado à tecnologia de criptografia, o sigilo dos dados fornece uma camada de proteção a mais.
4. Interoperabilidade – Possibilidade de aplicações poderem *rodar* em diferentes dispositivos e plataformas, de maneira agnóstica, não presa a um sistema operacional em específico e não fazendo diferenciação entre plataformas iOS ou Android, por exemplo.¹²
5. Ausência de permissão para participar da blockchain – tratando aqui das blockchains públicas¹³. Qualquer pessoa pode criar um endereço e interagir com a rede, não havendo barreiras geográficas, gênero ou qualquer outra restrição social ou

¹² Vale ressaltar que, apesar do objetivo cada vez maior do agnosticismo buscado pelas plataformas blockchain, ainda há certas limitações de interoperabilidade mesmo entre as diferentes soluções blockchain. Como exemplo de barreira ainda não superada, podemos citar a barreira de executar smart contracts da rede Ethereum dentro da rede Solana, por exemplo. No ano em que escrevo esta obra, iniciativas bem avançadas e promissoras neste sentido de interoperabilidade entre plataformas já têm sido desenvolvidas, mas ainda em fase de testes. Como estamos falando de tecnologias relativamente novas, é perfeitamente factível esperar uma padronização no setor que consiga nos levar a este alvo.

¹³ As blockchains públicas se diferenciam das blockchains privadas exatamente por este caráter de abertura para participação. Enquanto as públicas não fazem qualquer restrição para a entrada de um participante na rede, as blockchains privadas controlam seus participantes. Esta última abordagem tem sido utilizada principalmente por grandes empresas para integrar dentro de seu guarda-chuva tecnológico seus fornecedores e clientes para que haja um controle de toda a cadeia produtiva de modo fim-a-fim proporcionando automatização de processos e rastreabilidade de eventos, com agentes participando de todo o processo da cadeia e outros adentrando e saindo apenas em momentos específicos. Exemplo de plataforma que faz esta abordagem é a Hyperledger Fabric, iniciativa da Linux Foundation e amplamente utilizada pela IBM. Famoso vídeo do caso de parceria entre a gigante da tecnologia IBM e a gigante da logística Maersk utilizando blockchain como solução pode ser vista neste endereço eletrônico: <<https://www.youtube.com/watch?v=tdhpYQCWnCw>>.

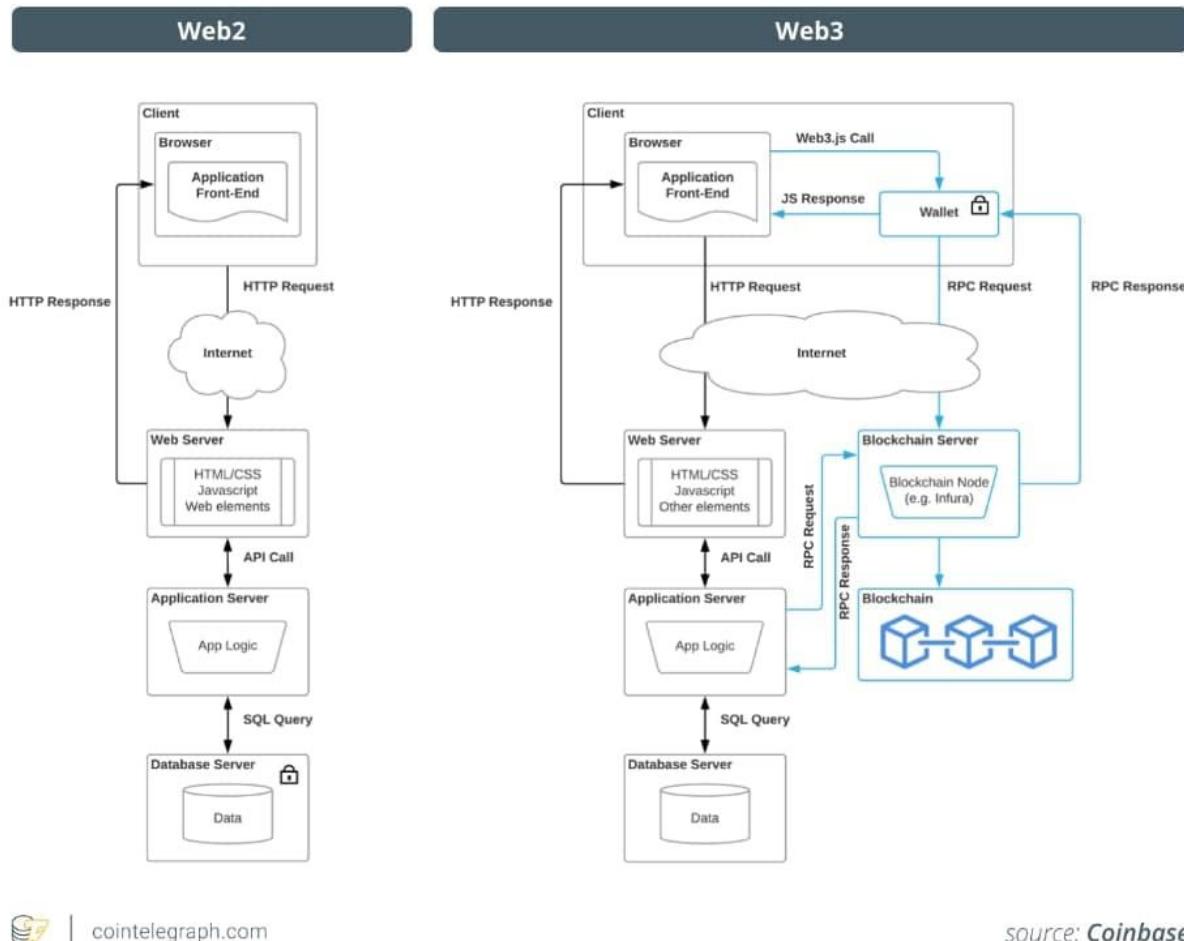
demográfica.

6. Funcionamento ininterrupto – Suspensão de conta e ataque de negação de serviço (DDOS) são dramaticamente reduzidos, pois não há um concentrador central, um ponto único de falha, os nós da rede garantem redundância de dados e proporcionam maior garantia da continuidade do serviço.

Pelas diferenças apresentadas, fica claro que as soluções propostas para a Web2 e Web3 se mostram bem diferentes e essa diferença acaba se desdobrando na forma como a arquitetura para cada uma destas duas soluções tradicionalmente se desenvolve. Em artigo publicado em 2018 TEKISALP na **coinbase**¹⁴ apresenta essas diferenças e a figura abaixo, mostra as diferenças arquiteturais em uma típica aplicação baseada em Web2 comparada a outra aplicação Web3 (TEKISALP, 2018).

¹⁴ **coinbase** - <https://www.coinbase.com/>

Architecture of a Web2 application vs that of a Web3 application



cointelegraph.com

source: Coinbase

Figura 52 – Aplicação Web2 vs aplicação Web3. [TEKISALP] 2018

7.3 SOLUÇÕES BLOCKCHAIN

Como mencionado na introdução desta obra, há diversas soluções no mercado que utilizam a tecnologia blockchain, e para tentar ilustrar este panorama, serão apresentadas nesta seção algumas alternativas já em mercado que fazem uso dessa tecnologia.

Para começar, a obra de MORAIS e LINS (2020) traz projetos na área de educação. Neste trabalho, os autores começam reconhecendo que o documento mais importante no

sistema educacional é o diploma e que para evitar fraudes, a tecnologia blockchain se apresenta como solução eficaz para evitar diplomas falsos e adulterações, garantindo a idoneidade das informações da instituição e alunos, permitindo que eles sejam validados a qualquer momento por um interessado.

Já existem ferramentas no mercado voltadas para esta aplicação, onde podemos destacar a Blockcerts, desenvolvida pelo MIT e a TrueRec, criada pelo grupo SAP. Outras propostas são: Smart Contracts na plataforma Ethereum, o UniCert, uma solução que usa validação em uma base de dados local em PostgreSQL e o registro do hash em uma blockchain, o SmartCert que utiliza a plataforma Ethereum e o CredenceLedger, outra aplicação baseada na rede Ethereum.

Muitas soluções além do registro de diplomas podem ser gerenciadas com a tecnologia blockchain. Histórico escolar, o desenvolvimento da aprendizagem, dados científicos, pagamentos, entre outros, são soluções em que esta tecnologia pode ser utilizada de forma bastante segura e garantindo a validação por qualquer interessado.

Vale lembrar que para o desenvolvimento de uma solução em blockchain é possível adotar uma arquitetura pública ou privada, onde esta permite a entrada apenas de pessoas autorizadas e aquela é aberta ao público em geral. Cada uma destas propostas tem suas vantagens e desvantagens e deve ser adotada aquela que fornece melhor alternativa ao projeto.

Outros cuidados que devem ser levados em consideração na implementação de uma solução blockchain para registro de informações educacionais é a capacidade da rede local para lidar com essa nova carga de dados, olhando aqui tanto para a capacidade e segurança necessária para tal, a capacidade de se integrar com sistemas legados, e a capacidade de integração de forma segura com ferramentas de aprendizagem educacional externas.

Na mesma esteira tecnológica, NIWA (2020) apresenta um sistema de voto eletrônico

baseado em blockchain. Em seu documento, o autor faz um estudo dos sistemas de votação passando pelo tradicional sistema de papel e em seguida apresenta os sistemas com votação eletrônica. Há as chamadas urnas de 1^a geração (DRE – *Direct Recording Electronic voting machine*), onde os votos são registrados na memória de um equipamento eletrônico e posteriormente contados pelo administrador do sistema e o desenvolvedor do software, sendo este o modelo utilizado no Brasil. Há, ainda, as urnas de 2^a geração (IVVR – *Independent Voter Verifiable Record*), que imprimem um comprovante em papel possibilitando a auditoria contábil dos votos e a de 3^a geração (*End-to-End verifiable*), que contam com dispositivos de rádio frequência possibilitando a conferência do eleitor independente de software.

Na sequência é apresentada a motivação de um novo sistema que inclui, principalmente, as falhas denunciadas por: CUNHA (2014) e demais anomalias que poderiam ser melhoradas denunciadas por ARRIAL (2018), TSE (2018) e RAMALHO (2018). E como solução, é apresentado um sistema de votação que utilize a tecnologia *blockchain*, sendo: “um banco de dados criado através de uma rede distribuída, descentralizada e com criptografia de chaves público-privada e algoritmos de hash seguros” (NIWA, 2020).

7.4 SMART CONTRACTS

Como proposta de nosso trabalho, iremos utilizar a estratégia de smart contracts para desenvolver a solução. Mas esta estratégia é válida? É reconhecida? O Direito a reconhece como instrumento viável? Na busca dessas respostas o trabalho de CARVALHO e ÁVILA (2019) nos ajuda a responder.

Em seu artigo, os autores discorrem sobre a teoria dos contratos e definem os smart contracts como nada mais que contratos codificados (representados em código de programação) que tem execução automática e autônoma.

Ao definir contrato, em sua forma tradicional, os autores trazem a visão deste conceito sob a ótica de renomados juristas e destaca como definição:

“contrato é negócio jurídico que necessita obrigatoriamente de ao menos duas partes para compor a relação”
(CARVALHO, 2022).

Mas ressalta que apesar da liberdade de contrato entre as partes, este contrato deve estar alinhado com preceitos legais estabelecidos no ordenamento.

Já quanto aos Contratos Inteligentes, conhecidos também como *Smart Contracts*, inferem a origem do termo a Nick Szabo, na obra *Smart Contracts: Building Blocks for Digital Free Markets*” de 1996. Nesta obra, Nick Szabo apresenta a verificabilidade, o acompanhamento, a privacidade e a exigibilidade como melhorias nos contratos trazidas pelos Contratos Inteligentes.

Após tratar sobre contratos, os autores abordam a tecnologia chave para esses Contratos Inteligentes, Blockchain, sendo esta, uma cadeia de blocos organizados através de uma rede *Peer-to-Peer* (Ponto-a-Ponto), que funciona como um banco de dados descentralizado das transações e dados dos blocos, organizados cronologicamente e criptografados através de *hash*.

Como exemplos de soluções que fazem uso desse novo modelo de contratos são apresentados cenários como na fabricação e venda de veículo, financiamento de automóvel e soluções como *OriginalMy*, que registra prova de autenticidade em custos muito mais competitivos que cartórios notariais, e *OpenLaw*, solução na área jurídica que fornece modelos de contratos que podem ser adaptados por seus signatários e que também fazem uso da tecnologia blockchain.

Com essa nova tecnologia a celeridade, a certeza do cumprimento e a solução da falta de confiança entre as partes são peças chave que colocam os Contratos Inteligentes como alternativa eficaz para adquirir, extinguir, modificar ou executar direitos e obrigações.

Insta mencionar o consenso entre as partes para pactuarem desta forma, sendo, então, a tecnologia blockchain alternativa viável, mas não única fonte para todos os modos de pactuação de contratos. Superadas as devidas ressalvas, este novo modelo, já nem tão novo assim, se apresenta como modelo viável e eficaz para atender diversos cenários onde a celeridade, objetividade e certeza de cumprimento se apresentam na realidade dos contratos.

7.5 ETHEREUM

Ethereum é uma blockchain pública não permissionada, de código aberto e de uso geral, idealizada por Vitalik Buterin e lançada oficialmente em 2015 (KUNTZ, 2022). Como diferencial, permite que clientes interajam com a rede através de linguagens de programação para a execução de Smart Contracts. Enquanto a rede Bitcoin se concentra em se consolidar como uma rede de pagamentos, a rede Ethereum se posiciona como um Marketplace.

Essas características possibilitaram que esta rede fosse palco para receber diversas aplicações nos mais diversos segmentos, desde o setor financeiro, saúde, moda, entretenimento etc. Alguns destaque sobre a rede Ethereum estão sobre a sua utilização para soluções DeFi (*Decentralized Finances*), DAO (*Decentralized Autonomous Organization*), NFT (*Non-Fungible Tokens*) entre outras.

Os dados da rede Ethereum impressionam. De acordo com dados da State of the dapps¹⁵, com dados de maio de 2022, há 2970 projetos utilizando a rede Ethereum, mais de

¹⁵ <https://www.stateofthedapps.com/stats/platform/ethereum#new>

50 milhões de Smart Contracts utilizando sua rede¹⁶ e mais de 1.6 bilhão de transações em sua rede¹⁷.

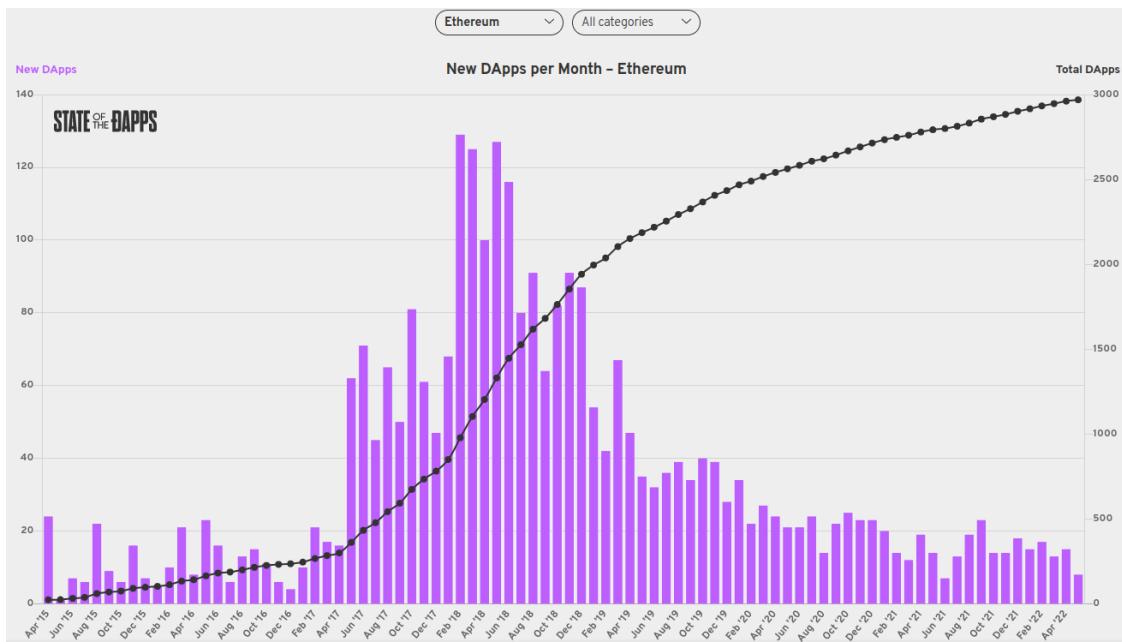


Figura 53 – Projetos utilizando Ethereum.



Figura 54 – Smart Contracts utilizando a rede Ethereum.

¹⁶ https://dune.com/sawmon_and_natalie/smart-contracts-on-ethereum

¹⁷ <https://etherscan.io/txs>

More than > 1,670,963,147 transactions found
(Showing the last 500k records)

Txn Hash	Method ⓘ	Block	Age	From	To	Value	Txn Fee
0xda89a9a39919622d20...	Transfer	15306903	51 secs ago	Ethermine	0x45f27558ea454f90ba4...	0.005912490058798 Ether	0.00017124 ⚡
0x1307fa3f4e6cd38726d...	Mint	15306903	51 secs ago	0x25c76a801f6c262921...	0x45f27558ea454f90ba4... Compound: cETH Token	40 Ether	0.00130377 ⚡
0xee831140bec60d6f11...	Transfer	15306903	51 secs ago	0x9abef270d5452af637...	0x45f27558ea454f90ba4... Centre: USD Coin	0 Ether	0.0003956 ⚡
0x7823e86a3ace5b392f...	Transfer	15306903	51 secs ago	0xc2c0df71da6f73cd077...	0x45f27558ea454f90ba4... Tether: USDT Stablecoin	0 Ether	0.00051544 ⚡
0xcb151c727a47942bde...	Transfer	15306903	51 secs ago	0x4d25a05059c0c0b529...	0x45f27558ea454f90ba4... 0x479e2cfa68c4a30eb3...	0 Ether	0.00017783 ⚡
0xbff4d6fe7a9cfa869376...	Transfer	15306903	51 secs ago	0x58e996377e9ed437a6...	0x45f27558ea454f90ba4... Centre: USD Coin	0 Ether	0.00039579 ⚡
0x37fdb8f5d61694f87d09...	Transfer	15306903	51 secs ago	Hiveon: Spreader	0x511a912cf1a1689bb3...	0.204971889 Ether	0.00017124 ⚡
0x7a7c1f36303271503cf...	Transfer	15306903	51 secs ago	Hiveon: Spreader	0x511a912cf1a1689bb3...	0.20498556 Ether	0.00017124 ⚡
0x0ec7a85f298035ff70e...	Transfer	15306903	51 secs ago	Hiveon: Spreader	0x511a912cf1a1689bb3...	0.204990817 Ether	0.00017124 ⚡

Figura 55 – Smart Contracts utilizando a rede Ethereum.

7.5.1 ETHEREUM VIRTUAL MACHINE (EVM)

A Máquina Virtual do Ethereum (*Ethereum Virtual Machine - EVM*) é uma máquina virtual nos padrões Turing-Complete, podendo executar qualquer programa expresso em código. No contexto da blockchain, a propriedade Turing-Complete é importante porque permite que contratos inteligentes sejam criados e executados na plataforma. O ponto de contato entre a EVM e os contratos inteligentes se dá nos *bytecodes*, que são códigos gerados pelo processo de compilação e carregados na EVM. Através dos *bytecodes* a EVM reconhece o conjunto de instruções, entradas e saídas de um contrato inteligente.

A EVM é ainda, uma máquina de estados distribuída onde cada nó da rede executa esta máquina e para isso, há protocolos próprios para controlar estes estados em diferentes pontos da rede. O estado no Ethereum é uma grande estrutura de dados que não tem apenas contas e saldos, como no caso do Bitcoin, mas também um estado da máquina, que pode mudar de bloco para bloco. Se o blockchain pode ser traduzido em uma cadeia de blocos a EVM pode ser traduzida em uma cadeia de transações executadas em uma máquina de estados.

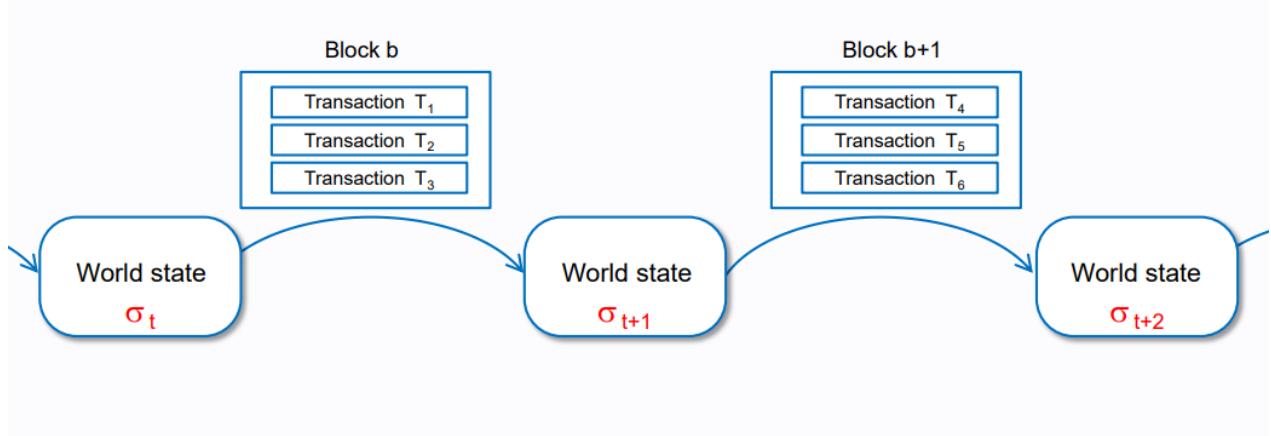


Figura 56 – Cadeia de estados na EVM. [TAKENOBU]

A EVM se comporta como uma função matemática em que de acordo com a entrada, ele produz uma saída determinística. Portanto, é bastante útil descrever mais formalmente o Ethereum como tendo uma **função de transição de estado** (Ethereum, EVM):

$$Y(S, T) = S'$$

Sendo:

Y → Uma função de transição

S → Um estado inicial

T → Um conjunto de transações válidas

S' → Um novo estado final de saída

Se olharmos para a estrutura da EVM poderemos perceber algo bem semelhante a uma estrutura computacional como conhecemos hoje, mas há um elemento importante nessa estrutura que age como um agente de custo das operações que é o Gas. O Gas é uma expressão computacional diretamente ligada ao esforço computacional dispendido para realizar uma operação solicitada à EVM. Se a operação for muito custosa, mais Gas será exigido, se for menos custosa, menos Gas será gasto. Em linhas gerais, podemos entender o Gas como uma espécie de pedágio para as operações da EVM. Essa característica é

importante pois estimula os desenvolvedores a escreverem códigos cada vez mais otimizados para fazerem uso da EVM da forma mais otimizada possível.

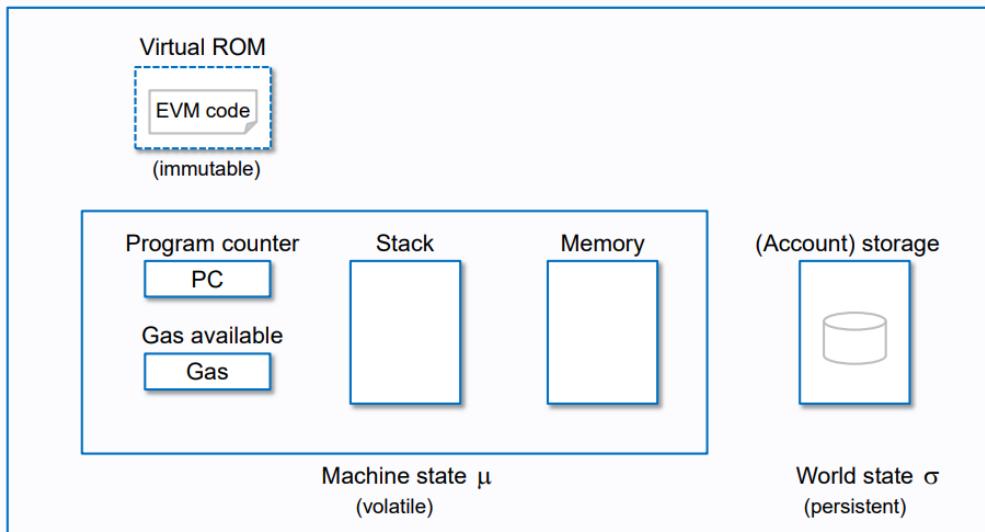


Figura 57 – Cadeia de estados na EVM. [TAKENOBU]

7.5.2 CONTAS

Outro elemento importante na estrutura do Ethereum é a conta. Conta, no mundo Ethereum é um objeto que se relaciona diretamente a um estado da EVM. Há dois tipos de conta: Conta de proprietário externa (*Externally Owned Account* –EOA) e Conta de Contrato (*Contract Account* – CC). Ambas são representadas por uma identificação de 20 dígitos em hexadecimal e esta identificação designa o endereço desta conta. As contas de proprietário são associadas a uma pessoa e são independentes da EVM, já as contas de contrato são autogerenciadas pela EVM. Somente contas de proprietário externa iniciam uma transação.

Enquanto as contas de proprietário externo possuem apenas duas informações associadas a ela que são a quantidade de moeda nativa associada a essa conta, chamada de **balance**, e um número decimal usado para o controle de transações referenciadas a esta conta que inicial em zero e é incrementado a cada transação, chamado de **nonce**, as Contas de Contrato possuem, além de **balance** e **nonce**, também código que a EVM pode executar

(o código programado no contrato - *bytecode*) e variáveis de estado que armazenam dados.

As contas de proprietário são controladas pela sua chave privada, ao passo que as contas de contrato são controladas pela EVM.

Outro detalhe importante que diferencia esses dois tipos de conta é que a conta de proprietário externa é composta por um par de chaves pública e privada (para maiores detalhes reveja a sessão 7.1.3 onde falamos sobre criptografia e validação) que identificam o proprietário de uma conta (identificação) e o autor de uma transação (não repúdio).

Quando o usuário da blockchain cria uma conta será gerado uma chave privada aleatória composta por 64 caracteres hexadecimal e a partir dela, utilizando um algoritmo de assinatura digital de curva elíptica será gerada a chave pública desta conta. Seu endereço público é formado com base nessa chave pública sendo os últimos 20 bytes do hash Keccak-256 da chave pública e adicionado **0x** no início (ETHEREUM).

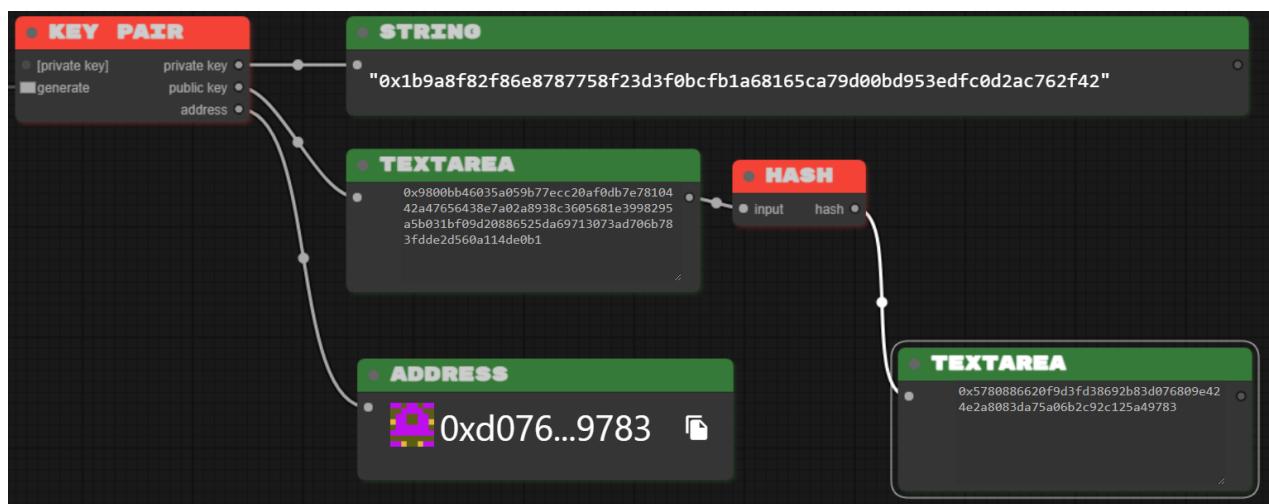


Figura 58 – Chave privada, pública e endereço de uma conta.¹⁸

¹⁸ **Chave privada, pública e endereço de uma conta.** Na geração dos dados de uma conta a geração da chave privada é completamente aleatória e após aplicação do algoritmo se obtém a chave pública. Perceba que o hash tem como entrada uma chave pública e como saída, pegando os últimos 20 dígitos e acrescentando 0x no início temos o endereço de uma conta – Contribuição de eth.build. Disponível em: <<https://sandbox.eth.build/>> . Acessado em: 11 de março de 2023.

