

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

**LUIZ HENRIQUE DOS SANTOS VIEIRA
NATHAN SCHNEIDER GAVENSKI**

**BLOCKCHAIN GOVERNAMENTAL
Uma solução para gestão de documentos**

Porto Alegre

2018

LUIZ HENRIQUE DOS SANTOS VIEIRA

NATHAN SCHNEIDER GAVENSKI

**BLOCKCHAIN GOVERNAMENTAL: Uma solução para gestão de
documentos**

Monografia de trabalho de conclusão de curso de graduação apresentado à Escola Politécnica da Pontifícia Universidade Católica do Rio Grande do Sul, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Professor Orientador: Prof. Me. Dilnei Venturini

Porto Alegre

2018

DEDICATÓRIA

Dedicamos este trabalho as nossas avós, que sempre estiveram do nosso lado e que, com muito carinho e apoio, não mediram esforços para que nossos sonhos se tornassem realidade. Elas sempre estarão conosco e por isso, o nosso muito obrigado.

AGRADECIMENTOS

Agradecemos ao nosso Prof. Dilnei Venturini pelos textos mais variados sobre *blockchain* e a orientação, que sem ela este trabalho não teria se tornado realidade.

Agradecemos ao nosso Prof. Avelino Zorzo por nos ajudar pela avaliação e pelas referências compartilhadas.

Agradecemos aos nossos pais e família por nos acompanharem durante toda esta trajetória e nos apoiarem em todos os momentos.

RESUMO

A criação e gerenciamento de documentos, assim como a comunicação entre diferentes sistemas do estado, apresentam diversos problemas que emendas constitucionais e projetos de lei tentam resolver. Com um movimento para um novo estilo de documentação, mais unificada e de difícil falsificação, este trabalho propõe uma abordagem mais tecnológica para esta situação.

O presente projeto sugere uma solução via a tecnologia de *blockchain*, que tem como objetivo reduzir o custo de manutenção e criação do vasto número de documentações hoje existentes e ir diretamente contra a falsificação de novos documentos. A proposta tem também como objetivo propor uma forma que apresenta maior durabilidade e maior transparência para o governo.

A plataforma foi desenvolvida via Ethereum e sua implementação de Smart Contract, Solidity. Também foi usada alguma linguagem de *front-end* para apresentação da solução. Esta solução visa melhor atender as necessidades da população brasileira.

Palavras-chave: Blockchain. Smart Contract. Ethereum. Documentação.

ABSTRACT

The creation and management of documents as well as communication between different systems of our state present several problems that our constitution and amendments try to solve. With a movement in favor of a new style of documentation, more unified and difficult to falsify, this paper proposes a more technological approach for such situation.

This project suggests a solution via blockchain technology, aiming to reduce the cost of maintenance and creation of the vast number of documents available today and fight directly against forgery of new ones. The proposal also aims to suggest a form that presents greater durability and transparency for our government.

The platform was developed on Ethereum and its implementation of Smart Contract, Solidity. Some front-end language was used to present the solution. This paper aims to better meet today's needs of the Brazilian population.

Keywords: Blockchain. Smart Contracts. Ethereum. Documentation.

LISTA DE ILUSTRAÇÕES

Imagen 1: Fluxograma de Emissão da Certidão de Nascimento	20
Imagen 2: Fluxograma de Emissão do Registro Geral	22
Imagen 3: Ilustração de blocos de uma blockchain	23
Imagen 4: Casos de uso para a blockchain	32
Imagen 5: Fluxograma do front-end.....	34
Imagen 6: Ciclo de vida de uma promessa.....	35
Imagen 7: Modelo serviço para comunicação com blockchain	36
Imagen 8: Diagrama de classe para serviço de serialização do usuário	37
Imagen 9: Serviço de orquestração de chamadas Firebase.....	39
Imagen 10: Diagrama do Registro Padrão.....	42
Imagen 11: Diagrama de classe para o contrato Registro Padrão	42
Imagen 12: Diagrama de classe para certidão de nascimento.	44
Imagen 13: Diagrama da Certidão de Nascimento	46
Imagen 14: Diagrama de atividade da Certidão de Nascimento.....	46
Imagen 15: Diagrama de classe Registro Geral	48
Imagen 16: Diagrama do Registro Geral	49
Imagen 17: Diagrama de atividade do Registro Geral	49
Imagen 18: Diagrama de classe do contrato de Notificação	50
Imagen 19: Diagrama de classe do contrato Parse	52
Imagen 20: Diagrama de classe de todos os contratos presentes	53
Imagen 21: Resultado dos testes unitários	54
Imagen 22: Diagrama de testes unitários para aplicações Angular	55
Imagen 23: Resultados lint do projeto.....	56
Imagen 24: Tela da ferramenta Remix.....	57
Imagen 25: Tela defeitos abertos no repositório de Smart Contracts	58
Imagen 26: Tela defeitos e tarefas do repositório front-end.....	58
Imagen 27: Tela de login	59
Imagen 28: Tela do formulário	60
Imagen 29: Fluxo de criação de registro padrão.....	60
Imagen 30: Tela da Home Page	61
Imagen 31: Fluxo telas RG	62
Imagen 32: Tela de Adicionar RG.....	63

