

CENTRO UNIVERSITÁRIO SERRA DOS ÓRGÃOS - UNIFESO  
CENTRO DE CIÊNCIA E TECNOLOGIA - CCT  
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

DOUGLAS ORNELAS DE SOUSA

**USO DE BLOCKCHAIN PARA REGISTROS DE DADOS DE CADEIAS DE PRODUÇÃO**

TERESÓPOLIS

2020

**CENTRO UNIVERSITÁRIO SERRA DOS ÓRGÃOS - UNIFESO**  
**CENTRO DE CIÊNCIA E TECNOLOGIA - CCT**  
**CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

**DOUGLAS ORNELAS DE SOUSA**

**USO DE BLOCKCHAIN PARA REGISTROS DE DADOS DE CADEIAS DE PRODUÇÃO**

Trabalho de Conclusão de Curso apresentado  
ao Centro Universitário Serra dos Órgãos como  
requisito obrigatório para obtenção do título de  
Bacharel em Ciência da Computação.

Orientador: Prof. Alberto Torres Angonese

**TERESÓPOLIS**

**2020**

S696 Sousa, Douglas Ornelas de.  
Uso de blockchain para registros de dados de cadeias de produção. / Douglas Ornelas de Sousa. – 2020.  
38f.

Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro Universitário Serra dos Órgãos, UNIFESO, Teresópolis, 2020.  
Bibliografia: f. 37-38.  
Orientador: Alberto Torres Angonese.

1. Ciências da Computação. 2. Blockchain. 3. Cadeia Produtiva. 4. Solidity. I. Título.


CDD 004

CENTRO UNIVERSITÁRIO SERRA DOS ÓRGÃOS - UNIFESO  
CENTRO DE CIÊNCIA E TECNOLOGIA - CCT  
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

**USO DE BLOCKCHAIN PARA REGISTROS DE DADOS DE CADEIAS DE PRODUÇÃO**

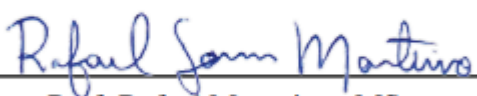
DOUGLAS ORNELAS DE SOUSA

Trabalho de Conclusão de Curso apresentado ao Centro Universitário Serra dos Órgãos como requisito obrigatório para obtenção do título de Bacharel em Ciência da Computação.



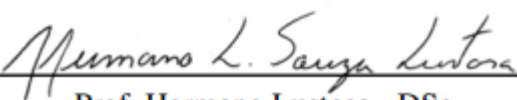
---

Prof. Alberto Torres Angonese - DSc.  
Orientador



---

Prof. Rafael Monteiro - MSc.



---

Prof. Hermano Lustosa - DSc.

*Dedico esse trabalho a todas as pessoas que pintaram um pedaço da minha vida comigo,  
graças a elas consigo ver o mundo com todas essas cores*

## **AGRADECIMENTOS**

Gostaria de agradecer a toda a minha família que me criou como a pessoa que sou hoje em dia, em especial a minha mãe e meus irmãos que sempre me apoiaram e fizeram de tudo por mim para que eu conseguisse ser hoje quem eu sou, sem eles eu não seria metade da pessoa que me tornei. Agradeço também a todos os professores que tive durante essa jornada, desde a Michelle que foi a primeira professora que me mostrou que aprender era divertido, assim como o Carlinhos, o Melo e todos os outros que sempre ensinaram com amor a seus alunos até eu chegar a minha escolha profissional. Agradeço ao professor José Roberto que foi a pessoa que me guiou para a escolha dessa carreira quando estava com dúvidas ao final do ensino médio, e de toda a parceria que tive com ele durante esse tempo de faculdade nos projetos que participamos. Agradeço ao Laion e a Samara, que sempre que necessário me ajudaram em qualquer problema que tive na faculdade, me sinto como se eles pertencessem a minha família com todos os carinhos e orientações que me deram. Agradeço também aos meus professores na faculdade, em especial ao Rafael Monteiro, que me incentivou tanto a me divertir programando com as maratonas de programação que participei e ao Hermano e ao Laion, professores que sempre ensinaram de forma divertida aos seus alunos. Agradeço também a todos os amigos que tive durante minha vida e durante a faculdade, e aos amigos que posso contar até hoje e sei que posso contar no futuro, como o Maycon Cuervo, o João Pedro Neves, o Daniel Tatagiba e o Lucas Amaral. Agradeço também a todos os amigos que eu tive que já se formaram, mesmo que tenha perdido o contato de alguns deles, como o Ariel Zimbrão, o Gustavinho e o Katreque. Gostaria de agradecer também ao Guimarães, Nívea, Mariane, Leonardo, Carol, Babi, Rocha, Anne e o Rafael, colegas de trabalho que sempre me trataram com respeito e carinho. Por fim agradeço ao meu orientador Alberto Angonese, que foi meu parceiro desde o início da faculdade quando o conheci na eletiva de robótica, a qual fui mentor e aprendi muito com ele graças a isso, e agradeço por me ajudar com essa ideia de monografia e ser sempre meu amigo em várias ocasiões e eventos, como a OBR.

*“Life is not complex. We are complex.  
Life is simple, and the simple thing is the right thing.”  
(Oscar Wilde)*

## RESUMO

No contexto da rastreabilidade de cadeias de produção de produtos existem várias informações que devem ser registradas e que são importantes para o conhecimento do consumidor. Essa informação deve ser pública e há a necessidade para ela ser garantida, para um consumidor com problemas como alergia possa ter segurança de que ele está consumindo um produto seguro. Este trabalho propõe uma solução para assegurar a segurança e confiabilidade dos dados da cadeia de produção usando *Blockchain* para gerenciar como esse dado é validado e como ele pode ser retornado ao usuário. Para isso foi desenvolvida uma aplicação *Web* utilizando o *framework* Django para o controle de registro e armazenamento de informações de etapas de uma cadeia de produção em rede *blockchain*. Foi desenvolvido um contrato inteligente para o registro desses dados referentes à etapa de produção em rede *blockchain*, através da rede Ethereum, utilizando a linguagem Solidity. Esses contratos foram registrados na rede Ethereum a partir do *framework* Infura. E com esses dados foi proposta e validada uma forma de registro de cadeias generalizadas usando a rede Ethereum.

**Palavras-chave:** *Blockchain*, cadeia de produção, Solidity.



## ABSTRACT

In the context of tracking supply chains there is a lot of information that should be registered and that are important for the customer knowledge. This information should be public and there is the need of it being ensured, so a customer with problems like allergies can trust that he is using a safe product. With this in mind, in this monography a solution is proposed to confirm the security and trust of data in a supply chain using Blockchain to manage how the data is validated and how it can be returned to the user. For this a web application using the Django framework was made to control how the information of steps of the supply chain is registered and stored. A smart contract was made to register these data referring to the supply chain steps on the blockchain, using the solidity programming language. These contracts were then registered on the Ethereum network from the Infura framework. With this data, a form of registering generalized chains using the Ethereum network was proposed and validated.

**Keywords:** *Blockchain*. Supply Chain. Solidity.

## LISTA DE ILUSTRAÇÕES

Figura 1 – Exemplo de código em Python . . . . .	13
Figura 2 – Exemplo de uso do <i>framework</i> Django . . . . .	14
Figura 3 – Interface do PGAdmin para consulta de dados do PostgreSQL . . . . .	15
Figura 4 – Funcionamento <i>Blockchain</i> . . . . .	17
Figura 5 – Ilustração dos blocos em uma rede <i>blockchain</i> . . . . .	17
Figura 6 – Estrutura de entrada de dados em Solidity . . . . .	18
Figura 7 – Exemplo de código em Solidity, na IDE Remix . . . . .	20
Figura 8 – Exemplo de sensor e RF tag, usados em um microcontrolador Arduino . . . . .	22
Figura 9 – Arquitetura Proposta pelo trabalho relacionado . . . . .	22
Figura 10 – Interface QuipoAgro . . . . .	23
Figura 11 – Comparação entre trabalhos relacionados . . . . .	23
Figura 12 – Estrutura de dados utilizada . . . . .	25
Figura 13 – Fluxo de exemplo de uma cadeia de produção . . . . .	26
Figura 14 – Tela de cadastro de usuários . . . . .	27
Figura 15 – Exemplo de dados no banco . . . . .	27
Figura 16 – Dados e seu <i>hash</i> gerado . . . . .	28
Figura 17 – Dados na rede <i>blockchain</i> . . . . .	28
Figura 18 – Fluxograma do cadastro de dados . . . . .	29
Figura 19 – Cadastro de usuário . . . . .	30
Figura 20 – Lista de usuários . . . . .	30
Figura 21 – Cadastro de tipos de cadeia de produção e suas etapas . . . . .	31
Figura 22 – Lista de tipos de cadeia de produção e suas etapas . . . . .	31
Figura 23 – Dados cadastrados para o envio das transações para a rede Ethereum . . . . .	32
Figura 24 – Exemplos de arquivos cadastrados para o envio das transações para a rede Ethereum . . . . .	32
Figura 25 – Lista de dados cadastrados para o envio das transações para a rede Ethereum . . . . .	33
Figura 26 – Lista de dados cadastrados para o envio das transações para a rede Ethereum no banco de dados . . . . .	33
Figura 27 – Página de envio de transações para a rede Ethereum . . . . .	34
Figura 28 – Dados do bloco da rede Ethereum no banco de dados . . . . .	34
Figura 29 – Dados do bloco na rede Ethereum . . . . .	35

## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO . . . . .</b>	<b>11</b>
1.1	Justificativa . . . . .	11
1.2	Motivação . . . . .	11
1.3	Objetivo . . . . .	12
1.3.1	Objetivos Gerais . . . . .	12
1.3.2	Objetivos Específicos . . . . .	12
<b>2</b>	<b>FUNDAMENTAÇÃO TEÓRICA . . . . .</b>	<b>13</b>
2.1	Python . . . . .	13
2.2	Django . . . . .	13
2.3	PostgreSQL . . . . .	14
2.4	<i>Blockchain</i> . . . . .	14
2.4.1	BitCoin . . . . .	16
2.4.2	Blocos . . . . .	16
2.4.3	Ethereum . . . . .	16
2.4.4	Contratos Inteligentes . . . . .	18
2.4.5	Solidity . . . . .	19
2.4.6	Remix IDE . . . . .	19
<b>3</b>	<b>TRABALHOS RELACIONADOS . . . . .</b>	<b>21</b>
3.1	<i>An Agri-food Supply Chain Traceability System for China Based on RFID Blockchain Technology</i> . . . . .	21
3.2	<i>How blockchain improves the supply chain: case study alimentary supply chain</i> . . . . .	21
3.3	QuipoAgro . . . . .	21
<b>4</b>	<b>METODOLOGIA E DESENVOLVIMENTO . . . . .</b>	<b>24</b>
4.1	Implementação utilizada . . . . .	24
4.2	Banco de dados . . . . .	24
4.2.1	Estrutura de dados . . . . .	24
4.3	Geração do arquivo <i>Hash</i> . . . . .	26
4.4	Desenvolvimento . . . . .	26
<b>5</b>	<b>EXPERIMENTOS E RESULTADOS . . . . .</b>	<b>29</b>
<b>6</b>	<b>CONCLUSÃO . . . . .</b>	<b>36</b>
<b>7</b>	<b>SUGESTÕES DE TRABALHOS FUTUROS . . . . .</b>	<b>37</b>
	<b>REFERÊNCIAS . . . . .</b>	<b>38</b>

# 1 INTRODUÇÃO

Com o avanço da tecnologia que vivemos, vemos novas formas de cultivo, transporte, e várias outras maneiras que nossos produtos são tratados (PEDERNEIRAS, 2020). Com isso surge uma necessidade de rastreabilidade de como o produto que consumimos é tratado antes de chegar ao consumo. Há diversos problemas que podem ser gerados a empresas com forma de coleta, armazenamento e transporte indevido que podem gerar vários riscos à saúde do consumidor, e há a necessidade que possamos assegurar a segurança desses produtos para a cadeia de produção de forma a não acarretar um prejuízo à empresa (BRASIL, 2020). Com isso, neste trabalho é proposta uma solução para que, a partir dos dados de uma cadeia de produção genérica, possamos realizar o registro desses dados em uma rede *blockchain*, garantindo a imutabilidade desse dado e gerando uma segurança na informação que vemos dessa cadeia. A proposta vem com a ideia de gerar uma arquitetura de dados de forma generalizada e, a partir de uma aplicação *web* desenvolvida em Django, realizar o registro de informações quaisquer dessa cadeia de produção na rede Ethereum.

## 1.1 JUSTIFICATIVA

Com o aumento que vem surgindo sobre necessidade de segurança de dados, nota-se uma necessidade cada vez maior em garantia de integridade dos dados que são encontrados sobre produtos (LEÃO, 2020). Em uma cadeia de produção, a origem e garantia de qualidade do produto durante toda sua produção até ele ser consumido é um fator de extrema importância na decisão de qual produto um consumidor deseja. Com o uso de uma tecnologia de *blockchain*, baseado em coleta de dados e seu devido registro durante uma cadeia de produção, pode-se garantir uma forma de armazenamento de informação que não pode ser adulterada, com isso gerando garantia de qualidade de produto, garantindo transparência, imutabilidade e confiabilidade.

## 1.2 MOTIVAÇÃO

O desafio de poder mostrar ao consumidor que seu produto possui uma melhor qualidade, ou que aplica alguma técnica diferente do seu concorrente é algo que todo produtor encontra durante seu trabalho. Por exemplo em um setor farmacêutico, a necessidade de garantia de qualidade vai desde a seleção do fornecedor do insumo, passando por toda a etapa de fabricação até a liberação final do produto para o mercado. E além disso, envolve todo o transporte do insumo por meio da transportadora, armazenamento, qualificação do cliente até chegar ao consumidor final. Com isso nota-se que essa garantia de qualidade do produto é extremamente importante. Contudo, ainda ocorrem questões de fraude de qualidade de um produto, ou adulteração de informações durante a cadeia de produção do mesmo, e por causa disso há a necessidade de garantia de integridade em todo o processo.

### 1.3 OBJETIVO

Nesta seção serão apresentados os objetivos gerais e específicos do trabalho

#### 1.3.1 OBJETIVOS GERAIS

Criar uma aplicação capaz de registrar dados de etapas e tipos de cadeias de produção generalizadas.

A partir desses dados, gerar um *hash* que será enviado a rede Ethereum e fazer seu devido registro em rede *blockchain*.

#### 1.3.2 OBJETIVOS ESPECÍFICOS

A partir do uso de aplicação *Web* desenvolvida, permitir cadastros e alterações de dados referentes à uma linha de produção.

Criar um modelo de dados generalista para que seja capaz de se adaptar a linhas de diferentes produtos.

Implementar um contrato inteligente em Solidity para registro dos dados na rede Ethereum.

## 2 FUNDAMENTAÇÃO TEÓRICA

Nessa seção serão apresentadas as ferramentas e tecnologias utilizadas no desenvolvimento da solução proposta.

### 2.1 PYTHON

Python é uma linguagem orientada a objetos de alto nível, com semântica dinâmica e que suporta módulos e pacotes de forma a permitir modularidade do programa e reuso de código. Seu interpretador e sua biblioteca estão disponíveis de forma gratuita para diversas plataformas (PYTHON, 2020). Por ser uma linguagem com diversos pacotes de código feitos pela comunidade disponíveis para serem instalados, é considerada uma ótima linguagem para desenvolvimento de projetos de pesquisa em diversas áreas, como Aprendizado de máquina, Inteligência Artificial, dentre outros. Na Figura 1 podemos ver um exemplo de código em Python utilizado pela aplicação, para a parte de envio de dados cadastrados para a rede *blockchain*.

**Figura 1 – Exemplo de código em Python**

```

network='mainnet',
cache_expire_after=5,

infura_url = "https://rinkeby.infura.io/v3/1544e22dd1614874bad2e7dd9fe37259"
web3 = Web3(Web3.HTTPProvider(infura_url))
address = "0xc79ECF81F950b03673dAa00C3e1fA15d92b2d71"
balance = web3.eth.getBalance(address)
abi = '[{"constant":true, "inputs":[], "name":"name", "outputs":[{"internalType":"string",
abi = json.loads(abi)
contract = web3.eth.contract(address=address, abi=abi)
transaction = contract.functions.transfer(address, 0x01, dado).buildTransaction({'chainId': 4, 'gas':70000, 'nonce': web3.eth.getTransactionCount(address)})
signed_txn = web3.eth.account.signTransaction(transaction, '0x0866deefaeb0618e034886509376744b5964e5c2c11378dc6136d4a195188b1c')
txn_hash = web3.eth.sendRawTransaction(signed_txn.rawTransaction)
resultado = binascii.hexlify(txn_hash)

return resultado

```

**Fonte: Autoria Própria**

### 2.2 DJANGO

Django é um *framework* de Python construído visando o rápido desenvolvimento de sites seguros e de fácil manutenção. O *framework* possibilita um desenvolvimento escalável, seguro, versátil e portátil entre diferentes sistemas. Sua estrutura permite a construção rápida de projetos para *web* com escalonamento e versatilidade (DJANGO, 2020).

Na Figura 2 podemos observar uma parte de um código em Python usando Django para a construção de modelos que serão usados para a aplicação *Web* criada.

**Figura 2 – Exemplo de uso do *framework* Django**

```
from django.db import models
from django.contrib.postgres.fields import JSONField
import uuid
import datetime
from TCC.ether_TCC import envio_transacao_etherium

class usuario(models.Model):
    id = models.UUIDField(primary_key=True, default = uuid.uuid4, editable = False)
    nome = models.CharField(max_length=200,blank=False, default='')
    login = models.CharField(max_length=200,blank=False, default='')
    senha = models.CharField(max_length=200,blank=False, default='')

class tipo(models.Model):
    id = models.UUIDField(primary_key=True, default = uuid.uuid4, editable = False)
    id_usuario = models.ForeignKey(usuario, on_delete=models.CASCADE, db_column='id_usuario')
    nome = models.CharField(max_length=200,blank=False, default='')

class etapa(models.Model):
    id = models.UUIDField(primary_key=True, default = uuid.uuid4, editable = False)
    id_tipo = models.ForeignKey(tipo, on_delete=models.CASCADE, db_column='id_tipo')
    nome = models.CharField(max_length=200,blank=False, default='')

class transacao(models.Model):
    id = models.UUIDField(primary_key=True, default = uuid.uuid4, editable = False)
    id_etapa = models.ForeignKey(etapa, on_delete=models.CASCADE, db_column='id_etapa')
    data = models.DateTimeField(default=datetime.date.today())
    dado = models.CharField(max_length=200,blank=False, default='')
    arquivo = models.FileField(upload_to='documents/%Y/%m/%d/', db_column='arquivo')

class resultado_transacao(models.Model):
    id = models.UUIDField(primary_key=True, default = uuid.uuid4, editable = False)
    id_transacao = models.ForeignKey('Transacao', on_delete=models.CASCADE, db_column='id_transacao')
```