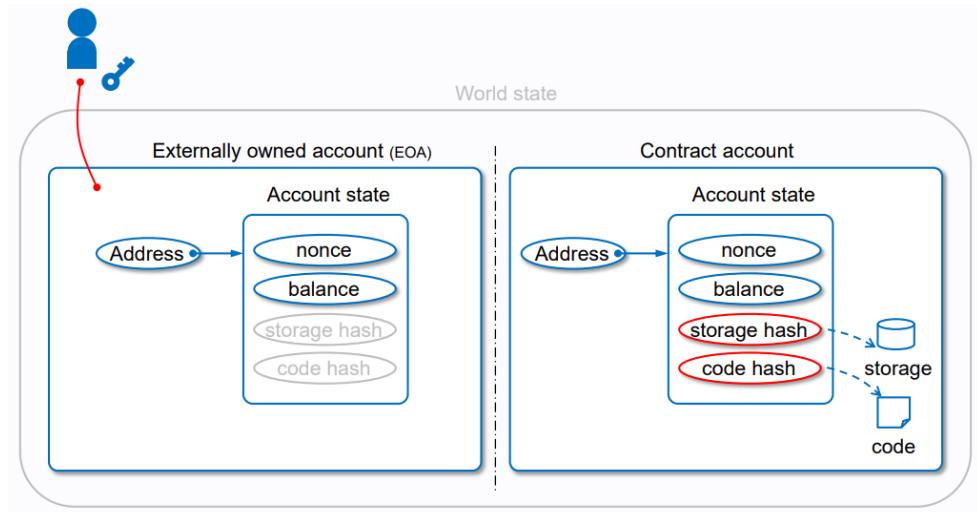


Figura 59 – Cadeia de estados na EVM. [TAKENOBU]

7.5.3 TRANSAÇÕES

Transações são instruções submetidas por um ator externo criptograficamente assinadas que alteram o estado da blockchain.

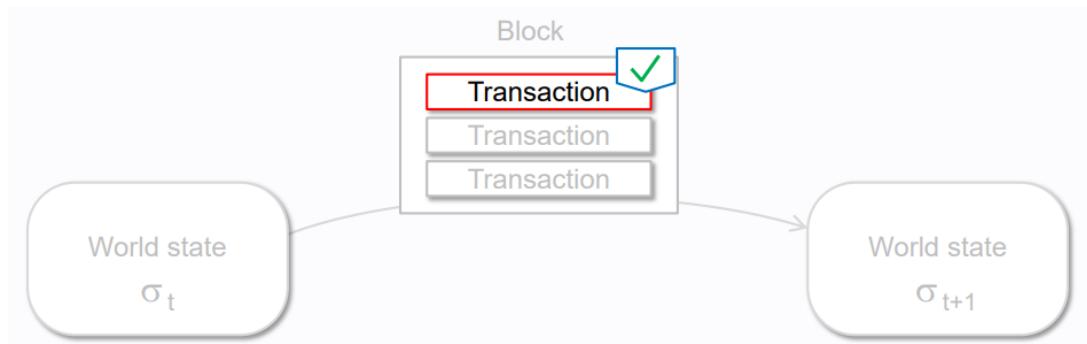


Figura 60 – Transação na EVM. [TAKENOBU]

As transações que alteram o estado da blockchain devem ser replicadas para toda a rede. Essas transações são solicitadas por uma conta e registradas em um bloco. Qualquer nó pode transmitir uma solicitação para que uma transação seja executada na EVM; depois que isso acontecer, um validador executará a transação e a propagará para toda a rede.

Uma transação enviada a EVM possui as seguintes informações:

- **Recipient**: se for uma conta externa, esta receberá o valor; se for um contrato, executará o código do contrato;
- **Signature**: o identificador do remetente mediante o uso de sua chave privada e mensagem;
- **Nonce**: um contador de incremento sequencial que indica o número de transações da conta;
- **Value**: quantidade de ETH (moeda do Ethereum) a ser transmitida para o recipient;
- **Data**: campo adicional que possui dados arbitrários;
- **GasLimit**: quantidade máxima de gas que pode ser consumida pela transação;
- **MaxPriorityFeePerGas**: quantidade máxima de gas a ser incluída como gorjeta para o validador;
- **MaxFeePerGas**: quantidade máxima de gas a ser paga pela transação.

Existem três tipos básicos de transações no Ethereum:

- **Transações regulares**: transação entre contas externas;
- **Transações de criação de contrato**: transações sem um endereço de destino onde são enviados os dados do código do contrato;
- **Transações de execução de contrato**: transações que chamam alguma função de um contrato criado na rede Ethereum.

Importante frisar que uma transação é atômica na EVM. Uma transação não pode ser dividida, nem interrompida. A estrutura blockchain como um todo pode ser considerado como uma estrutura que possui propriedades ACID (Atomicidade, Consistência, Isolamento e

Durabilidade).

7.5.4 PATRICIA MERKLE TREE

Em sessões anteriores falamos sobre a utilidade da *Merkle Tree* para guardar hashes das transações, mas como a rede Ethereum tem algumas complexidades a mais como dados para armazenamento, é interessante utilizar uma outra estrutura de dados capaz de atender a esta demanda, e a *Patricia Merkle Tree* tem esse propósito.

A *Patricia Merkle Tree* (PMT), também conhecida como Trie-based Merkle Tree, é uma estrutura de dados que combina as árvores de Patricia e as árvores de Merkle. É usada principalmente em sistemas distribuídos e blockchains para garantir a integridade dos dados e a segurança das transações. A árvore de Patricia é uma estrutura de dados de árvore digital que armazena chaves em seus nós internos, em vez de apenas em suas folhas. Isso permite uma rápida pesquisa de prefixo, que é útil em sistemas de indexação de palavras, por exemplo. Já as árvores de Merkle são árvores binárias hash-based, onde cada nó folha é o hash de uma transação e cada nó interno é o hash dos seus nós filhos.

A PMT combina essas duas estruturas, onde cada nó folha da árvore de Patricia armazena um hash do conteúdo correspondente, enquanto os nós internos armazenam hashes das suas sub árvores. Assim, a PMT permite que sejam verificadas as provas de autenticidade e integridade das transações e dos dados armazenados na árvore.

Dessa forma, a *Patricia Merkle Tree* oferece uma solução eficiente para a validação de dados em sistemas distribuídos e blockchains, permitindo que se verifique se as informações armazenadas em diferentes nós são iguais sem a necessidade de transferir grandes quantidades de dados pela rede.

8.0 SOLUÇÕES DE CONTROLE DE PONTO UTILIZADAS NO MERCADO

Conforme mencionado na seção 3 desta obra, o Registro Eletrônico de Ponto (REP) é regulado pela portaria 671/2021 do MTP e reconhece três tipos de soluções para registro eletrônico destas operações, REP-C, REP-A e REP-P. Essas soluções são desenvolvidas por grandes *players* do mercado, inclusive, com um deles, a DIMEP®, fundada em 1936 e desenvolvendo relógios de ponto desde 1959.



Figura 61 – Exemplos de equipamentos físicos para registro de ponto. Fonte: Internet



Figura 62 – Exemplo de software para registro de ponto.¹⁹

¹⁹ FONTE PONTOTEL. <https://www.pontotel.com.br/>

De uma forma geral, podemos dividir esses equipamentos em dois grandes grupos: um deles funcionando como um equipamento independente (*stand alone*) e o outro formado com software de gestão podendo fazer uso, ou não, de algum equipamento físico para os registros. Independente de qual dos dois tipos de solução adotada, todas elas fazem uso de uma arquitetura centralizada, sofrendo de todos os problemas típicos de aplicações desse tipo conforme já mencionamos nas sessões anteriores.

8.1 MODELO DE NEGÓCIO

O modelo de negócio utilizado para este tipo de produto conta basicamente com players de mercado que atuam no segmento de controle de acesso e controle de ponto. Alguns deles fornecem soluções integradas para estes dois cenários e outros não, mas não necessariamente são implementadas as soluções de um só fabricante. Em minha experiência profissional já me deparei em campo com empresas que utilizam sistema de controle de acesso e controle de ponto nas mais diversas associações.

Neste modelo de negócio o player pode vender, alugar ou comodatar o equipamento, podendo, ainda, oferecer um contrato de manutenção mensal para atendimentos ao equipamento e ao software de gestão.

9.0 REGRAS DE NEGÓCIO

Como fruto de determinações legais e dinâmica de uso da solução, descrevemos abaixo as regras de negócio que orientarão o desenvolvimento.

- **Regra de tipo de perfil:** O sistema deve possuir perfil de administrador (gestor) e funcionário.

- **Regra de privilégio de gestor:** Somente a pessoa com perfil de gestor poderá cadastrar as informações do empregador, do funcionário, inserir novos gestores, fazer buscas no sistema e emitir relatórios (salvo condições especiais onde o funcionário poderá ver alguns de seus relatórios). Somente administradores ativos poderão interagir com o sistema.
- **Regra de segurança:** Somente pessoas registradas no sistema poderão interagir com o mesmo para alterar e visualizar dados.
- **Regra de criação de administrador:** O administrador ao ser criado deve possuir uma identificação única (id), nome, número de documento e o estado (ativo ou inativo).
- **Regra de criação de empregador:** O empregador ao ser criado deve possuir informação que o identifique de forma única (id), nome, número de documento e endereço.
- **Regra de criação de funcionário:** O funcionário ao ser criado deve possuir informação que o identifique de maneira única (id), nome, número de documento, jornada de trabalho, empregador e o estado (ativo ou desativado).
- **Regra de criação de dados:** Qualquer dado novo ao ser inserido no sistema deve ser registrado a identificação de quem fez tal adição.
- **Regra de alteração de dados:** Os dados de qualquer entidade do sistema ao serem modificados (nome, documento etc.) devem ser registrados a identificação de quem fez tal modificação.
- **Regra de horário:** o sistema ao ser implantado deve aceitar o ajuste de fuso horário para adequação de acordo com o fuso horário desejado.
- **Regra de tipo de ponto:** Deve haver quatro marcações de ponto: início da jornada, início da pausa, fim da pausa, fim da jornada; e somente perfil funcionário poderá

register ponto.

- **Regra de marcação geral de ponto:** O ponto deve ser marcado mediante identificação do funcionário no sistema e posterior escolha do ponto a ser marcado, com o sistema, de forma automática, reconheça o horário e registre o ponto, devendo o ponto registrar o funcionário, o tipo de ponto marcado e o horário. Somente funcionários ativos marcarão pontos.
- **Regra de marcação de início de jornada:** O início da jornada deverá ser obrigatoriamente a primeira marcação do dia, não permitindo que seja feita uma marcação de outro tipo em seu lugar.
- **Regra de marcação de início da pausa:** A marcação de início da pausa só poderá ser realizada se houver um registro de marcação de início de jornada no mesmo dia.
- **Regra de marcação de fim de pausa:** A marcação de fim de pausa só poderá ser realizada se houver registros de início de jornada e início de pausa anteriores a ela no mesmo dia.
- **Regra de marcação de fim de jornada:** A marcação de fim de jornada só poderá ser marcada se houver registro de marcação de início de jornada no mesmo dia. Caso haja um registro de início de pausa e não haja fim de pausa o sistema deverá se comportar da seguinte forma:
 - Se o intervalo entre o início da pausa e a marcação de fim de jornada for **inferior a 1h**, o sistema deverá registrar primeiro o fim da pausa e logo em seguida o fim da jornada.
 - Se o intervalo entre o início da pausa e a marcação de fim de jornada for **superior a 1h**, o sistema deverá registrar primeiro o fim da pausa com um período de 1h entre o início da pausa e o fim da pausa e logo em seguida o fim da jornada.

- **Regra de emissão de relatório por gestor:** Apenas gestor poderá emitir relatórios fazendo busca por qualquer funcionário e qualquer data ou período.
- **Regra de emissão de relatório por funcionário:** O funcionário registrado poderá emitir relatórios apenas sobre seu registro.

10.0 MODELAGEM DE PROCESSOS

Observando a proposta do sistema observamos três processos distintos que descrevem a operação do sistema, a saber:

- Processo de Cadastro: responsável por cadastrar os perfis e utilizadores do sistema;
- Processo de Marcação de Ponto: responsável pelo funcionário realizar sua marcação de ponto;
- Processo de Emissão de Relatório: responsável pela emissão dos relatórios por parte do perfil administrador e do funcionário.

Na imagem abaixo vemos o processo de cadastro onde um solicitante faz o pedido de um novo cadastro ao administrador e após a validação das informações fornecidas, o administrador procede com a gravação do novo cadastro (administrador, empregador ou funcionário) e retorna com a confirmação para o solicitante.

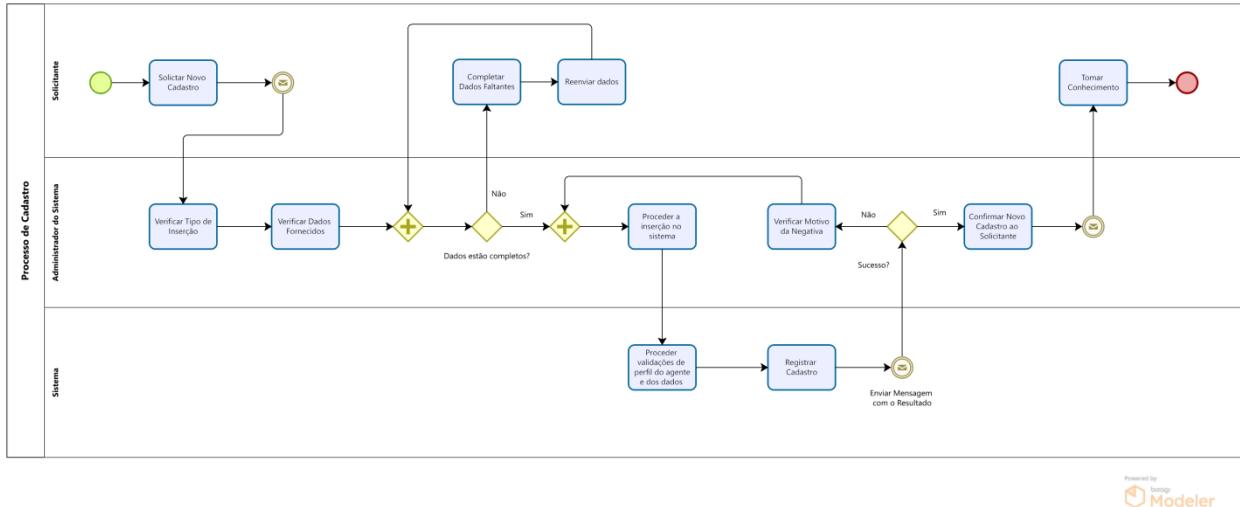


Figura 63 – BPMN - Processo de cadastro. Fonte: Autor

Na imagem seguinte descrevemos o processo de marcação de ponto onde um funcionário após realizar sua identificação e realizar sua validação, a marcação de ponto é realizada.

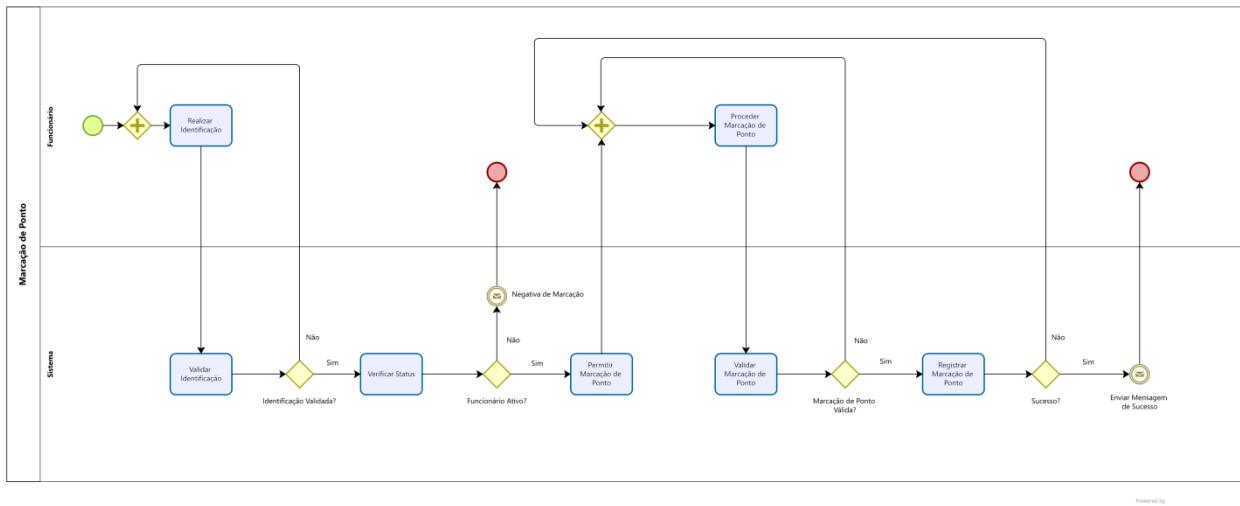


Figura 64 – BPMN - Processo de marcação do ponto. Fonte: Autor

E no último processo, temos a emissão de relatórios, que pode ser feita por administrador ou funcionário, onde o funcionário pode emitir seus relatórios e o administrador pode emitir relatório de qualquer funcionário e em qualquer período de data. Da mesma forma que os anteriores, é feita identificação e validação das ações e perfis.

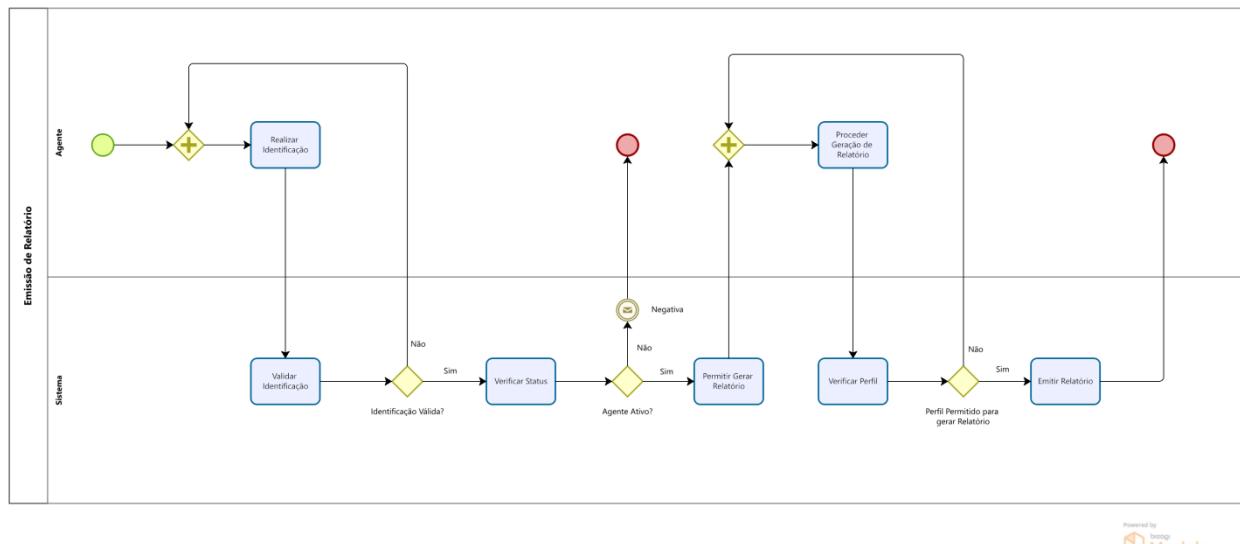


Figura 65 – BPMN - Processo de emissão de relatório. Fonte: Autor

11.0 REQUISITOS

Após reconhecer as regras de negócio e desenhar os processos envolvidos estamos aptos a descrever os requisitos do nosso sistema, assim temos a tabela abaixo:

REQUISITOS FUNCIONAIS	
REF	DESCRIÇÃO
RF01	O sistema deverá incluir pelo menos dois perfis de usuário, um sendo gestor e o outro funcionário, que realizará a marcação de pontos.
RF02	O sistema deverá ser capaz de identificar de forma única e inequívoca cada usuário que utiliza o sistema.
RF03	Somente o perfil administrador poderá cadastrar e atualizar participantes do sistema, devendo sempre registrar o autor dessas ações.
RF04	O sistema poderá aceitar múltiplos usuários de tipo gestor e funcionário, mas cada um terá sua identificação única.
RF05	O sistema deverá cadastrar o empregador com os dados de nome, documento e endereço legal, todos de caráter obrigatório.
RF06	O sistema deverá cadastrar funcionário com os dados de nome, documento, início e fim da jornada de trabalho, seu empregador associado e status (ativo ou inativo), todos de caráter obrigatório.
RF07	O sistema deverá cadastrar o gestor com os dados de nome, documento e status (ativo ou inativo), todos de caráter obrigatório.
RF08	O sistema deverá ser capaz de registrar o fuso horário em que será utilizado.
RF09	O sistema deverá reconhecer 4 tipos de marcação de ponto: início e fim da jornada e início e fim da pausa.
RF10	O sistema deverá permitir que apenas funcionários ativos realizem a marcação de ponto.
RF11	A primeira marcação do dia deverá ser, sempre, início da jornada e na sequência, início da pausa, fim da pausa e fim da jornada, não admitindo inversão de qualquer ordem nem replicações do mesmo tipo de marcação de ponto durante o dia, mas podendo admitir a ausência das marcações de pausa.
RF12	Se houver marcação de início de pausa, obrigatoriamente, deverá ser registrada a marcação de fim da pausa.
RF13	Se durante a marcação de fim de jornada houver ocorrência de início de pausa sem uma marcação de fim de pausa correspondente no mesmo dia, caso o intervalo entre o início da pausa e o fim da jornada seja maior que 1 hora, deverá ser registrada uma marcação de fim da pausa 1 hora após o início da jornada e posteriormente a marcação do fim da jornada do dia, caso esse intervalo seja menor que 1 hora, as marcações de fim de pausa e fim da jornada serão marcadas juntas.
RF14	Todo registro de marcação de ponto deverá conter de forma inequívoca o funcionário, o tipo da marcação e o horário.
RF15	O sistema deverá permitir a emissão de relatórios realizados por funcionário ou gestor onde, o gestor poderá emitir relatório de marcação de ponto realizado por qualquer funcionário em qualquer dia e o funcionário apenas as suas marcações.
RF16	O registros de marcação de ponto devem estar disponíveis a qualquer tempo para visualização de seus relatórios, independente do horário do dia.

REQUISITOS NÃO FUNCIONAIS	
REF	DESCRIÇÃO
RNF01	O sistema deverá ser desenvolvido em ambiente web para que possa ser utilizado por qualquer funcionário, dentro ou fora da empresa.
RNF02	Devem ser utilizadas soluções Open Source em seu desenvolvimento.
RNF03	Devem ser respeitadas as determinações da LGPD.
RNF04	O sistema deverá funcionar em ambiente independente da empresa de forma que mesmo que a sede da empresa esteja fechada ou inoperante por qualquer motivo seja possível a marcação do ponto.
RNF05	Deverão ser desenvolvidos testes unitários para validar cada uma das operações do sistema, seja de algum tipo de cadastro, marcações de ponto ou emissão de relatórios.
RNF06	O desenvolvimento do sistema deve ser feito em diferentes camadas de forma que as regras de negócio possam ser evoluídas de forma independente da interface web utilizada.
RNF07	Os dados devem ser persistidos em uma estrutura perene que garanta a gravação individual de cada registro e independente, alinhando-se às propriedades ACID.

12.0 CASOS DE USO

Casos de uso são ferramentas importantes que nos ajudam a perceber e definir a forma de interação dos usuários com o sistema proposto. De acordo com a modelagem dos processos, reconhecemos três principais tipos de interação com o sistema a saber: Cadastro, Marcação de Ponto e Emissão de Relatório, assim, apresentamos abaixo caso de uso com sua representação.

Durante nosso desenvolvimento serão utilizadas algumas nomenclaturas que nos ajudarão a simplificar, organizar e reconhecer os elementos no documento de análise, assim sendo, temos:

UCXX: Use Case (Caso de Uso) – Descrição de caso de uso. O **XX** representa uma sequência numérica de dois dígitos, sequencial e único para cada caso de uso, começando em 00. Exemplo: UC01 – Caso de uso 01.

AFXX-YY: Alternative Flow (Fluxo Alternativo) – Descrição de fluxo alternativo dentro de um caso de uso. O **XX** representa uma sequência numérica de dois dígitos, sequencial e única para o caso de uso em questão. O **YY** representa uma sequência numérica de dois dígitos e única indicando o número do fluxo alternativo para um caso de uso começando em 01. Exemplo: AF03-01 – Fluxo alternativo 01 do caso de uso 03.

Para o nosso sistema foram relacionados os seguintes casos de uso:

UC	NOME	DESCRIÇÃO
00	Geral	Visão geral do sistema
01	Manter Administrador	Visão das ações de cadastro de administrador
02	Manter Empregador	Visão das ações de cadastro de Empregador
03	Manter Funcionário	Visão das ações de cadastro de funcionário
04	Emitir Relatório	Visão das ações para emissão de relatório
05	Marcar Ponto	Visão das ações para marcação de ponto

12.1 UC00 - GERAL

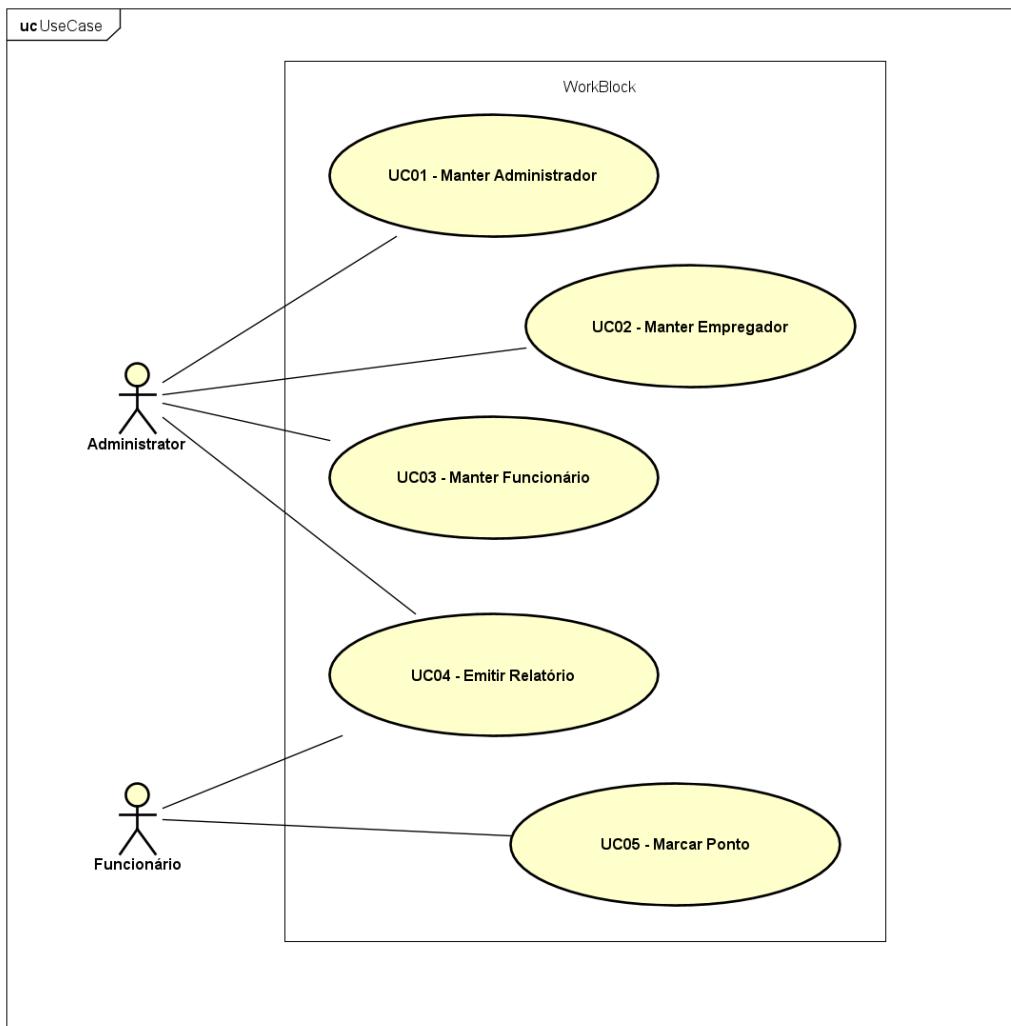


Figura 66 – UC00 – Diagrama de Casos de Uso. Fonte: Autor

Sumário: Estrutura geral e relacionamentos entre atores e caso de uso.