Imagen 33: Tela de Visualizar	64
Imagen 34: Tela de Renovar	65
Imagen 35: Fluxo telas Certidão de Nascimento	65
Imagen 36: Tela adicionar Certidão de Nascimento	66
Imagen 37: Fluxo de criação de Certidão de Nascimento	67
Imagen 38: Tela de visualização de Certidão de Nascimento	67
Imagen 39: Tela de renovação de Certidão de Nascimento	68
Imagen 40: Tela de informações de perfil.....	69
Imagen 41: Tela de Notificações.....	69
Imagen 42: Soluções criadas pela E-stonia.....	70
Imagen 43: Documento oficial sobre blockchain do Reino Unido	71

LISTA DE TABELAS

Tabela 1: Cronograma de Atividades TC I.....	98
Tabela 2: Cronograma de atividades TC II	99

LISTA DE SIGLAS

ACID - Atomicidade, Consistência, Isolamento e Durabilidade
API - *Applications Programming Interface*
BaaS - *Backend as a Service*
BaaS - *Blockchain-as-a-Service*
BPMN - *Business Process Model and Notation*
CNH - Carteira Nacional de Habilitação
CPF - Cadastro de Pessoa Física
CTPS - Carteira de Trabalho e Previdência Social
EGD - Estratégia de Governança Digital
EVM - *Ethereum Virtual Machine*
Gb - *Gigabytes*
GDPR - *General Data Protection Regulation*
Ghz - *Gigahertz*
ID - Identificador
IDE - *Integrated Development Environment*
IoT - *Internet of Things*
KSI - *Keyless Signature Infrastructure*
LGPD - Lei Geral de Proteção de Dados
MERCOSUL - Mercado Comum do Sul
P2P - *Peer-to-Peer*
POW - *Proof of Work*
RG - Registro Geral
RIC - Registro de Identificação Civil
RS - Rio Grande do Sul
SALT - *Sequential, Agreed, Ledgered, Temper-resistant*
SI - Sistema de Informação
TC 1 - Trabalho de Conclusão 1
TC 2 - Trabalho de Conclusão 2
TI - Tecnologia da Informação

SUMÁRIO

1. INTRODUÇÃO	15
2. FUNDAMENTAÇÃO TEÓRICA	18
2.1. Histórico da Documentação.....	18
2.2. Certidão de Nascimento	19
2.3. Registro Geral.....	20
2.4. A Blockchain.....	22
2.5. Smart Contracts.....	25
2.6. Ethereum	26
2.7. Fundamentação do Problema: Veracidade dos Documentos.....	27
3. OBJETIVOS	29
3.1. Objetivo Geral.....	29
3.2. Objetivo Específico	29
4. PROPOSTA DE SOLUÇÃO.....	30
5. ARQUITETURA DA SOLUÇÃO.....	34
5.1. Front-End.....	34
5.1.1. Web3:.....	35
5.1.2. Firebase / Angularfire2:.....	37
5.2. Back-end.....	38
5.2.1. Firebase:	38
5.2.2. Ethereum.....	39
5.3. Smart Contratcs.....	40
5.3.1. Truffle / Ganache:	40
5.3.2. Contrato de Registro Padrão:.....	41
5.3.3. Contrato de Certidão de Nascimento:	43
5.3.4. Contrato de Registro Geral:	46
5.3.5. Notification	49