**Fonte: Autoria Própria**

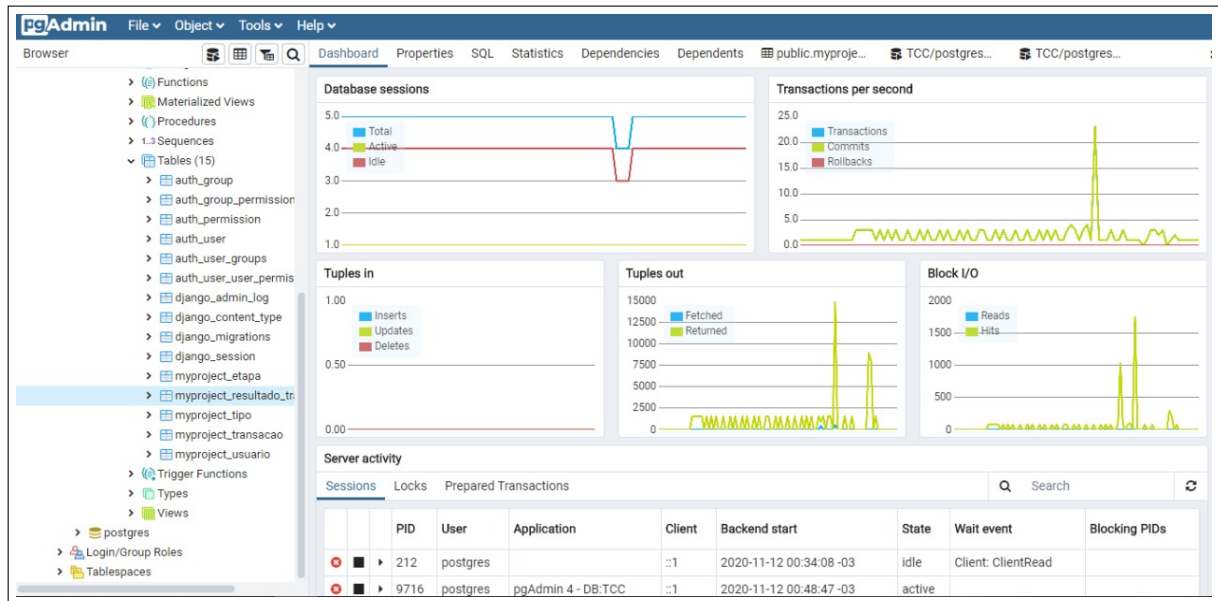
## 2.3 POSTGRESQL

PostgreSQL é um banco de dados relacional com suporte a consultas relacionais e não-relacionais. O mesmo é um banco altamente estável com mais de 20 anos de desenvolvimento em uma comunidade de código aberto (POSTGRESQL, 2020). Por ter uma base de usuários grande e um tempo de mercado muito longo, possui seu uso em diversas grandes empresas, como Spotify e Skype (ROMANOWSKI, 2020). Além disso, o mesmo possui um fácil uso comparado a outras linguagens relacionais e pode ser usada por vários gerenciadores de banco de dados, como o DBeaver e o PGAdmin. Na Figura 3 podemos ver o PGAdmin, que é uma interface para comunicação com o banco de dados PostgreSQL e visualização simplificada dos dados.

## 2.4 BLOCKCHAIN

O *Blockchain* tem sua definição como um "Livro de registro contábil (livro razão) distribuído, para registro de transações, inicialmente de criptomoedas, de forma transparente, confiável e imutável"(NAKAMOTO, 2008). Nas próximas subseções serão explicados os principais conceitos utilizados para o desenvolvimento deste trabalho. Para a explicação desses conceitos, será necessário a explicação de estruturas usadas para contratos inteligentes, sendo

**Figura 3 – Interface do PGAdmin para consulta de dados do PostgreSQL**



**Fonte: Autoria Própria**

elas:

- *Proof-of-Work*

O *Proof of Work* é um algoritmo de consenso usado por criptomoedas como o Ethereum e Bitcoin. O mesmo age do princípio de que para um nó ser adicionado a rede, o mesmo precisa concluir uma prova de trabalho para verificar todas as transações no bloco. Resumidamente o Proof of Work é um sistema que garante segurança e consenso em toda a rede *blockchain*.

- *Peer-to-peer*

Uma rede *peer-to-peer* pode ser entendida como uma rede de computadores onde cada um deles pode atuar como cliente ou como servidor, tendo como objetivo o aumento de disponibilidade do recurso e aumentar a banda de upload do sistema (CRIPTOFACIL, 2020).

- *Testnet*

Testnet é um alternativo a rede *blockchain*, utilizado para teste. Suas moedas são separadas e distintas de outras criptomoedas e não possuem valor. Isso permite com que programadores possam realizar testes envolvendo *blockchain* sem ter que usar criptomoedas reais (BITCOIN.IT, 2020).

- *Infura*

API utilizada para comunicação com a EVM (*Ethereum Virtual Machine* ou Máquina Virtual Ethereum, a mesma será explicada posteriormente), sua arquitetura é feita de forma



a facilitar as maneiras de comunicação com a EVM com uma interface para gerenciamento de recursos fácil de ser compreendida. A mesma foi feita para com que a conexão com a EVM seja centralizada por meio dela para facilitar o desenvolvimento de programas que utilizam a rede Ethereum (INFURA, 2020).

#### 2.4.1 BITCOIN

O bitcoin foi idealizado por Shatoshi Nakamoto em 2008 em artigo acadêmico (NAKAMOTO, 2008). É uma tecnologia criada para permitir pagamentos entre diferentes partes sem a passagem por uma instituição financeira, propondo uma solução com uma rede ponto a ponto que mantem seus arquivos criando *hashes* para ter com isso seu "*proof-of-work*", garantindo com que com isso seus dados não possam ser alterados, a não ser que todos os registros anteriores sejam alterados. Sua estrutura é definida como “uma rede que marca o tempo das transações, colocando-as em uma cadeia contínua no *hash*, formando um registro que não pode ser alterado sem refazer todo o trabalho”.(NUBANK, 2020).

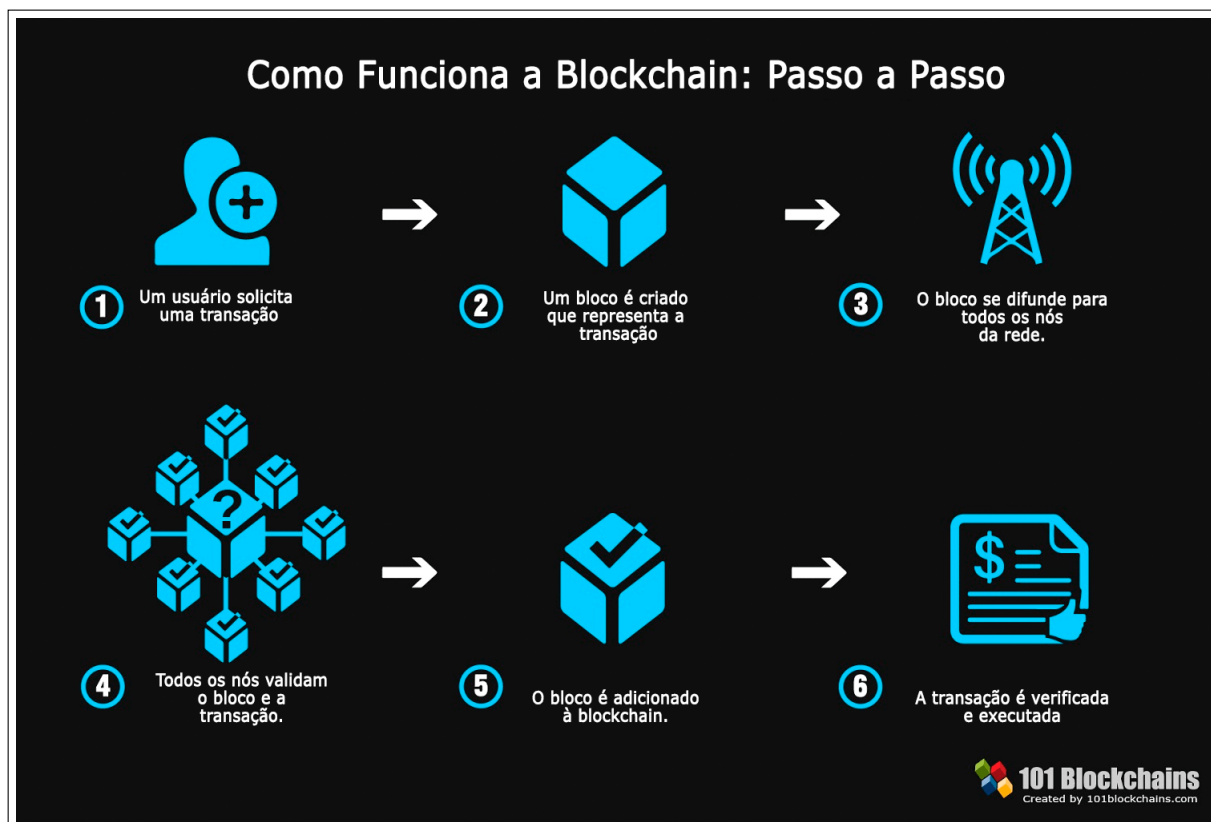
Conforme ilustrado na Figura 4, quando um usuário tenta fazer o registro de uma informação para a rede *blockchain*, essa informação é transformada em um bloco de informações que é difundida na rede para com que outras máquinas conectada a rede verifiquem esse bloco e valide o mesmo para que possa ser inserido a rede *blockchain*, e caso aconteça o sucesso, esse bloco é registrado e a transação é executada.

#### 2.4.2 BLOCOS

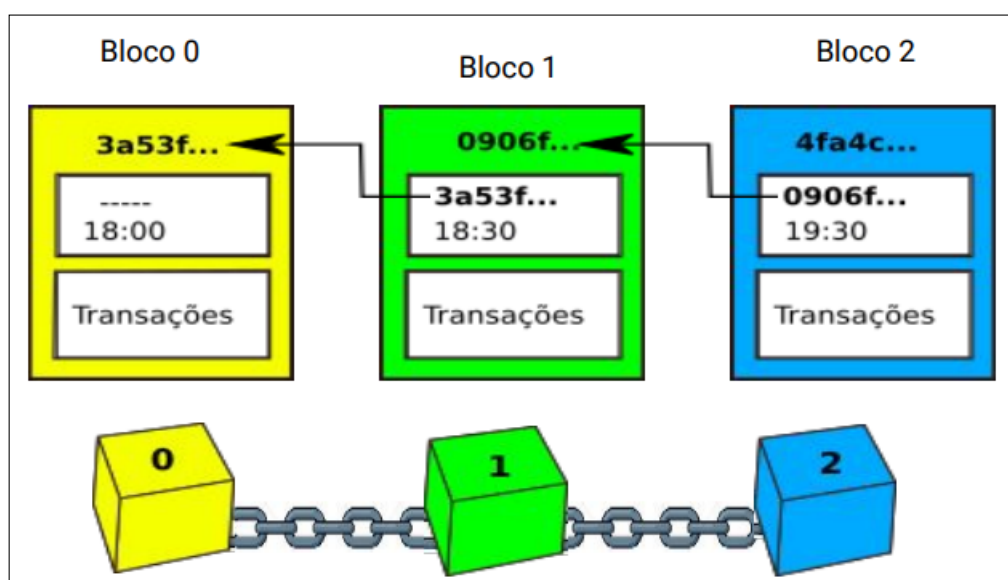
O *Blockchain* implementa uma estrutura que consiste em uma cadeia ordenada e consistente de blocos, esses blocos são implementados em uma rede *peer-to-peer*, ou seja, sempre que um novo nó é criado, ele é enviado a todos os blocos da rede. Cada nó é validado antes se pode ser inserido a rede, e cada um deles possui uma estrutura de forma a precisar de informações do último nó para serem devidamente cadastrados a rede. O nó pode ser dividido em duas partes principais, seu cabeçalho e seus dados. O cabeçalho é a parte responsável por identificar o nó. Para isso, ele precisa do *hash* do código anterior registrado na rede, um identificador único e seu tempo de registro, assim como uma assinatura que é feita para validar e inserir o nó. Para a parte de dado, são armazenadas as informações referentes à transação executada pelo bloco, podendo representar dados como documentos, links, dados sobre transferência entre valores, ou outras informações desejadas. Esses dados são transformados em um código *hash* para seu registro na transação podendo ser descriptografados a partir da recuperação dos seus dados. A Figura 5 ilustra como os registros na rede são representados para esses blocos.