Atores:

Administrador: Ator que ocupa posição de gestão cadastrando dados e emitindo relatórios.

Este ator se relaciona com os casos de uso:

- UC01 - Manter Administrador,
- UC02 - Manter Empregador,

- UC03 - Manter Funcionário,
- UC04 - Emitir Relatório

Fucionário: Ator que faz uso do sistema na ação de marcação de ponto e ocasionalmente emite seus relatórios de marcação de ponto. Este ator se relaciona com os casos de uso:

- UC04 - Emitir Relatório,
- UC05 - Marcar Ponto

12.2 UC01 – MANTER ADMINISTRADOR

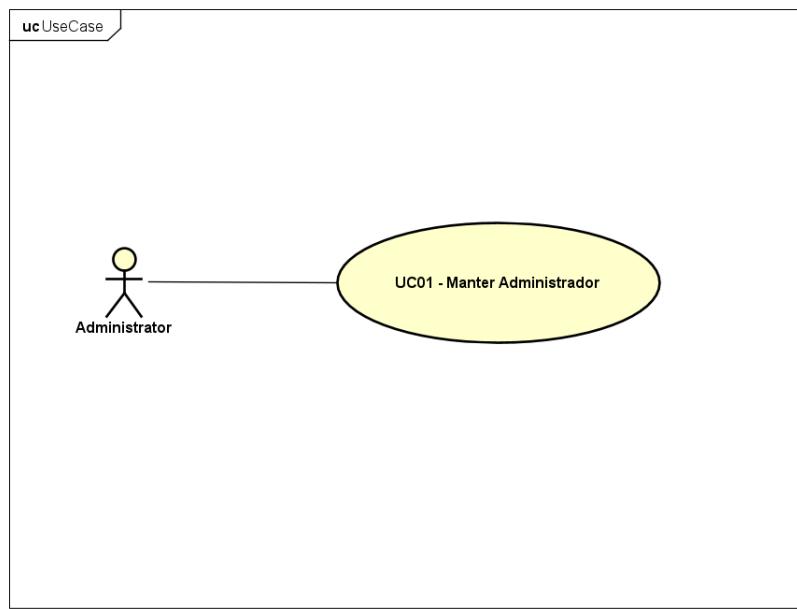


Figura 67 – UC01 – Casos de Uso Manter Administrador. Fonte: Autor

Sumário: O ator Administrador mantém administrador no sistema.

Autor principal: Administrador

Pré-condição:

1) Ator cadastrado no sistema e ativo

Pós-condições:

1) Administrador mantido no sistema e apto para executar ações.

Fluxo Principal:

1) O Ator principal coleta as informações necessárias para cadastro no sistema. Informações necessárias para cadastro:

i - nome completo.

i.a - Se pessoa física, o nome registrado no documento oficial, se pessoa jurídica, a razão social.

ii - numero de documento oficial válido.

ii.a - Se pessoa física, preferencialmente CPF, se pessoa jurídica, preferencialmente CNPJ.

Fluxo Alternativo [AF01-01]:

- 1) Ator pode buscar administradores cadastrados no sistema.

Fluxo Alternativo [AF01-02]:

- 1) Ator pode atualizar dados de administradores cadastrados no sistema.

12.3 UC02 – MANTER EMPREGADOR

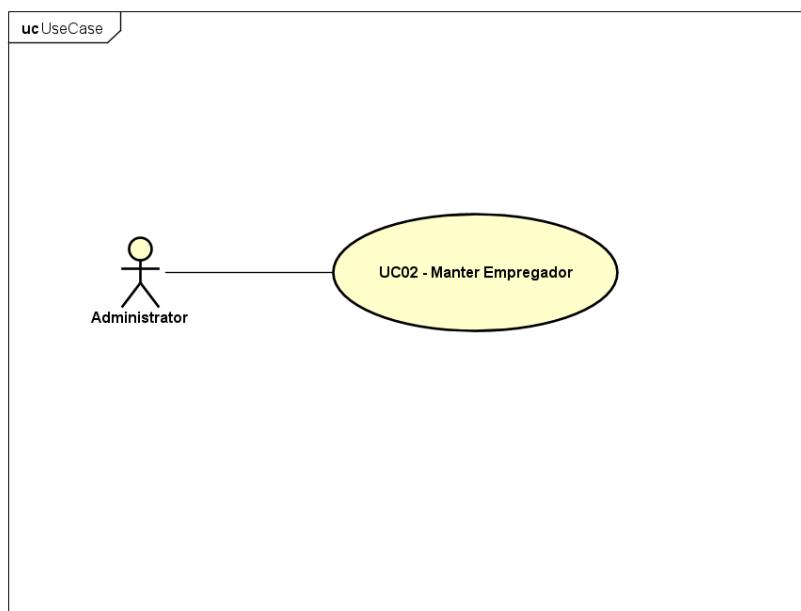


Figura 68 – UC02 – Casos de Uso Manter Empregador. Fonte: Autor

Sumário: O ator Administrador mantém empregador no sistema.

Ator principal: Administrador

Pré-condição:

1) Ator cadastrado no sistema e ativo

Pós-condições:

1) Empregador mantido no sistema.

Fluxo Principal:

1) O Ator principal coleta as informações necessárias para cadastro no sistema. Informações necessárias para cadastro:

i - nome completo.

i.a - Se pessoa física, o nome registrado no documento oficial, se pessoa jurídica, a razão social.

ii - numero de documento oficial válido.

ii.a - Se pessoa física, preferencialmente CPF, se pessoa jurídica, preferencialmente CNPJ.

iii - endereço, localização, do empregador

Fluxo Alternativo [AF02-01]:

1) Ator pode buscar empregadores cadastrados no sistema.

Fluxo Alternativo [AF02-02]:

1) Ator pode atualizar dados de empregadores cadastrados no sistema.

12.4 UC03 – MANTER FUNCIONÁRIO

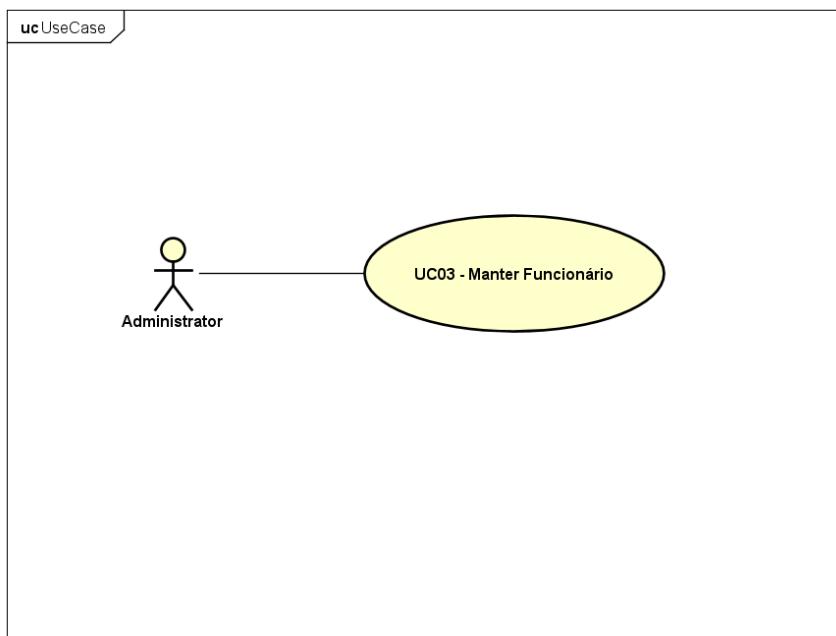


Figura 69 – UC03 – Casos de Uso Manter Funcionário. Fonte: Autor

Sumário: O ator Administrador mantém funcionário no sistema.

Autor principal: Administrador

Pré-condição:

1) Ator cadastrado no sistema e ativo

Pós-condições:

1) Funcionário mantido no sistema.

Fluxo Principal:

1) O Ator principal coleta as informações necessárias para cadastro no sistema. Informações necessárias para cadastro:

i - nome completo registrado no documento oficial.

ii - numero de documento oficial válido, preferencialmente PIS ou CPF.

iii - hora de início e fim da jornada de trabalho do funcionário.

iv - referência do empregador

Fluxo Alternativo [AF02-01]:

- 1) Ator pode buscar funcionários cadastrados no sistema.

Fluxo Alternativo [AF02-02]:

- 1) Ator pode atualizar dados de funcionários cadastrados no sistema.

12.5 UC04 – EMITIR RELATÓRIO

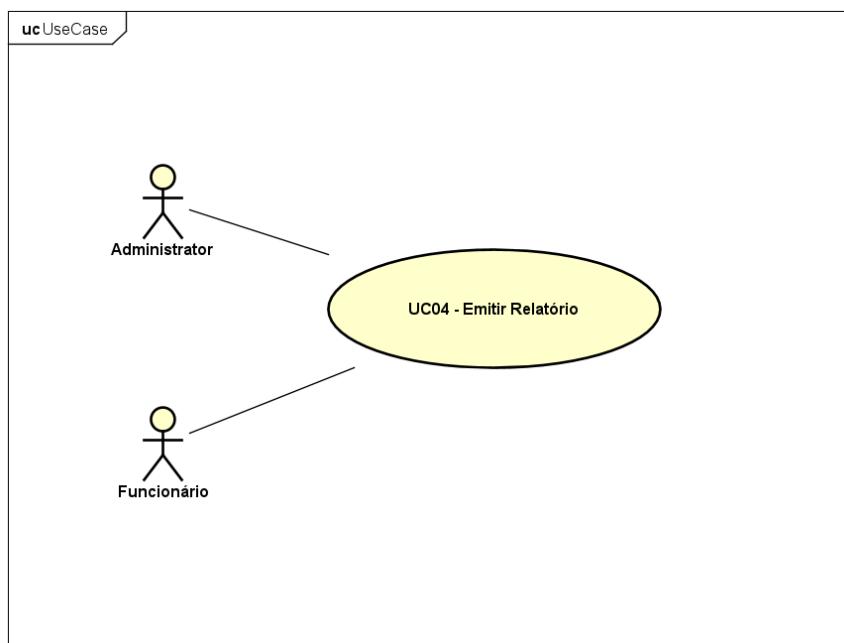


Figura 70 – UC04 – Casos de Uso Emitir Relatório. Fonte: Autor

Sumário: Os atores emitem relatório de marcação de ponto.

Ator principal: Administrador

Autor secundário: Funcionário

Pré-condição:

- 1) Atores cadastrados no sistema e ativos.
- 2.a) Ator secundário consulta apenas os seus registros

Pós-condições:

- 1) Relatório desejado emitido pelo sistema.

Fluxo Principal:

- 1) Os Atores entram com informação do funcionário e data buscada
- 2) O sistema emite o relatório desejado.

- i - Se o agente for o Ator Secundário, o sistema valida para que a busca seja permitida apenas para os dados referentes a seu registro.
- ii - Se o agente for o Ator Principal, o sistema permite a busca de registros de qualquer funcionário e qualquer data.

12.6 UC05 – MARCAR PONTO

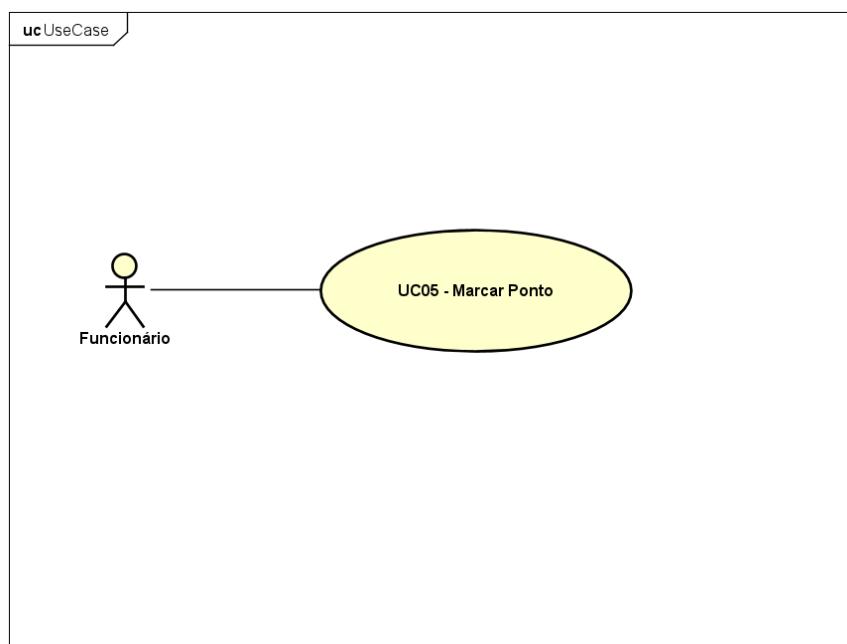


Figura 71 – UC05 – Casos de Uso Marcar Ponto. Fonte: Autor

Sumário: O ator faz a marcação de ponto.

Autor principal: Funcionário

Pré-condição:

- 1) Ator cadastrado no sistema e ativo.

Pós-condições:

- 1) Marcação de ponto registrada sistema.

Fluxo Principal:

- 1) Os Ator escolhe o ponto a ser marcado
- 2) O sistema valida a solicitação de marcação de ponto
- 3) O sistema registra a marcação de ponto realizada.

13.0 DIAGRAMA DE CLASSE

O Diagrama de Classe descreve as entidades do sistema com seus atributos e comportamentos e seus relacionamentos entre si. Durante nosso desenvolvimento serão utilizadas algumas nomenclaturas que nos ajudarão a simplificar, organizar e reconhecer os elementos no documento de análise, assim sendo, temos:

CDXX: Class Diagram (Diagrama de Classe) – Diagrama de Classe. O **XX** representa uma sequência numérica de dois dígitos, sequencial e único para cada diagrama de classe, começando em 00. Exemplo: CD01 – Diagrama de Classe 01.

Para o nosso sistema foram relacionados os seguintes diagramas de classe:

CD	NOME	DESCRIÇÃO
00	Contracts	Relacionamento entre as os Smart Contracts
01	UtilContract	Smart Contract UtilContract
02	AdministratorContract	Smart Contract AdministratorContract
03	EmployerContract	Smart Contract EmployerContract
04	EmployeeContract	Smart Contract EmployeeContract
05	PontoBlock	Smart Contract PontoBlock
06	PontoBlockReports	Smart Contract PontoBlockReports

13.1 CD00 - CONTRACTS

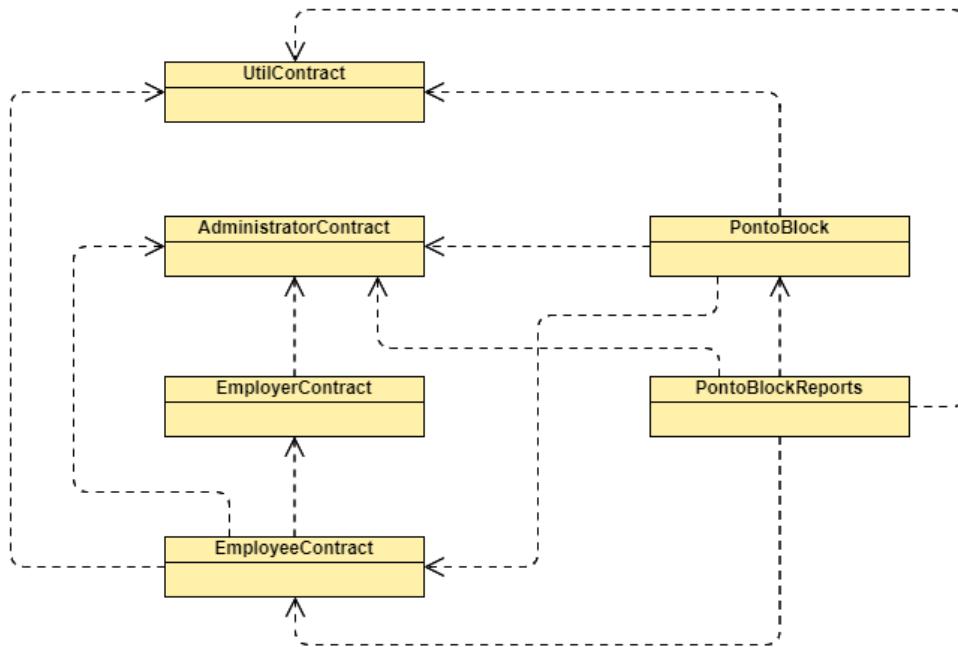


Figura 72 – CD00 – Contracts. Fonte: Autor

Sumário: Estrutura geral e relacionamentos entre os Smart Contracts.

Entidades:

UtilContract: Representação do Smart Contract UtilContract.

Administrator: Struct que representa as propriedades do administrador.

State: Enum que representa o status, inativo ou ativo.

EmployerContract: Representação do Smart Contract EmployerContract.

Employer: Struct que representa as propriedades do empregador.

EmployeeContract: Representação do Smart Contract EmployeeContract.

Employee: Struct que representa as propriedades do funcionário.

PontoBlock: Representação do Smart Contract PontoBlock.

EmployeeRecord: Struct que representa as propriedades do registro das gravações de ponto.

PontoBlockReports: Representação do Smart Contract PontoBlockReports.

13.2 CD01 – UTIL CONTRACT

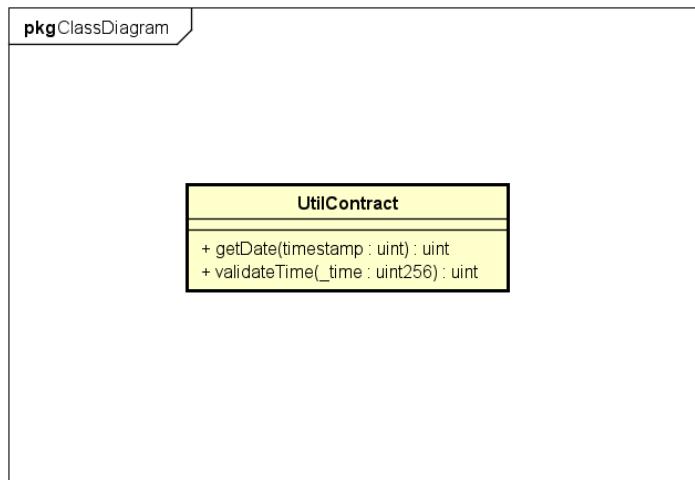


Figura 73 – CD01 – Util Contract. Fonte: Autor

O primeiro contrato desenvolvido é o **UtilContract**. Este contrato é responsável por executar funções genéricas que não tem uma relação específica a um contrato, mas funções de caráter geral.

As funções neste contrato são **getDate** e **validateTime**. A função **getDate** tem como parâmetro um *timestamp* e retorna um *uint* no formato AAAAMMDD onde, AAAA representa o ano de 4 dígitos, MM representa o mês com 2 dígitos e DD representa o dia com 2 dígitos. Já a função **validateTime** é responsável com base em um *uint256* de entrada no formato hhmm onde, hh representa a hora e mm representa o minuto, retorna um booleano informando se o formato passado é válido ou não.

13.3 CD02 – ADMINISTRATOR CONTRACT

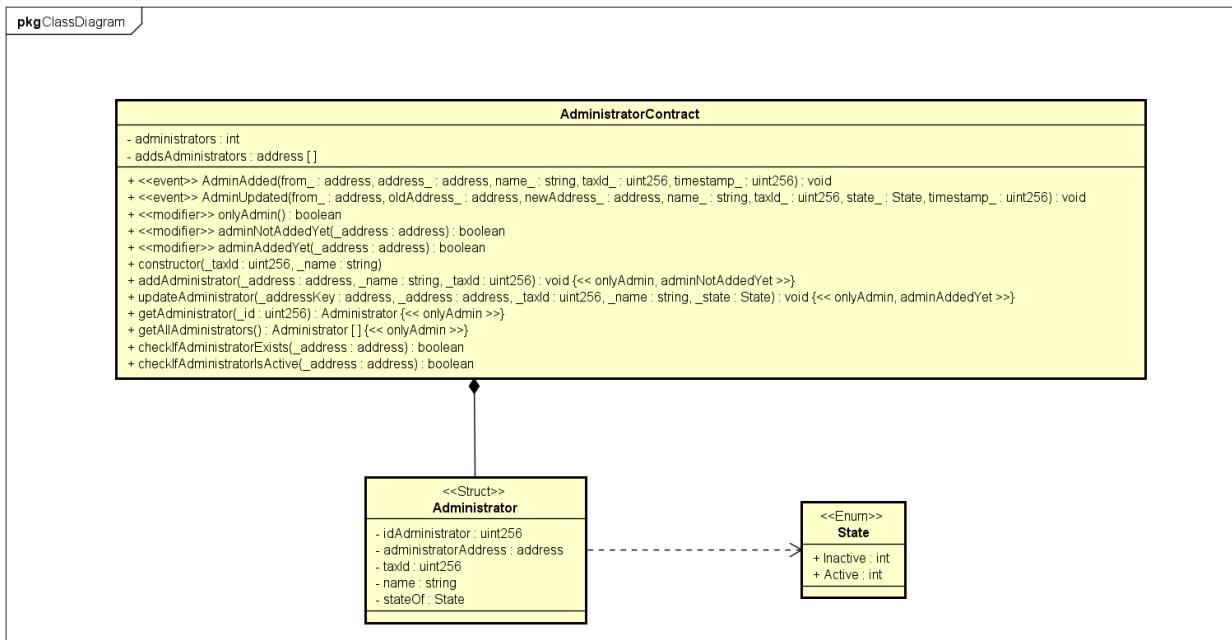


Figura 74 – CD02 – Administrator Contract. Fonte: Autor

O próximo contrato desenvolvido é o **AdministratorContract**. Este contrato é responsável manter os administradores do sistema. Vale lembrar que os contratos também são conhecidos por suas contas, assim como os usuários, assim sendo, este contrato guarda as contas (de usuário e de contrato) com perfil administrador. Para as contas de usuário, a referência ao **AdministratorContract** se faz necessária porque há algumas operações como adicionar funcionário, exibir relatório, entre outras, que só podem ser feitas por agente permissionado. E quanto aos contratos, como há relações entre as funções de um contrato com outro, eles também devem ser colocados nesta categoria.

Em sua composição há um Struct para armazenar os dados do administrador com:

- o *id* sendo um *uint256* que guarda um sequencial a partir de zero,
- o endereço do administrador sendo um *address*,

- o taxId representando o documento do administrador sendo um *uint256*,
- o nome, sendo uma *string*,
- o estado, representando a situação do administrador podendo ser inativo ou ativo.

Há também o Enum para representar a situação, Inativo ou Ativo.

Duas variáveis, um array de *address* e um *mapping* de *uint256* para o Struct de Administrador para armazenar a lista de Administradores.

Quanto aos métodos, há 12 funções neste contrato, sendo duas funções de evento, três funções modificadoras, um construtor e 6 funções de manipulação da entidade, sendo:

- Evento *AdminAdded*, sendo uma função especial, tipo evento, que atua como um log onde após um administrador ser adicionado registra esse log com os dados do agente da ação (*from_*), o endereço do administrador adicionado (*address_*), nome (*name_*), documento (*taxId_*) e timestamp da gravação (*timestamp_*);
- Evento *AdminUpdated*, sendo uma função especial, tipo evento, que atua como um log onde após um administrador ser atualizado, registra esse log com os dados do agente da ação (*from_*), o antigo endereço do administrador (*oldAddress_*), novo endereço do administrador (*newAddress_*), nome atualizado (*name_*), documento atualizado (*taxId_*), estado (*state_*) e timestamp da gravação (*timestamp_*);
- Modificador *onlyAdmin*, sendo uma função que faz uma validação verificando se o agente da ação está incluso na coleção de administradores. Seu resultado positivo permite que a função seja executada;
- Modificador *adminNotAddedYet*, sendo uma função que faz uma validação verificando se o endereço fornecido como parâmetro ainda não foi inserido na coleção de administradores. Seu resultado positivo, permite que a função seja executada;
- Modificador *adminAddedYet*, sendo uma função que faz uma validação verificando se

o endereço fornecido como parâmetro já foi inserido na coleção de administradores.

Seu resultado positivo, permite que a função seja executada;

- O *constructor*, que tem como parâmetros um *uint256* para guardar o documento do administrador e uma *string* para guardar seu nome;
- O *addAdministrator*, que tem como parâmetros um *address*, uma *string* e um *uint256* para armazenar o endereço, nome e documento do administrador, respectivamente;
- O *updateAdministrator*, tendo como parâmetros dois *address* para buscar o administrador e o seu novo endereço, *uint256* para seu documento, *string*, nome, e *State* para situação;
- O *getAdministrator*, que com base em um *uint256* busca um administrador no *mapping administrators*;
- O *getAllAdministrators*, que retorna um array com todos os administradores;
- O *checkIfAdministratorExists*, que com base em um *address* dado, verifica se ele existe no array *addsAdministrators*.
- Por último, *checkIfAdministratorIsActive*, que com base em um *address* dado, verifica se o administrador está ativo.

13.4 CD03 – EMPLOYER CONTRACT

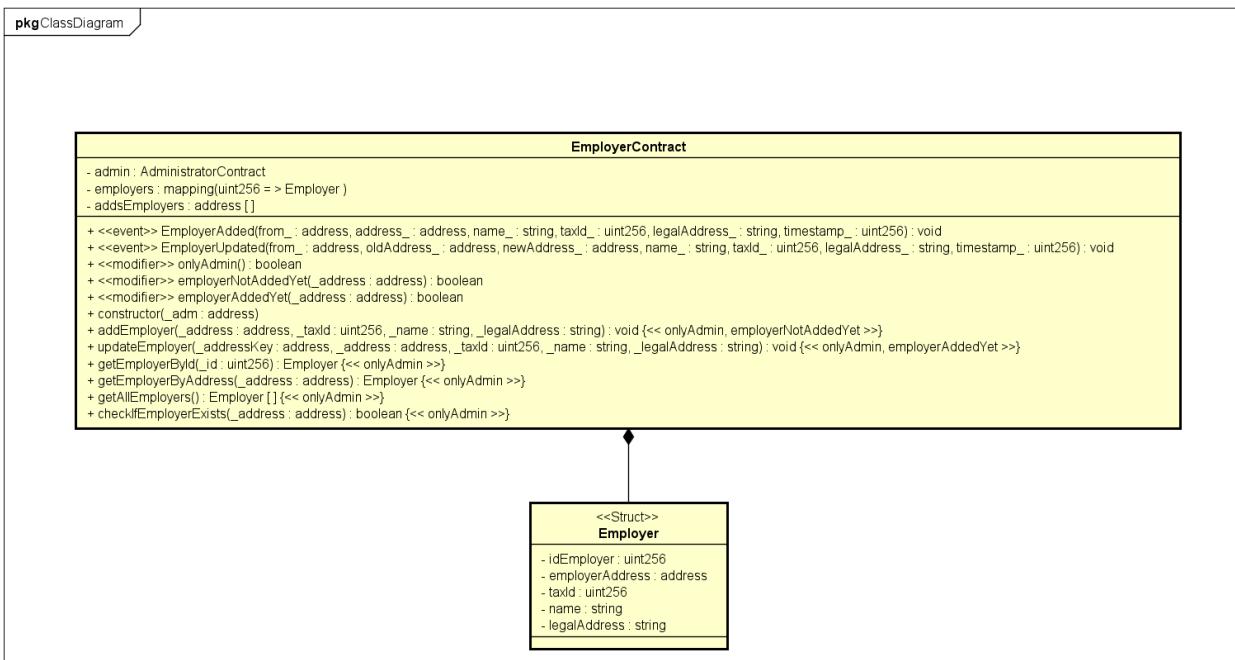


Figura 75 – CD03 – Employer Contract. Fonte: Autor

O **EmployerContract** é responsável manter os empregadores do sistema.

Em sua composição há um Struct para armazenar os dados do empregador com:

- o *id* sendo um *uint256* que guarda um sequencial a partir de zero,
- o endereço do empregador, sendo um *address*,
- o *taxId* representando o documento do empregador, sendo um *uint256*,
- o nome, sendo uma *string*,
- o *legalAddress*, representando o endereço físico de registro do empregador.

Duas variáveis, um array de *address* e um *mapping* de *uint256* para o Struct de Employer para armazenar a coleção de Employers, e *admin*, para referenciar o AdministratorContract, compõem as variáveis deste Smart Contract.

