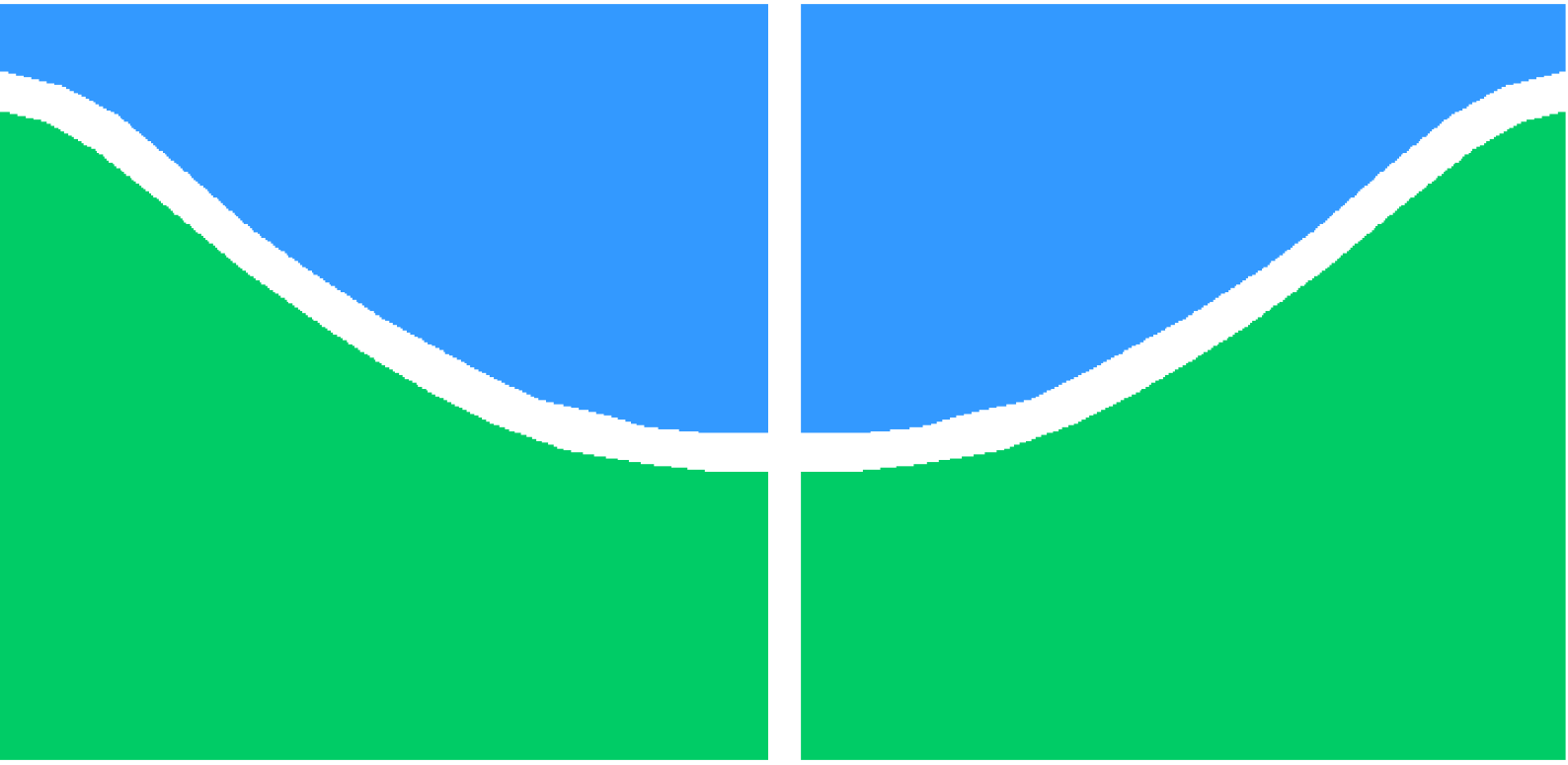

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de Software

Tecnologia promovendo solução para votações

Autor: Gustavo Moreira Nascimento Araujo
Orientador: Dr. Vandor Roberto Vilardi Rissoli

Brasília, DF
2019

Gustavo Moreira Nascimento Araujo

Tecnologia promovendo solução para votações

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do título de Bacharel em Engenharia de Software .

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Dr. Vandor Roberto Vilardi Rissoli

Brasília, DF

2019

Gustavo Moreira Nascimento Araujo
Tecnologia promovendo solução para votações/ Gustavo Moreira Nascimento Araujo. – Brasília, DF, 2019-
86 p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. Vandor Roberto Vilardi Rissoli

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2019.

1. Blockchain. 2. Votações. I. Dr. Vandor Roberto Vilardi Rissoli. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Tecnologia promovendo solução para votações

CDU 02:141:005.6

Gustavo Moreira Nascimento Araujo

Tecnologia promovendo solução para votações

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do título de Bacharel em Engenharia de Software .

Trabalho aprovado. Brasília, DF, :

Dr. Vandor Roberto Vilardi Rissoli
Orientador

Prof. MSc. Ricardo Ajax Dias Kosloski
Convidado 1

Prof. MSc. Giovanni Almeida Santos
Convidado 2

Brasília, DF
2019

Agradecimentos

Agradeço primeiramente à minha mãe, por todo o apoio e atenção durante a minha graduação e por ser um exemplo ímpar de ser humano, tanto no âmbito pessoal, quanto no profissional. Você é a minha maior inspiração.

Agradeço ao meu professor orientador, Dr. Vandor Rissoli, por me guiar nesta reta final do curso, oferecendo-me grandes oportunidades de estudo para que eu me torne um Engenheiro de Software.

Agradeço também aos meus amigos que fiz durante a graduação, em especial Michel Alba, Luan Noleto, Fabio Bassi, Sergio Daher, Diego Bony, João Guilherme Silva, João Guilherme Araruna, Nilton Cesar Araruna, Ruan Donato e Gabriel Araujo, por sempre me incentivarem durante a graduação, compartilhar momentos de alegria e compartilhar conhecimentos no decorrer dessa jornada.

Por fim, agradeço a todos que participaram da minha vida durante o percurso da minha primeira graduação.

Resumo

Este trabalho tem como principal objetivo propor um sistema capaz de possibilitar votações em diversos cenários. Para tal são utilizadas diversas tecnologias capazes de garantir tanto a segurança dos dados quanto a sua escalabilidade, podendo assim esta solução ser aplicada até ao cenário político, permitindo à população brasileira o exercício do direito de voto por meio de plebiscitos. Do ponto de vista da Engenharia de Software é utilizada uma metodologia adaptada do *Scrum* para documentar e desenvolver o "produto", buscando alcançar os conceitos de segurança e escalabilidade na sua implementação. A partir de um estudo aprofundado sobre as melhores alternativas tecnológicas para o desenvolvimento desta solução foi verificado que o *Blockchain* seria uma solução tecnológica que compreende a maioria dos requisitos de uma aplicação que projeta votações.

Palavras-chaves: Votação. Engenharia de Software. Blockchain.

Abstract

The main objective of this work is to propose a system capable of enabling voting in various scenarios, using various technologies capable of ensuring both data security and its scalability, thus this solution can be applied until the political scenario, allowing the Brazilian population to exercise the right to vote by means of plebiscites. From the Software Engineering point of view, a methodology adapted from Scrum is used to document and develop the "product", seeking the concepts of security and performance for an application. From an in-depth study on the best technological alternatives for the development of this solution it was verified that the Blockchain would be a technological solution that comprises most of the requirements of an application that projects votes.

Key-words: Voting. Software Engineering. Blockchain.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

ACID	Atomicidade, Consistência, Isolamento e Durabilidade
BPMN	Business Process Modeling Notation
BTC	Bitcoin
CAP	C: Consistência, A: Alta disponibilidade e P: Tolerância ao Particionamento de dados na rede
DDoS	Ataque de negação de serviço
DER	Diagrama Entidade-Relacionamento
DES	Data Encryption Standard
DLTs	Distributed Ledger Technologies
EOAs	Externally Owned Accounts
ETH	Ethereum
FGA	Faculdade do Gama
FIPS	Federal Information Processing Standard
IaaS	Infrastructure as a Service
IDE	Ambiente de Desenvolvimento Integrado
LDAP	Lightweight Directory Access Protocol
MVP	Model View Presenter
NIST	National Institute of Standards and Technology
No-SQL	Not Only SQL
ONU	Organização das Nações Unidas
PEC	Projeto de Emenda Constitucional
PO	Product Owner
P2P	Peer-to-peer
RSA	Rivest-Shamir-Adleman

SGBD	Sistemas Gerenciadores de Banco de Dados
SGBDR	Sistemas Gerenciadores de Banco de Dados Relacional
SHA	Secure Hash Algorithm
SQL	Structured Query Language
SRP	Single Responsibility Principle
TIC	Tecnologia de Informação e Comunicação
UnB	Universidade de Brasília
UML	Unified Modeling Language

Sumário

1 Introdução

Neste capítulo será abordada a contextualização do qual se insere o projeto, as questões de pesquisa a serem alcançadas, uma justificativa para a realização deste trabalho de conclusão de curso, os objetivos e ao final, como serão organizados os capítulos subsequentes.

1.1 Contextualização

A origem das votações se deu na cidade-Estado Grega, Atenas, por meio da institucionalização da democracia direta, onde o poder era exercido direta e indiretamente pelo povo (NILSON, 2008). A partir deste momento, a humanidade foi capaz de estender este conceito para os mais diversos contextos.

Ao longo de todos estes anos, a democracia foi evoluindo e atualmente no Brasil utiliza-se a democracia direta, prevista em sua constituição, utilizando plebiscitos, referendos e iniciativas populares. A democracia participativa também é prevista pela constituição, porém, essa denominação não é utilizada no texto legal. Em suma, a democracia participativa é forma de exercer a democracia com o exercício de poder direto do povo, inclusive na tomada de decisões políticas (NAVARRO, 2003).

A partir do momento em que a população encaminha-se para uma organização sobre a discussão a respeito de diversos temas, os representantes acabam sendo mais questionados sobre as suas decisões, logo pode-se concluir que nem todas as decisões tomadas pelos representantes da população atendem as demandas da sociedade. Com o passar do tempo as exigências da população estão se tornando mais complexas e é observada a nítida necessidade da cooperação entre os congressistas e a população na tomada de decisões (NAVARRO, 2003).

Uma das grandes preocupações em relação qualquer tipo de votação é sempre com a sua idoneidade. A tecnologia, no decorrer dos anos, vem se tornando uma forte alidada à transparência, permitindo obter vários pontos de vista em relação as decisões tomadas. A partir do crescimento exponencial dos dados, foram identificados vários desvios e pontos de corrupção. Logo, este fato induz os cidadãos a serem mais conectados, com isso eles se tornam menos tolerantes a corrupção e agora existem meios para avaliar e tornar as votações mais transparentes (SANTISO, 2018).

Como o procedimento de votação é a parte fundamental para qualquer tipo de processo decisivo, demonstrando assim o poder de direito ou a sua intenção a respeito de um determinado tema, vários estudos sobre tecnologias estão sendo desenvolvidos sobre

sistemas de votações confiáveis e seguros, abrangendo os conceitos de anonimato, justiça, confiabilidade e disponibilidade (GARG P. SARASWAT, 2019). Uma destas tecnologias é o Blockchain, uma tecnologia capaz de sanar as duas principais preocupações a respeito de votações, a fraude e a elegibilidade do eleitor, além de proporcionar a privacidade e auditoria dos dados (NIR, 2018).

Este trabalho apresenta diversas tecnologias, como o Blockchain e Hyperledger Fabric, para desenvolver um sistema de votações que possibilita um conjunto de pessoas a exercer o direito de voto em diversos cenários em que a eleição de algo seja relevante.

1.2 Questão de Pesquisa

O intuito deste trabalho de conclusão de curso (TCC) é proporcionar a um conjuto de pessoas uma solução informatizada para se realizar eleições, em que a votação em uma opção ou alternativa entre as que participem de um processo eleitoral possam ser computadas, sendo a vencedora pela quantidade maior de votos obtidos seja apresentada com segurança no processo. Assim se almeja elaborar nesta proposta uma solução robusta o suficiente para garantir uma eleição segura e genuína. Em síntese, este trabalho busca responder a seguinte questão:

É possível desenvolver um sistema capaz de auxiliar a uma população realizar eleições por meio de votações que sejam seguras e confiáveis empregando tecnologias computacionais?

1.3 Justificativa

Os processos eleitorais sempre estão com a sua integridade em questionamento pelo fato de que elas podem ser computadas ou calculadas erroneamente. Estes cálculos podem sofrer alterações através de imprecisões causadas por erro humano ou por falta de de supervisão capacitada. Outro ponto a considerar a respeito da contagem dos votos é a possibilidade de fraude, em que pode ocorrer uma votação dupla de um participante ou até uma votação ilegal de eleitores que não estejam envolvidos realmente em um processo eleitoral específico, por exemplo, algumas pessoas que não moram em um determinado condomínio votarem em um amigo que esteja concorrendo a ser síndico do condomínio em que vive. Já no processo de apuração da eleição o resultado pode ser alterado por um ator mal-intencionado (SCIENCES, 2018).

Dentro do processo da apuração dos votos, tanto as cédulas físicas quanto os métodos eletrônicos, podem ser destruídos, perdidos ou alterados, devido a um descuido ou atividade maliciosa, causando assim um erro na contagem dos votos. Estes meios de fraude podem se estender as evidências do voto, como por exemplo os boletins de votação,

prejudicando assim, a auditoria externa dos dados utilizados na votação (SCIENCES, 2018).

A partir dos pontos apresentados a respeito da segurança das votações e como a tecnologia, no passar dos anos, vem adquirindo maturidade suficiente para ser utilizada pela maior parte da população de maneira integrada, é possível imaginar soluções envolvendo a tecnologia nas decisões que sejam mais democráticas por meio de eleições que seriam acompanhadas por recursos tecnológicos (XAVIER, 2015).

Em vista disso, uma maneira de promover tais eleições poderia acontecer utilizando o Blockchain, uma tecnologia cuja fundamentação é utilizar os usuários contidos na rede para a validação das informações e manter os dados sincronizados entres os mesmos (ASTE P. TASCA, 2017). Esta tecnologia tem o potencial de limitar a fraude e tornar o processo de votação rastreável e verificável. Para tal, o mesmo possui características ímpares como a imutabilidade, verificabilidade e o consenso distribuído. Tais características são alcançadas por meio de criptografia avançada, fornecendo assim um nível de segurança maior do que qualquer sistema de manutenção de registros existente. Portanto, o Blockchain é uma tecnologia potencial para implementar um sistema moderno para se realizar processos eleitorais em que a votação precise ser segura e sua apuração confiável e ágil (HJALMARSSON G. K. HREIðARSSON, 2018).

1.4 Objetivos

1.4.1 Objetivo Geral

Desenvolver uma aplicação cuja função é possibilitar eleições por meio do voto individual e único dentre os mais diversos contextos no qual as opções de voto possam ser apresentadas e um processo eleitoral identifique a mais votada como vencedora (ou escolhida) entre os participantes dessa eleição. Uma vez que existem diversas ocasiões em que este cenário possa acontecer e será necessária a utilização de um sistema capaz de proporcionar resultados transparentes, destacando o estudo da segurança de sistemas com um alto grau de integridade e confiabilidade.

1.4.2 Objetivos Específicos

1. Disponibilizar um sistema capaz de realizar o acompanhamento de uma eleição.
2. Permitir aos eleitores de uma determinada eleição, um mecanismo para votar nos candidatos desta eleição.
3. Habilitar aos usuários do sistema a visualização do resultado da eleição.
4. Proporcionar a auditoria dos dados da eleição.

1.5 Organização do Documento

Os próximos capítulos deste trabalho de conclusão de curso, estão divididos da seguinte maneira:

Capítulo 2 - Referencial Teórico: apresenta uma visão sobre os conceitos, modelos e explicações acerca dos temas abordados por este trabalho, baseados na literatura.

Capítulo 3 - Metodologia: explicita as metodologias utilizadas para o desenvolvimento tanto do trabalho escrito, quanto da criação do software, além de caracterizar as principais ferramentas e tecnologias utilizadas para o desenvolvimento deste trabalho.

Capítulo 4 - Desenvolvimento: evidencia o produto final deste trabalho, esclarecendo a sua arquitetura, como o mesmo irá operar e o resultado da implementação do mesmo.

Capítulo 5 - Considerações Finais: expõe as considerações à respeito dos estudados realizados para produzir o sistema proposto e o futuro deste trabalho.

2 Referencial Teórico

Neste capítulo serão apresentados os fundamentos teóricos para que seja possível o desenvolvimento do software proposto por este trabalho. Serão abordados os principais temas relacionados com as partes mais relevantes do sistema que será construído.

Como o sistema proposto tratará de escolhas e desejos de um conjunto de pessoas através da coleta de seus respectivos votos (escolhas) sobre algum assunto que lhes sejam de interesse, o software que corresponderá ao produto desse trabalho precisará possuir um nível alto à respeito da segurança dos dados, impedindo possíveis alterações dos votos, protegendo o seu sigilo e possibilitando auditorias em seus dados.

2.1 Segurança

A segurança de sistemas informatizados vai além da garantia da integridade e proteção dos dados de uma organização ou país. A segurança nesse contexto busca não só a proteção dos dados em si, mas da aplicação, quanto ao ambiente físico em que o sistema está inserido (CASTRO, 2011).

Para projetar uma base de dados segura, é necessário analisar três pontos principais:

- Sigilo

  Informações só poderão ser visualizadas por usuários autorizados (RAMAKRISHNAN, 2008). Como por exemplo: um cidadão não pode ver o voto de outro eleitor

- Integridade

  A modificação dos dados não pode ser realizada por seus usuários depois de efetivada (RAMAKRISHNAN, 2008). Como, por exemplo: um eleitor não pode alterar o seu voto depois que ele foi confirmado.

- Disponibilidade

  Um sistema deve sempre estar disponível e os usuários autenticados não podem ter o seu acesso negado (STALLINGS, 2011).

Esses três pontos acima são os fundamentos dos objetivos de segurança para sistemas informatizados. Como já diz a *National Institute of Standards and Technology* (NIST) padrão *Federal Information Processing Standard* (FIPS) 199, o sigilo, a integridade e a disponibilidade são os três objetivos de segurança a se alcançar para um sistema ser considerado seguro (STALLINGS, 2011).