#### 2.4.3 ETHEREUM

Ethereum é uma rede *blockchain* que permite a execução de programas em um ambiente seguro, possuindo a EVM (*Ethereum Virtual Machine*, ou Máquina Virtual Ethereum) que permite

Figura 4 – Funcionamento *Blockchain*

Fonte: (101BLOCKCHAINS, 2020)

Figura 5 – Ilustração dos blocos em uma rede *blockchain*

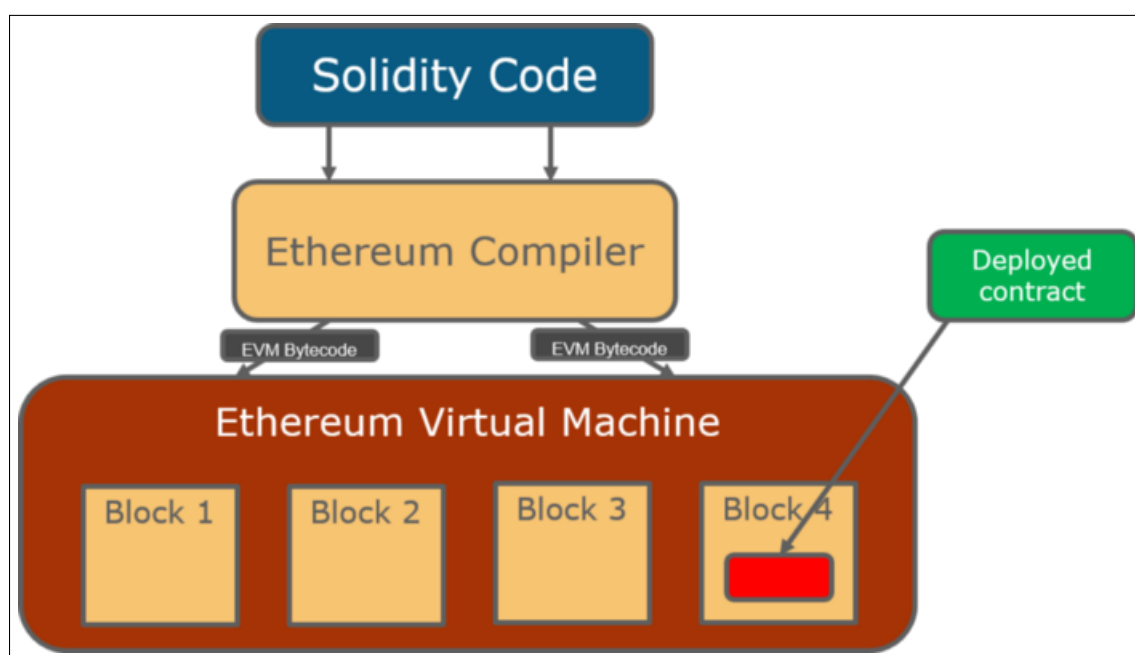
Fonte: (ANGONESE, 2020)

que códigos sejam verificados e executados na rede, garantindo que o resultado será executado de mesma forma para todas as máquinas, a partir de contratos inteligentes. O Ethereum também é responsável por manter a EVM e com isso processar integridade de novos nós e contratos que

são feitos na rede. Essa rede é baseada na criptomoeda Ether, que é usada para suas transações (TRUFFLESUITE, 2020).

A Figura 6 demonstra a forma como esses dados são passados para a rede Ethereum, tendo seu código em Solidity passado pelo compilador da Ethereum e com isso passado os binários com as informações para o cadastro de dados dos blocos, com isso os dados são passados para a EVM e seus registros são cadastrados em um bloco.

**Figura 6 – Estrutura de entrada de dados em Solidity**



Fonte: (GARG, 2020)

#### 2.4.4 CONTRATOS INTELIGENTES

Contratos Inteligentes são contratos digitais que se utilizam de tecnologia para garantir que os acordos firmados serão cumpridos, ou seja, é um código de programação que define regras, obrigações, benefícios e penalidades às devidas partes do contrato. A sua principal diferença de contratos tradicionais é que o mesmo, por ser digital e difundido na rede *blockchain*, não pode ser perdido ou adulterado. Os contratos inteligentes são montados em programação definindo suas cláusulas e consequências, e quando o acordo é finalizado as exigências do contrato se ativam automaticamente, e sua validação é feita pela rede *blockchain* que acompanha os dados e permite a comunicação direta e criptografada entre as partes do contrato. Suas informações inseridas são atualizadas automaticamente e suas execuções não possuem risco de fraude ou alteração, porque o mesmo é imutável, fazendo com que qualquer modificação, mesmo um erro de digitação, gere um novo contrato. Para um documento ser considerado um contrato inteligente ele precisa ter os princípios que são: observabilidade, que é a capacidade de verificar que todas as entidades do contrato cumpriram sua parte; verificabilidade, que é a habilidade de provar a

uma outra entidade que o contrato foi ou não cumprido; e a privacidade, que faz com que apenas os responsáveis possam ter acesso a execução dos processos do contrato (BITCOINTRADE, 2020).

No trabalho apresentado, o contrato inteligente é um código executado na EVM, podendo aceitar e guardar Ether e/ou dados, e usando a lógica programada no contrato, distribuir o ether para as contas ou para os outros contratos inteligentes. Esses contratos inteligentes são escritos em uma linguagem chamada Solidity.

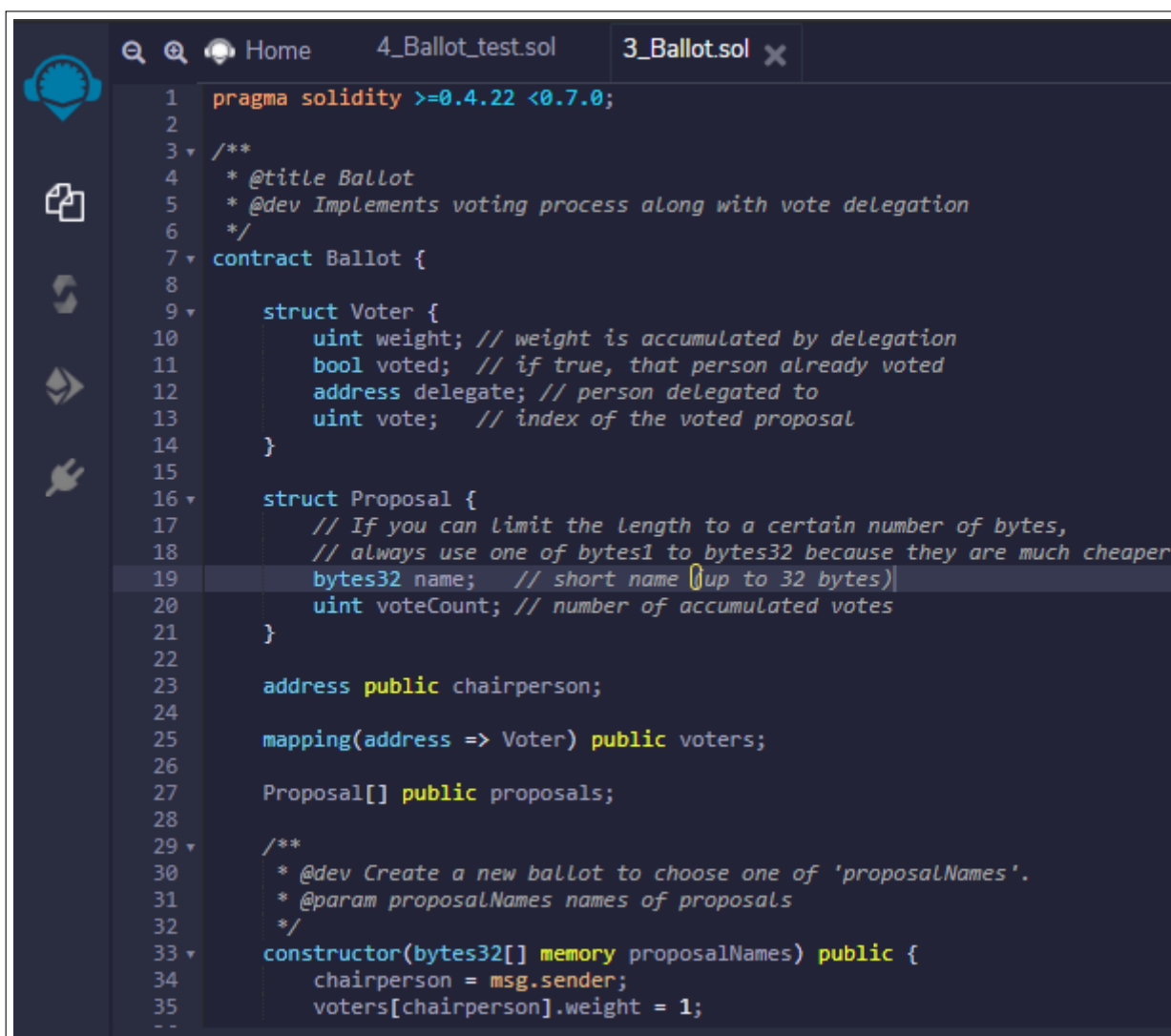
#### 2.4.5 SOLIDITY

Solidity é uma linguagem de alto nível orientada em objetos para a implementação de contratos inteligentes, sendo influenciada por linguagens como C++ e Python e designada para a Ethereum Virtual Machine (EVM). Sua sintaxe é semelhante à sintaxe usada em JavaScript, e o mesmo possui IDEs como o Remix para facilitar seu entendimento e uso para criação de contratos inteligentes (SOLIDITY, 2020).