Quanto aos métodos, há 12 funções neste contrato, sendo duas funções de evento, três funções modificadoras, um construtor e 6 funções de manipulação da entidade, sendo:

- Evento *EmployerAdded*, sendo uma função especial, tipo evento, que atua como um log onde após um empregador ser adicionado, registra esse log com os dados do agente da ação (*from_*), o endereço do empregador adicionado (*address_*), nome (*name_*), documento (*taxId_*); endereço legal (*legalAddress_*) e timestamp da gravação (*timestamp_*);
- Evento *EmployerUpdated*, sendo uma função especial, tipo evento, que atua como um log onde após um empregador ser atualizado, registra esse log com os dados do agente da ação (*from_*), o antigo endereço do empregador (*oldAddress_*), novo endereço do empregador (*newAddress_*), nome atualizado (*name_*), documento atualizado (*taxId_*), endereço legal (*legalAddress_*) e timestamp da gravação (*timestamp_*);
- Modificador *onlyAdmin*, sendo uma função que faz uma validação verificando se o agente da ação está incluso na coleção de administradores. Seu resultado positivo permite que a função seja executada;
- Modificador *employerNotAddedYet*, sendo uma função que faz uma validação verificando se o endereço fornecido como parâmetro ainda não foi inserido na coleção de Employers. Seu resultado positivo, permite que a função seja executada;
- Modificador *employerAddedYet*, sendo uma função que faz uma validação verificando se o endereço fornecido como parâmetro já foi inserido na coleção de Employers. Seu resultado positivo, permite que a função seja executada;
- O *constructor*, que tem como parâmetros um *address* para referenciar o endereço do contrato *AdministratorContract*;
- O *addEmployer*, que tem como parâmetros um *address*, um *uint256*, duas *string* para

armazenar o endereço, documento, nome e endereço legal do empregador, respectivamente;

- O *updateEmployer*, tendo como parâmetros dois *address* para buscar o administrador e o seu novo endereço, *uint256* para seu documento e duas *string* para o nome e endereço legal, atuando na atualização dos dados do empregador;
- O *getEmployerById*, que com base em um *uint256* busca um empregador na coleção de Employers;
- O *getEmployerByAddress*, que com base em um *address* busca um empregador na coleção de Employers;
- O *getAllEmployers*, que retorna um array com todos os empregadores;
- Por último, *checkIfEmployerExists*, que com base em um *address* dado, verifica se ele existe no array *addsEmployers*.

13.5 CD04 – EMPLOYEE CONTRACT

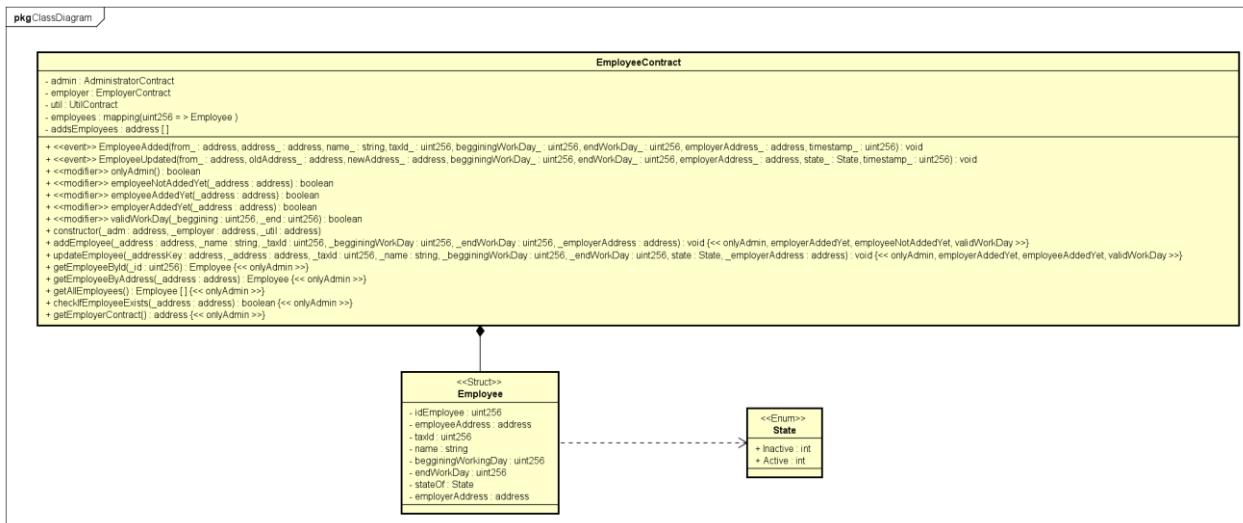


Figura 76 – CD04 – Employee Contract. Fonte: Autor

O **EmployeeContract** é responsável manter os funcionários do sistema.

Em sua composição há um Struct para armazenar os dados do funcionário com:

- o *id* sendo um *uint256* que guarda um sequencial a partir de zero,
- o endereço do funcionário, sendo um *address*,
- o *taxId* representando o documento do funcionário, sendo um *uint256*,
- o nome, sendo uma *string*,
- o início da jornada de trabalho, sendo um *uint256*,
- o fim da jornada de trabalho, sendo um *uint256*,
- O estado, inativo ou ativo,
- Endereço do empregador, sendo um *address* representando o endereço do empregador.

Cinco variáveis, um array de *address* e um *mapping* de *uint256* para o Struct de

Employee para armazenar a coleção de Employees, admin, para referenciar o AdministratorContract, employer, para referenciar o EmployerContract, e util, para referenciar o UtilContract, compõem as variáveis deste Smart Contract.

Quanto aos métodos, há 15 funções neste contrato, sendo duas funções de evento, cinco funções modificadoras, um construtor e 7 funções de manipulação da entidade, sendo:

- Evento *EmployeeAdded*, sendo uma função especial, tipo evento, que atua como um log onde após um funcionário ser adicionado, registra esse log com os dados do agente da ação (from_), o endereço do funcionário adicionado (address_), nome (name_), documento (taxId_); início da jornada de trabalho (begginingWorkDay_), fim da jornada (endWorkDay_), endereço do empregador (employerAddress_) e timestamp da gravação (timestamp_);
- Evento *EmployeeUpdated*, sendo uma função especial, tipo evento, que atua como um log onde após um funcionário ser atualizado, registra esse log com os dados do agente da ação (from_), o antigo endereço do funcionário (oldAddress_), novo endereço do funcionário (newAddress_), nome atualizado (name_), documento atualizado (taxId_), início da jornada de trabalho (begginingWorkDay_), fim da jornada (endWorkDay_), endereço do empregador (employerAddress_), estado (state_) e timestamp da gravação (timestamp_);
- Modificador *onlyAdmin*, sendo uma função que faz uma validação verificando se o agente da ação está incluso na coleção de administradores. Seu resultado positivo permite que a função seja executada;
- Modificador *employeeNotAddedYet*, sendo uma função que faz uma validação verificando se o endereço fornecido como parâmetro ainda não foi inserido na coleção de Employees. Seu resultado positivo, permite que a função seja executada;
- Modificador *employeeAddedYet*, sendo uma função que faz uma validação verificando

se o endereço fornecido como parâmetro já foi inserido na coleção de Employees. Seu resultado positivo, permite que a função seja executada;

- Modificador `employerAddedYet`, sendo uma função que faz uma validação verificando se o endereço fornecido como parâmetro já foi inserido na coleção de Employers. Seu resultado positivo, permite que a função seja executada;
- Modificador `validWorkDay`, sendo uma função que faz uma validação verificando se os dois parâmetros `_beginning` e `_end` compõem um horário válido. Seu resultado positivo, permite que a função seja executada;
- O `constructor`, que tem como parâmetros três `address` para referenciar o endereço dos contratos `AdministratorContract`, `EmployerContract` e `UtilContract`;
- O `addEmployee`, que tem como parâmetros dois `address`, uma `string` e três `uint256` para armazenar o endereço do funcionário e do empregador, nome, documento, início e fim da jornada de trabalho do funcionário;
- O `updateEmployee`, tendo como parâmetros dois `address` para buscar o funcionário e o seu novo endereço, `uint256` para seu documento, uma `string` para o nome, dois `uint256` para início e fim da jornada de trabalho, um `address` para endereço do empregador e `State` para o estado, atuando na atualização dos dados do funcionário;
- O `getEmployeeById`, que com base em um `uint256` busca um funcionário na coleção de Employees;
- O `getEmployeeByAddress`, que com base em um `address` busca um funcionário na coleção de Employees;
- O `getAllEmployees`, que retorna um array com todos os funcionários;
- O `checkIfEmployeeExists`, que com base em um `address` dado, verifica se ele existe no array `addsEmployees`;

- *getEmployerContract*, que fornece o endereço do EmployerContract.

13.6 CD05 – PONTO BLOCK

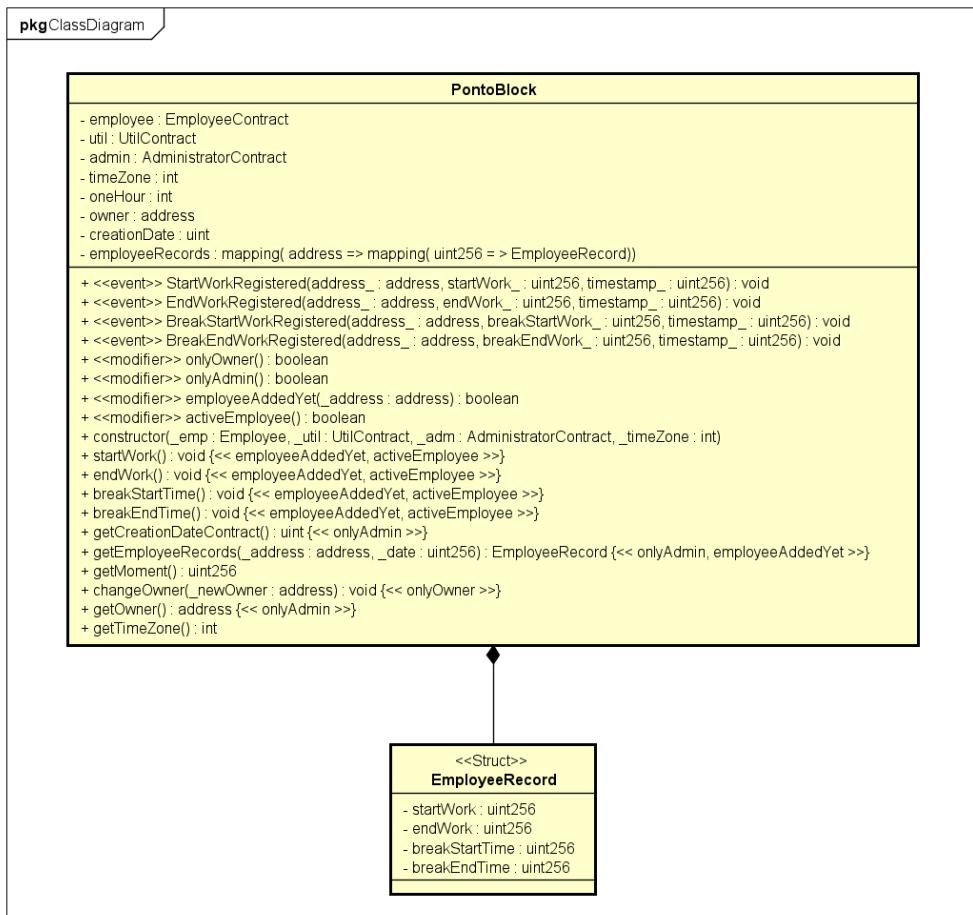


Figura 77 – CD05 – Ponto Block. Fonte: Autor

O **PontoBlock** é responsável manter os registros de marcação de ponto dos funcionários.

Em sua composição há um Struct para armazenar os dados da marcação de ponto com:

- o startWork, sendo um *uint256* que guarda o timestamp da hora da marcação de ponto do início da jornada,
- o endWork, sendo um *uint256* que guarda o timestamp da hora da marcação de ponto

do fim da jornada,

- o breakStartTime, sendo um *uint256* que guarda o timestamp da hora da marcação de ponto do início da pausa,
- o breakEndTime, sendo um *uint256* que guarda o timestamp da hora da marcação de ponto do fim da pausa.

Oito variáveis, um mapping de *address* de Employee que aponta para um mapping de *uint256* para armazenar uma data (YYYYMMDD) que aponta para o Struct EmployeeRecord, um *uint* para armazenar a data de criação do contrato, um *address* para o endereço do dono do contrato, *int* para a representação de 1h, *int* para o Time Zone, admin, util e employee para a referência dos contractos AdministratorContract, UtilContract e EmployeeContract, respectivamente, compõem as variáveis deste Smart Contract.

Quanto aos métodos, há 19 funções neste contrato, sendo quatro funções de evento, quatro funções modificadoras, um construtor e 10 funções de manipulação da entidade, sendo:

- Evento *StartWorkRegistered*, sendo uma função especial, tipo evento, que atua como um log onde após a marcação de início de jornada, registra esse log com os dados do funcionário (*address_*), timestamp da marcação (*startWork_*) e timestamp da gravação (*timestamp_*);
- Evento *EndWorkRegistered*, sendo uma função especial, tipo evento, que atua como um log onde após a marcação de fim de jornada, registra esse log com os dados do funcionário (*address_*), timestamp da marcação (*endWork_*) e timestamp da gravação (*timestamp_*);
- Evento *BreakStartWorkRegistered*, sendo uma função especial, tipo evento, que atua como um log onde após a marcação de início da pausa, registra esse log com os dados do funcionário (*address_*), timestamp da marcação (*breakStartWork_*) e timestamp da

gravação (`timestamp_`);

- Evento `BreakEndWorkRegistered`, sendo uma função especial, tipo evento, que atua como um log onde após a marcação de fim da pausa, registra esse log com os dados do funcionário (`address_`), timestamp da marcação (`breakEndWork_`) e timestamp da gravação (`timestamp_`);
- Modificador `onlyOwner`, sendo uma função que faz uma validação verificando se o agente da ação é o dono do contrato. Seu resultado positivo permite que a função seja executada;
- Modificador `onlyAdmin`, sendo uma função que faz uma validação verificando se o agente da ação está incluso na coleção de administradores. Seu resultado positivo permite que a função seja executada;
- Modificador `employeeAddedYet`, sendo uma função que faz uma validação verificando se o endereço fornecido como parâmetro já foi inserido na coleção de Employees. Seu resultado positivo, permite que a função seja executada;
- O `constructor`, que tem como parâmetros três `address` para referenciar o endereço dos contratos `AdministratorContract`, `EmployeeContract` e `UtilContract` e um `int` para referenciar o Time Zone do contrato;
- O `startWork` é a função que faz o registro da marcação do início da jornada;
- O `endWork` é a função que faz o registro da marcação de fim da jornada;
- O `breakStartTime` é a função que faz o registro de marcação de início da pausa;
- O `breakEndTime` é a função que faz o registro de marcação de fim da pausa;
- O `getCreationDateContract` fornece a data em que o contrato foi criado;
- O `getEmployeeRecords`, que com base em um `address` e um `uint256` dados, retorna a Struct `EmployeeRecord`;

- *getMoment*, que fornece o endereço o timestamp ajustado com o Time Zone;
- *changeOwner*, fornece o endereço do dono do contrato;
- E por último, *getTimeZone*, que retorna um *int* para o Time Zone.

13.7 CD06 – PONTO BLOCK REPORTS

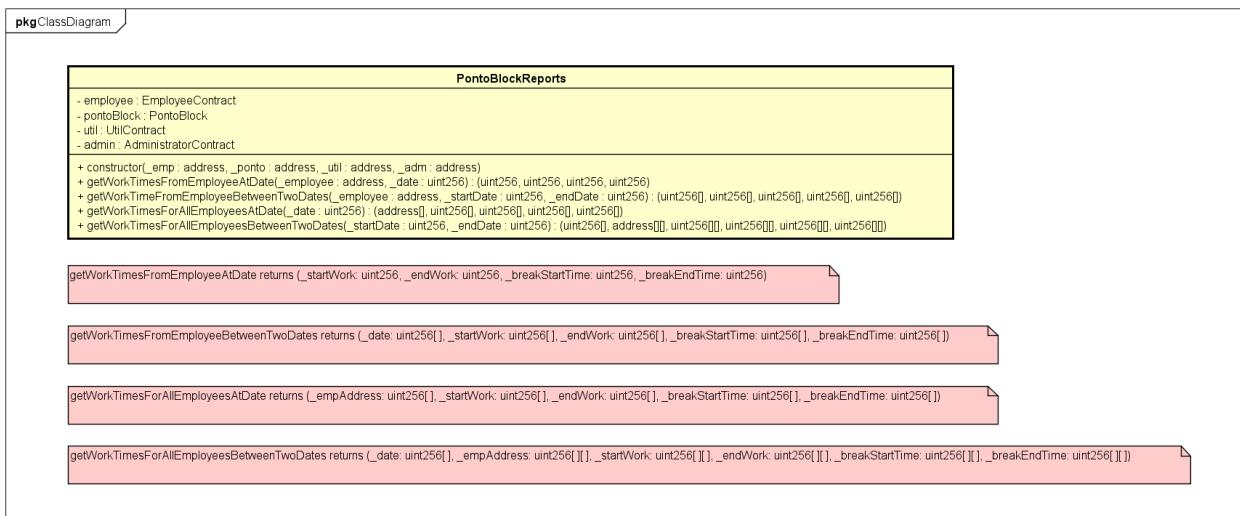


Figura 78 – CD06 – Ponto Block Reports. Fonte: Autor

O **PontoBlockReports** é responsável retornar os registros das marcações de ponto com base na pesquisa por endereços dos funcionários e a data.

- Em sua composição há quatro variáveis para referenciar os contratos `EmployeeContract`, `PontoBlock`, `UtilContract` e `AdministratorContract`.

Quanto aos métodos, há 5 funções neste contrato, sendo um construtor e 4 funções de manipulação da entidade, sendo:

- O *constructor*, que tem como parâmetros quatro *address* para referenciar o endereço dos contratos `AdministratorContract`, `EmployeeContract` e `UtilContract` e `PontoBlock`;
- O *getWorkTimesFromEmployeeAtDate*, que tem como parâmetros um *address* para referenciar o endereço de um funcionário e um *uint256* para referenciar uma data e retorna quatro *uint256* para representar o timestamp para cada uma das marcações de ponto do funcionário no dia;
- O *getWorkTimeFromEmployeeBetweenTwoDates*, que tem como parâmetros um *address* para referenciar o endereço de um funcionário e dois *uint256* para referenciar

uma data início e uma data fim e retorna cinco array de *uint256* para representar o dia e o timestamp para cada uma das marcações de ponto do funcionário no intervalo de dias;

- O *getWorkTimesForAllEmployeesAtDate*, que tem como parâmetros um *uint256* para referenciar uma data e retorna um array de *address* e quatro array de *uint256* para representar os endereços dos funcionários os timestamps das marcações de ponto do dia;
- O *getWorkTimesForAllEmployeesBetweenTwoDates*, que tem como parâmetros dois *uint256* para referenciar uma data início e uma data fim e retorna um array de *uint256* para representar o dia, uma matriz *address* de duas dimensões para representar os endereços no dia, e quatro matriz de *uint256* de duas dimensões para representar o timestamp de cada uma das marcações de ponto do funcionário no intervalo de dias.

14.0 DIAGRAMA DE SEQUÊNCIA

O diagrama de sequência mostra a troca de informações entre os objetos do sistema e como nossa solução tem uma estrutura particular com solidity, considero importante representar a relação entre as estruturas, muito em especial por conta da participação das funções auxiliares modificadoras que validam algumas ações e eventos que registram os logs. Abaixo a representação do Diagrama de Sequência para representar a ação de registro do início da jornada de trabalho.

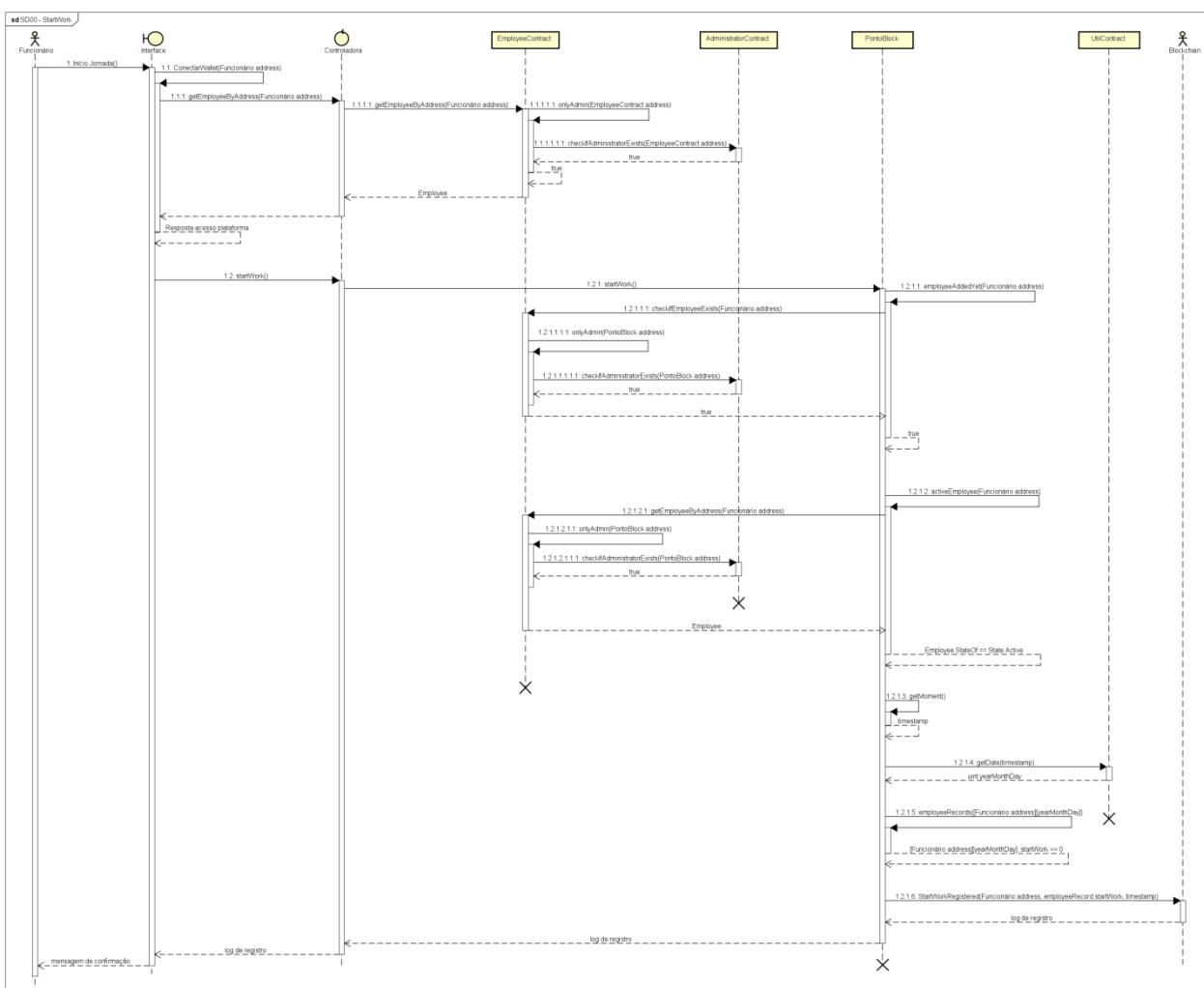


Figura 79 – SEQ – Diagrama de Sequência. Fonte: Autor

No diagrama acima temos a representação da ação desde a interação do usuário com a interface web do sistema, troca de mensagens e validação das ações entre os contratos,

envio de log para a rede blockchain e retorno da informação ao usuário.

Primeiramente, o funcionário ao interagir com o sistema precisa conectar seu endereço com a utilização de uma Wallet, após essa conexão, o sistema verifica se o mesmo é reconhecido como um funcionário registrado e ativo, passando pelo método `getEmployeeByAddress` e validação pela função modificadora `onlyAdmin`. Após essa validação é retornada a informação de cadastro do funcionário através do Struct `Employee`. As figuras 80, 81 e 82 ilustram esse fluxo.

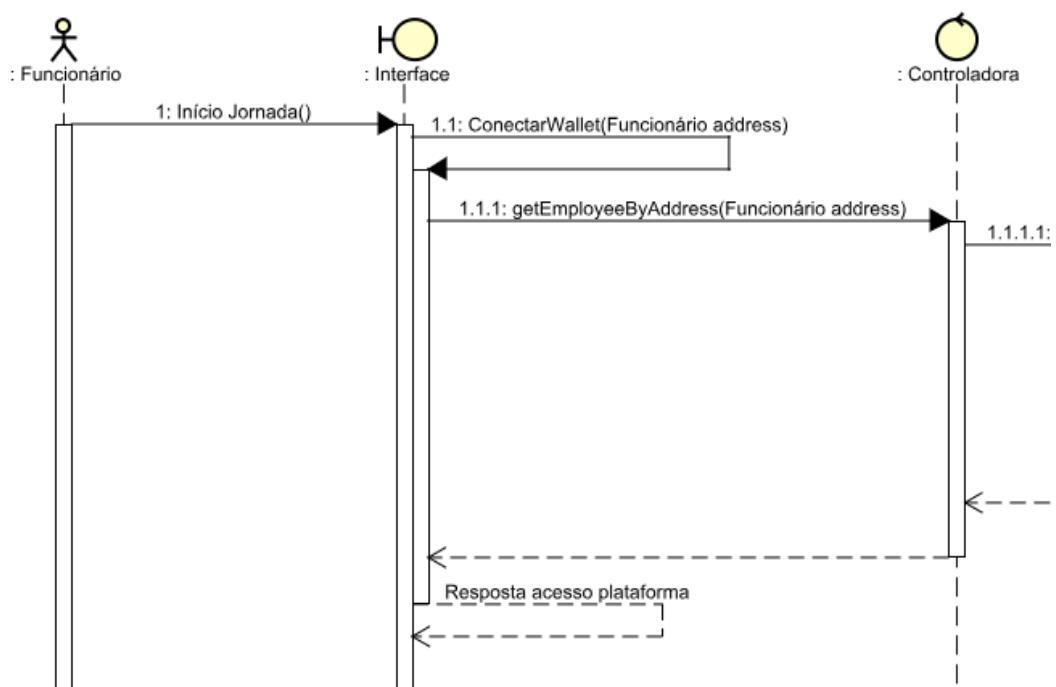


Figura 80 – SEQ – Verificação do funcionário. Fonte: Autor

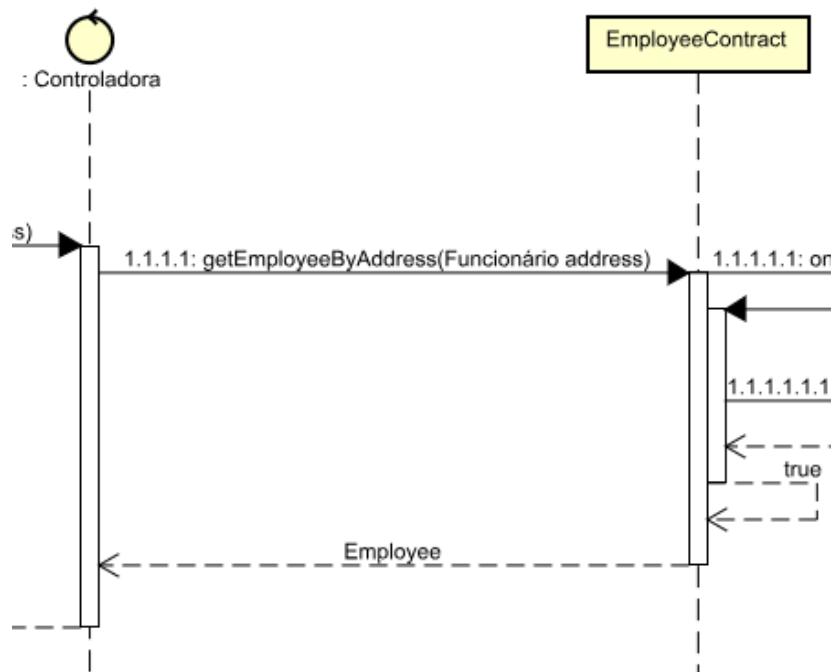


Figura 81 – SEQ – Verificação e retorno dos dados do funcionário. Fonte: Autor

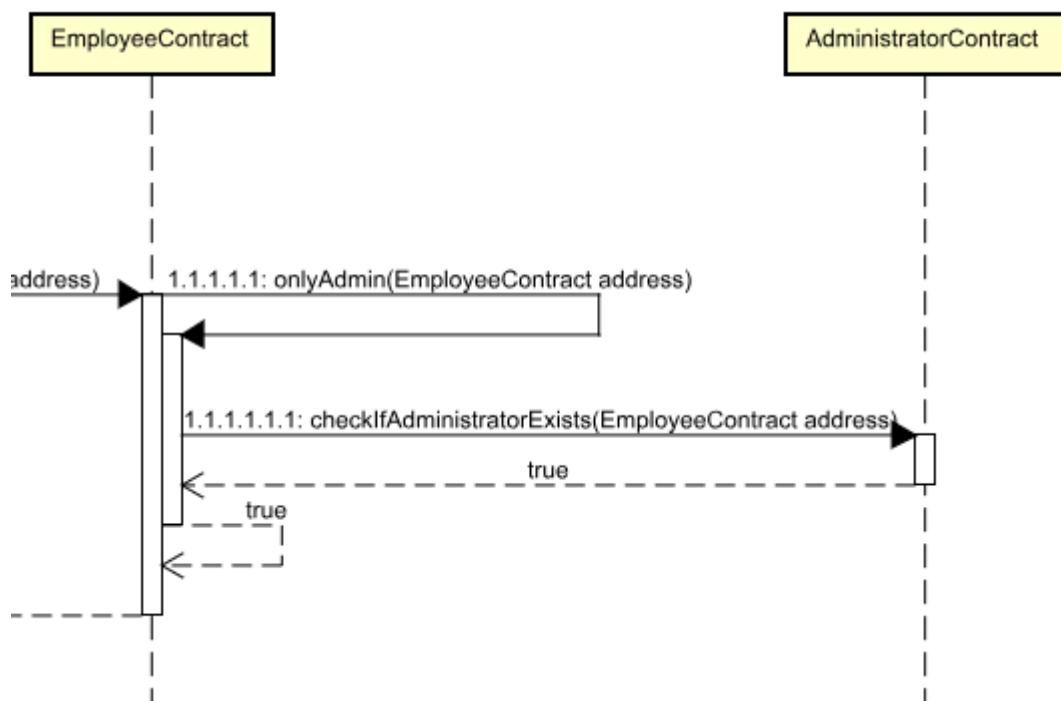


Figura 82 – SEQ – Validação dos dados do funcionário. Fonte: Autor

Ultrapassada a fase de identificação é realizada a ação de registro da jornada propriamente dita através da função *startWork*. Novamente são feitas validações do funcionário, se o mesmo existe no cadastro e está ativo para que seja dado seguimento ao processo reconhecendo a *epoch* desse registro fazendo consulta ao timestamp da rede e ajuste com o TimeZone do contrato. Essas validações sendo feitas com sucesso a gravação é registrada no Struct *employeeRecords* com o endereço do funcionário, a data e o timestamp da gravação com seu devido ajuste de timestamp e em seguida envio deste log para a rede com o resultado da gravação e retorno para o usuário. Todo o detalhamento deste fluxo está apresentado nas imagens 83 até 88.