Para implementar estes três pontos é necessário instaurar uma política de segurança clara e consistente. Outros pontos são necessários para garantir a segurança de um Sistema Gerenciador de Banco de Dados (SGBD), como mecanismo de segurança física do computador em que se encontra instalado o SGBD e de seu sistema operacional, visando sempre vários níveis de segurança para obter uma política mais concisa e segura. Para os sistemas que funcionam em um ambiente *web*, outros pontos são importantes à segurança, tais como a criptografia, protocolos de segurança e assinaturas digitais (RAMAKRISHNAN, 2008).

Segundo Stallings (2011), existem alguns desafios em relação à segurança de sistemas, sendo estes:

1. A segurança de sistemas não é tão simples quanto parece. Os nomes dos requisitos de segurança parecem ser descomplicados, como: confiabilidade, autenticação ou integridade. Porém, para implementar estes requisitos requer um esforço complexo.

2. Sempre é preciso considerar ataques maliciosos no desenvolvimento de um mecanismo de segurança, todos os pontos devem ser considerados, pois a maioria dos casos de ataques bem sucedidos exploram alguma fraqueza do mecanismo de segurança.

3. Em muitos casos um mecanismo de segurança não é óbvio, logo, para fazerem sentido, é necessário ponderar vários tipos diferentes de ameaça.

4. É necessário saber onde e quando utilizar um mecanismo de segurança, tanto em um ambiente físico, quanto no ambiente lógico.

5. Geralmente os mecanismos de segurança abrangem mais do que apenas algoritmos e protocolos. Comumente são utilizadas informações secretas, o que levanta o questionamento sobre a segurança na criação, distribuição e proteção destas informações. Outro ponto a levar em consideração é a dependência dos protocolos de comunicação, o que pode aumentar o grau de dificuldade no desenvolvimento de um mecanismo de segurança.

6. A interação entre um usuário malicioso, o qual tenta encontrar vulnerabilidades, e o projetista ou administrador, que tenta bloqueá-las, é uma questão bastante complicada a ser resolvida. Visto que, neste cenário o projetista está em desvantagem, pois é preciso identificar todas as vulnerabilidades antecipadamente para alcançar uma possível segurança impecável, enquanto o usuário malicioso precisa apenas identificar uma fraqueza para explorá-la.

7. Existe uma predisposição a pouco investimento em segurança até que aconteça alguma falha ou invasão.

8. A segurança de um sistema necessita de um monitoramento constante, o que é dificultado em um cenário de prazos curtos e a sobrecarga do responsável de segurança com outras funções.

9. Em muitos casos a segurança é implementada após o desenvolvimento do projeto, enquanto deveria ser uma parte integral do processo de design.

10. Muitos usuários enxergam a segurança em grande escala como um empecilho na utilização de um sistema de informação ou ao acesso as próprias informações.

A fim de garantir que exista uma comunicação segura por todo o sistema, assegurando não apenas a integridade dos dados, mas também a sua confiabilidade e a autenticação de seus usuários as técnicas de criptografia são empregadas com sucesso.

2.1.1 Criptografia

Segundo Coutinho (2011), a criptografia estuda os métodos para codificar uma mensagem de modo que só o seu destinatário legítimo consiga interpretá-la.

O princípio da criptografia é utilizar um algoritmo criptográfico para encriptar um determinado dado, utilizando chaves criptográficas. Estas chaves podem ser definidas por algum usuário ou pelos administradores do sistema. Para os dados serem descriptografados, existe um algoritmo para decifrar estes dados, igualmente utilizando uma chave para descriptografá-los, em que estes retornaram a sua forma original (STALLINGS, 2011).

2.1.1.1 Tipos de Criptografia

- Criptografia Simétrica

  A criptografia simétrica utiliza uma mesma chave para criptografar e decifrar os dados. O principal ponto fraco deste tipo de criptografia é o fato de que todos os usuários com autorização precisam conhecer a chave, o que pode levar ao vazamento da mesma. Um exemplo deste tipo de algoritmo é o *Data Encryption Standard* (DES) que trabalha com este tipo de criptografia (RAMAKRISHNAN, 2008). Geralmente ele é utilizado para ocultar o conteúdo de blocos ou fluxos de dados de qualquer tamanho, incluindo mensagens, arquivos, chaves de criptografia e senhas (STALLINGS, 2011).

- Criptografia Assimétrica

  A criptografia assimétrica, muitas vezes conhecida como criptografia de chave pública, é fundamentada pelo conceito de que cada usuário possui uma chave pública, conhecida por todos, e uma chave privada, conhecida apenas pelo próprio usuário. Um ponto fraco da criptografia assimétrica é a forma como são designadas estas

chaves. Um exemplo é o algoritmo *Rivest-Shamir-Adleman* (RSA) (RAMAKRISHNAN, 2008). Este tipo de criptografia é geralmente utilizado para ocultar pequenos blocos de dados, como chaves de criptografia e valores de função *hash*, que são usados em assinaturas digitais (STALLINGS, 2011).

- Algoritmos de integridade de dados

  Estes algoritmos são baseados em funções de *hash* para garantir a integridade dos dados, ou seja, o *hash* irá determinar se o dado foi modificado ou não. Eles são geralmente utilizados para proteção contra a alteração de blocos de dados, como mensagens (STALLINGS, 2011).

- Protocolos de Autenticação

  Os protocolos de autenticação são esquemas baseados em algoritmos criptográficos desenhados para identificar entidades como receptores e emissários (STALLINGS, 2011).

2.1.2 Assinaturas Digitais

A criptografia assimétrica, ou mais especificamente a criptografia de chave pública, é utilizada para produzir assinaturas digitais. Suponha que exista dois usuários, o usuário A e o usuário B, e estes dois usuários estão enviando mensagens entre eles. O usuário A pode verificar se a mensagem do usuário B foi realmente enviada por ele, e vice e versa (RAMAKRISHNAN, 2008).

Para um melhor entendimento do sistema de criptografia para assinaturas digitais suponha que o usuário A deseja verificar se a mensagem foi mesma enviada pelo usuário B. B criptografa a sua mensagem utilizando a sua chave privada e depois criptografa o resultado utilizando a chave pública de A. Quando A recebe a mensagem ele primeiramente descriptografa utilizando a sua chave privada e depois descriptografa o resultado utilizando a chave pública de B, resultando assim na mensagem original escrita por B. Como a chave privada de B é apenas conhecida por ele mesmo, um elemento malicioso não deve possuir esta chave privada, logo ele não irá conseguir falsificar esta mensagem e caso ele tente falsificar a mensagem, o resultado da descriptografia seria ilegível (RAMAKRISHNAN, 2008).

2.2 Banco de Dados Relacional

Segundo Heuser (1998), banco de dados é um conjunto de dados integrados que tem por objetivo atender uma comunidade de usuários. Existem duas maneiras de administrar este conjunto de dados, manualmente ou de modo automatizado. A maneira usada para criar e manter estes conjuntos de dados é empregando o uso dos SGBD's e como um banco

de dados é composto por várias tabelas e as mesmas se relacionam entre si, daí o nome banco de dados relacional (ELMASRI, 2005).

O usuário utiliza uma linguagem chamada de *Structured Query Language* (SQL) para se comunicar com essa tecnologia de banco de dados (SGBDR), constituindo-se de uma linguagem de *script*, declarativa e baseada na álgebra relacional. Os dados armazenados pelo SGBDR são registrados em diversas tabelas (estruturas de armazenamento dos dados), também chamadas de relação e compostas por um conjunto de tuplas (linhas) e colunas (atributos dos dados) (ELMASRI, 2005).

Alguns conceitos são importantes no uso dessa tecnologia de banco de dados, entre eles os relacionamentos entre as tabelas que exigem a implementação das chaves primária e estrangeira nas tabelas envolvidas no estabelecimento desse relacionamento (HEUSER, 1998).

2.2.1 Chaves

1. Chave primária: em geral, existe um atributo em uma certa relação (tabela), do qual os seus valores são únicos, comumente empregado para identificar o registro (tupla) em questão. Porém, existem relações das quais a chave primária não é o um único atributo e por isso possui uma combinação de atributos para identificar, unicamente, cada registro que será armazenado no banco de dados. Quando isso acontecer em uma tabela a denominação dessa chave passa a ser chave primária composta (DATE, 2004).

2. Chave estrangeira: esse tipo de chave corresponde a um atributo que faz referência a uma chave primária, ou primária composta, de uma outra tabela (por isso a expressão estrangeira, dando a ideia de vindo de fora). A chave estrangeira possibilita o relacionamento entre as tabelas do banco de dados, utilizando seus valores como associados aos atributos armazenados em outra relação como chave primária (RAMAKRISHNAN, 2008).

2.2.2 Propriedades ACID

A tecnologia dos bancos de dados relacionais possui diversas vantagens, considerando as formas anteriores utilizadas para armazenar dados. Algumas delas são a representação simples dos dados que facilitam na realização de consultas complexas (RAMAKRISHNAN, 2008).

Os SGBD's possuem mecanismos para garantir que uma transação (programa em execução que forma uma unidade lógica de processamento no banco de dados, incluindo neste uma ou mais operações de acesso, englobando operações de inserção, alteração ou

recuperação) seja completada com sucesso, persistindo os seus dados no banco de dados ou, em caso de falha, impedindo a inconsistência do banco ou de outras transações. Logo, para garantir todas essas características as transações em um SGBD possuem as propriedades conhecidas pela sigla ACID (ELMASRI, 2005).

Segundo Elmasri (2005), as propriedades ACID são definidas como:

- Atomicidade: uma transação é uma unidade atômica de processamento, tendo a característica da transação ser executada totalmente ou não será nada dela executada;
- Consistência: uma transação sempre preservará a consistência se sua execução se completar e o banco de dados passará de um estado consistente para outro estado consistente;
- Isolamento: uma transação deverá ser executada isoladamente das demais transações, ou seja, a execução de uma transação não deve sofrer interferência de nenhuma outra transação concorrente até ser completamente executada;
- Durabilidade: as mudanças aplicadas no banco de dados por uma transação efetivada com sucesso deverão persistir no banco de dados (característica de permanência).

2.2.3 Limitações

Dentre algumas desvantagens do banco de dados relacional estão a não obrigatoriedade da criação de uma estrutura coesa das tabelas de algum modelo em questão, sendo assim possível a existência de sistemas complexos sem a devida necessidade e até prejudicando uma possível evolução do projeto. Uma outra questão importante é o grande aumento no volume de dados, podendo ocorrer alguns cenários em que o desempenho do banco não seja mais satisfatório em relação ao tempo de consultas complexas. (BRITO, 2010).

Pode-se acrescentar as limitações dos bancos de dados na atualidade, sendo grande a demanda por aplicações Web, cujo grande volume de dados, requisitos diferenciados como escalabilidade e alto grau de disponibilidade, encorajaram a concepção de uma nova solução para o armazenamento de dados (LOSCIO H. R. DE OLIVEIRA, 2011).

2.3 Banco de Dados Não Relacional

O banco de dados não relacional, também conhecido como No-SQL ou NoSQL, pode ser definido como um grande conjunto de conceitos relacionados ao armazenamento de dados e a sua manipulação, empenhando-se em atingir uma maior performance (TIWARI, 2011). Este novo conceito foi concebido para realizar uma modelagem mais

flexível, em comparação ao modelo relacional, propondo soluções para reduzir a estruturação do banco de dados relacional (BRITO, 2010). O banco de dados não relacional proporciona a operação com uma grande quantidade de dados não estruturados, porém, a flexibilidade de indexação, realização de consultas e a integridade transicional são prejudicadas (BRITO, 2010).

Utilizar o No-SQL provê uma série de vantagens, como:

- Escalabilidade Horizontal

  Esta propriedade permite a cada sistema a possibilidade de aumentar o número de nós com a camada *Infrastructure as a Service* (IaaS), em outras palavras, aumentando a capacidade de processamento e armazenamento (KELLY, 2014).

- Baixa Latência

  Esta propriedade utiliza a propriedade anterior, escalabilidade horizontal, para otimizar o tempo de performance para um alto nível de processamento, distribuindo este grande processo em vários, sendo estes pequenos processos divididos em vários núcleos de processamento. Isso viabiliza uma enorme quantidade de transações simultâneas de processos de leitura e escrita ou uma disponibilidade de acesso elevada, melhor explicada na continuidade desse capítulo (KELLY, 2014).

- Grande Disponibilidade de Acesso

  Esta vantagem é relacionada a grande quantidade de usuários simultâneos, superando o modelo relacional, pelo fato do alto poder de processamento (KELLY, 2014).

Entretanto, os bancos de dados não relacionais apresentam algumas limitações, por exemplo, a ausência de auxílio a uma linguagem de consulta estruturada e escassez na administração das transações (HAN E. HAIHONG, 2011).

Devido a todas essas características o No-SQL é, geralmente, mais utilizado nos seguintes contextos: processamento em massa de imagens, dados públicos de *websites*, dados de sensores remotos, dados sobre *logs* de eventos, dados de *smarphones*, dados de jogos e dados abertos vinculados (KELLY, 2014).

2.3.1 Tipos de Banco de Dados Não Relacionais

Dentro deste vasto cenário, incluso ao modelo relacional, foram criados diversos modelos de armazenamento de dados envolvendo o No-SQL, cada um aperfeiçoado para um certo contexto. Os mais relevantes, hoje em dia, são:

- Chave / Valor

Este foi primeiro modelo de armazenamento concebido junto ao movimento do No-SQL. Em sua definição, todos os dados são endereçados por uma chave, exclusivamente, armazenando estes dados em uma sequência de bytes, como pode ser observado na Figura 1. O diferencial deste modelo é a velocidade com que os dados são buscados, pois ele é baseado em operações simples (HECHT, 2011).

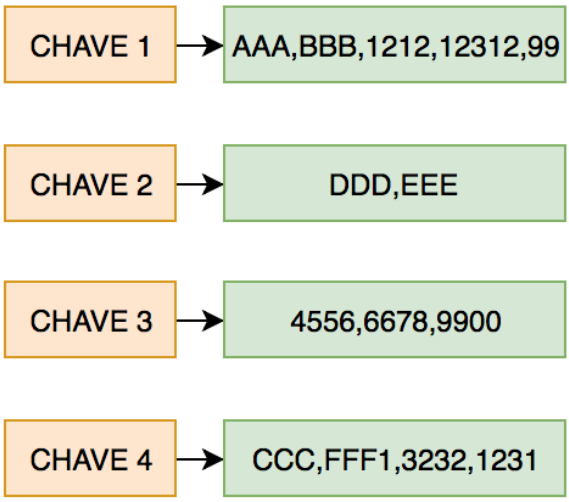

Figura 1 – Modelo Chave Valor. Fonte: (QUEIROZ, 2017).

- Orientado a Colunas

  Este modelo consiste no armazenamento dos dados em colunas, ou seja, os dados de uma tabela bidimensional são armazenados juntos, como podo ser observado na Figura 2, permitindo assim o não armazenamento de dados nulos, diferentemente do modelo relacional (REDMOND J. WILSON, 2012). O modelo orientado a colunas foi desenvolvido para lidar com uma grande quantidade de informações, visto que ele consiste em um mapa de dados amplo, persistente, distribuído e multidimensional (HECHT, 2011).

ID	NOME	ENDERECO	CIDADE
1	JOAO	R. ABC, 18	SALVADOR
2	JOSE	AV. BDJ 43	SÃO PAULO
3	MARIA	CD. ATOS	RIO DE JANEIRO
4	ANTONIO	ALM FGH	V. CONQUISTA

Figura 2 – Modelo Orientado a Colunas. Fonte: (QUEIROZ, 2017).

- Orientado a Documento

Segundo Wilson (2012), o modelo orientado a documento armazena documentos. Um documento é como um *hash*, com um ID único e valores de uma variedade de tipos, incluindo mais *hashes*, como é exemplificado na Figura 3. Estes documentos podem conter estruturas aninhadas e exibir um alto nível de flexibilidade, aplicada a vários domínios. Uma das vantagens do modelo orientado a documento é a facilidade da migração e integração entre diferentes gerenciadores de banco de dados e a obtenção de informações estatísticas (HECHT, 2011).

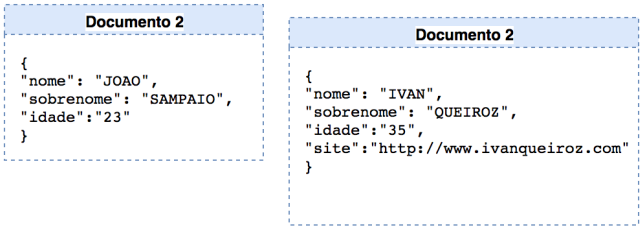

Figura 3 – Modelo Orientado a Documento. Fonte: (QUEIROZ, 2017).

- Orientado a Grafos

  O modelo orientado a grafos permite que um sistema elabore um grafo, em que cada nó possui uma chave e valores, como pode ser observado na Figura 4. Isso permite que uma aplicação com muitos dados interconectados realizem consultas mais rapidamente (HECHT, 2011).

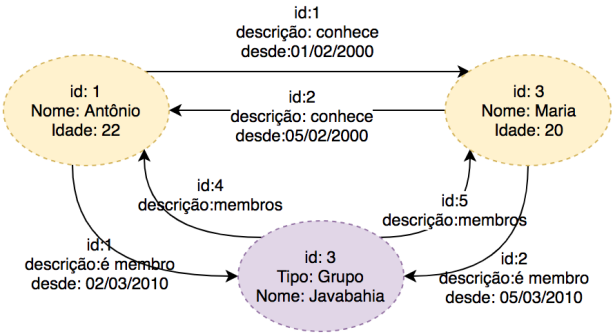

Figura 4 – Modelo Orientado a Grafos. Fonte: (QUEIROZ, 2017).

Com o intuito de evidenciar as limitações dos bancos de dados não relacionais torna-se importante esclarecer o teorema CAP (*Consistency, Availability* e *Network Partition Tolerance*), sendo *Consistency* relativo a propriedade de Consistência, *Availability* vinculado a Alta disponibilidade, enquanto *Network Partition Tolerance* seria a tolerância ao Particionamento de dados na rede. Utilizando estas três propriedades é possível trabalhar sobre a integridade transacional em grandes sistemas distribuídos e escaláveis (TIWARI, 2011).