#### 2.4.6 REMIX IDE

Remix é uma IDE de código aberto para aplicações *web* e *desktop*, a mesma é usada para o desenvolvimento de contratos como o ambiente de testes para aprender e ensinar sobre o uso de Ethereum. Sua IDE é fonte de ferramentas que auxiliam a escrita de contratos em Solidity diretamente do navegador, e possui módulos para teste e criação de contratos inteligentes (REMIX, 2020). Na Figura 7 podemos ver um exemplo de código em Solidity feito na IDE do Remix para registro de votos, delegações de votos e retorno do resultado de uma votação, a partir de contratos inteligentes.

Figura 7 – Exemplo de código em Solidity, na IDE Remix



```

1  pragma solidity >=0.4.22 <0.7.0;
2
3  /**
4   * @title Ballot
5   * @dev Implements voting process along with vote delegation
6   */
7  contract Ballot {
8
9      struct Voter {
10         uint weight; // weight is accumulated by delegation
11         bool voted; // if true, that person already voted
12         address delegate; // person delegated to
13         uint vote; // index of the voted proposal
14     }
15
16     struct Proposal {
17         // If you can limit the length to a certain number of bytes,
18         // always use one of bytes1 to bytes32 because they are much cheaper
19         bytes32 name; // short name (up to 32 bytes)
20         uint voteCount; // number of accumulated votes
21     }
22
23     address public chairperson;
24
25     mapping(address => Voter) public voters;
26
27     Proposal[] public proposals;
28
29     /**
30     * @dev Create a new ballot to choose one of 'proposalNames'.
31     * @param proposalNames names of proposals
32     */
33     constructor(bytes32[] memory proposalNames) public {
34         chairperson = msg.sender;
35         voters[chairperson].weight = 1;
36     }

```

Fonte: Autoria Própria

### 3 TRABALHOS RELACIONADOS

Neste capítulo será apresentada trabalhos relacionados encontrados durante o desenvolvimento deste trabalho.

#### 3.1 *AN AGRI-FOOD SUPPLY CHAIN TRACEABILITY SYSTEM FOR CHINA BASED ON RFID BLOCKCHAIN TECHNOLOGY*

Criado para ajudar com o controle de cadeia de produção de produtos agrícolas na China, a proposta tem como objetivo o uso de sensores e tecnologia *blockchain* para garantir a segurança de consumo desses alimentos. O trabalho propõe principalmente o uso de sensores RFID (Identificação por Radiofrequência), que são sensores conforme a Figura 8 que capturam dados a partir de etiquetas eletrônicas ou RF tags e permitem o controle e visualização dos mesmos. A partir destes sensores seria feita a confirmação dos dados e os mesmos passariam a ser registrados em uma rede *blockchain* (TIAN, 2016).

Assim como neste trabalho, o trabalho relacionado consiste em um registro de dados em cadeias de uma rede *blockchain* para com isso ter sua rastreabilidade, usando até mesmo de dados registrados por sensores, porém o mesmo foi desenvolvido para o uso do controle de uma produção agrícola somente.

#### 3.2 *HOW BLOCKCHAIN IMPROVES THE SUPPLY CHAIN: CASE STUDY ALIMENTARY SUPPLY CHAIN*

Proposta de modelo de cadeias de produção via *blockchain*, com isso habilitando um economia circular e eliminando alguns problemas encontrados em cadeias de produção. Sua arquitetura pode ser vista na Figura 9, que mostra que cada uma de suas camadas se comunica a partir dos contratos inteligentes (CASADO-VARA et al., 2018).

Embora a proposta seja semelhante à proposta levantada neste trabalho, a mesma não mostrava uma implementação de como seria feita, e sua estrutura de dados é baseada em um modelo circular para melhoria de registro de dados em qualquer cadeia de produção.

#### 3.3 QUIPOAGRO

Produto criado para rastreabilidade de cadeias de produção de café, o mesmo possui o armazenamento de dados de etapas de produção de café e faz o registro das mesmas em rede *blockchain*, conforme a Figura 10 (QUIPO, 2020).

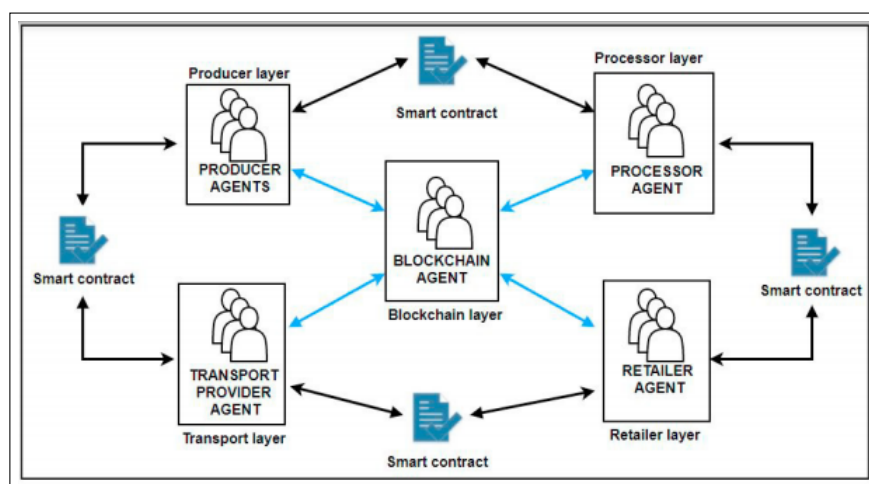
O produto apresenta um funcionamento que se assemelha a forma que é desejada para o protótipo desenvolvido durante este trabalho, porém com a principal diferença sendo a permissão

**Figura 8 – Exemplo de sensor e RF tag, usados em um microcontrolador Arduino**



Fonte: (MICROCONTROLANDOS, 2020)

**Figura 9 – Arquitetura Proposta pelo trabalho relacionado**



Fonte: (CASADO-VARA et al., 2018)

do envio de arquivos somente em um formato específico, enquanto o protótipo desenvolvido durante neste trabalho visa a permissão de dados para cadeias generalizadas.

**Figura 10 – Interface QuipoAgro**



Fonte: [agro.quipo.io/bt/1](http://agro.quipo.io/bt/1)

**Figura 11 – Comparação entre trabalhos relacionados**

	An Agri-food Supply Chain Traceability System for China Based on RFID & Blockchain Technology	How blockchain improves the supply chain: case study alimentary supply chain	QuipoAgro	Uso de Blockchain para registros de dados de cadeias de produção
Permite a recuperação dos dados	<b>X</b>		<b>X</b>	<b>X</b>
Uso em cadeias de produção generalizadas		<b>X</b>	<b>X</b>	<b>X</b>
Permissão de cadastro de dados gerais para o registro na cadeia de produção				<b>X</b>

Fonte: Autoria Própria



## 4 METODOLOGIA E DESENVOLVIMENTO

Neste capítulo será apresentada a metodologia que foi utilizada no desenvolvimento deste trabalho.

### 4.1 IMPLEMENTAÇÃO UTILIZADA

Para validação da ideia proposta, foi decidido a criação de uma aplicação *web* baseada em Django para a armazenar os dados cadastrados pelo cliente para permitir a comunicação com a rede *Blockchain* posteriormente.

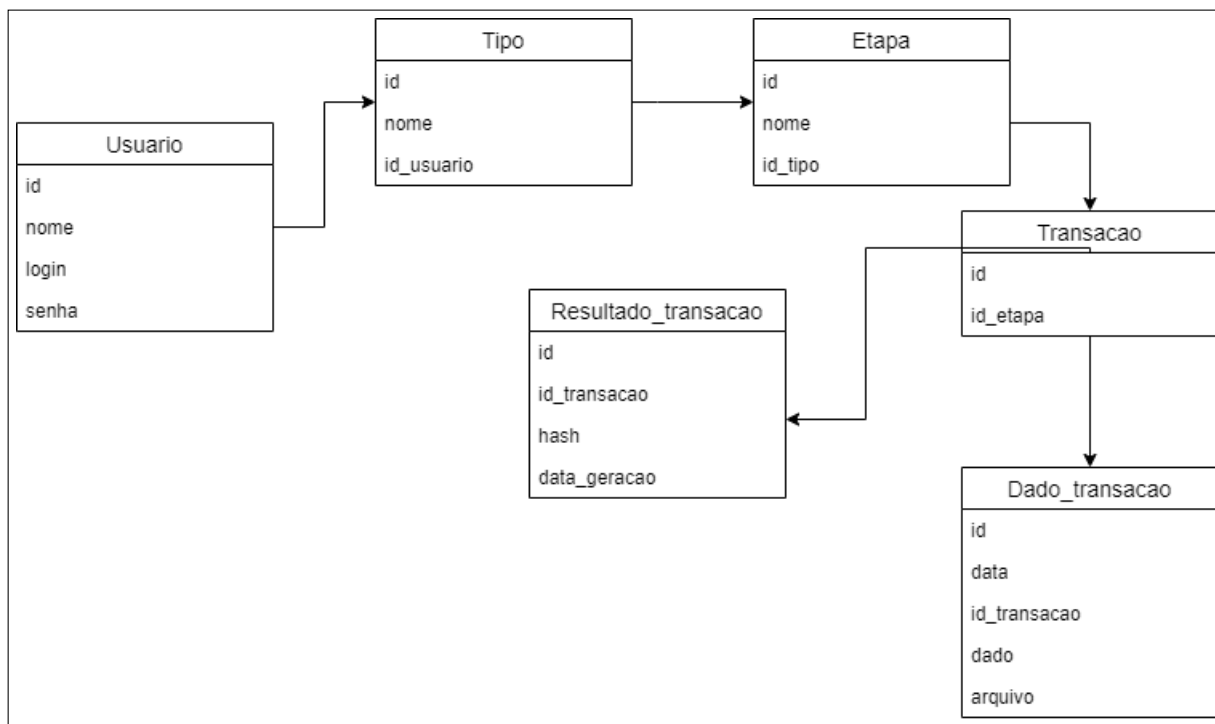
### 4.2 BANCO DE DADOS

No começo do projeto foi pensado no uso de um banco NOSQL, ou seja, um banco não relacional de dados que são modelados de forma diferente dos modelos relacionais que é mais comum de ser visto aplicações que o usam, para melhor generalização de dados. Porém devido a vários problemas encontrados para implementação com o Django. Após isso foi verificado que seria possível atender uma generalização com o Postgres, um banco com uma melhor documentação e com uma base maior de códigos, além de maior conhecimento sobre funcionamento e facilidade de uso, a ideia de uso de um banco NOSQL acabou sendo descartada e foi decidido o uso do banco PostgreSQL.