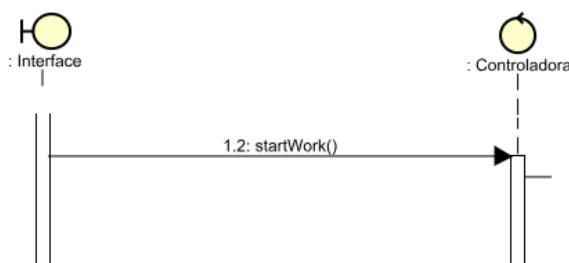


Figura 83 – SEQ – Chamada da função *startWork()*. Fonte: Autor

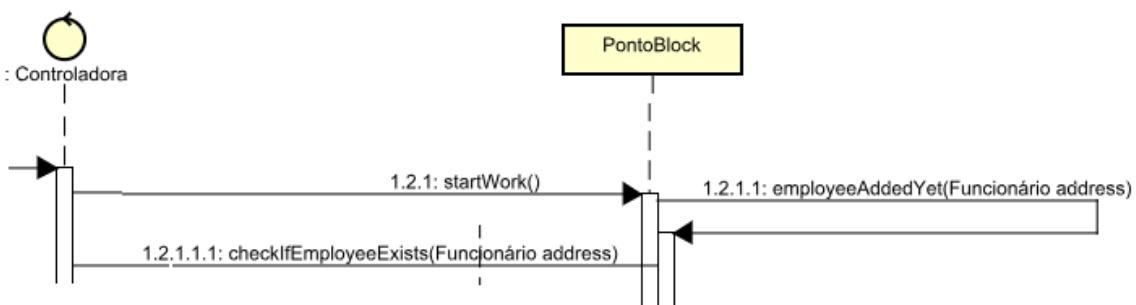


Figura 84 – SEQ – Verificação da existência do funcionário no sistema. Fonte: Autor

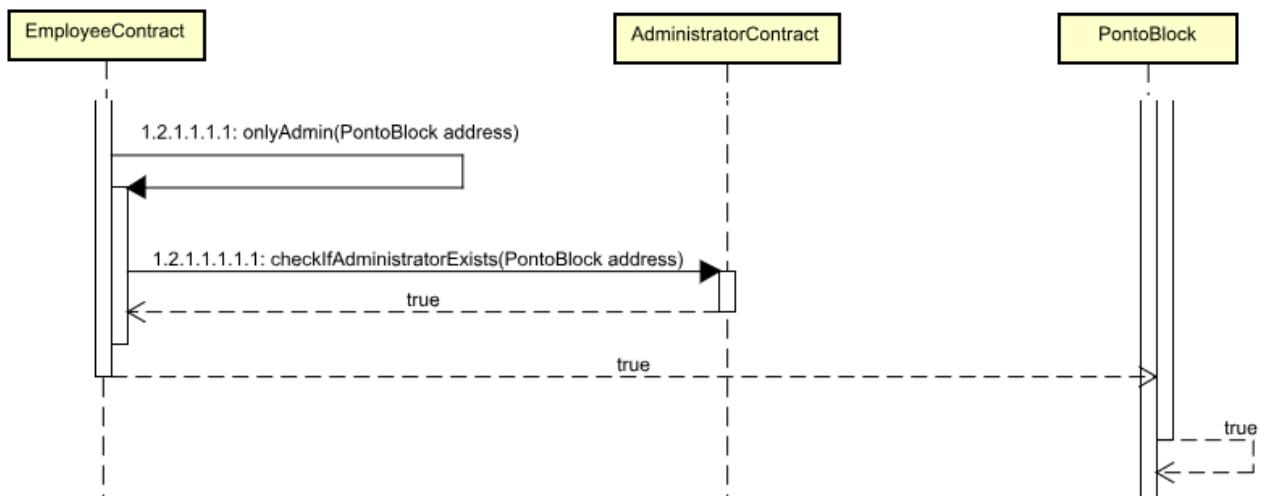


Figura 85 – SEQ – Verificação se o chamador (smart contract Ponto Block) está no roll de administradores.

Fonte: Autor

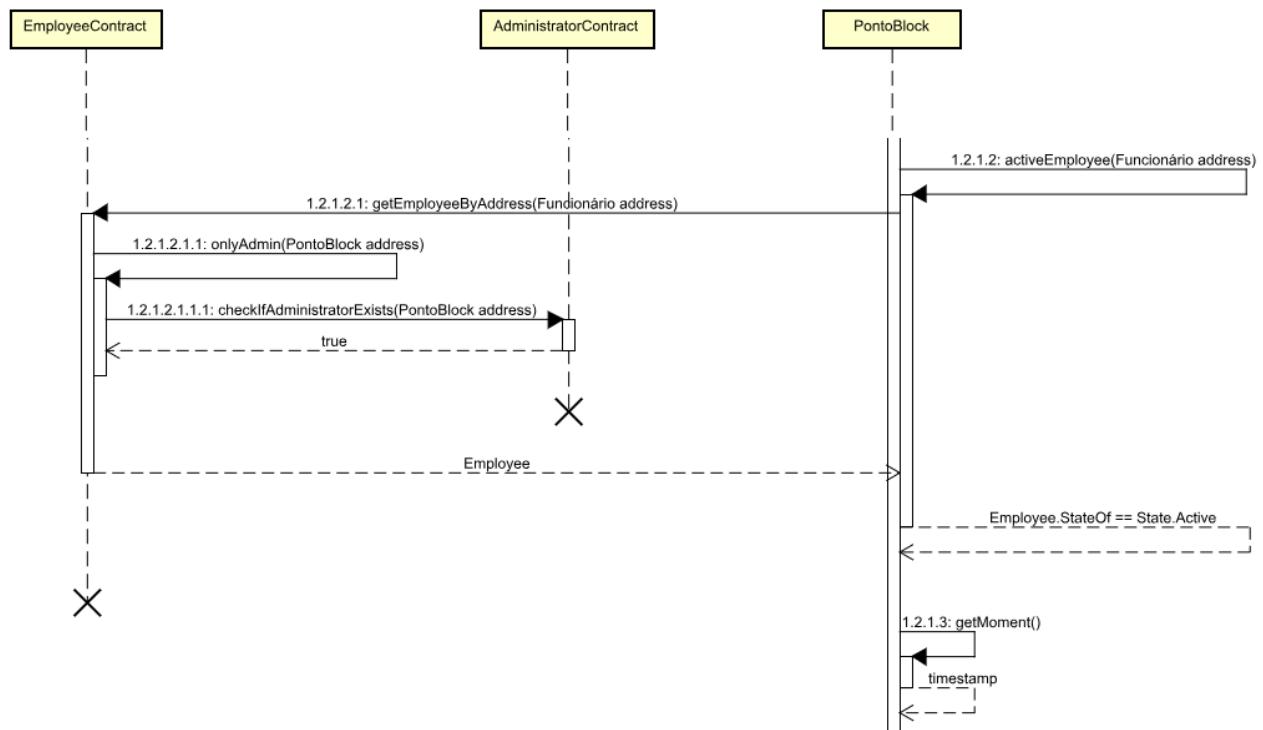


Figura 86 – SEQ – Envio dos dados do funcionário. Fonte: Autor

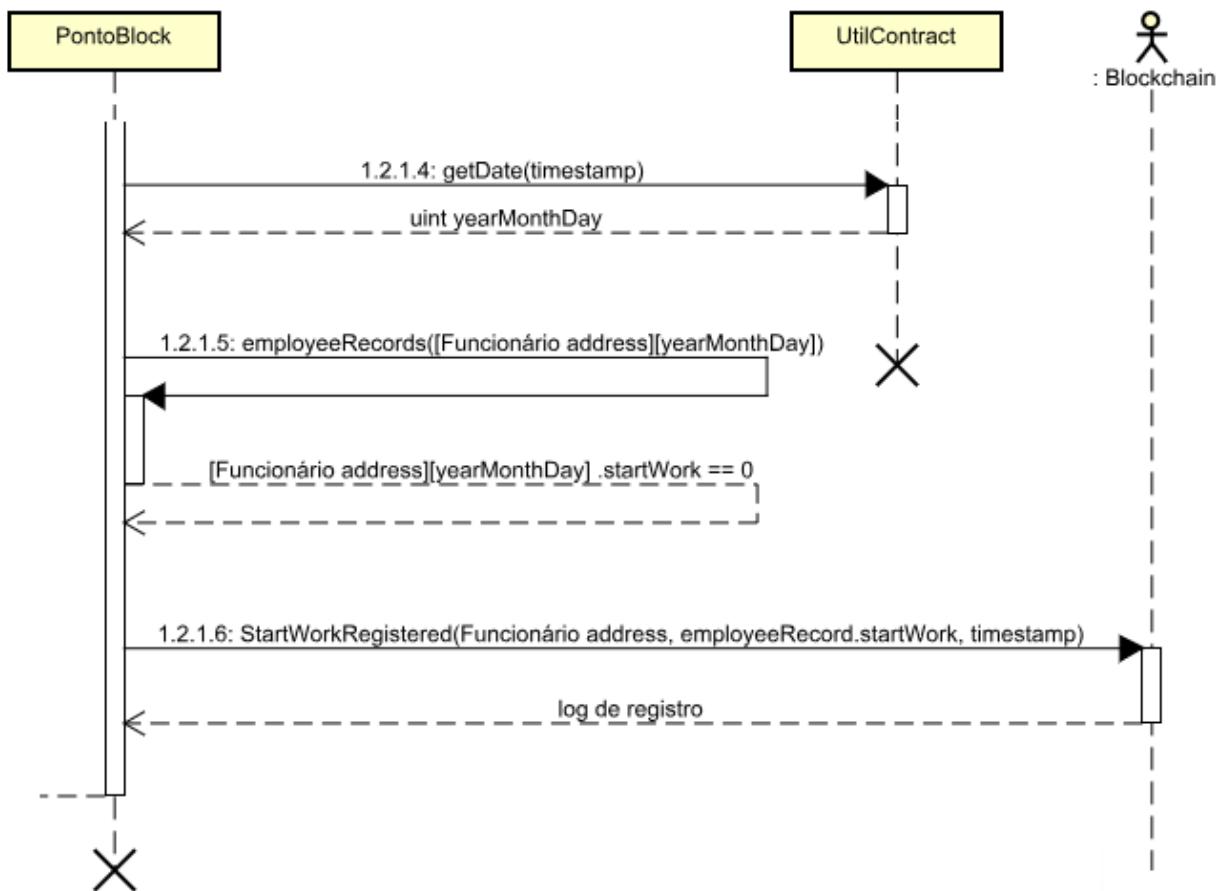


Figura 87 – SEQ – Tratamento do timestamp, execução da função startWork() e registro do evento através da função StartWorkRegistered(). Fonte: Autor

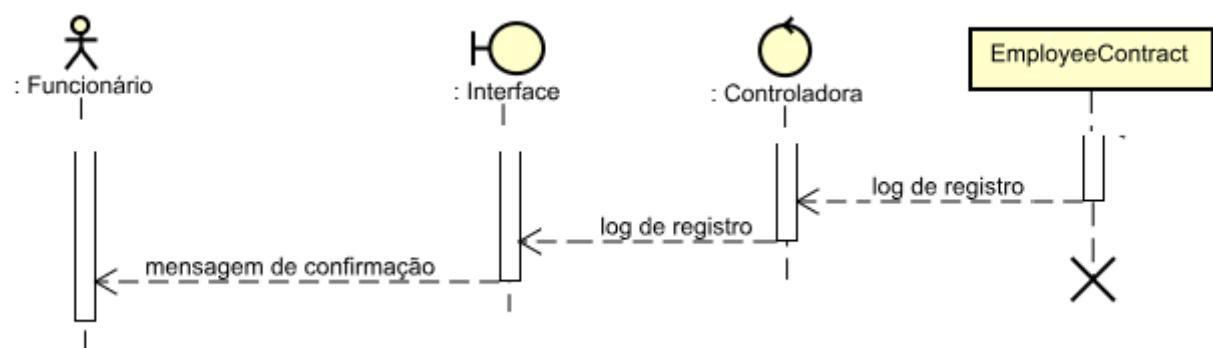


Figura 88 – SEQ – Retorno da mensagem de confirmação do registro de ponto. Fonte: Autor

15.0 PROTOTIPAÇÃO DE TELAS

Ultrapassada a fase de modelagem dos componentes do sistema, passamos agora para a fase de prototipação das telas, a parte visual, a interface onde os usuários irão interagir com o sistema.

Nossa primeira tela é a tela de cadastro. Aqui vemos um modelo inicial onde se referencia uma tela de cadastro de funcionário que se adapta aos outros cadastros: administrador e empregador.

The screenshot shows a software application window titled "WorkBlock". At the top, there is a navigation bar with five items: "Administrador", "Empregador", "Colaborador", and "Relatórios". Below the navigation bar, the main title "Lista de Colaboradores" is displayed. A table lists four employees with columns for Name, Carteira, PIS, Empregador, Estado, and Ações.

Nome	Carteira	PIS	Empregador	Estado	Ações
André Silva Teles	0xc32a...3eea1	23143243521	Companhia Sul de Navegação	Ativo	Alterar
Sandra Sousa	0xcae0...54a11	65214528456	Adrade Mattos Participações	Inativo	Alterar
André Silva Teles	0x231f...897dc	89551110025	Comandatuba Hotel e Resort	Ativo	Alterar

The screenshot shows a software application window titled "Cadastro de Colaborador". In the top right corner, there is a "Cancelar" (Cancel) button. The form contains the following fields:

- Nome: Antônio Silva Teles Ferreira Gomes
- Endereço: 0xcf1434054dcdea1681bf0806f46d669fd25a347f
- PIS: 202.57266.65-2
- Início Jornada: 07:30
- Fim Jornada: 16:30
- Endereço Empregador: 0xa1788a487e3c0e38a88b67711aa646a1cd733099
-

Figura 89 – Tela de cadastro de funcionário. Fonte: Autor

Nossa segunda tela é a tela de marcação de ponto, onde um funcionário registrado faz suas marcações de acordo com o horário desejado.

PontoBlock

conectar

Endereço da carteira:

Nome do Funcionário:

Nome do Empregador:

Jornada de Trabalho:

Marcação de Ponto

Entrada	Pausa	Retorno	Saída
Data: 23/04/2023 Hora: 07:23:15	Data: 23/04/2023 Hora: 12:17:10	Data: 23/04/2023 Hora: 13:15:40	Data: 23/04/2023 Hora: 16:30:28

Histórico

Data	Início	Pausa	Retorno	Saída
08/04/2023	07:33:18	13:01:00	13:58:50	16:25:38
09/04/2023	07:48:10	12:07:40	13:08:33	16:33:39
10/04/2023	07:20:00	13:21:00	14:10:55	16:30:11
11/04/2023	07:53:45	13:24:30	14:24:25	16:29:07

Figura 90 – Tela de marcação de ponto. Fonte: Autor

Nosso último modelo de referência, mas não menos importante, é a tela de relatório onde o administrador pode visualizar os relatórios de registro de ponto do sistema. No exemplo em tela temos um exemplo de tela onde é feita uma busca a partir de uma data e se obtém os registros de ponto gravados nesta referência.

WorkBlock

Carteira: 0x8ae4333778bc37743cf33ee32

Início: 02/02/2023 | Término: 09/02/2023

Buscar

Relatório

Data	Colaborador	Entrada	Pausa	Retorno	Saída
08/04/2023	0x8ae...3ee32	07:33:18	13:01:00	13:58:50	16:25:38
09/04/2023	0x8ae...3ee32	07:48:10	12:07:40	13:08:33	16:33:39
10/04/2023	0x8ae...3ee32	07:20:00	13:21:00	14:10:55	16:30:11
11/04/2023	0x8ae...3ee32	07:53:45	13:24:30	14:24:25	16:29:07

Figura 91 – Tela de relatório. Fonte: Autor

16.0 CONSIDERAÇÕES NO DESENVOLVIMENTO COM SOLIDITY

O core de nossa solução está representada nos smart contracts, que são os elementos responsáveis pela gestão dos recursos da aplicação, fazendo com que todos os dados sejam registrados na blockchain e posteriormente lidos e modificados.

Algumas considerações iniciais devem ser apresentadas quando lidamos com uma solução em blockchain utilizando a linguagem **Solidity**. A primeira delas é que as operações de leitura e gravação de dados da blockchain tem um custo e a otimização do código influencia diretamente no custo destas operações. Se tomarmos com exemplo um determinado valor para representar uma data com dia mês e ano, poderíamos ter três propostas para tal:

1. **Representar a data propriamente dita:** no caso da linguagem **Solidity**, como não possui os tipos **Date**, **Time** e **DateTime**, representá-la com o Unix Timestamp utilizando o tipo básico da linguagem **uint**.
2. **Representar a data como string:** neste caso, a linguagem possui o tipo **string**, então poderíamos representá-la com o formato **2023/03/23**.
3. **Representar a data como uint:** aqui, seria um misto das duas anteriores, representando a data em formato como **string** mas utilizando **uint**, ficando **20230323**.

Todas as três propostas acima conseguiriam representar a nossa data de exemplo, mas como mencionado, a **EVM** tem o fator de custo das operações na blockchain, e para efeito de custo a opção utilizando **uint** é a menos custosa, fazendo então que descartemos a opção 2. Com as opções 1 e 3 sobre a mesa resta a dúvida sobre qual utilizar, e em nossa proposta optamos pela opção 3 pois gravando a data nesse formato conseguimos utilizar uma um tipo básico da linguagem menos custoso para a **EVM** e já temos uma espécie de formatação, algo que otimiza algumas de nossas operações, fazendo também mais uma

opção pela operação menos custosa pois se utilizássemos o timestamp, em alguns cenários teríamos que fazer algumas conversões para comparar datas.

Na mesma visão de otimização de performance dos smart contracts, há uma pequena diferenciação com relação a uma classe tradicional do modelo de orientação a objetos. Enquanto as classes do modelo OO é comum serem divididas em atributos e métodos, nos smart contracts também podemos fazer esse mesmo tipo de divisão, mas para sua operação, se agruparmos as propriedades em um **Struct**, sua operação fica menos custosa para a EVM, daí, termos um **Struct** para representar um funcionário e não seus atributos discretos como nome, endereço e documento dentro do smart contract.

17.0 ARQUITETURA DO SISTEMA

Após ultrapassadas as etapas de conhecimento dos aspectos legais, soluções atuais de mercado, perspectivas para o desenvolvimento de uma solução com uma arquitetura inovadora como o blockchain e modelagem de regras de negócio, requisitos, processos e modelagem, começaremos a criar as definições para desenvolvimento do software para a solução de sistema de controle de ponto proposta.

Na elaboração da arquitetura do sistema, nossa proposta apresenta quatro projetos bem distintos, à saber:

1. **Desenvolvimento de Contratos:** Camada onde estão os smart contracts e todos os recursos para testes unitários e publicação na blockchain.
2. **API de Gestão:** Camada onde está a API que faz interação com aspectos administrativos de nossos smart contracts.
3. **Projeto Web de Gestão:** Camada onde está o projeto web que fará interação com a API para interagir com aspectos administrativos de nosso sistema.

4. Projeto Web de Marcação de Ponto: Este projeto é o responsável por fazer as interações diretamente com a blockchain e com a API para buscar os dados de cadastro do funcionário.

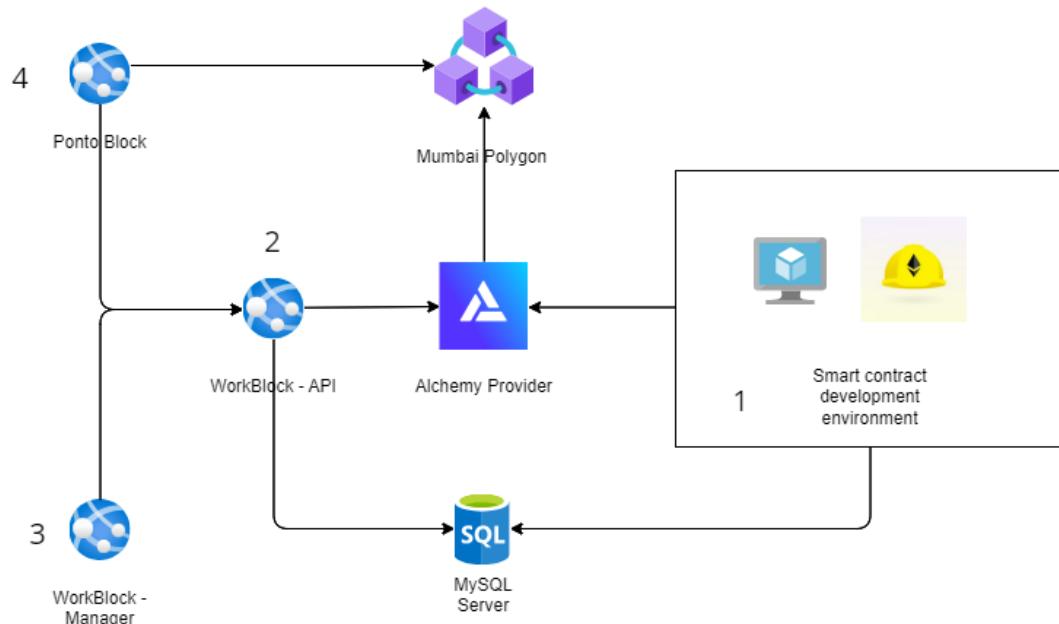


Figura 92 – Arquitetura do sistema. Fonte: Autor

Abaixo, temos a tabela com o endereço onde os projetos estão publicados.

PROJETO	URL	AMBIENTE
WorkBlock - API	https://workblock-api.azurewebsites.net	Azure – App Service
WorkBlock - Manager	https://workblock-manager.azurewebsites.net	Azure – App Service
PontoBlock	https://pontoblock.gilmarsantana.com	Hostgator

17.1 DESENVOLVIMENTO DE CONTRATOS

Na camada de desenvolvimento de contratos é utilizada a ferramenta **HardHat** para fazer compilação, testes, publicação e verificação dos contratos na rede.

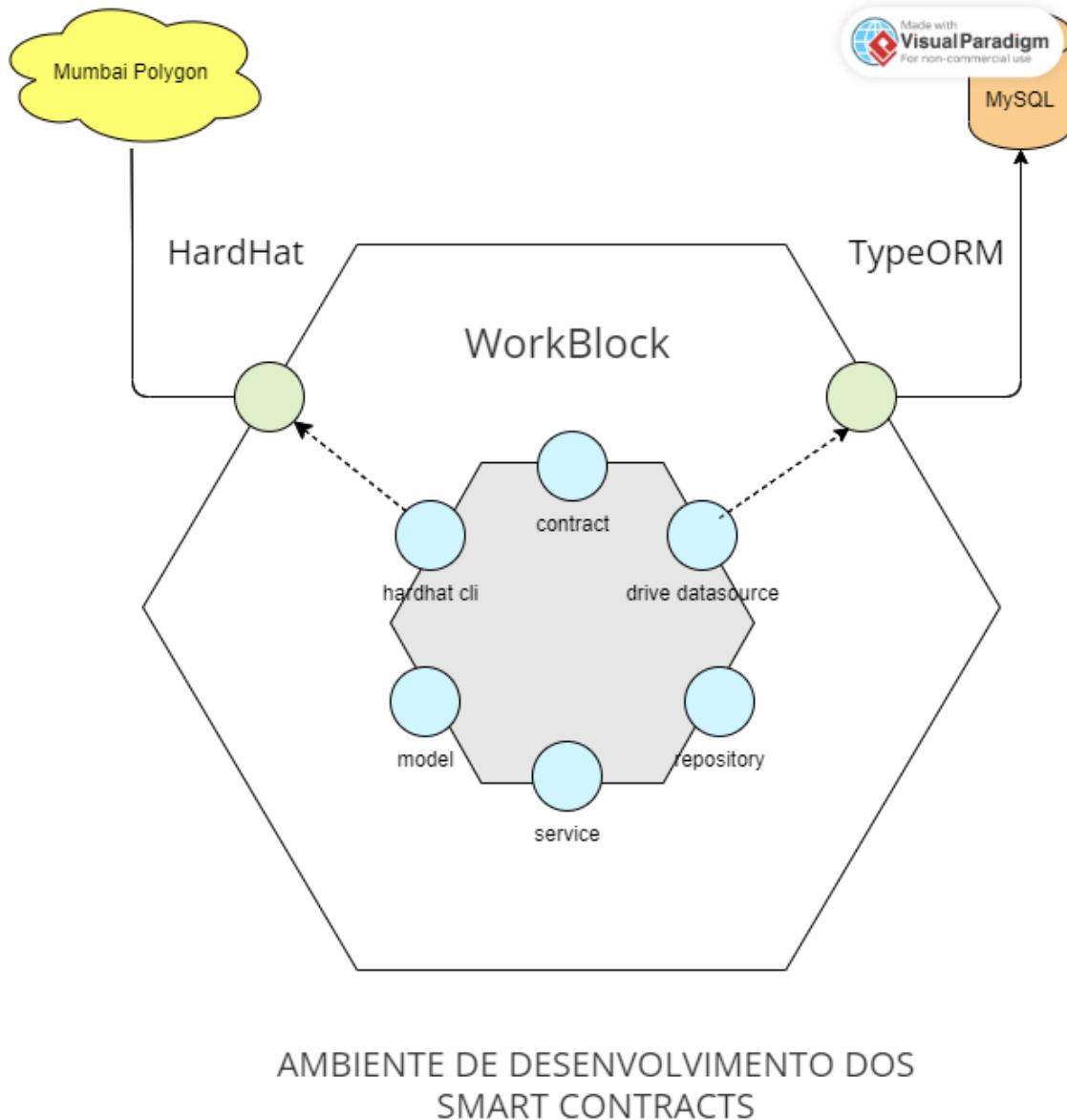


Figura 93 – Ambiente de desenvolvimento dos smart contracts. Fonte: Autor

Utilizando a ferramenta já citada, HardHat, e seguindo o ciclo de desenvolvimento de código publicação e verificação dos contratos, foram armazenados os dados de nome, endereço, abi, bytecode e data de criação em banco de dados MySQL utilizando a ferramenta

de mapeamento objeto relacional (ORM) **TypeORM**. Após este processo, temos os endereços abaixo referente a cada um dos contratos. Para a publicação de nossa solução utilizamos a plataforma Mumbai da Polygon.