- Consistência (*Consistency*): a consistência determina a situação do sistema após a consolidação de uma operação, ou seja, um sistema é considerado consistente quando todos os usuários que estão realizando a operação de leitura, visualizam ao mesmo tempo, uma atualização de uma operação de escrita em um sistema (FOWLER, 2012).
- Alta Disponibilidade (*Availability*): a disponibilidade determina que um usuário pode executar as operações de leitura e escrita a qualquer momento (FOWLER, 2012).
- Tolerância ao Particionamento (*Network Partition Tolerance*): a tolerância ao particionamento permite que um sistema execute normalmente as suas operações, apesar da divisão de suas tarefas, passando também pela realocação dinâmica de recursos entre nós de um grafo. Um sistema com uma tolerância ao particionamento pode ser descrito como um sistema escalável (FOWLER, 2012).

Diante dos esclarecimentos do teorema CAP, uma afirmação é válida: é impossível atingir estas três características em um sistema que deseja ser distribuído e escalável, sem ao menos um destes conceitos ser sacrificado (TIWARI, 2011).

2.4 Blockchain

O Blockchain pode ser considerado como um conjunto de tecnologias que funciona como um banco de dados, sendo somado a algumas outras funcionalidades que garantem a confiança entre os membros da rede que ele constitui. Formalmente, o Blockchain é uma estrutura de dados, ou seja, ele define como os dados são distribuídos e armazenados (LEWIS, 2015). Além de armazenar o registro de todas as transações de uma aplicação como um livro contábil, a tecnologia do Blockchain também é capaz de executar programas definidos pelo usuário (KOSBA A. MILLER, 2016). Uma linha do tempo da história do Blockchain pode ser traçada como sintetizado na Figura 5.

2.4.1 Contexto Histórico

No começo da década de 90 surgia a concepção de uma série de blocos criptografados para armazenar informações, baseada na utilização de funções de *hash* criptográficas, desenvolvidas por Adam Back, para garantir a ordem das informações e certificar que as informações anteriores não fossem modificadas (SHERMIN, 2017).

Em 2008, Satoshi Nakamoto implementou a primeira solução baseada em Blockchain, o Bitcoin. Ele utilizou o Blockchain para criar um registro contábil inviolável para a sua moeda virtual, visando resolver o problema do gasto duplo monetário, ou seja, impedir que uma mesma moeda fosse gasta mais de uma vez em situações diferentes (NAKAMOTO, 2008).

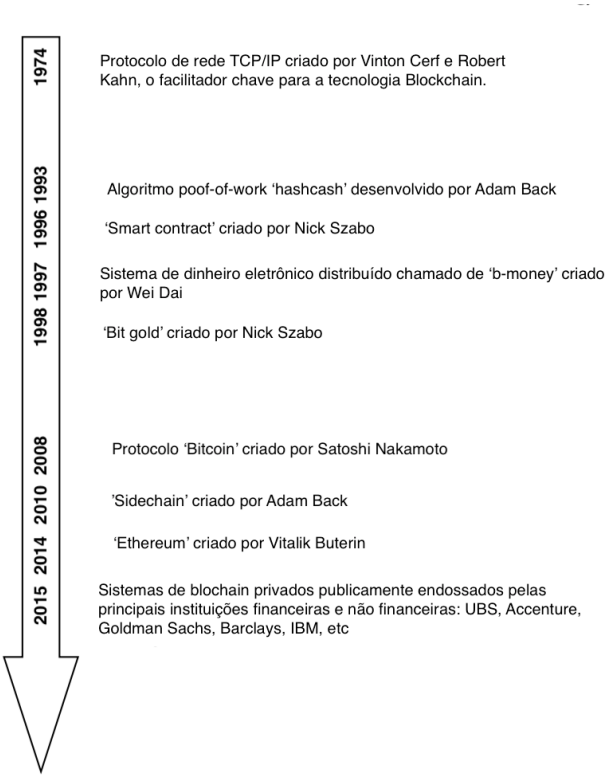

Figura 5 – Adaptado de história do tempo do Blockchain. Fonte: (OLLEROS, 2016).

2.4.2 Árvore de Merkle

A árvore de Merkle é um algoritmo cuja função é garantir a existência de uma certa informação em um determinado local. Este algoritmo basicamente divide uma grande quantidade de informações em pedaços, todos em ordem. A partir deste ponto, uma função de *hash* em cada um destes pedaços, o próximo passo é realizar a concatenação destes *hashes* e aplicar mais uma rodada de função *hash* a esta concatenação. O resultado desta série de passos será um nó da árvore de Merkle, os passos anteriores são repetidos continuamente até chegar ao nó raiz da árvore, como se pode notar na Figura 6 (MERKLE, 2000).

O algoritmo da árvore de Merkle é utilizado para realizar a prova de Merkle, cuja função é verificar onde está localizada uma informação dentro da árvore. No âmbito do Blockchain, a prova de Merkle diminui o tempo de busca da localização da informação em *log* de n (MERKLE, 2000).

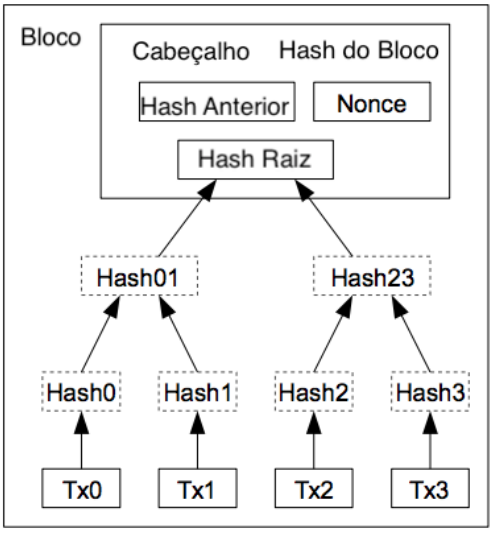

Figura 6 – Adaptado de transações em uma árvore de Merkle. Fonte: (NAKAMOTO, 2008).

2.4.3 Criptomoeda

Criptomoedas são moedas digitais que utilizam a criptografia para proteger seus dados, criar novos blocos e confirmar suas transações, através de cálculos computacionais realizados pelos computadores conectados na rede Blockchain de forma descentralizada. Os exemplos de criptomoedas mais conhecidos são o Bitcoin (BTC) e o Ethereum (ETH) (LI Q. LU, 2019).

2.4.4 Esclarecendo o Blockchain

O Blockchain utiliza uma rede *Peer-to-peer* (P2P) para conectar computadores em rede (cada computador pode ser chamado de um nó da rede). Cada nó possui um *backup* completo do banco de transações e é utilizado para validar e propagar novas transações aos outros computadores da rede (TAPSCOTT, 2016).

A principal vantagem do Blockchain é a impossibilidade de deletar ou alterar uma informação uma vez validada e armazenada no banco. Logo, estas informações são imutáveis, assim sendo é correto afirmar que este sistema emprega um alto grau de confiabilidade a qualquer projeto que o utilize. Porém, armazenar os dados em todos os computadores podem acabar gerando uma adversidade de armazenamento, pois as novas transações estão sempre acontecendo e sendo mantidas em todos os nós (TAPSCOTT, 2016).

As informações de um sistema que está utilizando o Blockchain são registradas em Blocos. O primeiro bloco é geralmente chamado de **bloco gênesis**, enquanto os próximos blocos são armazenados contendo um grupo de transações prontamente validadas. Todos os blocos possuem um *hash* do bloco anterior, como é impossível alterar alguma transa-

ção de algum bloco, sem alterar os próximos blocos da rede, pois o *hash* estaria sendo violado e todos os outros nós da rede estariam detectando esta tentativa de alteração e a invalidando, mantendo assim a base de dados íntegra (TAPSCOTT, 2016).

Os blocos são gerados pelos computadores da rede, contendo um conjunto de transações previamente validadas. Cada bloco é composto por algumas outras informações, além das transações, como o *hash* do bloco anterior e o *timestamp* da criação do bloco. Logo após a criação do bloco é obtido um *hash* do próprio para ser utilizada no bloco seguinte, sendo todas essas informações armazenadas no cabeçalho do bloco (KOSBA A. MILLER, 2016).

Geralmente, é utilizada a árvore de Merkle para estruturar as transações e armazenar a raiz da árvore em todas as *headers* dos blocos, com o intuito de gerar uma assinatura para um bloco, possibilitando a validação das transações empregando a prova de Merkle (MERKLE, 2000).

Para um melhor entendimento de como as transações, blocos e seus *hashes* são distribuídos em uma cadeia Blockchain, observe a Figura 7. Onde é possível identificar que o bloco 53 possui várias transações, um *hash* das suas transações, um *hash* do bloco anterior e o *hash* final do bloco em questão.

ID da Transação	BLOCO	HASH DO TRANSAÇÕES	HASH DO BLOCO ANTERIOR	HASH FINAL
148b2dfc47cf46be3840f80c826fea75	53			
84cd62a6425b38d228ef29869dfb0897	53			
aad6c875b15b126741c85c9fdd37a4f9	53			
fa59c2996d445ebce84710948aaac288	53	bdc5ef6b47177f56b18109aeb2317625	3346c4e052de08901daf6151c188c640	**000b7d7e471af96494565613141df45d**
01b8bedfc29bc61608b89e1ef300345e	53			
fba9ca1450a5477e8de461f5e6fa2af2	53			
6ceda1ffaa097890088163d32bf1aea0	53			
32b66448fed65314ce5d4c1eca917f6e	54			
b1d015975a200bbf0e4069932f89c5b9	54			
b35f941bd0942ffa64bfaa47b50d4123	54			
b3758bc58ded77a199f4148259c72be9	54	c33e8f2391c2e883dfd3a06f9030aad7	000b7d7e471af96494565613141df45d	**6c35b3e1e30a17ca14e997dc5ab11c16**
e3ca68c49f0e78acb7d8324202e85f4f	54			
4f5b7bb1d3f3e73121ff59538469d977	54			
a637e27540bafcdee51ca225d35cb754	54			

Figura 7 – Adaptado de um exemplo de uma cadeia em Blockchain. Fonte:(BORTOLINI, 2016).

Para a proteção contra fraudes à respeito criação dos blocos, são utilizados algoritmos para obter um consenso entre todos os nós da rede. Os mais comuns são a prova de trabalho (*Proof-of-Work*) e a prova de aposta (*Proof-of-Stake*) (OLLEROS, 2016).

O *Proof-of-Work* é baseado no tempo e gasto de recursos computacionais necessários para criar um bloco, de forma que seja computacionalmente inviável uma possível tentativa de criar uma cadeia de blocos falsa. Este algoritmo busca verificar um valor *hash*, que no caso do *Secure Hash Algorithm* (SHA) - 256, começa com um número de bits

O (zero) e o trabalho médio exigido para verificar um *hash* é o exponencial do número de bits zero necessários que podem ser verificados por um único *hash* (NAKAMOTO, 2008).

O *Proof-of-Stake* consiste na aleatoriedade da seleção dos computadores em rede para realizarem o consenso sobre a veracidade do bloco. Este algoritmo é considerado mais rápido do que o *Proof-of-Work* (OLLEROS, 2016).

2.4.5 Tipos de Blockchain

2.4.5.1 Blockchain Público

Os Blockchains estão abertos para todos no planeta poderem ler e enviar transações e participar da prova de consenso. Este tipo de Blockchain é considerado como totalmente descentralizados (BUTERIN, 2015).

2.4.5.2 Blockchain Privado

Para um Blockchain ser considerado privado a permissão para escrita é restrita a apenas os usuários da organização. A operação de leitura pode ser restrita ou aberta para pessoas fora da organização, como, por exemplo uma audiência externa (BUTERIN, 2015).

2.4.5.3 Blockchain de Consórsio

Este tipo de Blockchain possui uma pré-seleção dos computadores da rede para realizarem a validação de consenso. A permissão de leitura pode ser liberada ou restrita, sendo considerados Blockchains híbridos (BUTERIN, 2015).

2.4.6 Plataformas Blockchain

Uma das maneiras de se implementar o Blockchain em uma aplicação é utilizando a plataforma Ethereum ou a plataforma Hyperledger.

Para implementar o Blockchain em aplicações, exitem alguns *frameworks* para auxiliar o desenvolvedor a utilizar as suas funcionalidades, um exemplo, é o Truffle, um *framework* baseado na tecnologia Ethereum. Um outro *framework* é o Fabric, o mesmo utiliza o Hyperledger para desenvolver projetos empregando o uso de várias tecnologias envolvendo o Blockchain.

2.4.7 Ethereum

O Ethereum é uma plataforma *open-source* que possibilita a criação de aplicações utilizando a tecnologia Blockchain, sendo adaptativa e flexível para o desenvolvimento de novas soluções.

O Blockchain Ethereum armazena o estado de todas as contas do sistema e todas as transições de estado são enviadas através de dados por todas as contas. Existem dois tipos de contas no Ethereum:

- *Externally Owned Accounts* (EOAs), as quais são controladas por chaves privadas.
- Contas de Contratos, as quais são controladas por contratos inteligentes e são ativadas apenas por EOAs.

Os usuários do Ethereum devem "pagar"pequenas taxas de transações computacionais para a rede. Este mecanismo protege a rede Ethereum de ataques computacionais maliciosos, como o ataque de negação de serviço (DDoS) ou *loops* infinitos. Estes pagamentos são realizados pelos usuários que estão utilizando a função de escrita em cada passo ativado do programa.

2.4.7.1 Smart Contracts

Por definição um *Smart Contract* é um programa autoexecutável que desempenha exatamente o que o seu criador definiu, dentro do contexto do Blockchain. Eles são códigos que definem regras, obrigações, benefícios e penalidades que podem obter uma relação com contratos da vida real, em que são definidas regras e cláusulas a serem cumpridas (HERTIG, 2018).

No momento em que um *Smart Contract* é criado ele jamais pode ser alterado, nem mesmo o seu criador pode alterá-lo. Se alguma atualização for necessária, é mandatório a criação de uma nova rede (RANDE, 2018).

Consequentemente os *Smart Contracts* são autoexecutáveis, definem regras e um conjunto destes mesmos são utilizados para fundamentar sistemas distribuídos, adicionalmente ao fato destas regras serem imutáveis, um nível de segurança a mais é atribuído ao sistema que está utilizando estes contratos(RANDE, 2018).

2.4.8 Hyperledger Fabric

Para explicar os conceitos do Hyperledger Fabric, primeiramente é necessário definir o Hyperledger, sendo um esforço colaborativo, com o intuito de criar uma tecnologia de Blockchain avançada identificando e endereçando várias funcionalidades a respeito de *distributed ledger technologies*(DLTs), as quais podem transformar a maneira que as transações de negócio são conduzidas globalmente. Por fim, o Hyperledger é um conjunto de vários projetos (KUHRT, 2018).

O Hyperledger Fabric é um dos projetos do Hyperledger, consistindo em um projeto *open-source* de permissão do tipo privada a nível empresarial de uma DLT, oferecendo

diversas capacidades diferenciadas das tecnologias de contabilidade distribuída ou de plataformas baseadas no Blockchain (FERRIS, 2019).

Um diferencial do Hyperledger é que o mesmo foi estabelecido com base na Linux Foundation, a qual possui uma longa e bem-sucedida história participando no desenvolvimento de projetos *open-source* com supervisões governamentais, possuindo uma grande comunidade de apoiadores e ecossistemas prósperos. Um outro ponto apoiador ao anterior é que o projeto Hyperledger Fabric já é utilizado em mais de 35 empresas, aumentando a confiabilidade em uma tecnologia relativamente nova (FERRIS, 2019).

O Fabric possui uma arquitetura modularizada e configurável, permitindo então a inovação, versatilidade e optimização para uma grande parte da indústria. O próprio é a primeira plataforma DLT a oferecer suporte a contratos inteligentes desenvolvidos nas linguagens de programação mais utilizadas nos dias de hoje, como Node.js e Java, facilitando assim o desenvolvimento de smart contracts, pelo fato da maioria dos desenvolvedores já possuir o contato com tais tecnologias e não exigindo uma nova curva de aprendizado para adquirir o conhecimento de uma linguagem de programação específica (FERRIS, 2019).

A plataforma Hyperledger Fabric é do tipo Blockchain privado, significando que ao contrário dos Blockchains públicos (Bitcoin), os participantes da rede privada são conhecidos um pelo outro, em vez de anônimos. Estes por sua vez podem não ser confiáveis, ou seja, esse fato permite a rede operar sob um modelo de governança construído com base na confiança existente entre os participantes (FERRIS, 2019).

Um dos diferenciais mais importantes do Fabric é o seu suporte a protocolos de consenso conectáveis, possibilitando que a plataforma seja altamente customizável e permitindo assim a possibilidade de escolher o modelo de segurança mais condizente ao contexto desejado (FERRIS, 2019).

Outro ponto importante é que o Fabric permite a utilização de protocolos de consenso que não exigem a mineração de uma criptomoeda, como a plataforma do Ethereum, as quais atingem uma mineração cara, demandando mais poder computacional. Evitando o uso de criptomoedas, o risco de ataques é diminuído e a ausência de operações de mineração criptográfica significa que a plataforma pode ser implantada com aproximadamente o mesmo custo operacional que qualquer outro sistema distribuído (FERRIS, 2019).

O conjunto destes diferenciais da plataforma, torna o Fabric uma das plataformas existentes mais eficientes, em termos de processamento de transações e a latência para a confirmação da transação, permitindo também a privacidade e a confiabilidade das transações e dos *smart contracts* que o Fabric implementa.

2.4.8.1 Modularização

A arquitetura do Hyperledger Fabric foi desenvolvida para ser modularizada, podendo possuir um consenso conectável, protocolos de gerenciamento de identidade conectáveis, como (*Lightweight Directory Access Protocol*) *LDAP* ou *OpenID Connect*, protocolos de gerenciamento de chaves ou bibliotecas criptográficas. Pode-se concluir que a plataforma foi projetada para atender os mais diversos contextos e requisitos. Esta modularização está dividida em seis partes: *Assets, Chaincode, Ledger Features, Privacy, Security Membership Services e Consensus* (SYKES, 2019):

- **Assets:**

  São responsáveis por definir o que é necessário para realizar a troca de bens com valor monetário através da rede.

- **Chaincode:**

  A execução do Chaincode (a denominação dada a *smart contracts* pelo Hyperledger Fabric) é dividida na ordenação das transações, a limitação dos níveis de confiança, verificação através dos nós, otimização da escalabilidade da rede e performance.

- **Ledger Features:**

  O *ledger* (registro das transações) é imutável e compartilhado com cada canal de transmissão, contendo toda a história de transações. Um diferencial deste *ledger* é a capacidade de se realizar *queries* utilizando a linguagem SQL-like para auditoria.

- **Privacy:**

  Os canais de transmissão e a coleção de dados privados permitem a privacidade e a confiabilidade multilateral das transações, sendo estes dois requisitos geralmente solicitados por indústrias onde há a troca de ativos através de uma rede em comum.