#### 4.2.1 ESTRUTURA DE DADOS

A montagem da estrutura de dados foi planejada de forma que consiga se gerar uma generalidade na estrutura de dados para registro de uma cadeia de produção, conforme a Figura 12 mostra, a estrutura de dados permite que usuários possam ter várias etapas de cadeias de produção (como plantio, armazenamento, transporte, etc) enquanto cada uma destas pode possuir diversas transações também armazenando dados para o registro em *blockchain* (como documentos, medições de sensores, etc).

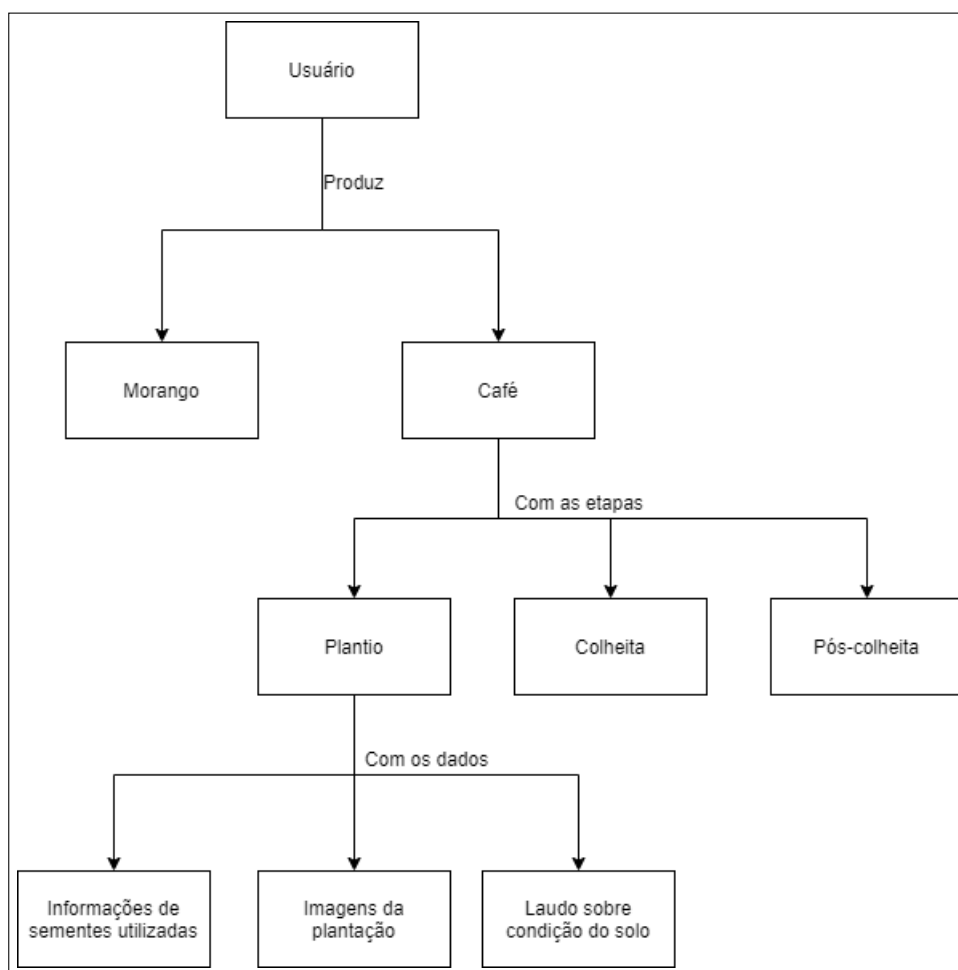
Dessa forma, conforme a estrutura demonstra, os usuários irão possuir seu cadastro e para cada um deles, ele pode ter  $n$  tipos diferentes de cadeias de produção (como café, soja, medicamentos, etc), e a partir desses tipos ele pode descrever cada etapa da linha de produção. Usando o processo de uma cadeia de produção de café por exemplo, teríamos etapas como a gestão da propriedade, colheita, pós-colheita, armazenamento, transporte, entre outros. A partir dessas etapas, podemos ter ainda  $n$  transações com dados referentes à cada etapa do processo, pegando as informações de plantio do café como exemplo novamente, teríamos dados como informações da propriedade usada para plantio, agroquímicos utilizados e suas informações de armazenamento, lote usado para plantação, informações referentes às sementes utilizadas,

**Figura 12 – Estrutura de dados utilizada****Fonte: Autoria Própria**

análise do solo, práticas para controle de pragas utilizadas. Cada uma dessas transações ainda pode possuir um texto ou arquivo que foi utilizado para seu cadastro.

Dessa forma, utilizando a estrutura de dados apresentada, conseguimos com isso gerar uma estrutura de dados que pode ser usada para a cadeia de produção de café, assim como outros tipos de cadeias de produção a partir de suas etapas e dos seus dados que precisam ser armazenados, com isso garantindo uma estrutura generalizada para o registro de informações para cadeias de produções diversas, tendo com isso escalabilidade para uso por diversas formas diferentes.

Seguindo por esse exemplo da cadeia de produção de café teríamos os usuários como os clientes que realizam o cadastro das informações, cada um deles poderia fazer o cadastro de quaisquer tipos de cadeias de produção e as mesmas também podem possuir quaisquer tipos de etapas de produção. Para os dados de transação para cada etapa, pode ser preenchido na tabela de transação arquivos ou texto referente a esse dado que será usado na transação, para ao final realizar a comunicação com a rede Ethereum e gerar no resultado da transação os dados referentes à transação que foi passada para a rede Ethereum, para com isso permitir sua consulta posteriormente. Na Figura 13 podemos ver um exemplo de uma cadeia de produção com a estrutura de dados proposta.

**Figura 13 – Fluxo de exemplo de uma cadeia de produção**

**Fonte: Autoria Própria**

### 4.3 GERAÇÃO DO ARQUIVO *HASH*

Após o registro dos dados no banco de dados, é necessário uma forma de se registrar os dados em *blockchain*. Para isso foi gerado em Solidity uma comunicação de forma para, assim do conjunto de dados passados, gerar um *hash* que será inserido na rede *Blockchain* e depois assim dele possa ser recuperado os dados que foram registrados. Para isso foi adaptada uma versão de código de uma Testnet chamada DAI (UNIVERSITY, 2020) para com ela poder realizar as operações necessárias de geração de um contrato inteligente permitindo que os dados fossem armazenados no bloco passado para a rede Ethereum.

### 4.4 DESENVOLVIMENTO

Foi feito um programa usando Django para o registro de dados que serão usados no cadastro e para o envio para a rede Ethereum, que é a rede *blockchain* selecionada para uso neste trabalho. Para isso foi criada uma interface simples para a exibição e cadastro desses dados, conforme a Figura 14

**Figura 14 – Tela de cadastro de usuários**

**Página de Inserção de Usuário**

- [Página Inicial](#)
- [Cadastrar Usuário](#)
- [Lista de Usuários](#)

Nome:  Login:  Senha:

**Fonte: Autoria Própria**

Com esses dados devidamente registrados no banco, podemos começar o envio dos mesmos para a rede Ethereum e com isso garantir o registro desses dados e o acesso a eles a partir do *hash* gerado. Um exemplo de acesso a um dado pode ser observado na Figura 15.

**Figura 15 – Exemplo de dados no banco**

Query Editor Query History

```

1 SELECT "id", id_etapa, data, dado
2 FROM public.myproject_transacao
3 where "id" = 'f0a002f3-9208-4dc6-8233-a27f62ad1032';

```

	id [PK] uuid	id_etapa uuid	data date	dado text
1	f0a002f3-9208-4dc6-8233-a27f62ad1032	860966ec-5efd-4961-9cbc-40ce70a3bdb9	2020-11-03	{ "Temperatura_media": 26, "Temperatura_minima": 20, "Temperatura_maxima": 30 }

**Fonte: Autoria Própria**

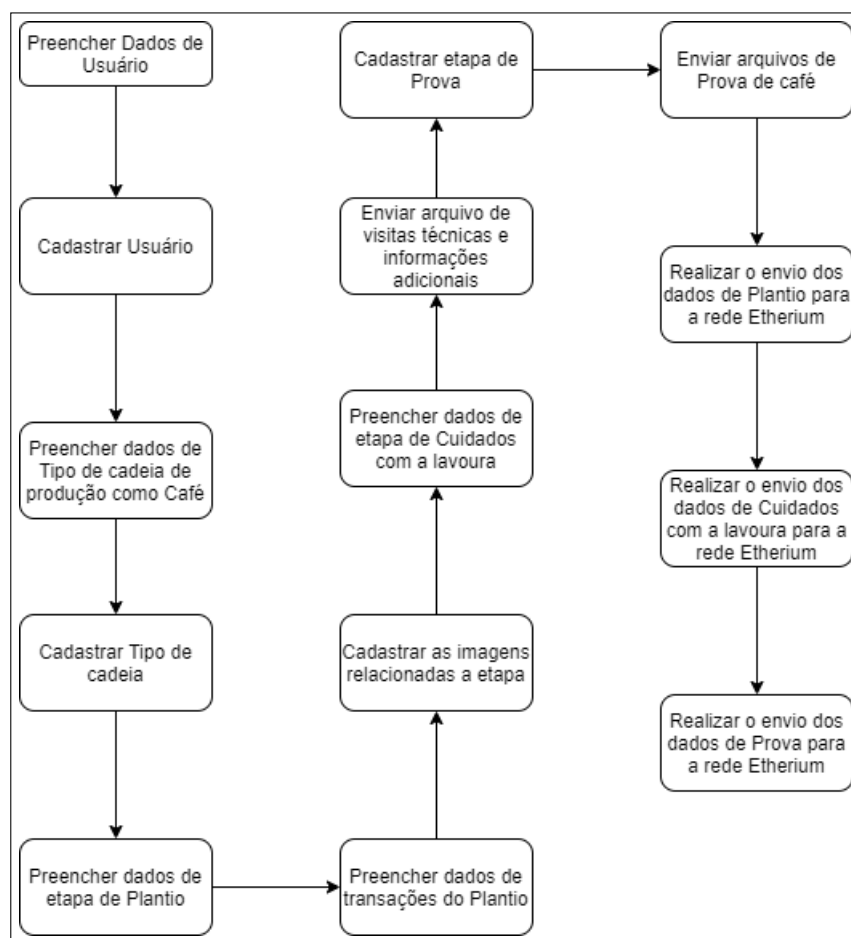
Com esses dados podemos gerar o *hash* para ser enviado para a rede Ethereum, conforme a Figura 16 que exhibe em verde as informações dessa transação que foram passadas para a rede Ethereum e logo abaixo o *hash* equivalente gerado dos dados da transação, assim que esses dados foram cadastrados na rede Ethereum. Com isso podemos fazer o envio desses dados para rede Ethereum para seu armazenamento e ter sua confiabilidade garantida pelos blocos da rede. O dado dessa transação também pode ser visualizado pela rede para ser descryptografado a partir da transação, para o retorno ao valor original, conforme a Figura 17 onde, a partir do bloco enviado,



## 5 EXPERIMENTOS E RESULTADOS

Para validação da ferramenta, foi utilizado o cenário de rastreabilidade da cultura do café. Para escolha das etapas do processo de produção foi adotada a metodologia proposta pelo Currículo de Sustentabilidade do Café (COOPERCITRUS, 2020), sobre o qual foi escolhido um número reduzido de etapas para facilitar a clareza e validação da proposta. Os mesmos dados são baseados em um caso de estudo do trabalho relacionado QuipoAgro. Para isso foi criado um fluxograma da forma de cadastro que pode ser visto na Figura 18 e com isso podemos detalhar o fluxograma e ver o funcionamento do cadastro dos dados na rede Ethereum.