SMART CONTRACT	ENDEREÇO
UtilContract	0x5C0e0C4100D838A70E0AF4378028bA66AB6aFBEd
AdministratorContract	0x1c7BC0748beE03B027da75B057040C8170C73d6F
EmployerContract	0xF43298d54cC69373F9349cbE025524e5F5f9D72f
EmployeeContract	0x599Cf0c748AD9F5340181A1F139a33EA1D7706f
PontoBlock	0x6BdD0030280da0170E9fe8544b18D2134925F3EE
PontoBlockReports	0x5A45be5c3D5BE4E286D8999E866C370Da6ec19e2

Uma vez publicados os smart contracts na blockchain, todos os seus dados e eventos podem ser monitorados por qualquer pessoa, bastando ter o endereço do contrato em questão. Nas imagens 96, 97 e 98, abaixo, temos o registro do contrato PontoBlock que pode ser visualizado através do endereço <https://mumbai.polygonscan.com/address/0x6bdd0030280da0170e9fe8544b18d2134925f3ee>.

Txn Hash	Method	Block	Age	From	To	Value	[Txn Fee]
0x869af0aa71ba2079d4...	End Work	37939423	1 hr 43 mins ago	0xe98806d8998cae1a45...	IN 0x6bdd0030280da0170e...	0 MATIC	0.000369592503
0x7bd55ce7111ef508eb...	Break End Time	37933403	5 hrs 29 mins ago	0xe98806d8998cae1a45...	IN 0x6bdd0030280da0170e...	0 MATIC	0.000310434003
0x425fe183bdd7e747dd...	Break Start Time	37931586	6 hrs 38 mins ago	0xe98806d8998cae1a45...	IN 0x6bdd0030280da0170e...	0 MATIC	0.000339999003

Figura 94 – Página principal do contrato no explorador de blocos na rede Mumbai Polygon. Fonte: Autor

Transactions	ERC-20 Token Txns	Contract	Events
			File 4 of 5 : PontoBlock.sol
Code	Read Contract	Write Contract	
 Contract Source Code Verified (Exact Match)			
Contract Name:	PontoBlock		
Compiler Version	v0.8.17+commit.8df45f5f		

Figura 95 – Código do contrato no explorador de blocos na rede Mumbai Polygon. Fonte: Autor

Figura 96 – Eventos do contrato no explorador de blocos na rede Mumbai Polygon. Fonte: Autor

Defendendo as boas práticas de desenvolvimento utilizamos a ferramenta **solidity-coverage**, que através de testes unitários são validados os nossos Smart Contracts. Abaixo, temos a imagem do resultado dos testes gerado pela ferramenta. Esta ferramenta apresenta, de forma muito útil, cria uma pasta para armazenar os resultados e cria páginas HTML para exibir toda a cobertura realizada, inclusive no código de cada arquivo coberto, de forma individualizada.

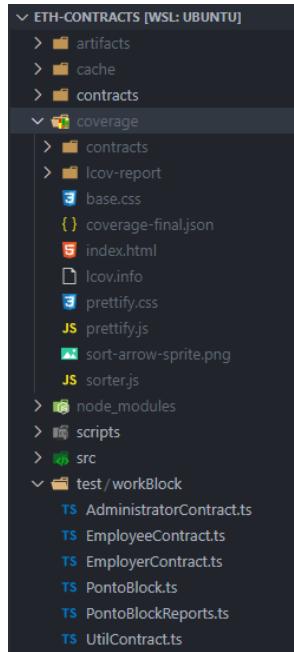


Figura 97 – Pasta criada pela ferramenta Coverage. Fonte: Autor

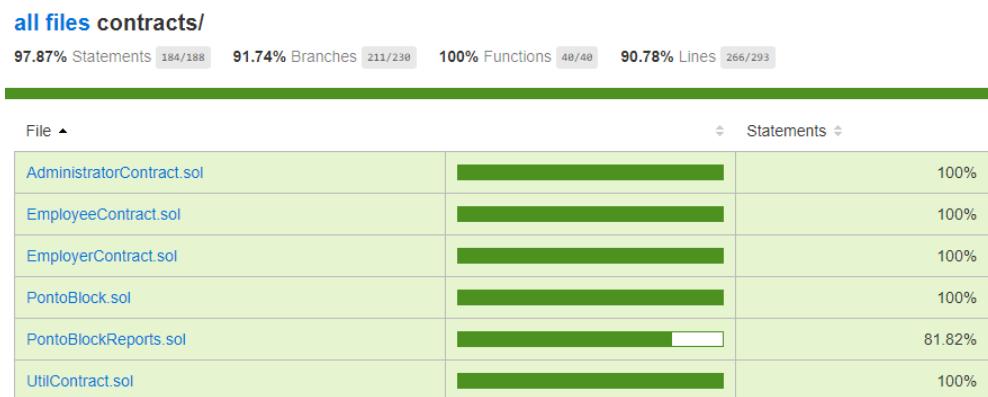


Figura 98 – Cobertura de testes dos smart contracts. Fonte: Autor

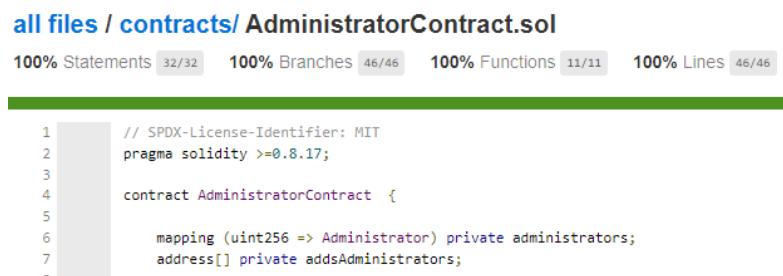


Figura 99 – Cobertura de testes em AdministratorContract. Fonte: Autor

17.2 API DE GESTÃO

Neste projeto fazemos interação com a blockchain para realizar as ações administrativas de manter administrador, empregador e funcionário e relatórios.

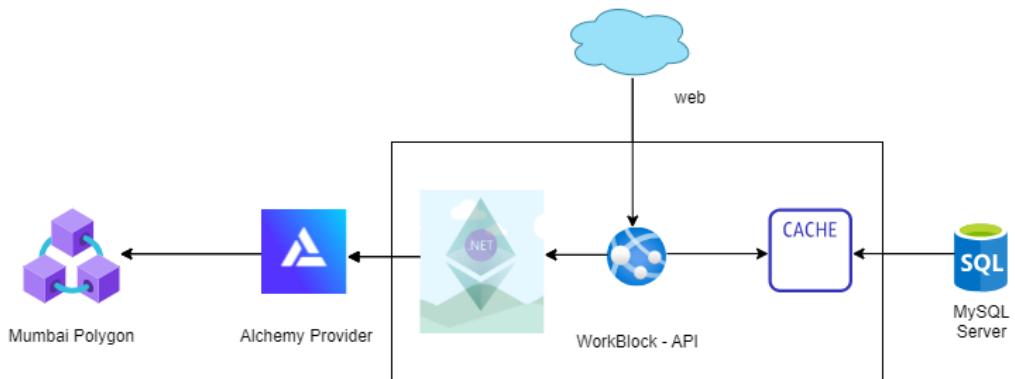


Figura 100 – Desenho de arquitetura da API. Fonte: Autor

Como tecnologia, utilizamos como base .NET e a biblioteca Nethereum, que possui recursos para criação, testes, deploy e interação com smart contracts no padrão Ethereum.

Para otimização das chamadas da aplicação utilizamos o recurso de cache do próprio .NET para armazenar os dados dos contratos. Além disso, o banco de dados MySQL armazena os dados dessas ações de gerenciamento onde são modificados dados de participantes. A imagem abaixo mostra a estrutura das tabelas utilizadas para registro dos eventos em nosso banco de dados.

As tabelas seguem as seguintes referências:

- __EFMigrationsHistory e migrations guardam a estrutura de relação do ORM, TypeORM, no caso do ambiente de desenvolvimento dos smart contracts, Entity Framework, para o projeto de API.
- Contracts armazena os dados de nossos smart contracts.
- AdminAddedEvents e AdminUpdatedEvents registram os eventos de adição e

atualização de dados dos administradores do sistema.

- EmployerAddedEvents e EmployerUpdatedEvents registram os eventos de adição e atualização de dados dos empregadores.
- EmployeeAddedEvents e EmployeeUpdatedEvents os eventos de adição e atualização dos funcionários.

The diagram illustrates the structure of tables in the WorkBlock database, categorized into four main sections:

- Admin:**
 - AdminAddedEvents**: Contains columns for Id, AddressFrom, AdministratorAddress, AdministratorName, AdministratorTaxId, Time, and HashTransaction.
 - AdminUpdatedEvents**: Contains columns for Id, AddressFrom, OldAddress, NewAddress, AdministratorName, AdministratorTaxId, State, Time, and HashTransaction.
- Employer:**
 - EmployerAddedEvents**: Contains columns for Id, AddressFrom, EmployerAddress, EmployerName, EmployerTaxId, EmployerLegalAddress, Time, and HashTransaction.
 - EmployerUpdatedEvents**: Contains columns for Id, AddressFrom, OldAddress, NewAddress, EmployerName, EmployerTaxId, EmployerLegalAddress, Time, and HashTransaction.
- Employee:**
 - EmployeeAddedEvents**: Contains columns for Id, AddressFrom, EmployeeAddress, EmployeeName, EmployeeTaxId, EmployeeBeginningWorkDay, EmployeeEndWorkDay, EmployerAddress, Time, and HashTransaction.
 - EmployeeUpdatedEvents**: Contains columns for Id, AddressFrom, OldAddress, NewAddress, EmployeeName, EmployeeTaxId, EmployeeBeginningWorkDay, EmployeeEndWorkDay, EmployerAddress, State, Time, and HashTransaction.
- Contracts:**
 - Contracts**: Contains columns for id, name, addressContract, abi, bytecode, and createdAt.

Figura 101 – Tabelas no banco de dados WorkBlock. Fonte: Autor

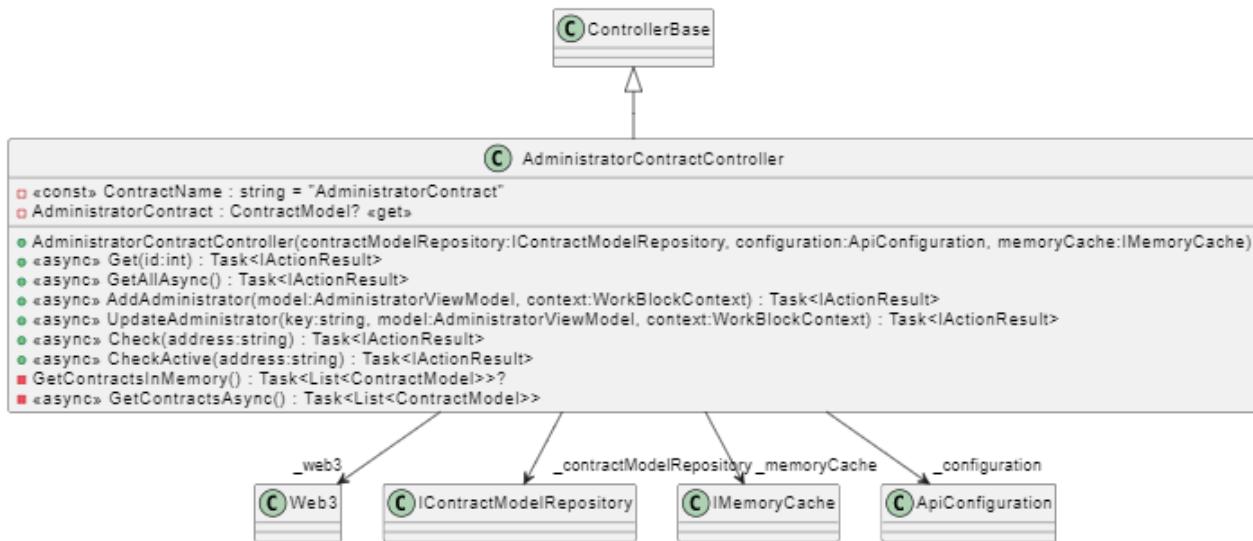


Figura 102 – Diagrama de classe de `AdministratorContractController`. Fonte: Autor

WorkBlockApi 1.0 OAS3

<https://workblock-api.azurewebsites.net/swagger/v1/swagger.json>

AdministratorContract ^

GET	/v1/contracts/administratorcontract/Get/{id}	▼
GET	/v1/contracts/administratorcontract/GetAll	▼
POST	/v1/contracts/administratorcontract/Add	▼
PUT	/v1/contracts/administratorcontract/Update/{key}	▼
GET	/v1/contracts/administratorcontract/Check/{address}	▼
GET	/v1/contracts/administratorcontract/CheckActive /{address}	▼

EmployeeContract ^

GET	/v1/contracts/employeecontract/Get/{id}	▼
GET	/v1/contracts/employeecontract/Get/{address}	▼
GET	/v1/contracts/employeecontract/GetAll	▼
POST	/v1/contracts/employeecontract/Add	▼
PUT	/v1/contracts/employeecontract/Update/{key}	▼
GET	/v1/contracts/employeecontract/Check/{address}	▼
GET	/v1/contracts/employeecontract/GetEmployerContract	▼

Figura 103 – Endpoints de AdministratorContract e EmployeeContract. Fonte: Autor

EmployerContract	
GET	/v1/contracts/employercontract/Get/{id}
GET	/v1/contracts/employercontract/Get/{address}
GET	/v1/contracts/employercontract/GetAll
POST	/v1/contracts/employercontract/Add
PUT	/v1/contracts/employercontract/Update/{key}
GET	/v1/contracts/employercontract/Check/{address}
PontoBlock	
GET	/v1/contracts/pontoblock/GetCreationDate
GET	/v1/contracts/pontoblock/GetMoment
GET	/v1/contracts/pontoblock/GetOwner
POST	/v1/contracts/pontoblock/ChangeOwner
GET	/v1/contracts/pontoblock/GetTimeZone
GET	/v1/contracts/pontoblock/GetEmployeeRecords

Figura 104 – Endpoints de EmployerContract e PontoBlock. Fonte: Autor

PontoBlockReports

GET	/v1/contracts/pontoblockreports /GetWorkTimesFromEmployeeAtDate	▼
GET	/v1/contracts/pontoblockreports /GetWorkTimeFromEmployeeBetweenTwoDates	▼
GET	/v1/contracts/pontoblockreports /GetWorkTimesForAllEmployeesAtDate	▼
GET	/v1/contracts/pontoblockreports /getWorkTimesForAllEmployeesBetweenTwoDates	▼

UtilContract

GET	/v1/contracts/utilcontract/getdate	▼
GET	/v1/contracts/utilcontract/validatehour	▼

Schemas

Address >
AdministratorViewModel >
EmployeeViewModel >
EmployerViewModel >

Figura 105 – Endpoints de PontoBlockReports e UtilContract. Fonte: Autor

The screenshot shows a Swagger UI interface for a RESTful API. At the top, there's a search bar and a dropdown menu. Below that, the main content area has sections for 'Path' and 'Operations'.

Path:

- Path: /v1/contracts/administratorcontract/{id}
- HTTP Method: GET
- Description: Get an administrator contract by id
- Responses:
 - 200: OK
 - 400: Bad Request
 - 404: Not Found
 - 500: Internal Server Error

Operations:

AdministratorContractController.cs

```
public class AdministratorContractController : ControllerBase
{
    private readonly IRepository<AdministratorContract> _repository;
    private readonly IMapper _mapper;

    public AdministratorContractController(IRepository<AdministratorContract> repository, IMapper mapper)
    {
        _repository = repository;
        _mapper = mapper;
    }

    [HttpGet]
    [Route("{id}")]
    public IActionResult Get(int id)
    {
        var contract = _repository.GetById(id);
        if (contract == null)
            return NotFound();
        var administratorContract = _mapper.Map<AdministratorContractModel>(contract);
        return Ok(administratorContract);
    }
}
```

Request Body:

Name	Description
id * required	integer(\$int32) (path)

Execute button and **Clear** button.

Responses

Curl command:

```
curl -X 'GET' \
  'https://workblock-api.azurewebsites.net/v1/contracts/administratorcontract/Get/0' \
  -H 'accept: */*'
```

Request URL:

<https://workblock-api.azurewebsites.net/v1/contracts/administratorcontract/Get/0>

Server response:

Code Details

Code	Details
200	Response body

Response body content:

```
{
  "data": {
    "address": "0xE98806d8998CAE1A45338C5F138438C708273c05",
    "taxId": "98667688053",
    "name": "GILMAR RIBEIRO SANTANA",
    "state": 1
  },
  "errors": []
}
```

Download button.

Response headers:

```
content-type: application/json; charset=utf-8
date: Tue, 11 Jul 2023 00:37:25 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses:

Code	Description	Links
200	Success	No links

Figura 106 – Exemplo de chamada de um endpoint exibido pelo Swagger. Fonte: Autor

Para facilitar a interação entre o cliente e nossa API adotamos a estratégia de utilizar uma classe especial para entregar os dados de forma padronizada. O objetivo dessa classe, **ResultViewModel**, é fornecer um modelo de dados para representar os resultados ou respostas de uma operação, onde, **Data** armazena o resultado principal e **Errors** mantém uma lista de erros associados. A classe é projetada para ser flexível, permitindo

diferentes maneiras de inicializar suas propriedades, dependendo dos dados disponíveis ou das necessidades do código que a utiliza.

```
namespace WorkBlockApi.ViewModels;

public class ResultViewModel<T>
{
    public T? Data { get; private set; }
    public List<string> Errors { get; private set; } = new();

    public ResultViewModel(T data, List<string> errors)
    {
        Data = data;
        Errors = errors;
    }

    public ResultViewModel(T data)
    {
        Data = data;
    }

    public ResultViewModel(List<string> errors) => Errors = errors;

    public ResultViewModel(string error) => Errors.Add(error);
}
```

Abaixo, temos, de forma ilustrativa, a chamada do método referenciado na figura 105, que busca um administrador pelo seu **id**.

```
[HttpGet("Get/{id:int}")]
public async Task<IActionResult> Get([FromRoute] int id)
{
    try
    {
        if (AdministratorContract is null)
            return NotFound(new ResultViewModel<string>("Contract Not Found"));

        if (id < 0)
            return NotFound(new ResultViewModel<string>("Invalid Format. Id must be equal to or greater than zero"));

        var service = new AdministratorContractService(_web3,
AdministratorContract.AddressContract);
        var admin = await service.GetAdministratorQueryAsync(id);
        var returnAdmin = new AdministratorModel
        {
            IdAdministrator = (uint)admin.ReturnValue1.IdAdministrator,
            Address = admin.ReturnValue1.AdministratorAddress,
            Name = admin.ReturnValue1.Name,
        };
    }
}
```