- **Security Membership Services:**

  Como o Hyperledger Fabric é do tipo privado (*permissioned*), ele promove uma rede confiável, onde os seus participantes conhecem todas as transações, permitindo o rastreamento dos dados a partir de reguladores e auditores.

- **Consensus:**

  Esta funcionalidade permite o consenso único, proporcionando flexibilidade e escalabilidade a qual o sistema necessita.

2.4.8.2 Arquitetura

A arquitetura do Fabric está divida em três categorias: *Membership, Blockchain e Chaincode*. Essas categorias são estruturas lógicas, não uma descrição física de como os

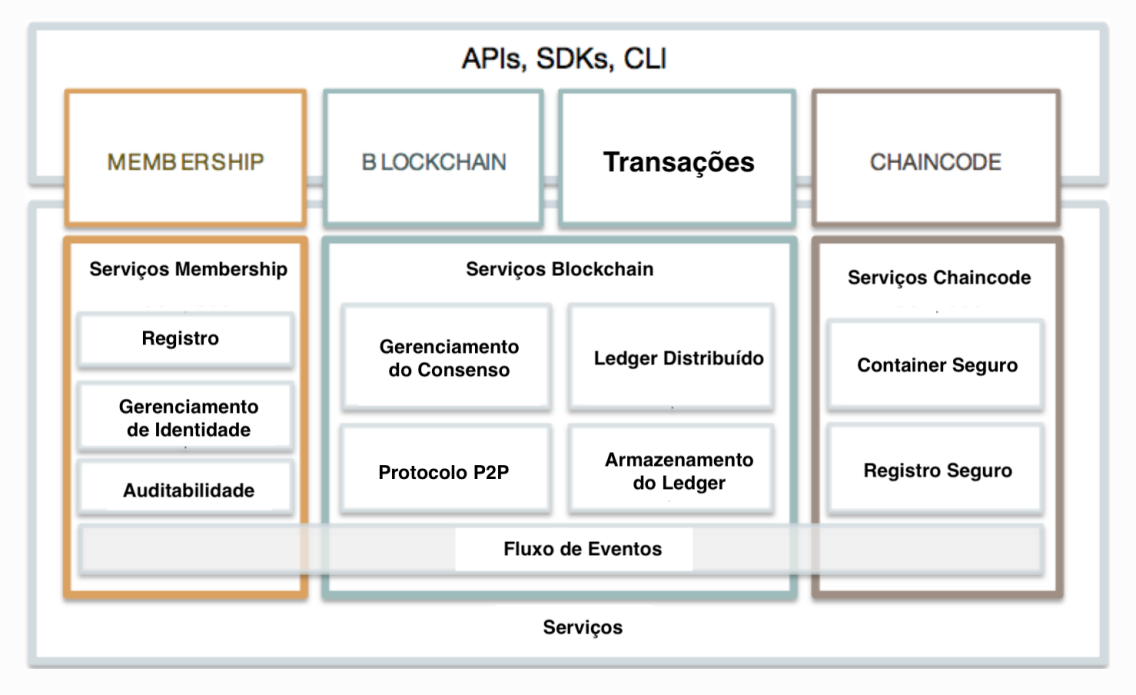

Figura 8 – Adaptado de arquitetura Hyperledger Fabric. Fonte: (CUOMO J. A. LATONE, 2016).

módulos estão disponibilizados em processos separados (CUOMO J. A. LATONE, 2016). A Figura 8 apresenta a arquitetura padrão do Hyperledger Fabric.

3 Metodologia

O capítulo sobre a metodologia tem por objetivo demonstrar os métodos utilizados para pesquisar e desenvolver este TCC. Na seção 4.1 será apresentada a metodologia utilizada para produzir a escrita do trabalho e a seção 4.2 irá exemplificar a metodologia empregada no desenvolvimento do software.

3.1 Metodologia referente ao desenvolvimento do trabalho

A técnica metodológica utilizada para guiar o desenvolvimento deste trabalho foi a Pesquisa-ação, a qual consiste em realçar o trabalho prático através da teoria (PINHEIRO, 2010).

A Pesquisa-ação é definida como uma pesquisa sistemática e cíclica, baseada em um problema a ser analisado. É utilizado o entendimento do pesquisador, englobando o seu conhecimento prévio para uma ação referente a bibliografia estudada (THIOLLENT, 2009).

3.1.1 Processo das atividades

O processo a seguir foi modelado considerando as melhores práticas para o desenvolvimento viável do projeto, como pode ser observado na Figura 9.

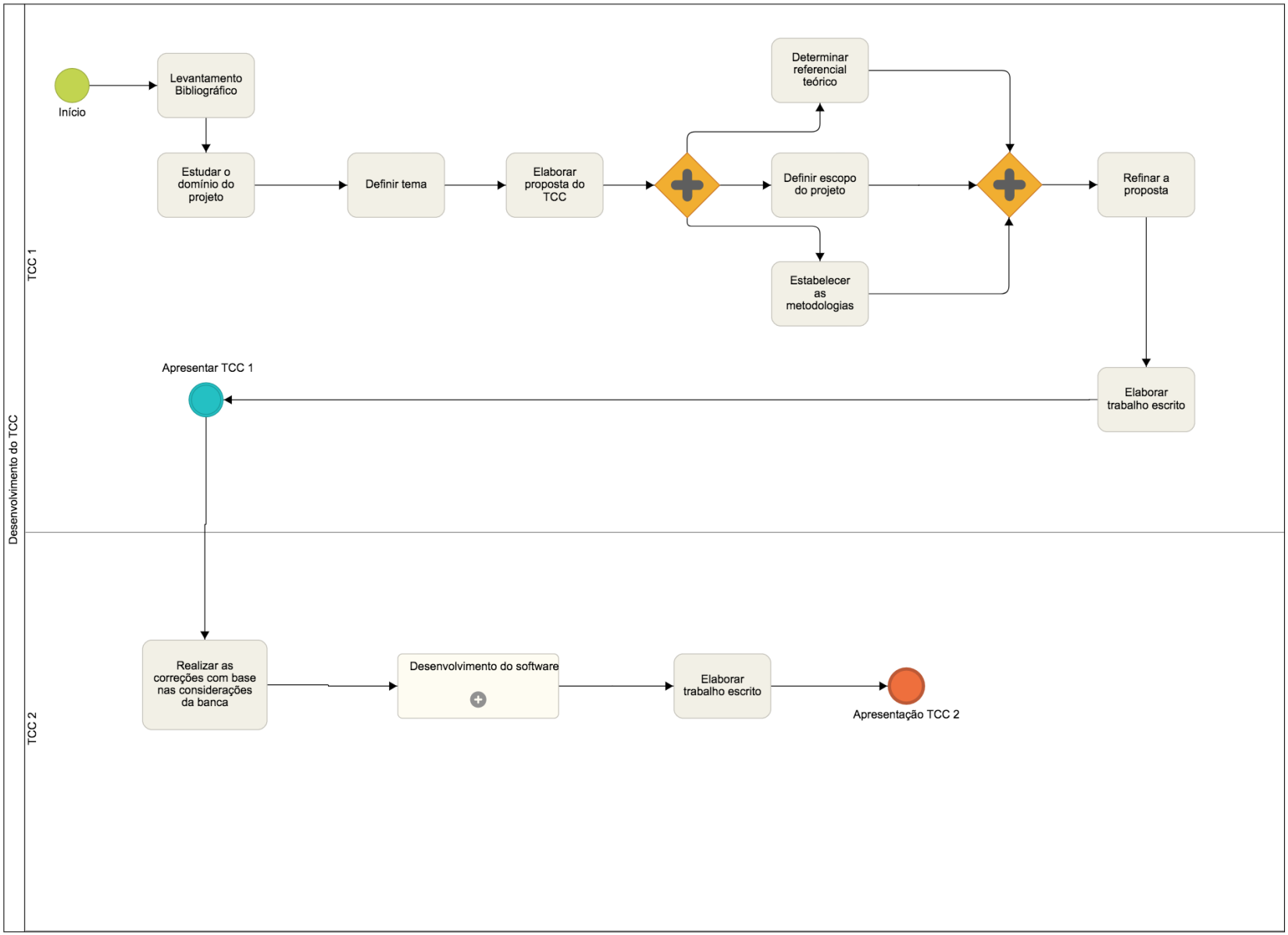

Figura 9 – Processo das atividades. Fonte: autor

A seguir são apresentados os esclarecimentos das atividades inerentes a elaboração da proposta inicial desse projeto (TCC 1):

- **Levantamento Bibliográfico**

  Realizar uma pesquisa sobre o tema a ser estudado e mapear uma lista de autores que fornecerão fundamentos as próximas atividades do processo.

- **Estudar o domínio do projeto**

  A partir de um conhecimento prévio, adquirido no levantamento bibliográfico, o objetivo será estudar o domínio do projeto de forma a contextualizar as ideias de alto nível.

- **Definir tema**

  O objetivo desta atividade é o alinhamento do conhecimento assimilado nas etapas anteriores com o *know how* do professor orientador e a partir disso definir um tema para este TCC. No caso foi definida uma solução que possibilitará a uma população participar de um processo eleitoral por meio do voto único entre as opções disponíveis.

- **Elaborar proposta do TCC**

  A proposta inicial do TCC compreende a formalização do tema definido junto com contexto, motivação, objetivos do trabalho e justificativas que encorajaram o desenvolvimento deste trabalho.

As atividades a seguir são executadas em paralelo.

- **Determinar referencial teórico**

  Definir uma base de textos, englobando artigos, livros e conhecimentos teóricos para embasar a elaboração textual do TCC.

- **Definir escopo do projeto**

  Esta atividade procura delimitar até onde o projeto irá prosseguir, a fim de estabelecer um projeto conciso para desenvolvimento e avaliação.

- **Estabelecer as metodologias**

  Tem por objetivo estabelecer as metodologias para o desenvolvimento tanto do trabalho teórico, quanto do trabalho prático, visando uma produção adequada do ponto de vista acadêmico, e com uma menor probabilidade da ocorrência de imprevistos na execução do processo.

- **Refinar a proposta**

  O refinamento da proposta visa identificar se o tema proposto, a contextualização tecnológica, metodologias utilizadas e o escopo estão bem definidos para a confecção do trabalho escrito.

- **Elaborar trabalho escrito**

  Esta atividade tem o objetivo de redigir o trabalho escrito, embasado por todas as atividades anteriores, com o intuito de ser apresentado à banca do TCC 1.

- **Apresentar TCC 1**

  Apresentar o trabalho escrito para a banca de professores avaliadores, a fim do trabalho ser avaliado e posto em discussão para possíveis evoluções.

A seguir são apresentadas as atividades inerentes a outra fase do projeto proposto como TCC (corresponde ao TCC2):

- **Realizar as correções com base nas considerações da banca**

  Esta primeira atividade do TCC 2 tem o objetivo de realizar as correções apontadas na apresentação do TCC 1.

- **Desenvolver o software**

  Este subprocesso é voltado para solução prática da proposta, onde será aprofundada na seção 5.

- **Testes e Manutenção**

  Atividade referente aos testes a serem realizados no software, os quais estão documentados na seção 5.6, onde é definido um plano de testes a fim de validar o software desenvolvido, além de realizar um teste do sistema em produção, em um contexto determinado. A manutenção busca realizar modificações no sistema desenvolvido no intuito de realizar melhorias e correções.

- **Elaborar trabalho escrito**

  Esta atividade possui o objetivo de detalhar o trabalho realizado no TCC2, em que na fase anterior o software foi realmente desenvolvido, além de se estabelecer como demonstrar os seus resultados.

- **Apresentação TCC 2**

  Esta é a última atividade do processo de desenvolvimento desse trabalho, em que a banca formada professores irá analisar o projeto.

3.2 Metodologia referente ao desenvolvimento do software

A metodologia utilizada para desenvolver o software corresponde a um subprocesso do Desenvolvimento do software, apresentado na parte relacionada a elaboração do TCC2, como pode ser observado na Figura 9.

Segundo Sommerville (2011), um processo de software é um conjunto de atividades relacionadas que levam à produção de um produto de software.

Não existe um processo ideal para o desenvolvimento de um software, pois os processos para o desenvolvimento de software são complexos e dependem dos indivíduos, os quais são responsáveis por colocar o processo em execução. Logo, eles estão diretamente ligados ao processo (SOMMERVILLE, 2011). Como não existe um processo ideal para a fabricação do software proposto será utilizado um processo híbrido entre as metodologias ágeis e tradicionais. Dentre as metodologias tradicionais foi selecionado o modelo incremental e dentre as metodologias ágeis foi escolhido o *Scrum* com algumas adaptações para melhor ser adequado ao contexto deste trabalho.

O processo de desenvolvimento do software está representado na Figura 10.

Dentro do processo adaptado do *Scrum*, os seguintes conceitos são definidos como:

Sprint's

O scrum define uma *sprint* de desenvolvimento como todas as atividades necessárias para entregar um incremento de softawre no final de um determinado período de tempo (SABBAGH, 2013).

Foram definidos *sprints* de 3 semanas, considerando o tempo para aprendizagem de novas tecnologias, a fim de alcançar o incremento de software ao final de cada *sprint*.

Papéis

- Scrum Master

  O *Scrum Master* busca assegurar que a equipe respeite e siga os valores e as práticas do *Scrum*, além de realizar o planejamento mais adequado para cada *sprint*. Disto posto, este papel irá ser desempenhado pelo próprio autor deste TCC e pelo seu professor orientador.

- Product Owner

  O *Product Owner* (PO) tem a responsabilidade de definir e priorizar as histórias de usuário, sempre se empenhando em manter a integridade conceitual de todas as funcionalidades. Este papel será desempenhado pelo autor deste trabalho.

- Time de Desenvolvimento

Responsável por desenvolver as *features* elicitadas para o desenvolvimento do software. Neste caso, o autor deste trabalho.

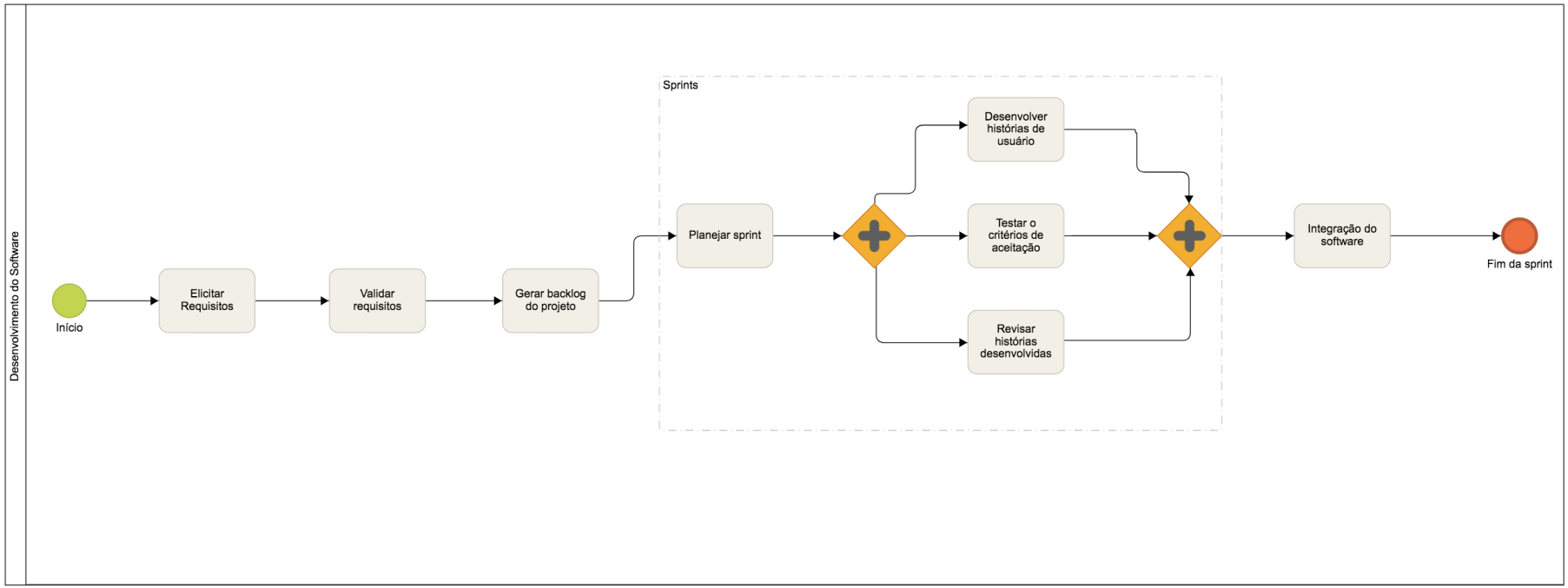

Figura 10 – Processo de desenvolvimento do software. Fonte: autor

As atividades presentes no processo de desenvolvimento do software proposto por este trabalho de conclusão de curso serão apresentadas a seguir.

- **Elicitar Requisitos**

  A elicitação de requisitos procura realizar a identificação dos requisitos para a elaboração do produto de software, utilizando os conceitos obtidos através do referencial teórico e do curso como um todo. A fim de um melhor detalhamento para a documentação do projeto foi definida a saída desta atividade, os artefatos: *Workflow*, Diagrama de Pacotes e o Diagrama de Classes para facilitar a manutenção e evolução do software, além de sua própria compreensão.

- **Validar Requisitos**

  Diante dos requisitos elicitados e documentos por meio de uma lista informal, estes serão validados com o professor orientador.

- **Gerar *backlog* do projeto**

  Esta atividade será destinada a documentar os requisitos elicitados, definindo os épicos e suas respectivas *features*, abrangendo todo o projeto e gerando então o *backlog* do produto. Vale ressaltar que ao início de cada *sprint* este documento pode ser atualizado e incrementado.

- **Planejar Sprint**

  O planejamento da *sprint* utiliza o *backlog* do projeto para dar insumo as histórias de usuários que serão desenvolvidas naquela *sprint*, as quais darão origem ao *backlog* da *sprint*. Como o time de desenvolvimento será composto por apenas um integrante, as atividades de avaliação do desempenho da equipe de desenvolvimento não fazem sentido neste contexto.

- **Desenvolver histórias de usuário**

  Nesta atividade o time de desenvolvimento produz as histórias de usuários contidas no *backlog* da *sprint*.

- **Testar os critérios de aceitação**

  Esta atividade busca realizar os testes dos critérios de aceitação da história de usuário que está sendo desenvolvida.

- **Revisar histórias desenvolvidas**

  Após a história de usuário ter sido desenvolvida e devidamente testada, deverá ser verificado se o que foi desenvolvido está de acordo com o contexto do produto final e com o restante do software.