**Figura 18 – Fluxograma do cadastro de dados**



**Fonte: Autoria Própria**

- **Preencher dados de usuário**

O primeiro passo necessário é o envio dos dados de usuário para a aplicação. Para isso foi usado uma interface simples de cadastro de usuário conforme podemos ver na Figura 19, que é semelhante às outras etapas de cadastro que veremos mais a frente.

- **Cadastrar e listar usuário**

**Figura 19 – Cadastro de usuário**
**Fonte: Autoria Própria**

Com os dados cadastrados, podemos realizar o envio desses dados de usuário e logo após visualizar os mesmos caso estejam corretos, conforme a Figura 20.

**Figura 20 – Lista de usuários**

Nome	Login	Senha	Atualizar	Excluir
teste 2	teste2@gmail.com	1234567	<a href="#">Atualizar</a>	<a href="#">Excluir</a>
Douglas	douglaso.escola@gmail.com	123456789	<a href="#">Atualizar</a>	<a href="#">Excluir</a>
teste	teste	teste	<a href="#">Atualizar</a>	<a href="#">Excluir</a>
Usuario de Teste	teste	teste	<a href="#">Atualizar</a>	<a href="#">Excluir</a>

**Fonte: Autoria Própria**

- Cadastro de outras informações das tabelas

Da mesma forma que o cadastro de usuário é feito, é feito o cadastro dos dados de qual tipo de cadeia de produção foi selecionada, que nesse caso foi selecionada o Café, suas etapas de produção, sendo Plantio, Cuidados com a Lavoura e Prova para com isso começarmos os cadastros dos dados de transações, que são as informações essenciais para o registro durante a cadeia de produção. A Figura 21 ilustra como esse cadastro é feito para essas etapas

- Visualização das etapas e tipos

E assim como o cadastro de usuários, esses dados podem ser visualizados conforme mostra a Figura 22 e com isso temos os dados necessários para começar os dados da transação que serão necessários para o envio para a rede Ethereum.

- Cadastro de dados para transação com a rede Ethereum

Agora com os dados necessários para especificar as informações dessa cadeia de produção devidamente cadastrados, podemos começar o cadastro dos dados que enviaremos para a rede Ethereum e com isso começar seus registros em *blockchain*. Conforme podemos

**Figura 21 – Cadastro de tipos de cadeia de produção e suas etapas**

### Página de Inserção de Tipo

- [Página Inicial](#)
- [Cadastrar Tipo](#)
- [Lista de Tipos](#)

Id usuario:  Nome:

---

### Página de Inserção de Etapa

- [Página Inicial](#)
- [Cadastrar Etapa](#)
- [Lista de Etapas](#)

Id tipo:  Nome:

---

### Página de Inserção de Etapa

- [Página Inicial](#)
- [Cadastrar Etapa](#)
- [Lista de Etapas](#)

Id tipo:  Nome:

Fonte: Autoria Própria

**Figura 22 – Lista de tipos de cadeia de produção e suas etapas**

### Lista de Tipos

- [Página Inicial](#)
- [Cadastrar Tipo](#)
- [Lista de Tipos](#)

Nome		
Produção de morango	<a href="#">Atualizar</a>	<a href="#">Excluir</a>
Produção de café	<a href="#">Atualizar</a>	<a href="#">Excluir</a>

---

### Lista de Etapas

- [Página Inicial](#)
- [Cadastrar Etapa](#)
- [Lista de Etapas](#)

Nome		
Armazenamento	<a href="#">Atualizar</a>	<a href="#">Excluir</a>
Plantio	<a href="#">Atualizar</a>	<a href="#">Excluir</a>
Cuidados com a Lavoura	<a href="#">Atualizar</a>	<a href="#">Excluir</a>
Prova	<a href="#">Atualizar</a>	<a href="#">Excluir</a>

Fonte: Autoria Própria

ver na Figura 23, os dados são cadastrados de forma a permitir um texto qualquer e um arquivo qualquer anexado a ele, podendo assim permitir o cadastro de dados genéricos, desde arquivos de vídeo, áudios, imagens, relatórios, etc ou somente textos de medições encontradas durante a lavoura. E com a Figura 24, podemos ver alguns desses tipos de dados anexados que estão sendo cadastrados, como imagens e relatórios do plantio do café, dentre as outras etapas.

- Lista de dados para transação com a rede Ethereum



**Figura 23 – Dados cadastrados para o envio das transações para a rede Ethereum**

**Página de Inserção de Transação**

- [Página Inicial](#)
- [Cadastrar Transação](#)
- [Lista de Transações](#)

Id etapa:  Data:  Dado:  Arquivo:

---

**Página de Inserção de Transação**

- [Página Inicial](#)
- [Cadastrar Transação](#)
- [Lista de Transações](#)

Id etapa:  Data:  Dado:  Arquivo:

---

**Página de Inserção de Transação**

- [Página Inicial](#)
- [Cadastrar Transação](#)
- [Lista de Transações](#)

Id etapa:  Data:  Dado:  Arquivo:

**Fonte: Autoria Própria**

**Figura 24 – Exemplos de arquivos cadastrados para o envio das transações para a rede Ethereum**



**Fonte: agro.quipo.io/bt/1**

Com esses dados gerados, podemos verificar o seu cadastro no banco e na aplicação conforme as Figuras 25 e 26 para verificar sua consistência para então com isso começar a

enviar essas transações para a rede *blockchain*.

**Figura 25 – Lista de dados cadastrados para o envio das transações para a rede Ethereum**

Lista de Transações			
<ul style="list-style-type: none"> <li><a href="#">Página Inicial</a></li> <li><a href="#">Cadastrar Transação</a></li> <li><a href="#">Lista de Transações</a></li> </ul>			
Data	Dado	Arquivo	
Nov. 15, 2020	Formação da Lavoura mais nova da propriedade : 8 mil pés de café adensado - 2018	documents/2020/11/16/Plantio1.jpeg	<a href="#">Atualizar</a> <a href="#">Excluir</a> <a href="#">Enviar</a>
Nov. 15, 2020	Formação da Lavoura mais nova da propriedade: 8 mil pés de café adensado - 2018.	documents/2020/11/16/Plantio2.jpeg	<a href="#">Atualizar</a> <a href="#">Excluir</a> <a href="#">Enviar</a>
Nov. 15, 2020	Nova lavoura	documents/2020/11/16/Plantio3.jpeg	<a href="#">Atualizar</a> <a href="#">Excluir</a> <a href="#">Enviar</a>
Nov. 15, 2020	Relatório de Visita Técnica	documents/2020/11/16/sabor_RELATORIO_DE_VISITA_TÉCNICA.pdf	<a href="#">Atualizar</a> <a href="#">Excluir</a> <a href="#">Enviar</a>
Nov. 15, 2020	Detalhe da Aducação		

Fonte: Autoria Própria

**Figura 26 – Lista de dados cadastrados para o envio das transações para a rede Ethereum no banco de dados**

<pre> 1 SELECT id, id_etapa, data, dado, arquivo 2 FROM public.myproject_transacao;</pre>						
Data	Output	Explain	Messages	Notifications		
	id [PK] uuid		id_etapa uuid	data date	dado text	arquivo text
1	122c62cb-36ef-4b31-bf87-00fb82406135		c373b8f2-1de6-4f0d-a086-bf58367b16c5	2020-11-15	Formação da Lavoura mais nova da...	documents/2020/11/16/Plantio1.jpeg
2	67c500cd-610e-4e61-be32-4fe1e21be382		c373b8f2-1de6-4f0d-a086-bf58367b16c5	2020-11-15	Formação da Lavoura mais nova da...	documents/2020/11/16/Plantio2.jpeg
3	96ac497b-a8c8-401d-89cc-9c4630a114f9		c373b8f2-1de6-4f0d-a086-bf58367b16c5	2020-11-15	Nova lavoura	documents/2020/11/16/Plantio3.jpeg
4	89e173de-77f8-47f5-930d-877879957ff6		2d3b4e7d-8fb3-45b8-a598-94ba5625838a	2020-11-15	Relatório de Visita Técnica	documents/2020/11/16/sabor_RELATORIO_DE_VISITA_TÉCNICA.pdf
5	9ce5676a-9988-431f-86e9-b3b0ac044e9e		2d3b4e7d-8fb3-45b8-a598-94ba5625838a	2020-11-15	Detalhe da Aducação diretamente n...	documents/2020/11/16/Cuidados_com_a_lavoura_1.jpeg
6	aa034146-0bb6-465a-af6c-c646fce4b0e3		2d3b4e7d-8fb3-45b8-a598-94ba5625838a	2020-11-15	Detalhe de cartaz distribuído pela P...	documents/2020/11/16/Cuidados_com_a_lavoura_2.jpeg
7	c58d0bab-033e-41b6-9384-9920e2c51424		ebaed0ef-8b4a-4eb8-b90f-09781e395635	2020-11-15	Laudo de Prova de café	documents/2020/11/16/Laudo_Prova_de_café.pdf

Fonte: Autoria Própria

- Envio de dados para a rede Ethereum

Após isso, é feito o envio desses dados para a rede Ethereum. Utilizando o botão "Enviar" visto na Figura 25 podemos abrir uma nova tela para confirmar o envio desses

dados de transações para a rede Ethereum e registrar o *hash* do bloco de transação que foi enviado, para consultas futuras caso necessário. Podemos ver na Figura 27 a tela usada para o envio dos dados e na Figura 28 podemos ver também como esses dados estão sendo cadastrados no banco de dados, com a informação do bloco da transação sendo devidamente armazenada no banco de dados.