```
        TaxId = admin.ReturnValue1.TaxId.ToString(),
        State = admin.ReturnValue1.StateOf
    };
    return StatusCode(200, new ResultViewModel<AdministratorModel>(returnAdmin));
}
catch (SmartContractRevertException e)
{
    return StatusCode(500, new ResultViewModel<AdministratorModel>(e.RevertMessage));
}
catch (Exception e)
{
    return StatusCode(500, new ResultViewModel<AdministratorModel>(e.Message));
}
}
```

17.3 PROJETO WEB DE GESTÃO

Este projeto é utilizado como ferramenta web para interação das ações administrativas com a nossa solução. Nela, fazemos as ações de manter administrador, empregador, funcionário e exibição de relatórios. A grande vantagem de utilizarmos este projeto de forma separada é que conseguimos manter as responsabilidades em projetos distintos e cada projeto pode escalar ou mudar de tecnologia sem afetar outra camada de nossa solução. Por exemplo, este projeto foi desenvolvido utilizando a tecnologia Razor Pages do .NET, mas nada impediria que fosse feito um outro projeto para o mesmo fim com Angular, React, Kotlin, React Native, etc.

Estruturalmente, continuamos utilizando a solução .NET para desenvolvimento do projeto. Adotamos a estruturação em camadas, utilizando a segregação por interfaces onde a nossa View, através de injeção de dependência, faz referência à sua Service, e esta, também utilizando injeção de dependência, faz referência à sua classe Repository, que faz a chamada para a API. As figuras 107, 108 e 109 mostram essa estrutura.

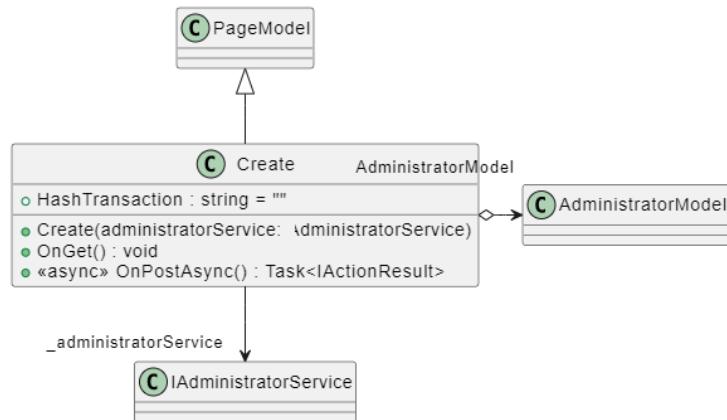


Figura 107 – Diagrama de classe da página Create do Administrador. Fonte: Autor

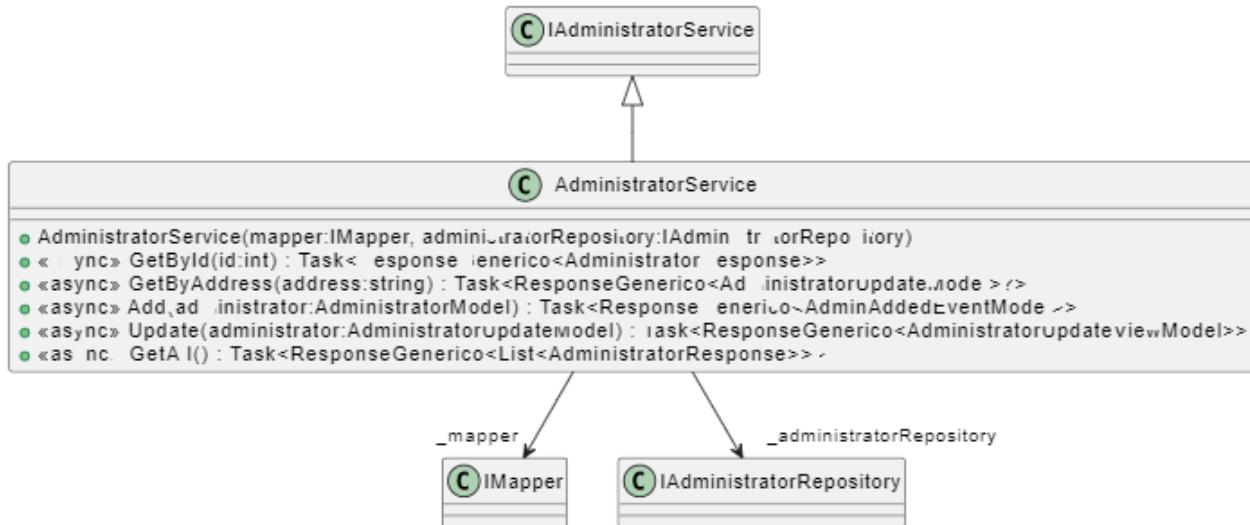


Figura 108 – Diagrama de classe de AdministratorService. Fonte: Autor

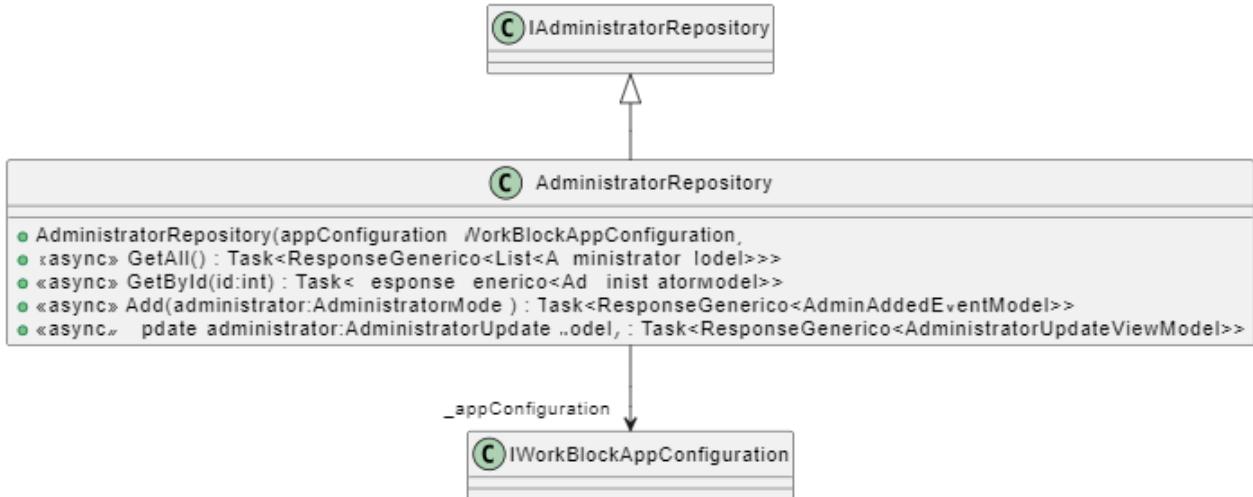


Figura 109 – Diagrama de classe de AdministratorRepository. Fonte: Autor

Quanto à parte visual, utilizamos o framework CSS Bootstrap para estilização dos componentes, algo que facilita muito na responsividade de nosso sistema e padroniza a visualização. As imagens 110 a 113 mostram o caminho desde a listagem dos colaboradores cadastrados no sistema até o relatório de marcação de ponto.

The screenshot shows a web application interface titled "Lista de Colaboradores". At the top, there is a navigation bar with links: "WorkBlockApp", "WorkBlock", "Administrador", "Empregador", "Colaborador", and "Relatórios". On the right side of the header is a blue button labeled "Cadastrar". The main content area displays a table with the following data:

Nome	Carteira	PIS	Empregador	Ações
GILMAR RIBEIRO SANTANA	0xE98...73c05	00.069.911/2196-22	ECO-EMPRESA DE CONSULTORIA E ORGANIZ SISTEMAS EDIT LTDA	Ativo Alterar
PAULO MASSILLON	0xb73...058e7	00.002.426/6254-51	FUNDACAO DE APOIO A ESCOLA TECNICA DO ESTADO DO RIO DE JANEIRO	Ativo Alterar
AYRTON SENNA	0x1C...4714c	00.029.799/4289-39	ECO-EMPRESA DE CONSULTORIA E ORGANIZ SISTEMAS EDIT LTDA	Ativo Alterar
EMPLOYEE	0x2f3...3C2Cc	00.017.019/3645-09	ECO-EMPRESA DE CONSULTORIA E ORGANIZ SISTEMAS EDIT LTDA	Ativo Alterar
NOVO EMPREGADO	0x7c3...31161	00.055.732/8972-35	ECO-EMPRESA DE CONSULTORIA E ORGANIZ SISTEMAS EDIT LTDA	Ativo Alterar

Figura 110 – Listagem de colaboradores. Fonte: Autor

Nome Completo
Nome do colaborador

Endereço de Carteira
0x71C7...9553

PIS
000.00000.00-0

Início de Jornada
08:00 AM

Fim de Jornada
17:00 PM

Carteira Empregador
0x71C7...9553

Cadastrar

Figura 111 – Página de cadastro de Colaborador. Fonte: Autor

Nome Completo
EMPLOYEE

Endereço de Carteira
0x2f3dA6AC9509E67cd259F50E9a5fa7C82cA3C2Cc

PIS
170.19364.50-9

Novo Endereço de Carteira
0x2f3dA6AC9509E67cd259F50E9a5fa7C82cA3C2Cc

Início de Jornada
09:00 AM

Fim de Jornada
06:00 PM

Estado
 Ativo

Carteira Empregador
0x7C4254F3367B92ff03CF1CCebC90740C88032f39

Atualizar

Figura 112 – Página de atualização de Colaborador. Fonte: Autor

WorkBlockApp WorkBlock Administrador Empregador Colaborador Relatórios

Registros de Ponto

Endereço de Carteira
0xE9806d8998CAE1A45338C5F138438C70B273c05

Período de pesquisa Período
15-June-2023 / 14-July-2023

Buscar

Data	Colaborador	Entrada	Início Pausa	Fim Pausa	Saída
08/07/2023	0xE98...73c05	--:--:--	--:--:--	--:--:--	--:--:--
09/07/2023	0xE98...73c05	--:--:--	--:--:--	--:--:--	--:--:--
10/07/2023	0xE98...73c05	07:02:23	13:15:42	14:09:56	--:--:--
11/07/2023	0xE98...73c05	07:58:10	12:44:44	13:52:50	--:--:--
12/07/2023	0xE98...73c05	08:12:27	14:10:54	14:11:28	16:41:19
13/07/2023	0xE98...73c05	07:48:54	--:--:--	--:--:--	--:--:--
14/07/2023	0xE98...73c05	08:03:43	13:55:10	15:04:46	18:49:53

Figura 113 – Página de Exibição de relatórios de marcação de ponto. Fonte: Autor

Para garantia de envio das informações de forma correta, criamos validações com responsabilidades no frontend e backend. Para exibir esse processo de validação, vamos utilizar a página de atualização de dados do colaborador para ilustrar esse comportamento.

Ao selecionar um colaborador, ele é carregado com seus dados na página de atualização.

Nome Completo
NOVO EMPREGADO

Endereço de Carteira
0x7c326Dbf40DbC6dB80F440A54d209EA396631161

PIS
557.32897.23-5

Novo Endereço de Carteira
0x7c326Dbf40DbC6dB80F440A54d209EA396631161

Início de Jornada
08:00 AM

Fim de Jornada
05:00 PM

Estado
 Ativo

Carteira Empregador
0x7C4254F3367B92ff03CF1CCebC90740C88032f39

Atualizar

Figura 114 – Dados do colaborador. Fonte: Autor

Se tentarmos atualizar os dados desse colaborador sem informar o endereço de carteira do empregador, por exemplo, a validação do frontend irá nos alertar que essa informação é obrigatória.

Nome Completo
NOVO EMPREGADO ✓
nome informado corretamente

Endereço de Carteira
0x7c326Dbf40DbC6dB80F440A54d209EA396631161 ✓
PIS
557.32897.23-5 ✓
PIS informado corretamente

Novo Endereço de Carteira
0x7c326Dbf40DbC6dB80F440A54d209EA396631161 ✓
Endereço informado corretamente

Início de Jornada
08:00 AM ✓
Início de jornada informado corretamente

Fim de Jornada
05:00 PM ✓
Fim de jornada informado corretamente

Estado
 Ativo

Carteira Empregador
0x71C7...9553 ⓘ
Você deve informar o endereço da carteira do empregador para continuar

Atualizar

Figura 115 – Teste de validação frontend (inválido). Fonte: Autor

Na sequência, se informarmos um endereço de carteira em formato incorreto, agora, nosso frontend reconhecerá que o campo que antes não tinha valor, agora é preenchido, mas backend da página irá nos alertar do erro ao submeter o formulário e não enviará a requisição para a API.

Nome Completo
NOVO EMPREGADO ✓
nome informado corretamente

Endereço de Carteira
0x7c326Dbf40DbC6dB80F440A54d209EA396631161 ✓
Novo Endereço de Carteira
0x7c326Dbf40DbC6dB80F440A54d209EA396631161 ✓
Endereço informado corretamente

Início de Jornada
08:00 AM ✓
Início de jornada informado corretamente

Fim de Jornada
05:00 PM ✓
Fim de jornada informado corretamente

Estado
 Ativo

Carteira Empregador
32453gdvcsacascd ✓
Carteira do empregador informado corretamente

Atualizar

Figura 116 – Teste de validação frontend (válido). Fonte: Autor

Nome Completo
NOVO EMPREGADO

Endereço de Carteira
0x7c326Dbf40DbC6dB80F440A54d209EA396631161 PIS
557.32897.23-5

Novo Endereço de Carteira
0x7c326Dbf40DbC6dB80F440A54d209EA396631161

Início de Jornada
08:00 AM
Fim de Jornada
05:00 PM
Estado
 Ativo

Carteira Empregador
32453gdvcsacascd

O endereço de carteira deve estar em formato correto

Atualizar

Figura 117 – Teste de validação backend (inválido). Fonte: Autor

Na mesma esteira, se informarmos um endereço de carteira em formato válido, mas este endereço não for um endereço de carteira de empregador cadastrado no sistema, agora, nossa API retornará o erro e não fará a atualização do registro. Como resultado, é exibido um toaster no canto superior direito da página com a mensagem: “Erro na atualização do colaborador. *Employer not exists.*”

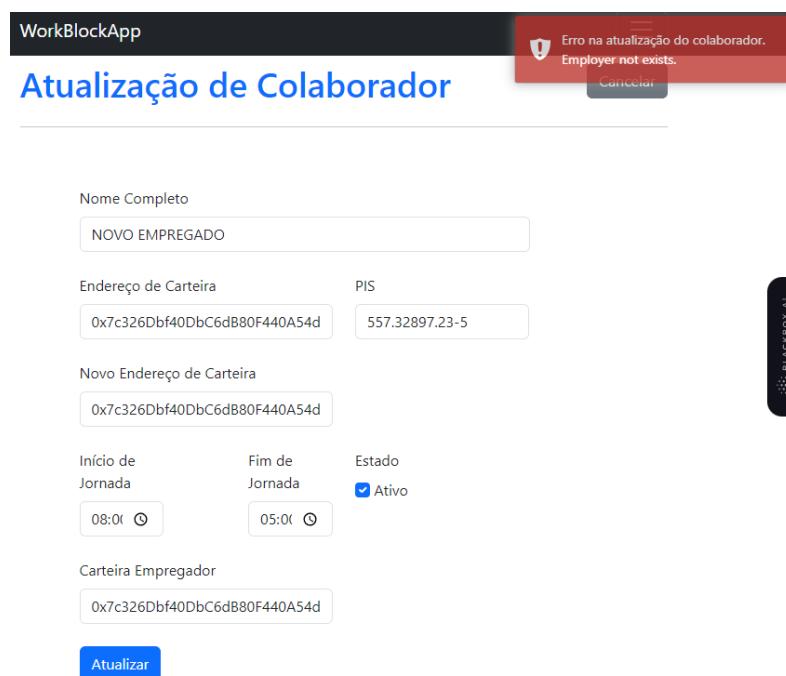


Figura 118 – Teste de validação resposta API (inválido). Fonte: Autor

Finalmente, se todos os dados de nosso formulário forem válidos, o registro é atualizado e redirecionado para a lista de colaboradores e exibido toaster confirmando a operação e o hash da transação realizada.

The screenshot shows the 'Colaborador' section of the WorkBlockApp. A green success message at the top right reads: 'Colaborador atualizado com sucesso.' followed by a transaction hash: '0xd7ffaa73be5cd2cb8454b991b5b5e17230'. Below the message is a button labeled 'Cadastrar'. The main area displays a table of employees with columns: Nome, Carteira, PIS, Empregador, and Ações. The table rows include:

Nome	Carteira	PIS	Empregador	Ações
GILMAR RIBEIRO SANTANA	0xE98...73c05	00.069.911/2196-22	ECO-EMPRESA DE CONSULTORIA E ORGANIZ SISTEMAS EDIT LTDA	Ativo <input checked="" type="checkbox"/> Alterar
PAULO MASSILLON	0xb73...058e7	00.002.426/6254-51	FUNDACAO DE APOIO A ESCOLA TECNICA DO ESTADO DO RIO DE JANEIRO	Ativo <input checked="" type="checkbox"/> Alterar
AYRTON SENNA	0x1Cc...4714c	00.029.799/4289-39	ECO-EMPRESA DE CONSULTORIA E ORGANIZ SISTEMAS EDIT LTDA	Ativo <input checked="" type="checkbox"/> Alterar
EMPLOYEE	0x2f3...3C2Cc	00.012.332/2132-11	ECO-EMPRESA DE CONSULTORIA E ORGANIZ SISTEMAS EDIT LTDA	Ativo <input checked="" type="checkbox"/> Alterar
NOVO EMPREGADO	0x7c3...31161	00.055.732/8972-35	FUNDACAO DE APOIO A ESCOLA TECNICA DO ESTADO DO RIO DE JANEIRO	Ativo <input checked="" type="checkbox"/> Alterar

Figura 119 – Resultado de atualização de registro. Fonte: Autor

Confirmando esta operação podemos ver no nosso banco de dados a ocorrência desse registro de atualização do colaborador, no explorador de blocos do contrato que mantem os dados do colaborador e é exibido um evento desta ação.

abc Id	abc Address	abc OldAddress	abc NewAddress	abc EmployeeName	abc Employee	Empl	Ei	abc EmployerAddress	128	Time	abc HashTransaction
08db84e3	0xE98806d899	0x7c326Dbf40DbC	0x7c326Dbf40DbC6	NOVO EMPREGADO	55732897235	08:00:00	17:00:00	0xb8ce9948b6c92038De	1	2023-07-15 03:25:09.028462	0xd7ffaa73be5cd2cb8454b991b5b5e17230

Figura 120 – Registro da atualização no banco de dados. Fonte: Autor

Note que o hash da transação gravado no banco de dados é o mesmo registrado na blockchain. 0xd7ffaa7....3e17230

Transactions	ERC-20 Token Txns	Contract	Events
Latest 25 from a total of 48 transactions			
Txn Hash	Method	Block	Age
0xd7ffaa73be5cd2cb845...	Update Employee	37948167	1 hr 13 mins ago

Figura 121 – Registro da transação no explorador de blocos. Fonte: Autor

The screenshot shows a transaction detail page for a Polygon PoS Testnet transaction. Key details include:

- Transaction Hash: 0xd7ffaa73be5cd2cb8454bb991b5b58e9f35b06e4d0ed6f1c92b8158183e17230
- Status: Success
- Block: 37948167 (2094 Block Confirmations)
- Timestamp: 1 hr 14 mins ago (Jul-15-2023 03:25:12 AM +UTC)
- From: 0xe98806d8998cae1a45338c5f138438c70b273c05
- To: Contract 0x599cf0c748ad9f5340181a1f139a33ea1d77065f
- Value: 0 MATIC (\$0.00)
- Transaction Fee: 0.000195678002087232 MATIC (\$0.00)
- Gas Price: 0.000000001500000016 MATIC (1.500000016 Gwei)

Figura 122 – Detalhe da transação no explorador de blocos. Fonte: Autor

The screenshot shows a transaction detail page with an 'Events' tab selected. It displays an event named 'EmployeeUpdated' triggered by a transaction. The event parameters are:

```

    updateEmployee
    (address,address,uint256,uint256,uint256,uint256,uint256,uint256)
    address from_
    0xe98806d8998cae1a45338c5f138438c70b273c05
    address oldAddress_
    0x7c326dbf40dbc6db80f440a54d209ea396631161
    address newAddress_
    0x7c326dbf40dbc6db80f440a54d209ea396631161
    uint256 beginningWorkDay_
    800
    uint256 endWorkDay_
    1700
    address employerAddress_
    0xb8ce9948b6c92038de09db42c3ae27a6e744f85d
    uint8 state_
    1
    uint256 timestamp_
    1689391512
  
```

[topic0] 0xc692ce3fffc6b3255d2d6a69015fcc188252f258b07ae35dbde03d07bba41c08
 [topic1] 0x00000000000000000000000000000000e98806d8998cae1a45338c5f138438c70b273c05
 [topic2] 0x000000000000000000000000000000007c326dbf40dbc6db80f440a54d209ea396631161

Figura 123 – Registro do evento da operação registrado no smart contract. Fonte: Autor

Pronto, com isso temos o ciclo completo de nossa aplicação desde a validação dos dados do frontend, backend, API, funções dos smart contracts e gravações dos registros na blockchain, geração de log e gravação desta operação em nosso banco de dados.

A tecnologia .NET é uma ferramenta fantástica que ajuda muito no desenvolvimento

de projetos, e entre seus diversos recursos, utilizamos na entidade um recurso chamado de *Data Annotations*. Com este recurso diversas propriedades e ações podem ser adicionadas de forma distinta a cada propriedade. Abaixo, coloco um exemplo com a propriedade Empregador:

```
[Required(ErrorMessage = "O campo carteira do Empregador é obrigatório")]
[RegularExpression("^0x[a-fA-F0-9]{40}$", ErrorMessage = "O endereço de carteira deve
estar em formato correto")]
[Display(Name = "Carteira Empregador", Prompt = "0x71C7...9553"), StringLength(42)]
[JsonPropertyName("employerAddress")]
public string Empregador { get; set; } = null!;
```

O Código acima adiciona novas funcionalidades à propriedade Empregador da classe EmployeeModel. Vamos descrever cada uma dessas anotações.

```
[Required(ErrorMessage = "O campo carteira do Empregador é obrigatório")]
```

→ Define que esta propriedade é obrigatória em nosso formulário e caso não seja fornecida exibirá a mensagem descrita em *ErrorMessage*.

```
[RegularExpression("^0x[a-fA-F0-9]{40}$", ErrorMessage = "O endereço de carteira deve
estar em formato correto")]
```

→ Define que este campo tem uma formatação específica que deve começar com 0x seguido de 40 caracteres e estes caracteres poder ser letras entre ‘a’ e ‘f’ (maiúsculas ou minúsculas) e números entre 0 e 9 e caso não seja respeitada esta formatação, exibirá a mensagem descrita em *ErrorMessage*.

```
[Display(Name = "Carteira Empregador", Prompt = "0x71C7...9553"), StringLength(42)]
```

→ Define que este campo tem uma etiqueta (*label*) associada definida na anotação *Name* e na propriedade *placeholder* de seu campo de texto será exibido o valor definido na anotação *Prompt*. Por fim, este campo de texto admitirá no máximo 42 caracteres.

Conforme definido pela anotação *StringLength*.

```
[JsonPropertyName("employerAddress")]
```

- ➔ Esta anotação está associada à Serialização desta classe onde a chave "employerAddress" do JSON da API estará relacionado com nossa propriedade Empregador da classe EmployeeModel.

Na renderização da página, a tecnologia .NET continua nos ajudando no desenvolvimento e podemos ver a junção das definições das anotações citadas acima trabalhando junto os elementos HTML associados a *TagHelpers* do ASP.NET.

Para ilustrar um pouco deste comportamento, vamos exibir abaixo o trecho de código utilizado pela renderização da propriedade Empregador.

```
<div class="col-12 col-md-7">
    <label asp-for="EmployeeModel.Empregador" class="form-label"></label>
    <input asp-for="EmployeeModel.Empregador" class="form-control" required>
    <span asp-validation-for="EmployeeModel.Empregador" class="text-danger"></span>
    <div class="invalid-feedback">
        Você deve informar o endereço da carteira do empregador para continuar
    </div>
    <div class="valid-feedback">
        Carteira do empregador informado corretamente
    </div>
</div>
```

Da mesma forma que fizemos com as anotações da classe, vamos descrever abaixo a função destes componentes, linha por linha.

```
<label asp-for="EmployeeModel.Empregador" class="form-label"></label>
```

- ➔ Esta referência fará associação da anotação *Display.Name* definido na propriedade, no caso, "Carteira Empregador"

```
<input asp-for="EmployeeModel.Empregador" class="form-control" required>
```

- Esta referência fará associação de diversas anotações, definindo que este campo é um campo de texto, e associando a ele o *placeholder* e as demais definições de tamanho e formatação do campo.

```
<span asp-validation-for="EmployeeModel.Empregador" class="text-danger"></span>
```

- Associação das mensagens de erro de validação. Este campo só será exibido se ocorrer algum erro de validação capturado pelo backend e definido nas anotações.

```
<div class="invalid-feedback">  
    Você deve informar o endereço da carteira do empregador para continuar  
</div>  
<div class="valid-feedback">  
    Carteira do empregador informado corretamente  
</div>
```

- Estas duas ‘divs’ estão relacionadas com a validação realizada pelo frontend, onde, ao submeter o formulário se o campo estiver de acordo com as regras de validação definidas pela camada de frontend, será exibida a div “**valid-feedback**”, caso contrário, será exibida a div com a classe “**invalid-feedback**”.

17.4 PROJETO WEB DE MARCAÇÃO DE PONTO

Da mesma forma que o projeto web de gerenciamento, o projeto web de marcação de ponto também foi desenhado para ser uma aplicação distinta das demais permitindo seu desenvolvimento e escalabilidade de forma distinta.

A filosofia por traz desta solução é ser o mais leve possível pois, entendendo que o usuário do sistema pode utilizar a aplicação em locais remotos e com internet móvel com

acesso mais lento e difícil, é de fundamental importância que ele consiga realizar este registro, mesmo em cenários de difícil conexão.

A tecnologia utilizada para este o funcionamento deste módulo conta com HTML e JavaScript puro, importando biblioteca Ethers.js para interação com a blockchain. É de fundamental importância informar que para que o usuário consiga interagir com este módulo da aplicação, ele tenha algum recurso que o possibilite utilizar uma carteira com o endereço de sua conta registrada a aplicação como funcionário. Depois de conectado à sua carteira, ele poderá fazer as marcações de ponto.

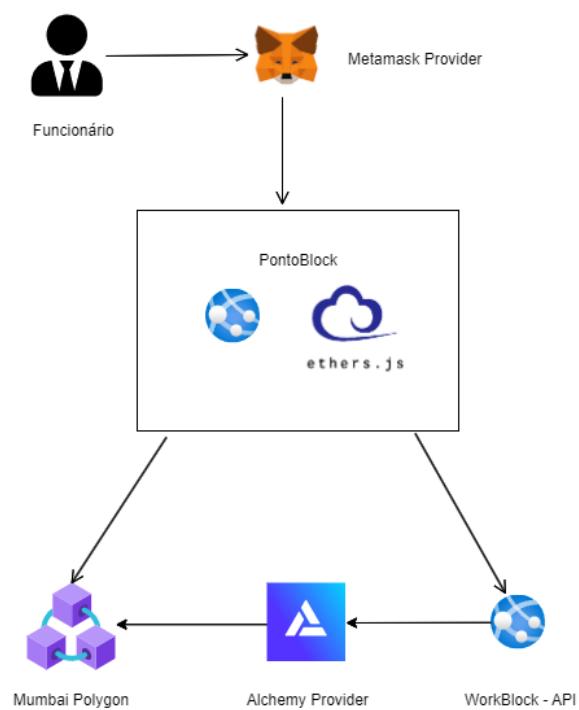


Figura 124 – Representação da arquitetura do projeto PontoBlock. Fonte: Autor

Conforme vemos na imagem, temos dois fluxos depois de conectados com a aplicação. Um deles nos leva à API e outro nos conecta diretamente à blockchain. A comunicação com a API é para nos trazer os dados de cadastro do funcionário, com seu nome, jornada e empregador. Já a comunicação com a blockchain nos permite realizar a

marcação de ponto.

A estratégia de adotar esses dois caminhos é porque queremos preservar a ação de registro de ponto de qualquer interferência fazendo com que essa gravação seja a mais limpa possível, sem que nenhum intermediário interfira, valide, controle ou se intrometa de alguma forma. Esta é a essência das soluções web3, executar ações sem a interferência de um terceiro confiável. A única ação aqui é do funcionário diretamente com o coração de nossa solução, que é a blockchain, onde lá há um smart contract, ou vários, que vai realizar nossa ação de registro de ponto.

A interface da aplicação Ponto Block é bem simples, onde temos um botão conectar, que identificará o provider e fará a conexão, campos somente leitura com a identificação dos dados do funcionário e, logicamente, os botões para marcação de ponto. A imagem 125 nos mostra a visualização dessa página.

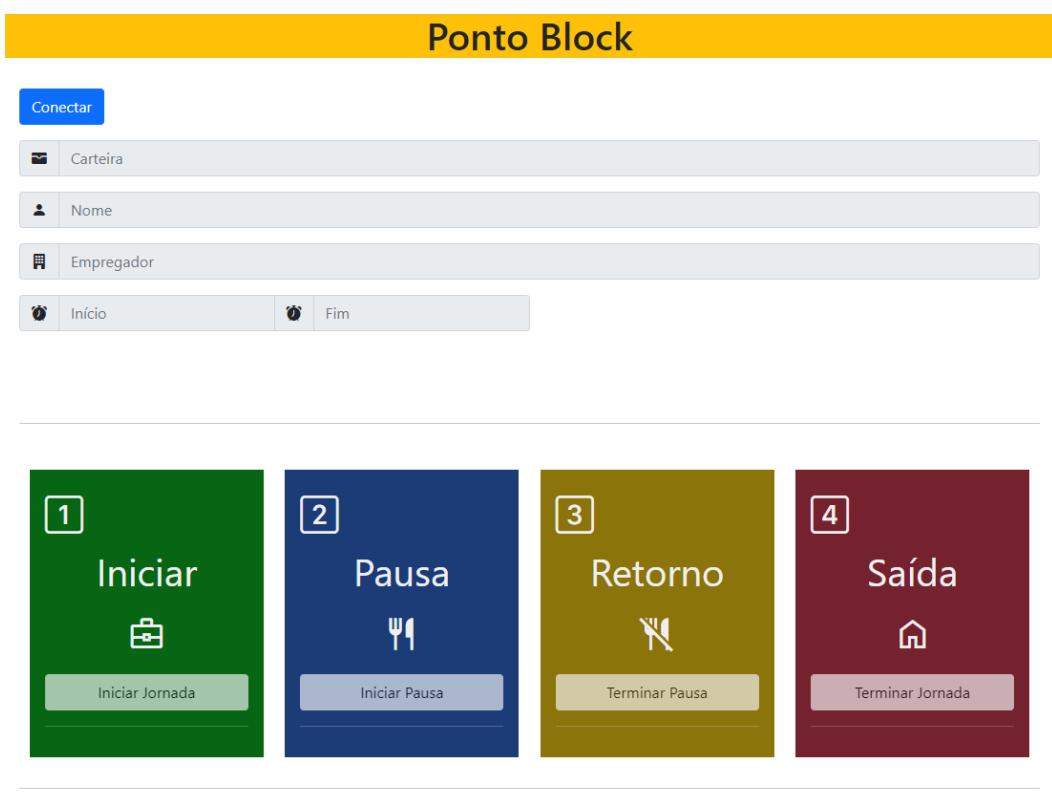


Figura 125 – Página principal da página Ponto Block. Fonte: Autor

Caso o usuário não tenha o MetaMask instalado, não será possível interagir com a aplicação, será exibido um toaster com a mensagem pedindo para instalar o MetaMask e o botão de conectar se transformará em um hiperlink que levará o usuário para a página de download da extensão.

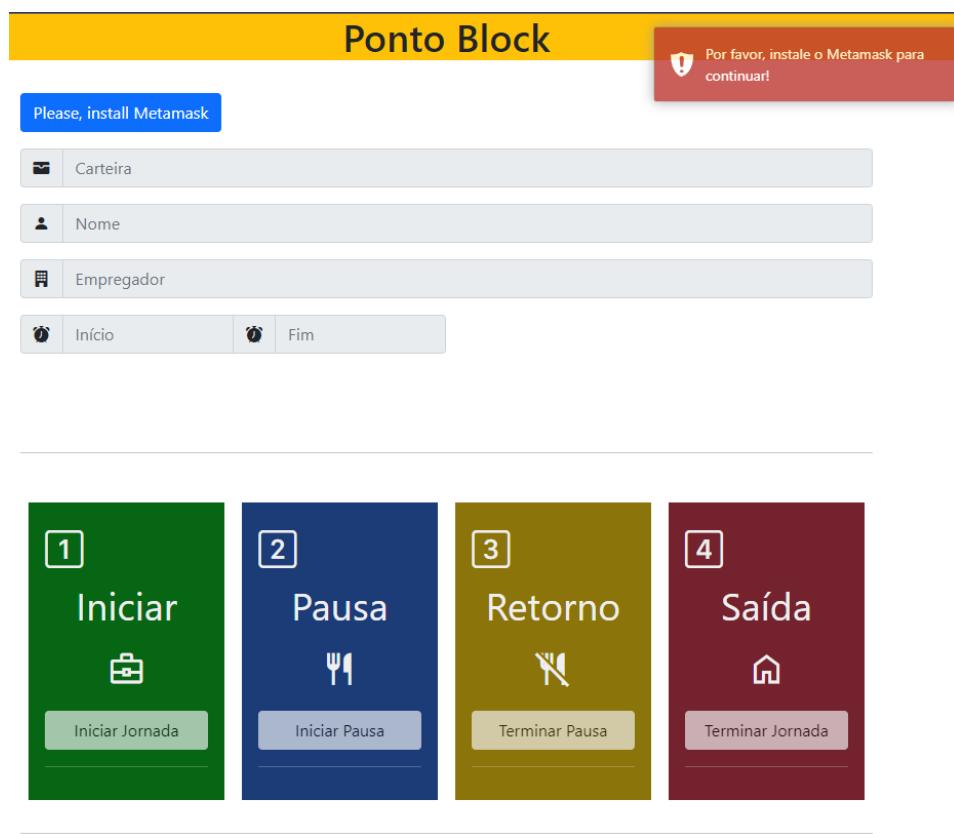


Figura 126 – Mensagem de erro caso não possua o MetaMask. Fonte: Autor

Caso o usuário possua a extensão MetaMask, será conectado com sucesso à extensão e na sequência o Ponto Block fará uma requisição para a API para exibir os dados do funcionário. Caso essa resposta seja positiva, os dados serão carregados e será possível interagir com os botões de marcação de jornada. As figuras 127 e 128 nos mostram essa resposta em caso de sucesso e em caso de erro, sucessivamente.

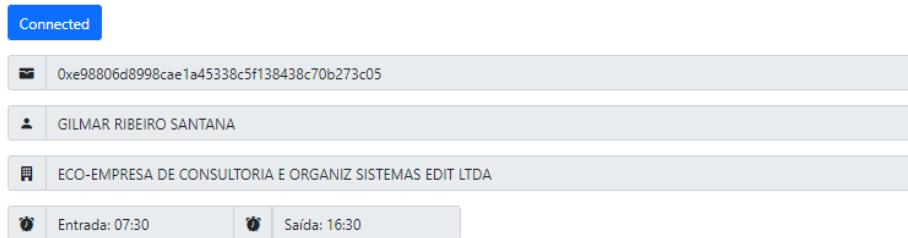


Figura 127 – Conexão com sucesso na aplicação. Fonte: Autor



Figura 128 – Conexão sem sucesso na aplicação. Fonte: Autor

Depois de conectado com sucesso, os botões para marcação de ponto são habilitados para interação. A aplicação pode receber diversos aperfeiçoamentos, entre eles, comunicações da empresa com o funcionário podem ser feitas de forma personalizada. Para ilustrar esse recurso, implementei um card com uma mensagem para que o colaborador atualize seus dados junto ao RH. Foi criado, também, um botão de consulta para os últimos registros do funcionário.



Notícias

The screenshot shows a digital clock interface with the following sections:

- Cadastro:** Non-English text: "Não esqueça de fazer o recadastramento de seus beneficiários no RH." with a "Saiba Mais" button.
- Time Tracking Options:**
 - Iniciar:** Green button with icon and "Iniciar Jornada" button.
 - Pausa:** Blue button with icon and "Iniciar Pausa" button.
 - Retorno:** Yellow button with icon and "Terminar Pausa" button.
 - Saída:** Red button with icon and "Terminar Jornada" button.
- Consultar Histórico:** A grey button for viewing historical data.

Figura 129 – Visão completa após a conexão com sucesso. Fonte: Autor

O smart contract que controla nossas interações com a marcação de ponto tem algumas validações, entre elas, só é possível marcar o fim da jornada se houver um início de jornada registrado para o dia em questão. O mesmo raciocínio se aplica ao término do intervalo. Note que esta validação é feita diretamente pelo smart contract que está publicado na blockchain e como não é uma ação válida, esse registro não será gravado na rede. A imagem 130 mostra esse comportamento. Da mesma forma que as ações anteriores, será exibido um toaster com o erro em questão.

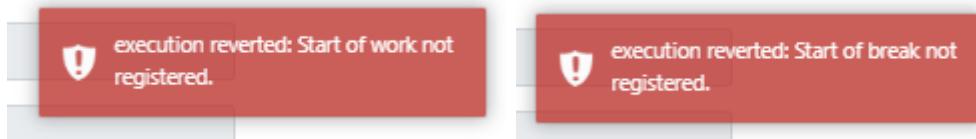


Figura 130 – Toaster com mensagem de erro para marcações inválidas. Fonte: Autor

De outra forma, se a ação for válida, o MetaMask identificará a ação de interação com a blockchain e exibirá uma janela com as informações do tipo de interação, o seu destinatário, o custo da operação e aguarda sua confirmação para que a ação em questão seja enviada para o smart contract.

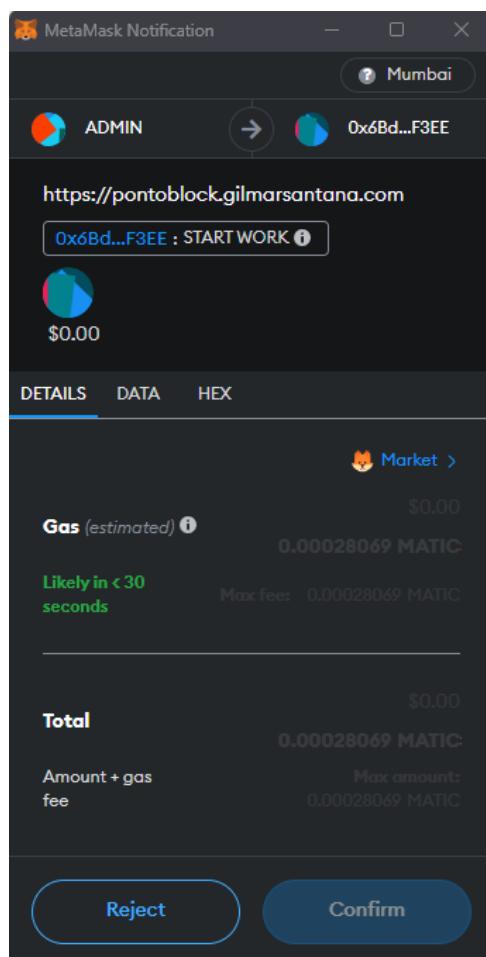


Figura 131 – Janela de confirmação MetaMask. Fonte: Autor

Vamos entender as informações que o MetaMask exibe para esta operação.

Na parte superior da janela MetaMask temos as seguintes informações:

- ➔ Em Vermelho: A rede na qual estamos conectados (Mumbai – Polygon).
- ➔ Em Azul: os atores da ação em que, ADMIN é a minha conta que fará uma ação para a conta 0x6Bd...F3EE, que é o endereço do smart contract.
- ➔ Em Amarelo: O endereço do site na qual estamos conectados com nossa carteira MetaMask.
- ➔ Em Roxo: A função que está sendo acionada para ação.
- ➔ Em Verde: O valor transferido entre as contas. Como não estamos transferindo nenhum valor monetário, é exibido o valor \$0.00.

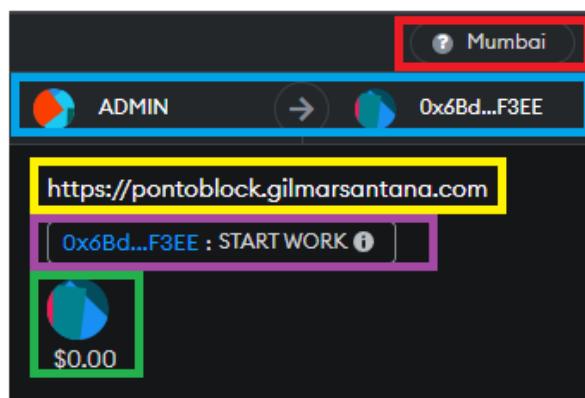


Figura 132 – Detalhes de conta na janela de confirmação MetaMask. Fonte: Autor

Na parte central da janela temos as informações de custo de nossa operação com os valores transferidos entre as duas contas e as taxas gastas na operação. Essas taxas, como já falados anteriormente em nosso documento, são uma espécie de pedágio pelo uso do recurso computacional da operação na blockchain, assim, temos:

- ➔ Em Amarelo: O custo do Gas da operação, que pode ser personalizado escolhendo uma resposta mais rápida ou mais lenta, influenciando no custo do Gas.
- ➔ Em azul: O Custo total que é o somatório do custo da transferência entre contas e o Gas dispendido.
- ➔ Em Vermelho: Os botões para interação, podendo confirmar ou rejeitar a operação.

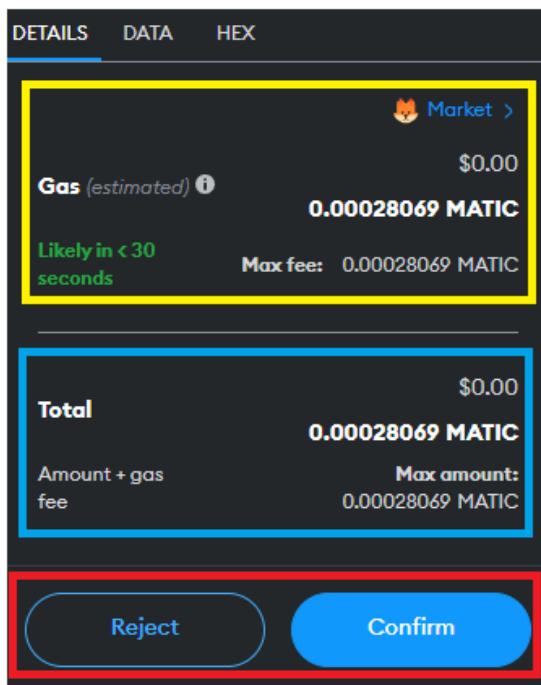


Figura 133 – Detalhes de transação na janela de confirmação MetaMask. Fonte: Autor

Com tudo dando certo em nossa operação, a solicitação será enviada, os dados serão gravados, receberemos o toaster de confirmação, o horário será escrito no botão da ação e o botão será desabilitado para interação. A figura 134 mostra todos estes dados em tela. A cada registro de ponto gravado suas informações de data e hora serão gravadas em cada um de seus respectivos botões. A figura 135 mostra esse comportamento.

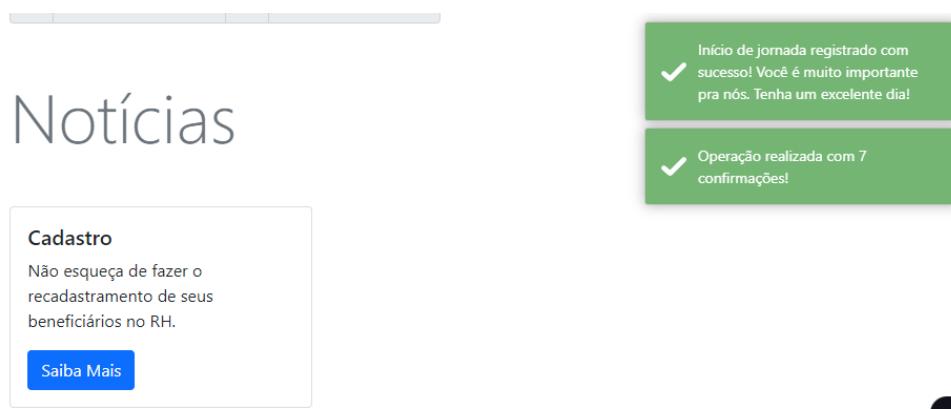


Figura 134 – Registro de início de jornada. Fonte: Autor



Figura 135 – Registros de ponto de um dia de jornada. Fonte: Autor

Com este fluxo podemos ver o ciclo completo de marcação de ponto desde a chamada da ação até o seu retorno para o usuário. As imagens 136, 137, 138 e 139 nos mostram essa evidência registrada na blockchain.



Figura 136 – Registro da transação no explorador de blocos da rede. Fonte: Autor

⑦ Transaction Hash:	0x75d7adfca54463673a8110207491e5bddeb884d8c3c1ea0e8853a08bf4094da
⑦ Status:	Success
⑦ Block:	37975348 286 Block Confirmations
⑦ Timestamp:	⑧ 11 mins ago (Jul-15-2023 08:56:10 PM +UTC)
⑦ From:	0xe98806d8998cae1a45338c5f138438c70b273c05
⑦ To:	Contract 0x6bdd0030280da0170e9fe8544b18d2134925f3ee
⑦ Value:	0 MATIC (\$0.00)
⑦ Transaction Fee:	0.00028069500280695 MATIC (\$0.00)
⑦ Gas Price:	0.000000001500000015 MATIC (1.500000015 Gwei)

Figura 137 – Detalhe da transação de início de jornada. Fonte: Autor