- **Integração do software**

  Esta atividade busca integrar a história de usuário implementada com o restante do software, caracterizando um incremento de software.

- **Fim da sprint**

  Com todas as atividades concluídas, o fim da *sprint* é caracterizado. Caso ainda exista *sprints* a serem desenvolvidas, se voltará ao início de uma nova *sprint*, caso seja a última *sprint*, a atividade de desenvolvimento é encerrada.

3.3 Cronograma

Foram elaborados dois cronogramas com o intuito de realizar o planejamento da realização das atividades do processo de desenvolvimento deste trabalho de conclusão de curso, como podem ser observados nas Tabelas 1 e 2.

Foi elaborado um cronograma com o intuito de realizar o planejamento da realização das atividades do processo de desenvolvimento deste trabalho de conclusão de curso, como se pode observar na tabela 1.

Tabela 1 – Cronograma TCC 1 - 2018/1

Atividades	**Março**	**Abril**	**Maio**	**Junho**	**Julho**
Levantamento Bibliográfico	X	X	X	X	X
Estudar o domínio do projeto		X	X	X	
Definir tema		X	X		
Elaborar proposta do TCC		X	X		
Determinar referencial teórico		X	X	X	
Definir escopo do projeto			X		
Estabelecer as metodologias			X		
Refinar a proposta				X	X
Elaborar trabalho escrito			X	X	X
Apresentar TCC 1					X

Tabela 2 – Cronograma TCC 2 - 2019/2

Atividades	**Setembro**	**Outubro**	**Novembro**	**Dezembro**
Realizar as correções no trabalho escrito	X	X		
Desenvolver o software		X	X	
Testes e Manutenção		X	X	
Elaborar o trabalho escrito			X	
Apresentar o TCC 2				X

3.4 Suporte Tecnológico

Esta seção tem a finalidade de apresentar o suporte tecnológico utilizado para o desenvolvimento teórico e prático deste trabalho de conclusão de curso.

3.4.1 Ferramentas

3.4.1.1 Visual Studio Code

O Visual Studio Code[1] é um poderoso editor de código fonte disponível para todas as plataformas e possui diversas funções úteis para um desenvolvimento mais rápido e fácil, como o realce da sintaxe, autocompletar de palavras, *debug* de código, *git* integrado e a possibilidade de instalar extensões externas para ampliar as funcionalidades do editor.

O Visual Studio Code foi utilizado para desenvolver o código fonte do projeto.

3.4.1.2 IBM Blockchain Platform Extension for VS Code

Esta extensão desenvolvida pela IBM para o Visual Studio Code ajuda os desenvolvedores a criar, testar e apurar *smart contracts* conectados a ambientes desenvolvidos utilizando o Hyperledger Fabric, além de compilar as aplicações desenvolvidas em uma rede Blockchain local.

3.4.1.3 Git

O Git[2] é um sistema de controle de versão, utilizado para o controle de versões do código-fonte do projeto desenvolvido. Com a utilização do Git é possível o registro da evolução do software, com base nesse princípio a evolução e manutenção do software é facilitada.

3.4.1.4 GitHub

O GitHub[3] é uma aplicação web cuja função é a hospedagem de repositórios Git, além disso, o mesmo possibilita a visualização de funcionalidades (*Issues*) a serem desenvolvidas, revisão de código e a possibilidade do desenvolvimento colaborativo.

3.4.1.5 Kanban

O Kanban é um sistema que mapeia um fluxo em quadros, sendo esses quadros divididos por colunas representando o status de desenvolvimento do trabalho. O trabalho em si é representado por cartões que são distribuídos pelas colunas do quadro, sendo

[1] https://code.visualstudio.com/
[2] https://git-scm.com/
[3] https://github.com/

disponibilizados por colunas a partir do seu status em relação ao seu desenvolvimento (POPPENDIECK, 2012). O Kanban foi utilizado neste trabalho para dividir o fluxo de desenvolvimento de atividades por *sprint*, cujo conceito é mais explorado no capítulo 3 deste trabalho.

3.4.1.6 ZenHub

O ZenHub[4] é um *plug-in* para o GitHub, o qual proporciona a utilização do Kanban para a organização das histórias de usuário, além de possibilitar a obtenção das métricas sobre o time de desenvolvimento dentro da abordagem ágil, como por exemplo o alcance do *burndown* da equipe.

3.4.2 Heflo

O Heflo[5] é uma plataforma web, gratuita, da qual possibilita o gerenciamento de processos, utilizando o BPMN (*Business Process Modeling Notation*) para a modelagem de processos. Esta ferramenta foi utilizada para modelar o processo das atividades relacionadas ao desenvolvimento deste trabalho e para modelar o processo de desenvolvimento do software proposto.

3.4.2.1 Astah

O Astah[6] é um programa multiplataforma de modelagem UML (Linguagem de Modelagem Unificada). Ele foi utilizado para modelar os diagramas de pacote e classes do produto final deste trabalho.

3.4.3 Tecnologias para Desenvolvimento

3.4.3.1 Blockchain

O Blockchain foi a tecnologia adotada para o armazenamento das informações, visto que ela permite o armazenamento distribuído dos dados. É confiável e transparente bem como não pode ser adulterado. Isso tudo garante transações confiáveis, atingindo um alto nível de segurança e permitindo uma auditoria transparente do resultado das votações.

3.4.3.2 Hyperledger Fabric

O *framework* escolhido para aplicar o Blockchain no desenvolvimento da aplicação proposta por este TCC foi o Hyperledger Fabric pelo fato do mesmo prover uma solução

[4] https://www.zenhub.com/
[5] https://www.heflo.com/
[6] https://www.http://astah.net/

que faz o uso do Blockchain privado, em que os participantes são conhecidos por todos, além de permitir a customização dos *smart contracts* e consensos de segurança, da maneira mais adequada ao contexto do sistema.

3.4.3.3 Node.js

O Node.js[7], é um interpretador assíncrono de código JavasScript, desenvolvido para criar aplicações altamente escaláveis. Esta foi a principal linguagem de programação utilizada para o desenvolvimento do projeto.

3.4.3.4 Vue.js

O Vue.js[8] é um *framework* projetado para produzir interfaces adaptativas de usuário. O núcleo desta ferramenta tem o foco apenas na camada da visualização da interface de usuário e é facilmente integrada a partir de outras bibliotecas ou componentes previamente implementados. Esta ferramente foi utilizada para desenvolver o *front-end* do sistema proposto por este TCC.

Os componentes do Vue.js são um diferencial importante em que o seu funcionamento se baseia no desenvolvedor poder separar a página em componentes que possuem, cada um, seu próprio código em JavaScript, HTML e CSS, permitindo assim a reutilização destes componentes em outras partes da aplicação.

Uma das características mais distintas do Vue é seu sistema de reatividade não obstrutivo. Os modelos de dados são simplesmente objetos JavaScript puros e quando os modificam a camada visual se atualiza. Isto torna o gerenciamento de estado simples e intuitivo. Em comparação com o JavaScript puro ou até mesmo o jQuery, utilizar a reatividade do Vue é bem mais simples.

[7] https://nodejs.org/en/
[8] https://vuejs.org/

4 Desenvolvimento

Este capítulo irá apresentar o VotingSys, o sistema proposto por este trabalho.

4.1 Visão Geral

O VotingSys é um sistema que busca possibilitar à população uma solução capaz de permitir votações, dentro dos mais diversos contextos, por meio do voto direto. Para tal, ele necessita de uma arquitetura bastante segura e escalável, com a finalidade de obter a intenção de voto de uma população de maneira mais confiável possível, sempre visando a não identificação pública do voto (anonimato) e sua integridade.

Para obter a confiabilidade, autenticidade e integridade do sistema, a tecnologia escolhida foi o Blockchain, pois esta praticamente impossibilita a alteração da cadeia dos votos do banco de transações e toda inserção de um voto deverá ser matematicamente provada para ser validada.

O desenvolvimento do VotingSys foi divido em duas partes, a primeira responsável pelo *back-end* do sistema onde será utilizado a linguagem de programação Node.js, além de empregar o uso do *framework* Hyperledger Fabric, para incorporar o Blockchain como tecnologia para o armazenamento das transações. A segunda parte do sistema é responsável pela interface do usuário, desenvolvida utilizando a biblioteca Vue.js.

A mudança de escolha do Hyperledger Fabric, ao invés do Ethereum, se deu pelo fato da não necessidade de se utilizar algum esforço computacional para minerar uma criptomoeda com destino a obter um consenso em um estado de contabilidade, (O livro inteiro precisa ser validado antes que as transações sejam aprovadas), um exemplo de algoritmo que utiliza o esforço computacional é o *proof-of-work*, este fato acaba deixando a aplicação mais lenta com o tempo, já que a medida que o número de blocos aumenta, o tempo computacional aumenta proporcionalmente. O Hyperledger chega a um consenso no nível transacional, isso significa que para uma transação ser considerada válida, o bloco inteiro não precisa ser validado, mas apenas a transação, esta validação se da pela escolha do algoritmo de validação adotado pelo desenvolvedor.

Outro ponto fundamental para a escolha do Hyperledger foi a confiabilidade, pois o todas as transações utilizando o Ethereum e podem ser visualizadas através do website <https://etherscan.io/>, já o Hyperledger possui uma rede privada onde todos os participantes são conhecidos.

4.2 Backlog do Produto

Esta seção é destinada a documentar os requisitos do projeto, definindo os Épicos, *Features* e Histórias de Usuário a serem desenvolvidas.

4.2.1 Épicos

1. **EP01 - Exercer o direito de voto em uma eleição**

   Possibilitar que a população vote (participe diretamente) em uma eleição

2. **EP02 - Gerar relatórios com base nos resultados dos votos**

   Computar os votos sobre determinada eleição e gerar relatórios com base nos resultados, evidenciando pontos a serem explorados.

4.2.2 Features

1. **EP01.FT01 - Cadastrar usuários**

   Esta funcionalidade visa o cadastramento dos diversos tipos de usuário, como os usuários comuns (cidadãos que desejam votar em uma eleição) e usuários administradores (responsáveis pela administração do sistema).

2. **EP01.FT02 - Votar**

   Funcionalidade que permite um usuário comum a votar em uma eleição.

3. **EP02.FT01 - Apresentar relatórios**

   A funcionalidade deverá expor os resultados das votações e os dados dos inseridos no Blockchain para auditorias.

4.2.3 Histórias de Usuário

1. **EP01.FT01.US01 - Eu, como administrador, desejo criar um perfil de administrador a fim de utilizar as funcionalidades reservadas ao mesmo.**

2. **EP01.FT01.US02 - Eu, como usuário, desejo criar um perfil que corresponde a minha pessoa, para ser capaz de votar em um candidato.**

3. **EP01.FT01.US03 - Eu, como administrador, desejo ser capaz de cadastrar candidatos em uma eleição.**

4. **EP01.FT02.US01 - Eu, como administrador, desejo ser capaz de cadastrar uma eleição.**

5. **EP01.FT02.US02 - Eu, como usuário, desejo ser capaz de votar em qualquer candidato cadastrado no sistema.**
6. **EP02.FT01.US01 - Eu, como usuário, desejo ser capaz de visualizar o resultado da eleição, demonstrando o quantitativo de votos por candidato.**
7. **EP02.FT01.US02 - Eu, como administrador, desejo ser capaz de visualizar um relatório apresentando todos os blocos do Blockchain.**
8. **EP02.FT01.US03 - Eu, como administrador, desejo ser capaz de visualizar um relatório apresentando os blocos do Blockchain referentes a uma palavra de busca.**

4.3 Arquitetura

Esta seção busca apresentar a arquitetura do VotingSys, explicitando como o mesmo foi planejado e como o seu funcionamento interno opera, com objetivo de facilitar o entendimento do software elaborado.

4.3.1 Workflow

A fim de documentar o ciclo de vida desse software foi utilizado os conceitos de *workflow*, pois os mesmos são utilizados para modelar, melhorar e automatizar os processos de negócio de um determinado sistema. Por meio dele é possível demonstrar uma automatização total ou parcial de um processo de negócio, neste caso o ciclo de vida do VotingSys (SILVA L. S. SOARES, 2006).

Na Figura 11, é possível visualizar a camada do usuário interagindo com a interface web, desenvolvida utilizando a biblioteca Vue.js. Essa camada interage com o servidor *back-end* implementado em Node.js e por fim com o servidor que mantém os dados utilizando o Hyperledger Fabric.

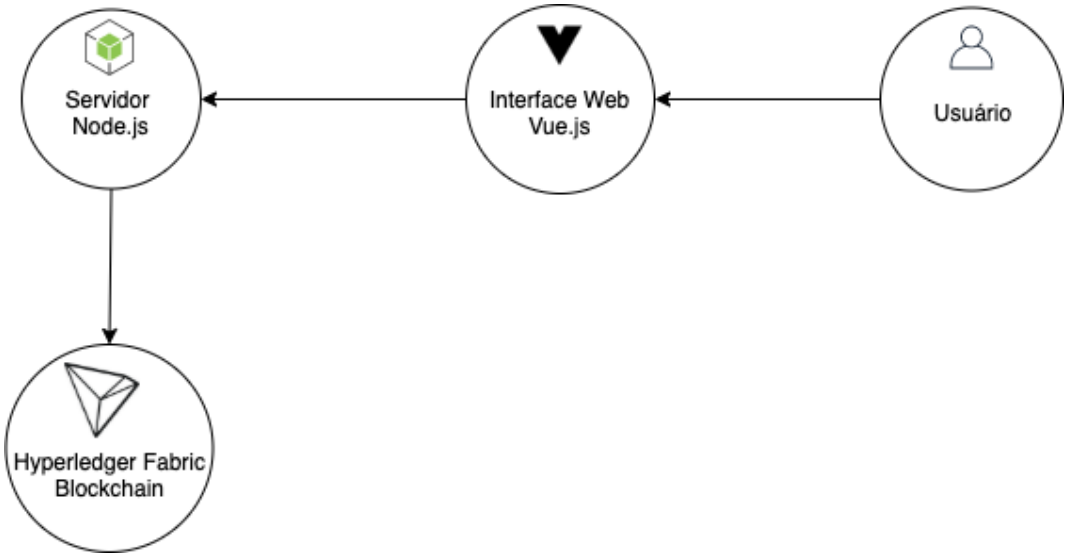

Figura 11 – Workflow do VotingSys. Fonte: autor

4.3.2 Visão de Pacotes

Uma visão de pacotes é apropriada para aprofundar a assimilação do sistema como um todo, pois é possível verificar a conectividade entre o Blockchain e o restante do software, além de demonstrar a conectividade entre o *front-end* e *back-end* do sistema. O diagrama de pacotes foi escolhido para demonstrar esta visão e está evidenciado na Figura 12.

É importante ressaltar que a arquitetura escolhida para o desenvolvimento do projeto foi a *Model View Controller Blockchain*(MVC-B), onde (CUOMO J. A. LATONE, 2016):

- **Model**: interface que gerencia os dados fora da *chaincode*, incluindo documentos e arquivos.
- **View**: é uma interface cuja função é interagir com a lógica obtida a partir da controladora através interface de usuário web ou aplicação *mobile*.
- **Controller**: coordena os dados entre a interface de usuário, modelo de dados e APIs para conduzir as transações e o *chaincode*.
- **Blockchain**: esta camada é uma extensão da lógica de controle e do modelo de dados, dentro do domínio do Blockchain. A lógica de controle é aprimorada pelo *chaincode* e o modelo de dados é aprimorado através das transações no Blockchain.

Este padrão de desenvolvimento adapta o padrão MVC padrão para a utlização do Blockchain em uma aplicação, mesmo com essa adaptação, o padrão alcança os cinco princípios da programação orientada a objetos, o SOLID (FUCOLO, 2017):

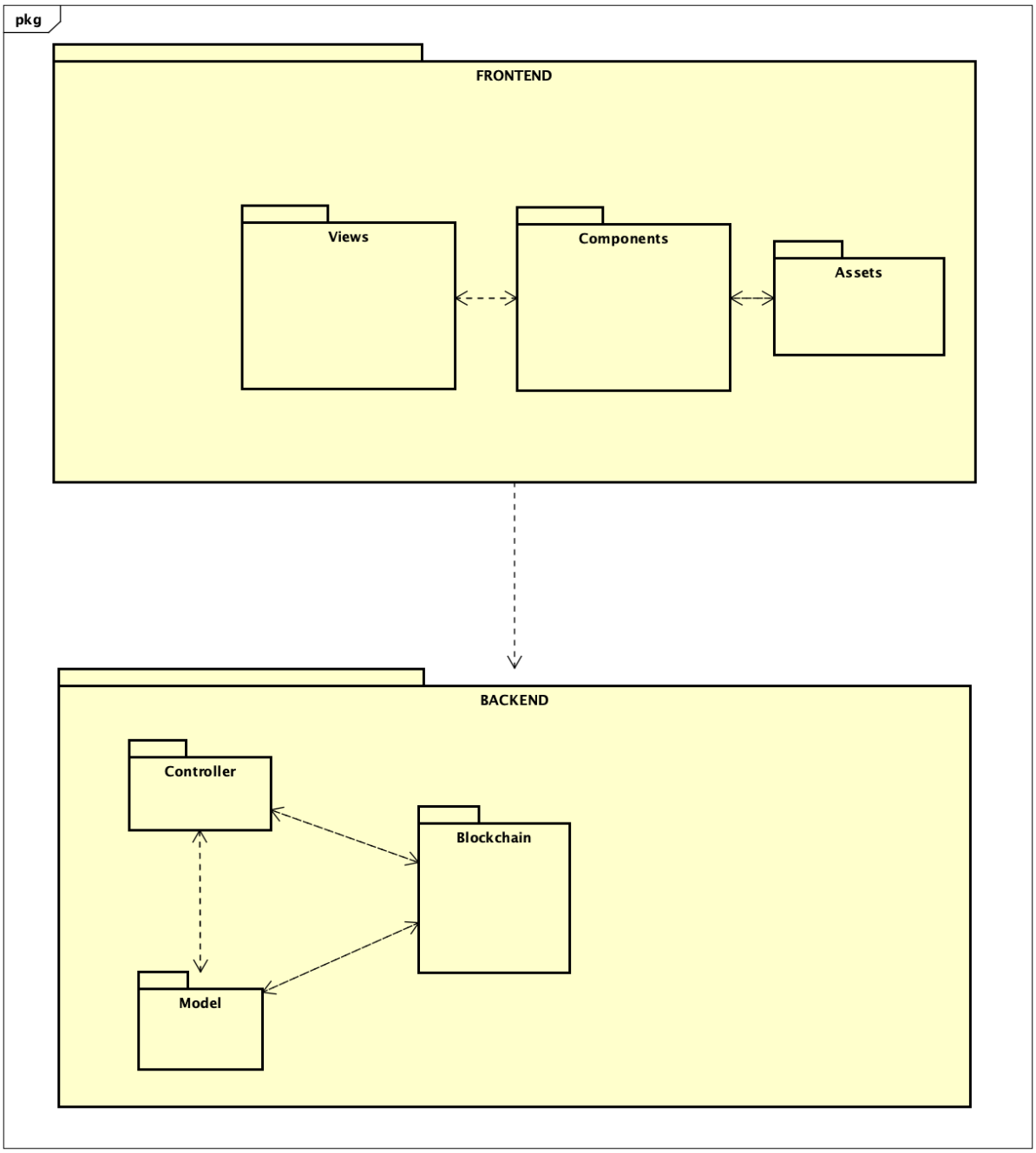

Figura 12 – Diagrama de Pacotes do VotingSys. Fonte: autor

S : SRP — Single Responsibility Principle

O princípio da responsabilidade única estabelece que uma classe deve possuir apenas uma finalidade, e realizá-la corretamente.