**Figura 27 – Página de envio de transações para a rede Ethereum**

**Página de Envio de Transação para a rede Ethereum**

- [Página Inicial](#)
- [Cadastrar Transação](#)
- [Lista de Transações](#)

Id transacao: 
Data:

Fonte: Autoria Própria

**Figura 28 – Dados do bloco da rede Ethereum no banco de dados**

```

1 SELECT id, id_transacao, hash, data
2 FROM public.myproject_resultado_transacao;
3

```

Data Output	Explain	Messages	Notifications
id [PK] uuid	id_transacao uuid	hash text	data date
e4bc4222-c135-4220-864f-d30de34f719f	122c62cb-36ef-4b31-bf87-00fb82406135	b'a1211226d0457011e9aa5dedd9f738e58b3dc8e01d4945ec0a45c1de61345a24'	2020-11-16
a787ffd3-89c9-4aa6-b802-afb09597a0c5	67c500cd-610e-4e61-be32-4fe1e21be382	b'97890cd4f333bd250de469863e73e55b3c058e4b9fb5f5c257e5cc1a6b547611'	2020-11-16
a284d829-4cfc-4cb3-9d91-91abd27e362d	96ac497b-a8c8-401d-89cc-9c4630a114f9	b'a02a355cd29b411a1021a68aa777a9f428d8e27954303281f0490390aea51854'	2020-11-16
569ee32f-f1de-41e9-bc1e-2c2b9a584ad7	89e173de-77f8-4f75-930d-877879957ff6	b'48bc6f6eaa704df74835e40744a6199b3b7e4a61f77ce0038351f2189c4e22be'	2020-11-16
dac6a414-6318-4b69-929c-e872ac03898c	9ce5676a-9988-431f-86e9-b3b0ac044e9e	b'72dbada3644f9c86ad1d02b1a5c6e78c135f8b3c426de146c8bb654d1ae17906'	2020-11-16
08ce4936-21aa-46dd-adea-a7de5091adcf	aa034146-0bb6-465a-af6c-c64f6ce4b0e3	b'9576a598bd5e43011043fca4e0393e78f1fde66f9a70096cd0159d4336b455e9'	2020-11-16
657a7a3c-b4b7-4388-b06e-d2130be0b466	c58d0bab-033e-41b6-9384-9920e2c51424	b'f3a307a901a2204bb587f459af847377256855d235bfff3af5c876eac7636f4fb'	2020-11-16

Fonte: Autoria Própria

- Visualização dos dados na rede Ethereum

Por fim com os dados devidamente cadastrados e registrados na rede Ethereum, podemos validar os mesmos consultando na rede Ethereum a partir do bloco informado, e verificar que os dados foram devidamente cadastrados. Conforme mostra a Figura 29, os dados foram cadastrados com sucesso para a primeira imagem de plantio passada no envio para a rede Ethereum, e com ela conseguimos verificar todos os dados que foram cadastrados e que agora são imutáveis para a rede *blockchain*.

Ao final de todas essas etapas podemos validar que é possível a criação de uma aplicação que registra dados de uma cadeia de produção na rede Ethereum conforme a estrutura de dados proposta no trabalho, garantindo com isso a imutabilidade do dado a partir do momento que sua transação é enviada para a rede *blockchain*.

**Figura 29 – Dados do bloco na rede Ethereum**

Overview State	
Transaction Hash:	0xa1211226d0457011e9aa5dedd9f738e58b3dc8e01d4945ec0a45c1de61345a24
Status:	Success
Block:	7560355 3 Block Confirmations
Timestamp:	41 secs ago (Nov-16-2020 09:57:54 PM +UTC)
From:	0xc79ecf81f950b033673daa00c3e1fa15d92b2d71
To:	0xc79ecf81f950b033673daa00c3e1fa15d92b2d71
Value:	0 Ether (\$0.00)
Transaction Fee:	0.000028924 Ether (\$0.000000)
Gas Price:	0.000000001 Ether (1 Gwei)
Gas Limit:	70,000
Gas Used by Transaction:	28,924 (41.32%)
Nonce	22
Input Data:	<pre>V,Ç\$ÇBİBÜP°3g=²ÄäÜBÜ+-qB`B.{ 'id_usuario': UUID('45ae82c6-5e49-4317-a95d-1bdeb9c98ba0'), 'id_tipo': UUID('9c450639-e1f4-46ae-8a32-9806bf8e35fe'), 'id_etapa': UUID('c373b8f2-1de6-4f0d-a086-bf58367b16c5'), 'dado': { 'id_transacao': UUID('122c62cb-36ef-4b31-bf87-00fb82406135'), 'data': datetime.date(2020, 11, 15), 'dado': 'FormaãSãfo da Lavoura mais nova da propriedade : 8 mil nães de café@ adensado - 2018'. 'arquivo':</pre>

**Fonte: Autoria Própria**

## 6 CONCLUSÃO

Neste trabalho foi apresentada a implementação de uma ferramenta *Web* para registro, de dados da cadeia de produção de produtos em rede *blockchain*. Foi proposta uma estrutura de dados genérica de forma a permitir com que essa implementação possa ser utilizada por distintas formas de tipos de cadeias de produção, para com isso garantir a segurança e imutabilidade dos dados de tais cadeias ao fim de seus registros.

Foi descrito um experimento simulando a inserção dos dados de uma cultura de café, cuja definição das etapas se baseou no CSC, validando a proposta apresentada.

## 7 SUGESTÕES DE TRABALHOS FUTUROS

Implementação de um programa por parte do consumidor para recuperação dos dados registrados durante este trabalho, para que o acesso às devidas informações das cadeias de produção seja público para qualquer usuário, sem a necessidade da consulta pela interface da rede *blockchain* utilizada.

Implementação de uma interface para o protótipo utilizado para a validação do trabalho.

Criar suporte para recepção de dados a partir de sensores IoT.

## REFERÊNCIAS

- 101BLOCKCHAINS. *101Blockchains*. 2020. Disponível em: <<https://101blockchains.com/pt/tecnologia-blockchain-guia>>.
- ANGONESE, A. **Inteligência Artificial, Blockchain e IOT: Possíveis cenários e desafios de aplicações**. 2020. Apresentação SNCT UNIFESO.
- BITCOIN.IT. **TestNet**. 2020. Disponível em: <<https://en.bitcoin.it/wiki/Testnet>>.
- BITCOINTRADE. **Smart Contracts: entenda o que são e como funcionam**. 2020. Disponível em: <<https://blog.bitcointrade.com.br/smart-contract/>>.
- BRASIL, G. **Você sabe o que é rastreabilidade?** 2020. Disponível em: <<https://blog.gs1br.org/rastreabilidade/>>.
- CASADO-VARA, R. et al. How blockchain improves the supply chain: Case study alimentary supply chain. **Procedia computer science**, Elsevier, v. 134, p. 393–398, 2018.
- COOPERCITRUS. **Currículo de Sustentabilidade do Café**. 2020. Disponível em: <<http://www.coopercitrus.com.br/upload/cafe-4C/Curriculo-de-Sustentabilidade-do-Cafe.pdf>>.
- CRIPTOFACIL. **O que é e como funciona o Proof of Work?** 2020. Disponível em: <<https://www.criptofacil.com/o-que-e-e-como-funciona-o-proof-of-work/>>.
- DJANGO. **Django Overview**. 2020. Disponível em: <<https://www.djangoproject.com/>>.
- GARG, A. **Ethereum Tutorial**:. 2020. Disponível em: <[https://medium.com/@abhinav.garg\\_90821/ethereum-tutorial-ba0e89cde09a](https://medium.com/@abhinav.garg_90821/ethereum-tutorial-ba0e89cde09a)>.
- INFURA. **Infura**. 2020. Disponível em: <<https://infura.io/>>.
- LEÃO, T. **Garantia da qualidade: o que é e o que você está perdendo se não tiver**. 2020. Disponível em: <<https://www.nomus.com.br/blog-industrial/garantia-da-qualidade/>>.
- MICROCONTROLANDOS. **Sensor RFID MFRC32**. 2020. Disponível em: <<http://microcontrolandos.blogspot.com/2014/02/pic-rfid-mfrc522.html>>.
- NAKAMOTO, S. Re: Bitcoin p2p e-cash paper. **The Cryptography Mailing List**, 2008.
- NUBANK. **O que é blockchain?** 2020. Disponível em: <<https://blog.nubank.com.br/o-que-e-blockchain/>>.
- PEDERNEIRAS, G. **Segurança da Informação na indústria 4.0**. 2020. Disponível em: <<https://www.industria40.ind.br/artigo/18129-seguranca-da-informacao-na-industria-40>>.
- POSTGRESQL. **PostgreSQL Overview**. 2020. Disponível em: <<https://www.postgresql.org/docs/9.6/plpgsql-overview.html>>.
- PYTHON. **What is Python? Executive Summary**. 2020. Disponível em: <<https://www.python.org/>>.
- QUIPO. **QuipoAgro**. 2020. Disponível em: <<https://www.quipo.io/quipoagro>>.
- REMIX. **Remix Overview**. 2020. Disponível em: <<https://remix-ide.readthedocs.io/en/latest/>>.

ROMANOWSKI, J. **Which Major Companies Use PostgreSQL? What Do They Use It for?** 2020. Disponível em: <<https://learnsql.com/blog/companies-that-use-postgresql-in-business/>>.

SOLIDITY. **Solidity Documentation**. 2020. Disponível em: <<https://solidity.readthedocs.io/en/v0.5.3/>>.

TIAN, F. An agri-food supply chain traceability system for china based on rfid & blockchain technology. In: IEEE. **2016 13th international conference on service systems and service management (ICSSSM)**. [S.l.], 2016. p. 1–6.

TRUFFLESUITE. **Ethereum Overview**. 2020. Disponível em: <<https://www.trufflesuite.com/tutorials/ethereum-overview>>.

UNIVERSITY, D. **Web3.py Intro**. 2020. Disponível em: <<https://www.dappuniversity.com/articles/web3-py-intro>>.