```
65  
66     function startWork()  
67         public  
68             employeeAddedYet(msg.sender)  
69             activeEmployee() {  
70  
71                 require(employeeRecords[msg.sender][util.getDate(block.timestamp)].startWork == 0, "Start of work already registered.");  
72                 employeeRecords[msg.sender][util.getDate(block.timestamp)].startWork = block.timestamp;  
73  
74             emit StartWorkRegistered(msg.sender, employeeRecords[msg.sender][util.getDate(block.timestamp)].startWork, block.timestamp);  
75 }
```

Figura 138 – Código da função de início de jornada. Fonte: Autor

Figura 139 – Registro do evento na blockchain. Fonte: Autor

Para terminar, temos na página o botão de consulta de histórico do funcionário. Da mesma forma que dito antes, será feita uma chamada da ação à blockchain para ter os registros gravados do funcionário. A figura 140 mostra esta ação.

Data	Início	Pausa	Retorno	Saída
16/06/2023	--:--:--	--:--:--	--:--:--	--:--:--
17/06/2023	--:--:--	--:--:--	--:--:--	--:--:--
18/06/2023	--:--:--	--:--:--	--:--:--	--:--:--
19/06/2023	--:--:--	--:--:--	--:--:--	--:--:--
20/06/2023	--:--:--	--:--:--	--:--:--	--:--:--
21/06/2023	--:--:--	--:--:--	--:--:--	--:--:--
22/06/2023	--:--:--	--:--:--	--:--:--	--:--:--
23/06/2023	--:--:--	--:--:--	--:--:--	--:--:--
24/06/2023	--:--:--	--:--:--	--:--:--	--:--:--
25/06/2023	21:57:53	22:35:51	12:54:59	20:40:58
26/06/2023	07:25:32	16:16:36	16:18:52	16:39:26
27/06/2023	08:17:46	12:05:04	13:00:36	16:30:22
28/06/2023	07:36:09	12:15:12	13:36:34	16:35:22
29/06/2023	07:12:36	15:51:46	16:01:28	17:09:50
30/06/2023	07:44:00	13:11:04	13:53:26	19:11:48
01/07/2023	--:--:--	--:--:--	--:--:--	--:--:--
02/07/2023	--:--:--	--:--:--	--:--:--	--:--:--
03/07/2023	07:35:04	12:49:00	16:21:25	16:52:11
04/07/2023	08:38:00	12:08:46	12:58:56	16:28:22
05/07/2023	07:54:38	12:38:35	12:58:09	16:29:55
06/07/2023	07:26:04	13:10:39	14:07:01	17:32:07
07/07/2023	08:45:51	13:44:29	14:39:05	16:59:35
08/07/2023	--:--:--	--:--:--	--:--:--	--:--:--
09/07/2023	--:--:--	--:--:--	--:--:--	--:--:--
10/07/2023	07:02:23	13:15:42	14:09:56	--:--:--
11/07/2023	07:58:10	12:44:44	13:52:50	--:--:--
12/07/2023	08:12:27	14:10:54	14:11:28	16:41:19
13/07/2023	07:48:54	--:--:--	--:--:--	--:--:--
14/07/2023	08:03:43	13:55:10	15:04:46	18:49:53
15/07/2023	17:56:10	18:02:05	18:02:25	18:02:51

Figura 140 – Histórico de marcação de ponto. Fonte: Autor

Para demostrar a mobilidade da aplicação, temos uma aplicação mobile para o MetaMask e caso o usuário instale essa aplicação, é possível fazer a interação com a solução a partir de smartphone. A imagem 141 mostra exemplos de tela dessa interação e a 142 mostra o MetaMask na

loja de aplicativos do Google.

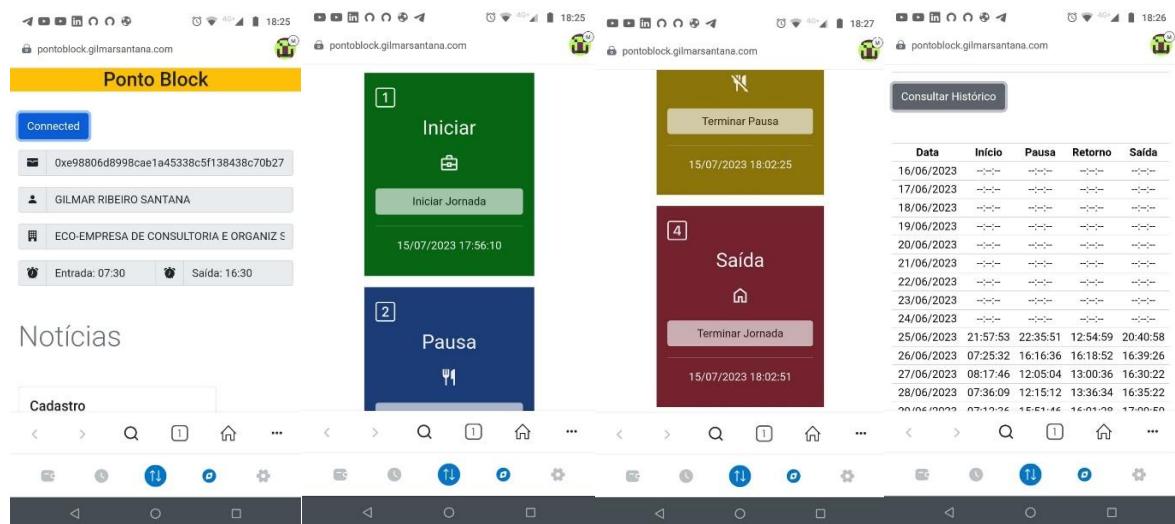


Figura 141 – Ponto Block em dispositivo móvel. Fonte: Autor

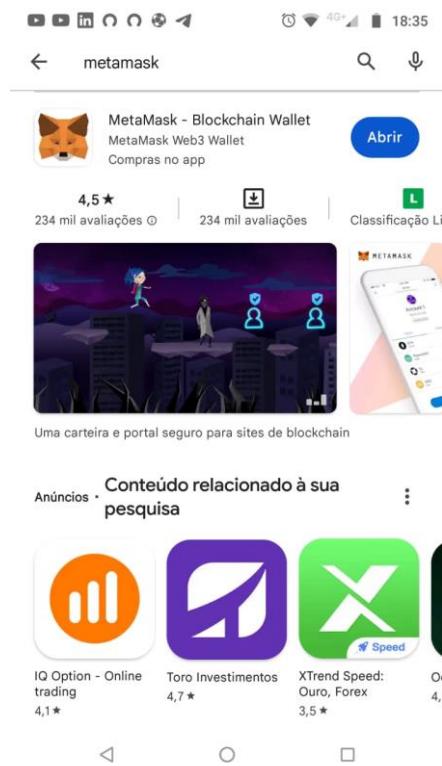


Figura 142 – MetaMask Google Play Store. Fonte: Autor

18.0 POTENCIALIDADES DE MERCADO

Na proposta que expomos aqui, entendemos que seria muito interessante um modelo onde a cobrança seria feita contemplando duas etapas:

1. Implantação do contrato inteligente e toda a infraestrutura necessária para iniciar o serviço do cliente com publicação dos smart contracts e dos serviços de consumo da aplicação para gestão e utilização do sistema.
2. Uma cobrança por cada registro feito no sistema (entrada, saída, início da pausa e fim da pausa).

Com levantamento inicial utilizando ambiente de teste para desenvolvimento da aplicação conseguimos custos bem interessantes tanto para a publicação do smart contract na rede de testes MUMBAI da POLYGON, como para a utilização das funcionalidades. Considerando a cotação da moeda nativa da rede POLYGON, MATIC para Real Brasileiro (BRL) a:

$$\mathbf{1 \text{ MATIC} = 5,77 \text{ BRL}}$$

Temos o seguinte quadro:

DEPLOY		
CONTRATO	MATIC	BRL
UTIL	0,0009	0,005193
ADMINISTRATOR	0,0041	0,023657
EMPLOYER	0,0042	0,024234
EMPLOYEE	0,0042	0,024234
PONTO BLOCK	0,0048	0,027696
PONTO BLOCK		
REPORTS	0,0052	0,030004
TOTAL	0,0234	0,135018

UTILIZAÇÃO DO SISTEMA		
MARCAÇÕES DE PONTO	MATIC	BRL
INÍCIO DE JORNADA	0,0002	0,001154
INÍCIO DE PAUSA	0,0003	0,001731
FIM DA PAUSA	0,0005	0,002885
FIM DA JORNADA	0,0006	0,003462
TOTAL	0,0016	0,009232

EVOLUÇÃO DE CUSTOS		
	MATIC	BRL
CUSTO DIÁRIO POR FUNCIONÁRIO	0,0016	0,009232
FUNCIONÁRIO MÊS (22 DIAS)	0,0352	0,203104
100 FUNCIONÁRIOS MÊS	3,52	20,3104
100 FUNCIONÁRIOS ANO	42,24	243,7248

PREVISÃO DE FATURAMENTO	BRL
EMPRESA COM 100 FUNCIONÁRIOS (ASSINATURA MENSAL)	1.230,00
FATURAMENTO ANUAL	14.760,00
CUSTO ANUAL	- 243,72
SALDO ANUAL	14.516,28

De acordo com as tabelas acima, notamos custos bem competitivos para a implantação do sistema. No cenário proposto, num universo de 100 funcionários em uma empresa conseguimos o resultado de R\$ 14.516,28 como custo anual para funcionamento do sistema.

19.0 CONCLUSÃO

O WORKBLOCK foi desenvolvido para, em resumo, oferecer meios para que uma companhia possa fazer gestão de controle de ponto de seus funcionários utilizando a tecnologia blockchain.

Com a implementação da solução, atividades de gestão de controle dos utilizadores e gestores do sistema, além, é claro, das marcações de ponto poderão ser realizadas permitindo gerar relatórios de marcação de ponto e acompanhamento dos registros.

O desenvolvimento desta solução foi um tremendo desafio, principalmente por tratar de uma tecnologia que apesar de sólida ainda não é tão difundida como solução empresarial ganhando maior destaque em soluções financeiras. Inicialmente apresentamos o problema a ser atacado, propomos o blockchain como solução para lidar com o problema e navegamos por vários preceitos teóricos da tecnologia. Conhecemos algumas soluções de mercado, especificamos os requisitos para desenvolvimento e publicamos o projeto e abordamos os custos para análise mercadológica e viabilidade econômica para seu uso.

Enfim, a jornada foi longa, e entendemos que há melhorias a ser aplicadas no sistema como utilização de validações de login com implantação de JWT, por exemplo, integração com sistema de folha de pagamento e homologação do sistema junto às entidades governamentais para sua implementação no mercado, mas creio que já avançamos grandemente na solução desenvolvida até aqui.

Entendo que os objetivos iniciais pretendidos foram alcançados e, principalmente, o alvo maior de desenvolver uma solução de negócio tendo um software como ferramenta foi alcançado. Além de diversas tecnologias aprendidas, um crescimento intelectual e

profissional foi atingido.

Deixo, então, meus sinceros agradecimentos à toda FAETERJ pela excelência em educação, e as amizades aqui criadas, algo que levamos para toda a vida e guardamos em nosso coração.

REFERÊNCIAS

ARRIAL, T. **Estudo Técnicos** - junho de 2018. Confederação Nacional de Municípios, 2018. Disponível em: <<https://www.cnm.org.br/cms/biblioteca/Eleitorado-2018.pdf>>.

ATTARAN, Mohsen e **GUNASEKARAN**, Angappa. **Applications of Blockchain Technology in Business - Challenges and Opportunities**. p. 14. Texas: Springer, 2019.

Bitcoin.org. What is Bitcoin. Disponível em: <<https://bitcoin.org/en/faq#general>>. Acesso em: 21 de maio de 2022.

BSC Army. Node and how to Run It. Disponível em: <<https://bscarmy.com/node-and-how-to-run-it/>>. Acesso em: 01 de agosto de 2022.

CARVALHO, Carla Arigony de; **ÁVILA**, Lucas Veiga. **A TECNOLOGIA BLOCKCHAIN APLICADA AOS CONTRATOS INTELIGENTES**. **Revista Em Tempo**, [S.I.], v. 18, n. 01, p. 159, dec. 2019. ISSN 1984-7858. Disponível em: <<https://revista.univem.edu.br/emtempo/article/view/3210>>. Acesso em: 19 june 2022.

Contas Ethereum. Disponível em: <<https://ethereum.org/pt-br/developers/docs/accounts/>>. Acessado em: 11 de março de 2023.

CUNHA, S. S. d. et al. **Relatório sobre o Sistema Brasileiro de Votação Eletrônica**. COMITÊ MULTIDISCIPLINAR INDEPENDENTE, 2014. Disponível em: <<http://www.brunazo.eng.br/voto-e/textos/CMind-1-Brasil-2010.pdf>>.

DAVIDSON, All. **Increasing trust in criminal evidence with blockchains.** MOJ Digital & Technology. Disponível em: <<https://mojdigital.blog.gov.uk/2017/11/02/increasing-trust-in-criminal-evidence-with-blockchains/>>. Acesso em: 21 de maio de 2022.

ENCCLA-2020. Estratégia Nacional de Combate à Corrupção e Lavagem de Dinheiro. Blockchain no setor público: Guia de conceitos e usos potenciais. Agosto de 2020. Disponível em: <<http://enccla.camara.leg.br/acoes/arquivos/resultados-enccla-2020/blockchain-no-setor-publico-guia-de-conceitos-e-usos-potenciais>>. Acesso em: 21 de maio de 2022.

ESCOBAR, Matheus Garcia. **Contextualização e Introdução ao Blockchain.** 2021. Disponível em: <<https://www.ufsm.br/pet/sistemas-de-informacao/2021/11/29/contextualizacao-e-introducao-ao-blockchain/>>. Acesso em: 22 de maio de 2022.

Ethereum. Disponível em: <<https://ethereum.org/pt-br/>>. Acessado em: 21 de maio de 2022.

FACHINI, Tiago. **Projuris.** **Smart contracts: o que é, como funciona e aspectos legais.** Disponível em: <<https://www.projuris.com.br/smart-contract/>>. Acesso em: 21 de maio de 2022.

HARDHAT. Disponível em: < Documentation | Ethereum development environment for professionals by Nomic Foundation (hardhat.org)>. Acessado em: 21 de maio de 2022

KUNTZ, João. Blockchain Ethereum – Fundamentos de Arquitetura, desenvolvimento de contratos e aplicações. p. 21-26. São Paulo: Casa do Código, 2022.

Lei de Liberdade Econômica – Lei Nº 13.874 de 20 de setembro de 2019, Art. 74 parágrafo segundo. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2019-2022/2019/lei/L13874.htm>. Acessado em: 10 de março de 2023.

Máquina virtual do Ethereum (EVM). Disponível em: <<https://ethereum.org/pt-br/developers/docs/evm/>>. Acessado em: 11 de março de 2023.

MORAIS, Anderson Melo de, LINS, Fernando Antonio Aires. Uso de Blockchain na Educação: Estado da arte e desafios

NAKAMOTO, Satoshi. Bitcoin: A Peer-to-Peer Eletronic Cash System. Bitcoin.org. Disponível em: <<https://bitcoin.org/bitcoin.pdf>>. Acesso em: 21 de maio de 2022.

NIWA, Henrique; "UM SISTEMA DE VOTO ELETRÔNICO BASEADO EM BLOCKCHAIN", p. 2879 . In: **Anais do XIX Simpósio de Pesquisa Operacional & Logística da Marinha.** São Paulo: Blucher, 2020. Disponível em: <<https://www.proceedings.blucher.com.br/article-details/um-sistema-de-voto-eletrônico-baseado-em-blockchain-34623>>. Acesso em: 20 de junho de 2022.

O, S. K., XU, X., CHIAM, Y. K. and LU, Q., "Evaluating Suitability of Applying Blockchain," 2017 22nd International Conference on Engineering of Complex Computer Systems (ICECCS), 2017, pp. 158-161, doi: 10.1109/ICECCS.2017.26.

OLIVEIRA, F. R.; MAZIERO, R. C.; ARAÚJO, L. S. de. UM ESTUDO SOBRE A WEB 3.0: evolução, conceitos, princípios, benefícios e impactos. Revista Interface Tecnológica, [S. I.], v. 15, n. 2, p. 60–71, 2018. DOI: 10.31510/infa.v15i2.492. Disponível em: <<https://revista.fatectq.edu.br/interfacetecnologica/article/view/492>>. Acesso em: 8 jun. 2022.

Polygon. Disponível em: <<https://coinext.com.br/criptomoedas/polygon>>. Acessado em: 10 de março de 2023.

Portaria Nº 1.510/2010 MTE. Disponível em:

<https://www.trt2.jus.br/geral/tribunal2/ORGaos/MTE/Portaria/P1510_09.html>. Acessado em: 10 de março de 2023.

Portaria Nº 373/2011 MTE. Disponível em:

<https://www.trt2.jus.br/geral/tribunal2/ORGaos/MTE/Portaria/P373_11.html>. Acessado em: 10 de março de 2023.

Portaria Nº 671/2021 MTP. Disponível em: <<https://www.in.gov.br/en/web/dou/-/portaria-359094139>>. Acessado em: 10 de março de 2023.

RAMALHO, R. Prazo para eleitor pedir voto em trânsito termina nesta quinta. 2018.

Disponível em: <<https://g1.globo.com/politica/eleicoes/2018/noticia/2018/08/22/termina-nesta-quinta-feira-prazo-para-eleitor-pedir-voto-em-transito.ghtml>>.

RODRIGUES, Carlo Kleber da Silva. Uma análise simples de eficiência e segurança da Tecnologia Blockchain. Revista de Sistemas e Computação, Salvador, v. 7, n. 2, p. 147-162, jul./dez. 2017. Disponível em:

<<https://repositorio.uniceub.br/jspui/bitstream/235/11373/1/Uma%20an%c3%a1lise%20simples%20de%20efici%c3%aancia%20e%20seguran%c3%a7a%20da%20Tecnologia%20Blockchain.pdf>>. Acesso em: 31 de julho de 2022.

SCHULTZ, Felix. Entenda 5 vantagens do Blockchain para empresas. Milvus. 11 de janeiro de 2019. Disponível em: <<https://blog.milvus.com.br/entenda-5-vantagens-do-blockchain-para-as-empresas/>>. Acesso em: 21 de maio de 2022.

SILVA, Euber Chaia Cotta e, e MARQUES, Rodrigo Moreno. Blockchain no setor público: uma revisão sistemática de literatura.

AtoZ:novaspráticaseminformaçaoeconhecimento,10(3),1-11, set./dez.2021. Disponível em: <<https://revistas.ufpr.br/atoz/article/view/79903/44241>>. Acesso em: 21 de maio de 2022.

STALLINGS, William. Criptografia e segurança de redes. p. 181 - 227. 4. ed. São Paulo: Pearson Prentice Hall, 2008.

TAKENOBU, T. – Ethereum EVM Illustrated. Disponível em: <https://takenobu-hs.github.io/downloads/ethereum_evm_illustrated.pdf>. Acessado em: 11 de março de 2023.

TEKISALP, Emre. **Understanding Web3 – A User Controlled Internet.** Disponível em: <<https://www.coinbase.com/blog/understanding-web-3-a-user-controlled-internet>>. Acessado em 10 de março de 2023.

TSE. **Eleições 2018: confira as datas do calendário eleitoral.** 2018. Disponível em: <<https://g1.globo.com/politica/eleicoes/2018/noticia/eleicoes-2018-datas.ghtml>>.

UML – UNIFIED MODELING LANGUAGE. Disponível em: <<https://www.uml.org/what-is-uml.htm>>. Acessado em: 19 de junho de 2022.

WÜST, Karl e GERVAIS, Arthur. **Do you need a blockchain?** 2018. Disponível em: <<https://www.law.berkeley.edu/wp-content/uploads/2018/08/Do-you-need-a-Blockchain-Karl-Wust-and-Arthur-Gervais.pdf>>. Acesso em: 21 de maio de 2022

ZAGO, Mateo. **Why the Web 3.0 Matters and you should know about it.** 2018. Disponível em: <<https://medium.com/@Matzago/why-the-web-3-0-matters-and-you-should-know-about-it-a5851d63c949>>. Acesso em: 08 de junho de 2022.

ZHENG, Zibin, **XIE**, Shaoan, **DAI**, Hongning, **CHEN**, Xiangping, **WANG**, Huaimin. "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends," 2017 IEEE International Congress on Big Data (BigData Congress), 2017, pp. 557-564, 558-559, doi: 10.1109/BigDataCongress.2017.85.

.NET. Disponível em: <What is .NET? An open-source developer platform. (microsoft.com)>. Acessado em: 21 de maio de 2022.