O : OCP — Open/closed principle

O princípio aberto/fechado define que uma classe deve ser aberta para extensões e fechada para modificações.

L : LSP — Liskov substitution principle

O princípio da substituição de Liskov recomenda que uma classe filha possa ser substituída por sua classe mãe.

I : ISP — Interface segregation principle

O princípio da segregação da interface indica para não forçar os usuários a utilizarem interfaces que não utilizam.

D : DIP — Dependency inversion principle

O princípio da inversão de dependência aconselha que classes dependam de classes abstratas e não concretas.

Aplicando os conceitos do SOLID é possível desenvolver um código extensível, coeso e de fácil manutenção.

O pacote do *Blockchain* irá armazenar os dados criados e modificados pela camada da *model* e disseminá-los em uma rede distribuída.

4.3.3 Diagrama de Classes

Buscando uma visão mais aprofundada do projeto, exibindo as entidades, de cada pacote já evidenciado e como eles se relacionam. Com o diagrama de classes é possível visualizar como as classes do sistema irão se comportar em relação ao software como um todo, como pode ser observado na Figura 13.

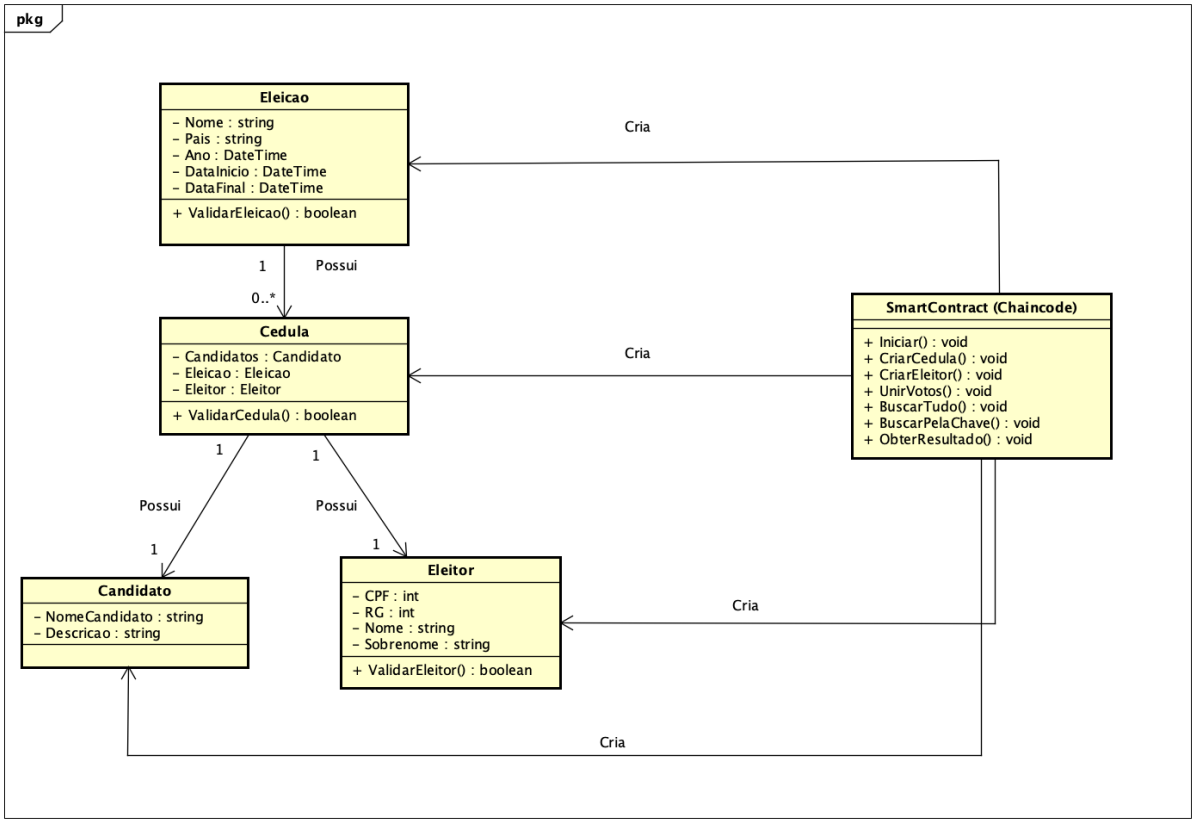

Figura 13 – Diagrama de Classes do VotingSys. Fonte: autor

- **Eleição**

  A classe Eleição é responsável por definir um tipo de objeto referente a uma eleição, possuindo os atributos Nome (atributo do tipo texto relacionado ao título da eleição), País (atributo do tipo texto relacionado ao país onde a eleição será realizada), Ano (atributo do tipo data relacionado ao ano em que a eleição será realizada), DataInicio (atributo do tipo data relacionado a data de início da eleição) e DataFinal (atributo do tipo data relacionado a data de término da eleição). Os atributos DataInicio e DataFinal definem o intervalo de tempo permitido para um eleitor enviar o seu voto. A função para validar esta classe verifica se os atributos inseridos para instanciar uma Eleição são válidos.

- **Candidato**

  A classe Candidato é responsável por definir um tipo de objeto referente a um candidato, possuindo os atributos NomeCandidato (atributo do tipo texto relacionado ao nome do candidato) e Descricao (atributo do tipo texto relacionado a uma descrição à respeito do candidato). Esta classe não possui uma função para validar o candidato, pois os candidatos são definidos através da função Iniciar da classe SmartContract, ou seja, os candidatos são cadastrados pelo administrador do sis-

tema no momento em que o Blockchain é instanciado, lembrando do fato de que a base de transações não pode ser modificada, uma eleição não pode modificar os seus candidatos.

- **Eleitor**

  A classe Eleitor é responsável por definir um tipo de objeto referente a um eleitor, possuindo os atributos CPF (atributo do tipo inteiro relacionado ao CPF do eleitor), RG (atributo do tipo inteiro relacionado ao documento de identidade do eleitor), Nome (atributo do tipo texto relacionado ao nome do eleitor) e Sobrenome (atributo do tipo texto relacionado ao Sobrenome do eleitor). A função para validar esta classe verifica se os atributos inseridos para instanciar um Eleitor são válidos.

- **Cédula**

  A classe Cédula é responsável por definir um tipo de objeto referente a uma cédula, possuindo os atributos Candidatos (atributo referente a um *array* do tipo Candidato representando o candidato escolhido pelo eleitor), Eleição ((atributo referente a eleição no qual a cédula se encontra) e Eleitor (atributo referente ao eleitor autenticado pelo sistema). A função para validar a cédula verifica se o eleitor já votou e se o mesmo está cadastrado no sistema.

- **SmartContract**

  Esta classe é responsável por construir o Blockchain e instanciar todas as outras classes do sistema. A função Iniciar gera o bloco gênesis criando um objeto do tipo Eleição e cadastra os candidatos. A função CriarCedula gera uma nova cédula e atualiza a contagem de votos do candidato. A função CriarEleitor gera um novo Eleitor baseado nos seus atributos. A função UnirVotos primeiramente verifica se o Eleitor ainda não votou e depois verifica se a data da eleição é válida, a partir dos dados da eleição e data atual do computador, em caso de sucesso um novo voto é contabilizado para o candidato escolhido. Por fim, o estado da aplicação é atualizado. A função BuscarTudo retorna todos os pares chave e valor dentro do Blockchain.A função BuscarPelaChave retorna os dados solicitados pela *string* de busca. E a função ObterResultados envia a contagem de votos por candidatos à interface de usuário.

4.4 Auditoria

Com as tecnologias utilizadas no projeto será possibilitada as auditorias tanto da contabilização dos votos, quanto do código-fonte do sistema.

A auditoria da contabilização dos votos, será possibilitada pela aplicação do Blockchain na arquitetura do projeto. Onde cada nó da rede irá guardar uma cópia dos dados,

garantindo assim que todos os nós possam criar um consenso sobre a veracidade das votações.

De modo que o sistema foi implementado utilizando o *framework* Hyperledger Fabric, este possibilita a utilização da linguagem SQL-like para realizar operações de busca nos dados armazenados pelo Blockchain. A partir do momento que um usuário administrador realize o login no VotingSys, o mesmo é habilitado a utilizar esta funcionalidade, a qual facilita a realização de auditorias, porém esta auditoria deve ser realizada por indivíduos capacitados, possuindo o conhecimento necessário para interpretar os dados obtidos.

A auditoria do código-fonte do projeto será possibilitada pelo fato do código estar disponível na plataforma de código aberto GitHub, possibilitando assim, a verificação das técnicas de programação que foram utilizadas corretamente e a identificação de falhas ou códigos que não fazem parte do propósito do sistema.

Aliado ao fato do software estar disponibilizado no GitHub e possuir uma licença *open-source*, ele poderá ser auditado por qualquer pessoa, possibilitando o *feedback* em relação a todos os pontos em seu desenvolvimento.

4.5 Consenso

Como o Hyperledger Fabric é um *framework* modularizado, o algoritmo de consenso para realizar a validação das transições foi o *Raft Order*, este algoritmo é um consenso tolerante a falhas de parada. O Raft utiliza a replicação de registros utilizando a replicação de máquina de estados que, em conjunto com o protocolo de consenso, garante a existência dos mesmos comandos na mesma ordem em todos as máquinas de estados dos nós, garantindo assim a integridade dos registros e o funcionamento do consenso com até f participantes em estado de falha de parada dos n = 2 f +1 participantes da rede (SCHNEIDER, 1990).

4.6 Validação

O objetivo da validação é validar se um produto de software atenderá a seu objetivo quando colocado no ambiente para o qual o mesmo foi desenvolvido (SOMMERVILLE, 2011).

O MPS.BR vê de forma geral que o processo de validação tem seu foco em avaliar a qualidade de um produto ou componente de produto, assegurando que os objetivos e ou necessidades dos clientes sejam atendidas quando colocado em um ambiente de produção, ou seja, o objetivo da validação é garantir que o produto correto está sendo desenvolvido (SOFTEX, 2016).

A ferramenta a ser utilizada para realizar os testes funcionais unitários será o *framework* de testes Mocha. Ele por sua vez é uma ferramenta de testes de Javascript, rodando em Node.js. Os testes Mocha são executados em série, permitindo relatórios flexíveis e precisos, enquanto mapeiam exceções não capturadas para os casos de teste corretos. Como o *back-end* do sistema foi desenvolvido em Node.js, o Mocha se encaixa perfeitamente no cenário m que o VotingSys se encontra para executar os seus testes unitários.

Para validar o VotingSys foi utilizada uma técnica dinâmica de avaliação, sendo criado um plano de teste cujo conceito é funcionar como um guia para a execução e implementação dos testes, além de ser também um guia para o próprio desenvolvimento do código. O objetivo deste artefato é definir quais serão as técnicas, as ferramentas e os critérios aceitáveis para a saída de cada um dos testes que será desenvolvido (BOURQUE, 2004).

Este plano de testes contém as seguintes informações:

1. Qual o objetivo do teste, real valor da funcionalidade do ponto de vista do cliente.
2. Foco de qualidade
3. O nível de teste
4. A técnica utilizada
5. O ambiente em qual será realizado os testes
6. As pré-condições para executar o teste
7. A descrição para realizar o teste
8. Os resultados esperados
9. Um campo que contém a aprovação ou não do teste.

4.6.1 Plano de Testes

Para organizar o plano de testes foram priorizadas algumas das histórias de usuário definidas na seção 5.2.3 para serem testadas, a partir do grau de importância em relação à execução do sistema como um todo. Logo, foram escolhidas as seguintes histórias:

- **EP01.FT01.US02** - Eu, como usuário, desejo criar um perfil que corresponde a minha pessoa, para ser capaz de votar em um candidato.
- **EP01.FT02.US02** - Eu, como usuário, desejo ser capaz de votar em qualquer candidato cadastrado no sistema.

- **EP01.FT01.US03** - Eu, como administrador, desejo ser capaz de cadastrar candidatos a uma eleição.
- **EP02.FT01.US01** - Eu, como usuário, desejo ser capaz de visualizar o resultado da eleição, demonstrando o quantitativo de votos por candidato.
- **EP01.FT02.US01** - Eu, como administrador, desejo ser capaz de cadastrar uma eleição.
- **EP02.FT01.US02** - Eu, como usuário, desejo ser capaz de visualizar um relatório apresentando todos os blocos do Blockchain.

A partir das histórias priorizadas, foi definido os seguintes casos de teste:

- **TC01 - Criar um usuário eleitor**

Tabela 3 – TC01 - Criar um usuário eleitor

Objetivo	O objetivo deste teste é cadastrar um eleitor no sistema com sucesso
Foco de Qualidade	Funcional
Nível	Sistema
Técnica	Caixa preta
Ambiente	Ambiente de desenvolvimento
Pré-Condições	Usuário dentro da página inicial
Descrição	1º. O usuário acessa a página inicial do sistema 2º. O usuário preenche o formulário de cadastro com os campos CPF, RG, Nome e Sobrenome.
Saídas	O sistema irá informar a mensagem: Eleitor com o CPF XXX foi adicionado. Utilize o seu CPF para realizar o login acima.
Status	OK
Defeito	N/A

- **TC02 - CPF já cadastrado**

Tabela 4 – TC02 - CPF já cadastrado

Objetivo	O objetivo deste teste é tentar cadastrar um usuário com um CPF já cadastrado no sistema
Foco de Qualidade	Funcional
Nível	Sistema
Técnica	Caixa preta
Ambiente	Ambiente de desenvolvimento
Pré-Condições	Usuário dentro da página inicial
Descrição	1º. O usuário acessa a página inicial do sistema 2º. O usuário preenche o formulário de cadastro com os campos CPF, RG, Nome e Sobrenome.
Saídas	O sistema irá informar a mensagem: Error! Já existe um usuário com o CPF XXX registrado. Favor inserir um CPF diferente.
Status	OK
Defeito	N/A

- **TC03 - Votar em um candidato**

Tabela 5 – TC03 - Votar em um candidato

Objetivo	O objetivo deste teste é um eleitor votar em um candidato com sucesso
Foco de Qualidade	Funcional
Nível	Sistema
Técnica	Caixa preta
Ambiente	Ambiente de desenvolvimento
Pré-Condições	Usuário dentro da página inicial
Descrição	1º. O usuário acessa a página inicial do sistema 2º. O usuário digita o seu CPF para realizar o login 3º. O usuário seleciona o candidato no qual deseja votar 4º. O usuário informa o seu CPF novamente 5º. O usuário clica em votar
Saídas	O sistema irá informar a mensagem: Voto registrado com sucesso para o candidato XXX votado pelo candidato de CPF XXX. Obrigado por participar da votação!
Status	OK
Defeito	N/A

- **TC04 - Eleitor já enviou o seu voto**

Tabela 6 – TC04 - Eleitor já enviou o seu voto

Objetivo	O objetivo deste teste é tentar votar novamente com um eleitor que já votou
Foco de Qualidade	Funcional
Nível	Sistema
Técnica	Caixa preta
Ambiente	Ambiente de desenvolvimento
Pré-Condições	Usuário dentro da página inicial
Descrição	1º. O usuário acessa a página inicial do sistema 2º. O usuário digita o seu CPF para realizar o login
Saídas	O sistema irá informar a mensagem: Este eleitor já votou, é aceito apenas um voto por eleitor!
Status	OK
Defeito	N/A

- **TC05 - Fora do período de eleição**

Tabela 7 – TC05 - Fora do período de eleição

Objetivo	O objetivo deste teste é tentar votar fora do período de eleição
Foco de Qualidade	Funcional
Nível	Sistema
Técnica	Caixa preta
Ambiente	Ambiente de desenvolvimento
Pré-Condições	Usuário dentro da página inicial
Descrição	1º. O usuário acessa a página inicial do sistema 2º. O usuário digita o seu CPF para realizar o login
Saídas	O sistema irá informar a mensagem: Eleição fechada!
Status	OK
Defeito	N/A

- **TC06 - Visualizar resultado das eleições**

Tabela 8 – TC06 - Visualizar resultado das eleições

Objetivo	O objetivo deste teste é o usuário ser capaz de visualizar o resultado das eleições
Foco de Qualidade	Funcional
Nível	Sistema
Técnica	Caixa preta
Ambiente	Ambiente de desenvolvimento
Pré-Condições	Usuário dentro da página inicial
Descrição	1º. O usuário acessa a página inicial do sistema 2º. O usuário clica no botão: Resultados 3º. O usuário clica em Obter Resultado
Saídas	O sistema irá renderizar um gráfico com a contagem dos votos por candidato
Status	OK
Defeito	N/A

- **TC07 - URL inválida para os resultados**

Tabela 9 – TC07 - URL inválida para os resultados

Objetivo	O objetivo deste teste é tentar acessar os resultados com uma URL inválida
Foco de Qualidade	Funcional
Nível	Sistema
Técnica	Caixa preta
Ambiente	Ambiente de desenvolvimento
Pré-Condições	Usuário dentro da página inicial
Descrição	1º. O usuário acessa a página inicial do sistema 2º. O usuário tenta acessar a URL inválida dos resultados
Saídas	O sistema irá mostrar a mensagem do erro 404.
Status	OK
Defeito	N/A

- **TC08 - Visualizar um relatório apresentando todos os blocos do Blockchain**

Tabela 10 – TC08 - Visualizar um relatório apresentando todos os blocos do Blockchain

Objetivo	O objetivo deste teste é o usuário ser capaz de visualizar todos os blocos do Blockchain
Foco de Qualidade	Funcional
Nível	Sistema
Técnica	Caixa preta
Ambiente	Ambiente de desenvolvimento
Pré-Condições	Usuário dentro da página inicial
Descrição	1º. O usuário acessa a página inicial do sistema 2º. O usuário clica no link: Buscar 3º. O usuário clica no botão: Buscar
Saídas	O sistema irá apresentar uma tela com todos os blocos do Blockchain
Status	OK
Defeito	N/A

- **TC09 - URL inválida para os dados do Blockchain**

Tabela 11 – TC09 - URL inválida para os dados do Blockchain

Objetivo	O objetivo deste teste é tentar acessar os dados do Blockchain com uma URL inválida
Foco de Qualidade	Funcional
Nível	Sistema
Técnica	Caixa preta
Ambiente	Ambiente de desenvolvimento
Pré-Condições	Usuário dentro da página inicial
Descrição	1º. O usuário acessa a página inicial do sistema 2º. O usuário tenta acessar a URL inválida dos dados do Blockchain
Saídas	O sistema irá mostrar a mensagem do erro 404.
Status	OK
Defeito	N/A

- **TC10 - Criar um objeto do tipo Eleitor**

Tabela 12 – TC10 - Criar um objeto do tipo Eleitor

Objetivo	Verificar se a função responsável por criar um eleitor, o cadastra com todos os atributos desejados
Foco de Qualidade	Funcional
Nível	Unitário
Técnica	Caixa preta
Ambiente	Ambiente de desenvolvimento
Pré-Condições	N/A
Entradas	- 02954492141 - 2968544 - Gustavo - Moreira
Saídas	- Objeto Eleitor criado com sucesso, com todos os seus atributos
Status	OK
Defeito	N/A

- **TC11 - Criar um objeto do tipo Eleição**

Tabela 13 – TC11 - Criar um objeto do tipo Eleição

Objetivo	Verificar se a função responsável por criar uma eleição, o cadastra com todos os atributos desejados
Foco de Qualidade	Funcional
Nível	Unitário
Técnica	Caixa preta
Ambiente	Ambiente de desenvolvimento
Pré-Condições	N/A
Entradas	- Eleicao2 - Eleição para diretor da FGA - Brasil - 2019 - 01/11/2019 - 29/12/2019
Saídas	- Objeto Eleição criado com sucesso, com todos os seus atributos
Status	OK
Defeito	N/A

- **TC12 - Criar um objeto do tipo Cédula**

Tabela 14 – TC12 - Criar um objeto do tipo Cédula

Objetivo	Verificar se a função responsável por criar uma cédula, o cadastra com todos os atributos desejados
Foco de Qualidade	Funcional
Nível	Unitário
Técnica	Caixa preta
Ambiente	Ambiente de desenvolvimento
Pré-Condições	N/A
Entradas	- Candidato1 - Eleição para diretor da FGA - 02954491153
Saídas	- Cédula criada com sucesso!
Status	OK
Defeito	N/A

- **TC13 - Criar um objeto do tipo Candidato**

Tabela 15 – TC13 - Criar um objeto do tipo Candidato

Objetivo	Verificar se a função responsável por criar uma Candidato, o cadastra com todos os atributos desejados
Foco de Qualidade	Funcional
Nível	Unitário
Técnica	Caixa preta
Ambiente	Ambiente de desenvolvimento
Pré-Condições	N/A
Entradas	- Vandor - Candidato de Software
Saídas	- Candidato criado com sucesso!
Status	OK
Defeito	N/A

- **TC14 - Criar um SmartContract**

Tabela 16 – TC14 - Criar um SmartContract

Objetivo	Verificar se a função responsável por criar o SmartContract, o cria com todos os seus atributos
Foco de Qualidade	Funcional
Nível	Unitário
Técnica	Caixa preta
Ambiente	Ambiente de desenvolvimento
Pré-Condições	- Contrato 1 criado - Contrato 2 criado
Entradas	- 3 - Contrato 3
Saídas	- Contrato criado com sucesso!
Status	OK
Defeito	N/A

- **TC15 - Criar um SmartContract já existente**

Tabela 17 – TC15 - Criar um SmartContract já existente

Objetivo	Verificar se a função responsável por criar o SmartContract, cria contratos repetidos
Foco de Qualidade	Funcional
Nível	Unitário
Técnica	Caixa preta
Ambiente	Ambiente de desenvolvimento
Pré-Condições	- Contrato 1 criado - Contrato 2 criado
Entradas	- 1 - Contrato 1
Saídas	- Contrato já existente!
Status	OK
Defeito	N/A

- **TC16 - Ler um SmartContract**

Tabela 18 – TC16 - Ler um SmartContract

Objetivo	Verificar se a função responsável por retornar um SmartContract, o retorna
Foco de Qualidade	Funcional
Nível	Unitário
Técnica	Caixa preta
Ambiente	Ambiente de desenvolvimento
Pré-Condições	- Contrato 1 criado
Entradas	- 1
Saídas	- Retorno: Contrato 1
Status	OK
Defeito	N/A

- **TC17 - Deletar um SmartContract**

Tabela 19 – TC17 - Deletar um SmartContract

Objetivo	Verificar se a função responsável por deletar um SmartContract, o deleta
Foco de Qualidade	Funcional
Nível	Unitário
Técnica	Caixa preta
Ambiente	Ambiente de desenvolvimento
Pré-Condições	- Contrato 1 criado
Entradas	- 1
Saídas	- Contrato 1 deletado!
Status	OK
Defeito	N/A

4.7 Metodologias

Em relação as metodologias empregadas para o desenvolvimento do trabalho, foi determinada uma base o conhecimento adquirido a partir do referencial teórico, explicitando o conhecimento necessário para a produção da solução proposta, conjunto com uma metodologia híbrida entre os métodos ágeis e tradicionais com o intuito de produzir o VotingSys de maneira eficaz e bem documentada, bem como as melhores práticas da Engenharia de Software as descrevem.

4.8 Aplicação Desenvolvida

Após o período de implementação definido no cronograma do TCC2, foram desenvolvidos as seguintes histórias de usuário:

- EP01.FT01.US01
- EP01.FT01.US02
- EP01.FT01.US03
- EP01.FT02.US01
- EP01.FT02.US02
- EP02.FT01.US01
- EP02.FT01.US02
- EP02.FT01.US03

Pode-se encontrar algumas imagens da aplicação desenvolvida no Apêndice A deste trabalho.

4.8.1 Requisitos e Arquitetura

Para a documentação e auxílio a compreensão dos requisitos e arquitetura do projeto foram desenvolvidos os seguintes artefatos:

Backlog do produto: presente na seção 5.2

WorkFlow: presente na seção 5.3.1

Diagrama de Pacotes: presente na seção 5.3.2

Diagrama de Classes: presente na seção 5.3.3

4.8.2 Pré-Requisitos

Os pré-requisitos para executar o sistema são:

- Visual Studio Code
- Extensão IBM Blockchain Platform para o Visual Studio Code
- Node v8.0 ou superior
- Docker

4.8.3 Especificações Técnicas

As especificações técnicas da máquina, a qual foi utilizada para a realização do desenvolvimento e execução do sistema proposto por este trabalho foram, um processador Intel Core i7, modelo 4770HQ de 2.2Ghz e 16Gb de memória RAM. O sistema operacional utilizado foi o macOS Catalina, versão 10.15.1 de 64 bits.

4.8.4 O Código

O código do sistema foi dividido em dois módulos, um referente ao *back-end* desenvolvido em Node.js e o outro referente ao *front-end* desenvolvido em Vue.js.

- O *back-end* é responsável por definir todas as classes modelo utilizadas no sistema, além de definir o contrato cuja responsabilidade é organizar todo o sistema para o Blockchain ser implantado e executar as suas funções.
- O *front-end* é responsável por definir as páginas de interface do usuário, componentes utilizados pelo Vue.js e *plug-ins* aplicados a experiência de usuário.

A classe responsável por definir o *smart-contract* é a mais importante do sistema, a mesma estende as funcionalidades da classe *Contract*, a classe base implementada pelo *framework* Hyperledger Fabric, implementando assim o algoritmo de consenso utilizado para autorizar um novo voto a ser computado e todos os outros dados que são inseridos no Blockchain. Esta implementação possibilita a iniciação do sistema, definindo os parâmetros (Eleição e Candidatos) para o sistema ser iniciado.

4.8.5 Testes Unitários

Foram realizados testes de sistema e unitários para garantir o funcionamento dos métodos do código-fonte e da utilização do sistema como definido no plano de testes, disponibilizado na seção 5.5.1. A Figura 14 mostra a cobertura de código em 77,78%.

```
> nyc mocha --recursive



  SmartContract
    #CriarContrato
      ✓ Contrato criado com sucesso
      ✓ Contrato já existente!
    #LerContrato
      ✓ Deve retornar o contrato
    #DeletarContrato
      ✓ Deve deletar o contrato
    #iniciar
Chamada da instância do contrato
query String
"{\"selector\":{\"type\":\"election\"}}"
      1) Deve retornar um array com dois valores
    #Eleitor
      ✓ Objeto Eleitor criado com sucesso, com todos os seus atributos
    #Eleicao
      ✓ Objeto Eleição criado com sucesso, com todos os seus atributos
    #Cedula
Não existe um CPF cadastrado para este eleitor.
      ✓ Cédula criada com sucesso!
    #Candidato
      ✓ Candidato criado com sucesso!


  8 passing (212ms)
  1 failing

  1) SmartContract
       #iniciar
         Deve retornar um array com dois valores:
     TypeError: Cannot read property 'next' of undefined
      at MyAssetContract.queryWithQueryString (lib/voterContract.js:117:90)
      at process._tickCallback (internal/process/next_tick.js:68:7)

------------------------------ Coverage summary ------------------------------
Statements   : 43.84% ( 96/219 )
Branches     : 32.26% ( 20/62 )
Functions    : 77.78% ( 14/18 )
Lines        : 44.24% ( 96/217 )
------------------------------------------------------------------------------
```

Figura 14 – Resultado dos testes unitários. Fonte: autor

4.8.6 Tutorial

Um tutorial foi elaborado para orientar o usuário de como executar e configurar o VoptingSys, podendo o mesmo ser acessado no endereço virtual <https://github.com/gustavonascimento/VotingSys>

4.9 Experimento com o Sistema

Para realizar um experimento no sistema foi idealizado o seguinte contexto: eleger o que os participantes do processo de votação prefeririam fazer no período de férias escolas,

estando disponibilizadas as três opções de voto: "Viajar", "Descansar"e "Realizar um curso de aperfeiçoamento em Banco de Dados".

Após o contexto definido foi determinada uma data de eleição e a turma de alunos destinada a particpar desse processo. A turma escolhida foi da disciplina de Sistemas de Banco de Dados 2 da Universidade de Brasília no Gama (UnB/FGA) do semestre corrente (2019/2).

Para realizar este teste a máquina descrita na seção 5.8.3 foi utilizada como o ambiente de produção.

4.9.1 Resultados do Teste

4.9.1.1 Interface de Usuário

Em uma versão alfa do sistema os formulários de registro e acesso estavam na mesma página, sendo verificado que esta experiência do usuário poderia dificultar o entendimento e escolha dos formulários corretos na interação como o mesmo, além de muita navegação incorreta para os objetivos do eleitor (mais esforço desnecessário). Já na versão beta, utilizada neste experimento, foi utilizada uma interface diferente e melhor elaborada para atender as características dos usuários (eleitores e administrador). Assim, existia uma interface específica para os eleitores votarem e outra para o registro do usuário, como pode ser observada nas Figuras 16 e 17, respectivamente.

Porém, no decorrer do teste foi observada uma dificuldade dos usuários no preenchimento do campo "CPF"para realizar o acesso a página de votação, após o cadastramento do mesmo na página de registro. Essa situação produziu reflexões ao desenvolvedor que procurou tornar a interface mais instrutiva e significativa as ações dos usuários.

4.9.1.2 Realização das Votações

Todos os participantes desse experimento conseguiram enviar os seus votos na respectiva opção desejada, obtendo assim o resultado da eleição da preferência do que fazer nas férias, como pode ser observado na Figura 15.

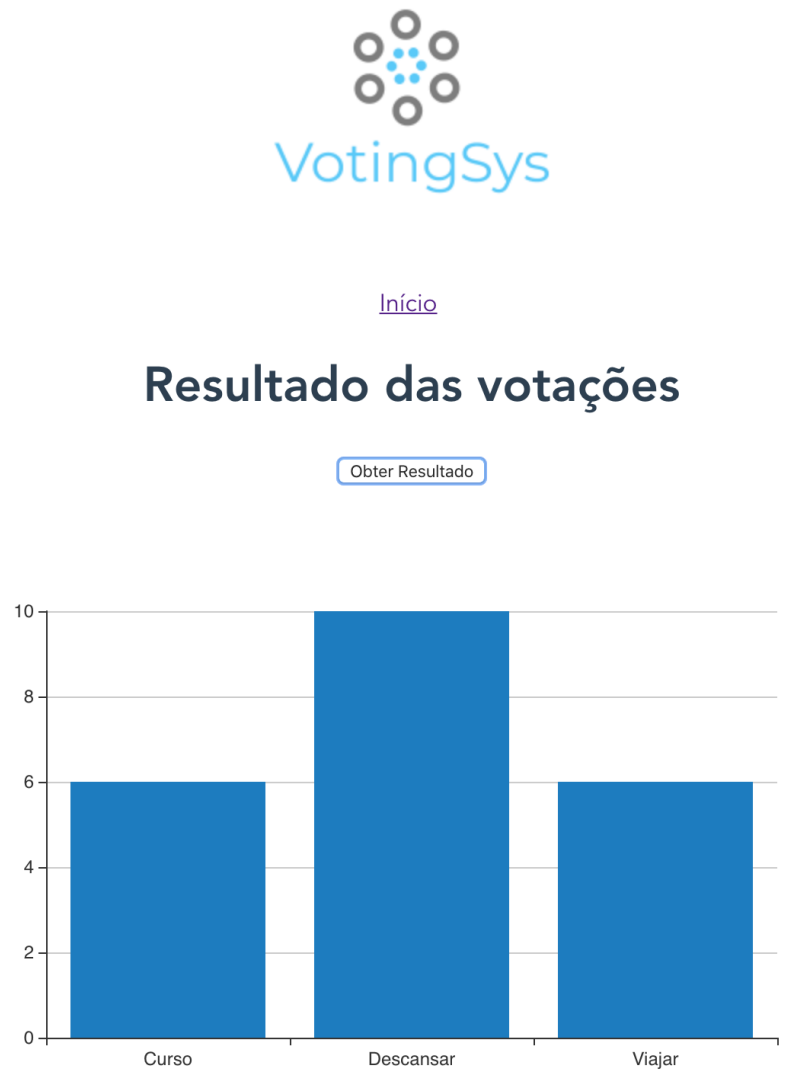

Figura 15 – Resultado da eleição. Fonte: autor

Um ponto a destacar foi a imediata ação do usuário em visualizar o resultado da eleição, após o cadastramento do seu respectivo voto. Com o intuito de não tendenciar a escolha do eleitor a votar nos candidatos que estão vencendo a eleição, esta funcionalidade foi modificada, onde a mesma só pode ser executada ao fim da eleição, ou para os usuários administradores do sistema.

4.9.1.3 Auditoria dos Dados

A auditoria dos dados atendeu as expectativas informando a participação de cada eleitor no processo, sendo estes dados auditados apresentados somente para o perfil de usuário administrador do sistema. No entanto, para o perfil de eleitor participante desse processo eleitoral não é possível a utilização das funcionalidades sobre a auditoria do sistema, apenas a visualização dos resultados da mesma. Essa funcionalidade protegeria a

privacidade de outro eleitor não poder apurar se aconteceu a participação de outro eleitor e qual teria sido o seu voto, sendo ela possível somente para o perfil de administrador do sistema.

A nível de curiosidade e confiança no sistema é interessante relatar que o professor da turma, em que foi realizado este experimento de eleição, informou a turma que a presença na aula do dia em que efetuado este teste beta teria como indicativo de presença a participação nesse processo eleitoral.

Assim, após concluído o processo eleitoral foi solicitado ao administrador do sistema o fornecimento dos dados de auditoria nessa eleição para apuração dos eleitores participantes que teriam a sua presença registrada na aula do referido dia.

5 Considerações Finais

5.0.1 Conclusão

Neste Trabalho de Conclusão de Curso buscou-se responder a pergunta se é possível desenvolver um sistema capaz de auxiliar uma população em processo de eleição segura envolvendo um universo determinado de opções de escolha que produziria um resultado confiável? A fim de responder esta questão foi apresentado o sistema VotingSys, um software capaz de realizar eleições nos mais diferentes contextos empregando uma tecnologia com recursos interessantes a segurança (Blockchain).

No contexto da Engenharia de Software, foram utilizados vários conhecimentos adquiridos durante o curso, tanto os conceitos de gestão quanto os conceitos de desenvolvimento, passando por vários dos conteúdos explorados por disciplinas como Requisitos, Métodos de Desenvolvimento, Desenho, Técnicas de Programação, Testes de Software e outras.

Alguns destaques relatados na documentação desse projeto poderiam citar no desenvolvimento dos requisitos algumas das adaptações das metodologias de desenvolvimento de software que foram propostas por este trabalho envolvendo métodos ágeis e características dos métodos tradicionais. No desenvolvimento do software houve a adaptação do Scrum para a realidade deste projeto, atendendo os aspectos de produção de um software e da elaboração de um trabalho com características científicas (TCC).

A implementação do VotingSys, utilizando a tecnologia Blockchain, foi uma escolha arriscada, visto que essa tecnologia é muito recente, não sendo tão simples encontrar materiais teóricos e técnicos confiáveis para realizar a implementação desse tipo de sistema. No entanto, a elaboração desse sistema colabora com a ampliação da comunidade do Blockchain, fornecendo um trabalho seguro e com resultados no âmbito teórico e prático com o desenvolvimento do VotingSys.

Dessa forma, foi possível elaborar um software que permitiria um processo eleitoral seguro empregando a tecnologia Blockchain com o *framework* Hyperledger Fabric, tornando assim possível o acompanhamento de uma eleição que permitiria aos seus eleitores escolherem entre as opções predefinidas para o seu voto, além destes visualizarem o resultado final da eleição, somente após o encerramento do período eleitoral, bem como forneceria ao administrador do software visualizar e analisar os dados de auditoria nesse processo eleitoral.

5.1 Trabalhos Futuros

Com o intuito de continuar a evolução desse projeto foram observadas algumas possibilidades interessantes a serem implementadas como trabalhos futuros, sendo estas:

- Métricas de Código-fonte: novas métricas para garantir a qualidade serão adicionadas. Identificando a complexidade ciclomática e duplicidades encontradas no código.
- Desenvolver interfaces de usuário para que o administrador seja capaz de executar funções relacionadas a ele.
- Analisar a possibilidade de utilizar o sistema proposto para a votação de plebiscitos pela população, para os mesmos serem capazes de exercer o seu direito de voto à respeito de um determinado tema em eleição

Referências Bibliográficas

ASTE P. TASCA, T. D. M. T. Blockchain technologies: The foreseeable impact on society and industry. 2017. Citado na página 16.

BORTOLINI, R. Como funciona o blockchain em 4 passos. https://blog.smlbrasil.com.br/2017/05/12/como-funciona-o-blockchain-em-quatro-passos/, 2016. Citado 2 vezes nas páginas 7 e 30.

BOURQUE, R. E. F. P. Software engineering body of knowledge – swebok. 2004. Citado na página 58.

BRITO, R. W. Bancos de dados nosql x sgbds relacionais:análise comparativa. In: . [S.l.: s.n.], 2010. Citado 2 vezes nas páginas 23 e 24.

BUTERIN, V. On public and private blockchains. https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/, 2015. Citado na página 31.

CASTRO, V. S. R. Segurança em cloud computing: Governança e gerenciamento de riscos de segurança. 2011. Citado na página 18.

COUTINHO, S. C. *Números inteiros e criptografia RSA*. [S.l.: s.n.], 2011. Citado na página 20.

CUOMO J. A. LATONE, C. C. G. Protocol specification. https://openblockchain.readthedocs.io/en/latest/protocol-spec/, 2016. Citado 3 vezes nas páginas 7, 35 e 52.

DATE, C. J. Introdução a sistemas de banco de dados. In: . [S.l.: s.n.], 2004. Citado na página 22.

ELMASRI, S. N. R. *Sistemas de banco de dados*. [S.l.: s.n.], 2005. Citado 2 vezes nas páginas 22 e 23.

FERRIS, C. Hyperledger fabric introduction. https://hyperledger-fabric.readthedocs.io/en/latest/whatis.html, 2019. Citado na página 33.

FOWLER, P. J. S. M. *NoSQL distilled: a brief guide to the emerging world of polygot persistence*. [S.l.: s.n.], 2012. Citado na página 27.

FUCOLO, I. M. S.o.l.i.d no android. https://medium.com/android-dev-br/s-o-l-i-d-no-android-d55e23f1c72d, 2017. Citado na página 52.

GARG P. SARASWAT, S. B. S. A. S. K. K. S. G. K. A comparitive analysis on e-voting system using blockchain. 2019. Citado na página 15.

HAN E. HAIHONG, G. L. J. D. J. Survey on nosql database. in pervasive computing and applications (icpca). In: . [S.l.: s.n.], 2011. Citado na página 24.

HECHT, S. J. R. Nosql evaluation: A use case oriented survey. in cloud and service computing (csc). 2011. Citado 2 vezes nas páginas 25 e 26.

HERTIG, A. How do ethereum smart contracts work? 2018. Citado na página 32.

HEUSER, C. A. *Projeto de banco de dados*. [S.l.: s.n.], 1998. Citado 2 vezes nas páginas 21 e 22.

HJALMARSSON G. K. HREIðARSSON, M. G. H. F. P. Blockchain-based e-voting system. 2018. Citado na página 16.

KELLY, D. M. A. Making sense of nosql : A guide for managers and the rest of us by ann kelly and dan mccreary. In: . [S.l.: s.n.], 2014. Citado na página 24.

KOSBA A. MILLER, E. S. Z. W. C. P. A. The blockchain model of cryptography and privacy-preserving smart contracts. 2016. Citado 2 vezes nas páginas 27 e 30.

KUHRT, T. Welcome to hyperledger fabric. https://wiki.hyperledger.org/display/fabric/Welcome+to+Hyperledger+Fabric, 2018. Citado na página 32.

LEWIS, A. A gentle introduction to blockchain technology. https://bitsonblocks.net/2015/09/09/a-gentle-introduction-to-blockchain-technology/, 2015. Citado na página 27.

LI Q. LU, S. C. Y. L. X. X. Z. A landscape of cryptocurrencies. 2019. Citado na página 29.

LOSCIO H. R. DE OLIVEIRA, H. R. d. P. B. F. Nosql no desenvolvimento de aplicações web colaborativas. In: . [S.l.: s.n.], 2011. Citado na página 23.

MERKLE, R. C. *A Digital Signature Based on a Conventional Encryption Function*. [S.l.: s.n.], 2000. Citado 2 vezes nas páginas 28 e 30.

NAKAMOTO, S. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, 2008. Citado 4 vezes nas páginas 7, 27, 29 e 31.

NAVARRO, L. A. Z. A inovação democrática no brasil. In: . [S.l.: s.n.], 2003. Citado na página 14.

NILSON, S. *O SISTEMA POLÍTICO BRASILEIRO: Continuidade ou Reforma?* [S.l.: s.n.], 2008. Citado na página 14.

NIR, V. J. K. Blockchain-enabled e-voting. 2018. Citado na página 15.

OLLEROS, M. Z. F. X. *Research Handbook on Digital Transformations*. [S.l.: s.n.], 2016. Citado 4 vezes nas páginas 7, 28, 30 e 31.

PINHEIRO, J. M. S. *Da Iniciacao Cientifica ao TCC: Uma abordagem Para os Cursos de Tecnologia*. [S.l.: s.n.], 2010. Citado na página 36.

POPPENDIECK, M. A. C. M. Lean software development: A tutorial. 2012. Citado na página 47.

QUEIROZ, I. O que devo saber sobre nosql? http://blog.ivanqueiroz.com/2017/01/o-que-devo-saber-sobre-nosql.html, 2017. Citado 3 vezes nas páginas 7, 25 e 26.

RAMAKRISHNAN, J. G. R. Sistemas de gerenciamento de banco de dados. In: . [S.l.: s.n.], 2008. Citado 5 vezes nas páginas 18, 19, 20, 21 e 22.

RANDE, T. C. Smart contracts: o que são e como funcionam? 2018. Citado na página 32.

REDMOND J. WILSON, J. C. E. *Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement*. [S.l.: s.n.], 2012. Citado 2 vezes nas páginas 25 e 26.

SABBAGH, R. *Scrum Gestao Agil para Projetos de Sucesso*. [S.l.: s.n.], 2013. Citado na página 40.

SANTISO, C. Will blockchain disrupt government corruption? https://ssir.org/articles/entry/will$_b$*lockchain*$_d$*isrupt*$_g$*overnment*$_c$*orruption*, 2018.*Citadonapágina*14

SCHNEIDER, F. B. *Implementing Fault-tolerant Services Using the State Machine Approach*. [S.l.: s.n.], 1990. Citado na página 57.

SCIENCES, N. A. of. *Securing the Vote: Protecting American Democracy*. [S.l.: s.n.], 2018. Citado 2 vezes nas páginas 15 e 16.

SHERMIN, V. Disrupting governance with blockchains and smart contracts*. 2017. Citado na página 27.

SILVA L. S. SOARES, J. L. B. R. A. C. Workflow aplicado a engenharia de software baseada em processos: Uma visão geral. 2006. Citado na página 51.

SOFTEX. Mps.br - melhoria de processo do software brasileiro - parte 9. 2016. Citado na página 57.

SOMMERVILLE, I. *Engenharia de Software*. [S.l.: s.n.], 2011. Citado 2 vezes nas páginas 40 e 57.

STALLINGS, W. *CRYPTOGRAPHY AND NETWORK SECURITY PRINCIPLES AND PRACTICE*. [S.l.: s.n.], 2011. Citado 4 vezes nas páginas 18, 19, 20 e 21.

SYKES, C. F. M. Hyperledger fabric model. https://hyperledger-fabric.readthedocs.io/en/latest/fabric$_m$*odel.html*, 2019.*Citadonapágina*34.

TAPSCOTT, A. T. D. *Blockchain Revolution: How the tecnology behind bitcoin is changing money, business, and the world*. [S.l.: s.n.], 2016. Citado 2 vezes nas páginas 29 e 30.

THIOLLENT, M. *Metodologia da Pesquisa-acao*. [S.l.: s.n.], 2009. Citado na página 36.

TIWARI, S. Professional nosql. programmmer to programmer. In: . [S.l.: s.n.], 2011. Citado 3 vezes nas páginas 23, 26 e 27.

XAVIER, L. E. C. Tecnologias da informação e comunicação (tic) no sector público da raem. In: . [S.l.: s.n.], 2015. v. 28, p. 777–779. Citado na página 16.

Apêndices

APÊNDICE A – Imagens da Aplicação

A Figura 16 apresenta um formulário, no qual o usuário eleitor pode efetuar o acesso as votação e o usuário administrador pode realizar o acesso à sua página de administração do sistema. Além de apresentar o link para o resultado das eleições e um botão para o registro de um novo usuário eleitor.

Figura 16 – Tela Inicial. Fonte: autor

O registro de usuários eleitores é realizado através de um formulário, onde é necessário informar um CPF válido, o RG, nome, sobrenome e senha, como pode-se observar na Figura 17.

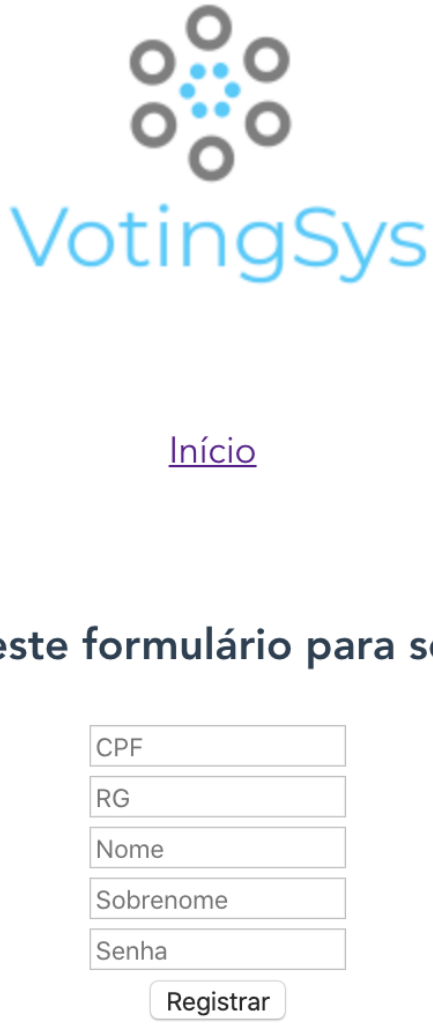

Figura 17 – Tela de Registro. Fonte: autor

É possível identificar os candidatos cadastrados no sistema e selecionar um dos candidatos para indicar o seu voto, o confirmando através do seu CPF, como ilustra a Figura 18.

Figura 18 – Tela de Votação. Fonte: autor

Os resultados da eleição podem ser visualizados, após a data final da eleição, através de um gráfico renderizando a contagem de votos por candidato, como se pode observar na Figura 19.

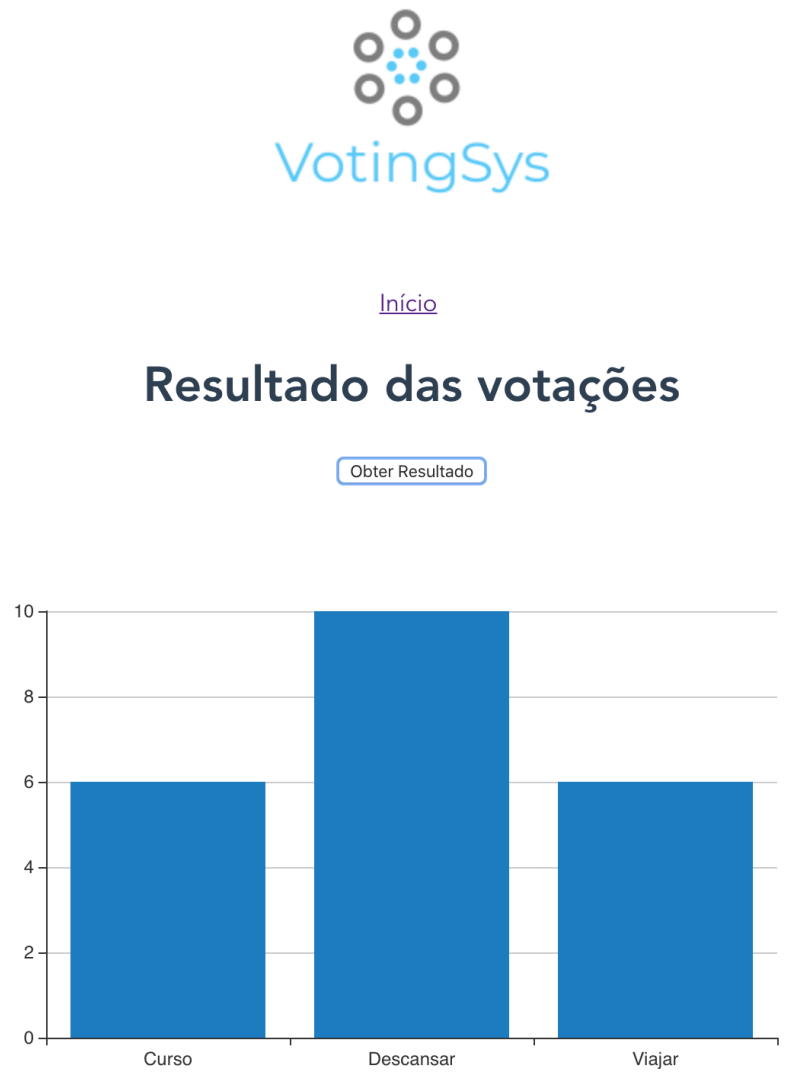

Figura 19 – Resultados. Fonte: autor

As funções de auditoria e possibilidade de acompanhar os resultados da eleição em tempo real, só podem ser efetuadas pelo administrador do sistema, o qual é criado no momento de instanciação do sistema, onde é informado o e-mail do administrado e suas credenciais lhe são enviadas. Após, o acesso do administrador no sistema a sua interface é carregada como ilustra a Figura 20.

Figura 20 – Resultados. Fonte: autor

A auditoria dos dados pode ser realizada através da função de Busca, em que todas as transações do Blockchain são disponibilizadas, ou então através da função Buscas pela Chave, onde é possível buscar uma transações através de uma palavra chave, como ilustram as Figuras 21 e 22.

Na Figura 21 é possível visualizar todas as transações inseridas na rede Blockchain do sistema, como por exemplo a eleição criada. Na Figura 22 pode-se observar a pesquisa das transações inseridas no Blockchain, através da busca através de uma *string*, no exemplo foi utilizada a *string* "Viajar"e o resulltado foi demonstrado na Figura em questão.

Início

Todos os dados do Blockchain

Buscar

02954492171 | { "ballot": "9r4466fs4o5cp9ythxiqdj", "ballotCreated": true, "firstName": "Gustavo", "lastName": "Moreira Nascimento Araujo", "password": "gama", "registrarId": "2968511", "type": "voter", "voterId": "02954492171" }

2o95aukueehj67lohypmwo | { "country": "Brasil", "electionId": "2o95aukueehj67lohypmwo", "endDate": "2019-11-30T00:00:00.000Z", "name": "Eleições FGA", "startDate": "2019-11-29T00:00:00.000Z", "type": "election", "year": 2019 }

79ol5t45qgstdlfp011a2 | { "ballotCast": false, "ballotId": "79ol5t45qgstdlfp011a2", "election": { "country": "Brasil", "electionId": "2o95aukueehj67lohypmwo", "endDate": "2019-11-30T00:00:00.000Z", "name": "Eleições FGA", "startDate": "2019-11-29T00:00:00.000Z", "type": "election", "year": 2019 }, "type": "ballot", "votableItems": [{ "count": 0, "description": "Viajar", "type": "votableItem", "votableId": "Viajar" }, { "count": 0, "description": "Descansar", "type": "votableItem", "votableId": "Descansar" }, { "count": 0, "description": "Curso de aperfeiçoamento em banco de dados", "type": "votableItem", "votableId": "Curso" }], "voterId": "V1" }

7a9graptkqww5ahyqce43b | { "ballotCast": false, "ballotId": "7a9graptkqww5ahyqce43b", "election": { "country": "Brasil", "electionId": "2o95aukueehj67lohypmwo", "endDate": "2019-11-30T00:00:00.000Z", "name": "Eleições FGA", "startDate": "2019-11-29T00:00:00.000Z", "type": "election", "year": 2019 }, "type": "ballot", "votableItems": [{ "count": 0, "description": "Viajar", "type": "votableItem", "votableId": "Viajar" }, { "count": 0, "description": "Descansar", "type": "votableItem", "votableId": "Descansar" }, { "count": 0, "description": "Curso de aperfeiçoamento em banco de dados", "type": "votableItem", "votableId": "Curso" }], "voterId": "administrador" }

9r4466fs4o5cp9ythxiqdj | { "ballotCast": false, "ballotId": "9r4466fs4o5cp9ythxiqdj", "election": { "Key": "2o95aukueehj67lohypmwo", "Record": { "country": "Brasil", "electionId": "2o95aukueehj67lohypmwo", "endDate": "2019-11-30T00:00:00.000Z", "name": "Eleições FGA", "startDate": "2019-11-29T00:00:00.000Z", "type": "election", "year": 2019 } }, "type": "ballot", "votableItems": [{ "Key": "Curso", "Record": { "count": 0, "description": "Curso de aperfeiçoamento em banco de dados", "type": "votableItem", "votableId": "Curso" } }, { "Key": "Descansar", "Record": { "count": 0, "description": "Descansar", "type": "votableItem", "votableId": "Descansar" } }, { "Key": "Viajar", "Record": { "count": 0, "description": "Viajar", "type": "votableItem", "votableId": "Viajar" } }], "voterId": "02954492171" }

Curso | { "count": 0, "description": "Curso de aperfeiçoamento em banco de dados", "type": "votableItem", "votableId": "Curso" }

Descansar | { "count": 0, "description": "Descansar", "type": "votableItem", "votableId": "Descansar" }

Figura 21 – Buscar. Fonte: autor

Figura 22 – Buscar por Palavra. Fonte: autor