5.3.6. Parse.....	51
5.3.7. Estrutura final.....	52
5.4. Testes.....	53
5.4.1. Camada front-end	53
5.4.2. Camada <i>back-end</i>	56
5.4.3. Gerenciador de defeitos	57
6. DESCRIÇÃO TELAS DA SOLUÇÃO	59
6.1. Tela Inicial.....	59
6.2. Registro	59
6.3. Home	61
6.4. Registro Geral.....	61
6.4.1. Adicionar	62
6.4.2. Visualizar.....	63
6.4.3. Renovar.....	64
6.5. Certidão de nascimento.....	65
6.5.1. Adicionar	66
6.5.2. Visualizar.....	67
6.5.3. Renovar.....	68
6.6. Perfil	68
6.7. Notificações	69
7. TRABALHOS SEMELHANTES	70
7.1. E-stonia - ESTÓNIA.....	70
7.2. BaaS - REINO UNIDO	71
7.3. Distributed Access Control Evaluation on IoT Ledger-based Architecture	72
8. PROBLEMAS ENCONTRADOS.....	73
8.1. Imutabilidade	73

8.1.1.	Mutabilidade de parte dos dados de um bloco	73
8.1.2.	Capacidade de alterar os dados e recriar o hash.....	74
8.1.3.	Dados mutáveis serem alocados dentro de uma estrutura dinâmica	74
8.2.	Permissão de acesso aos dados	75
8.3.	Versionamento.....	75
8.4.	Emissão de Certidão de Nascimento e Registro Geral.....	76
9.	Benefícios e malefícios encontrados durante o desenvolvimento	77
9.1.	Benefícios	77
9.1.1.	Modelo SALT:	77
9.1.2.	Log transacional:.....	78
9.1.3.	Capacidade de ter a lógica no contrato:.....	78
9.1.4.	Simplicidade na comunicação:.....	79
9.1.5.	Praticidade na criação dos contratos:	79
9.1.6.	Visibilidade dos dados:.....	80
9.2.	Malefícios.....	80
9.2.1.	Número máximo de parâmetros:.....	80
9.2.2.	Controle transacional:	81
9.2.3.	Linguagem limitada:	81
9.2.4.	Determinístico versus não determinismo:	82
9.2.5.	Serialização e desserialização:	83
9.2.6.	Verbosidade:	83
9.2.7.	Versionamento:	83
10.	CONCLUSÃO	85
	REFERÊNCIAS	88
	GLOSSÁRIO	94
	APÊNDICE A - Definição do Cronograma TC I e TC II	97

APÊNDICE B - Recursos Necessários	100
APÊNDICE C - Testes Unitários.....	102
APÊNDICE D - Avaliação de Código	107
APÊNDICE E - Testes Funcionais.....	108

1. INTRODUÇÃO

Ao longo dos anos, a população vem crescendo em níveis cada vez maiores. Nas últimas quatro décadas e meia o número de habitantes dobrou na face da terra, passando de quatro bilhões para oito bilhões. Quando analisa-se que levou 1802 anos para chegar ao primeiro bilhão e apenas duzentos e dezesseis para multiplicar a população por oito, é fácil lembrar alguns problemas sociais e governamentais criados por este crescimento exacerbado [30] [54]. A sociedade cresceu e expandiu-se, novos continentes foram descobertos, novos países foram criados, instituições governamentais foram construídas e dizimadas, a noção de indivíduo mudou, liberdades e direitos foram concedidos e as barreiras de gênero e raça foram diminuídas. Muitas coisas evoluíram e com todas essas mudanças a tecnologia da informação foi criada. Com ela foi possível realizar atividades antes impossíveis. Foram criados computadores cada vez menores, até eles caberem nas carteiras e bolsos. Computadores capazes de equacionar o que as pessoas pensam, capazes de dizer a previsão do tempo com maior precisão, de entender o contexto de diferentes pacientes e ajudar médicos a sugerir possíveis tratamentos. E assim nossa entidade evoluiu. A população deixou de ser *homo economicus* e tornou-se um acumulado de informações diferentes.

Atualmente, como indivíduos e cidadãos somos representados de diferentes formas [10] [11]. Têm documentos legais, como o Registro Geral (RG), a Carteira Nacional de Habilitação (CNH), passaporte, título de eleitor, a Carteira de Trabalho e Previdência Social (CTPS), o Cadastro de Pessoa Física (CPF) a certidão de nascimentos, entre outros. Possuem também os documentos pessoais, tais como cartões de crédito, seguros de saúde, cartões de fidelidade, carteiras de vacinação, carteira da faculdade, biblioteca e muitos outros. E para terminar há as contas sociais, que não tão importantes para o estado, mas os definem mais do que qualquer outro registro, como por exemplo o Facebook, Instagram, Reddit, Snapchat, Tinder ou qualquer outra mídia ou aplicativo. Enumerando desta forma fica fácil de identificar alguns problemas que existem com a quantidade excessiva de documentos e informações pessoais que são carregados ou guardados. De clonagens de

cartões a perfis falsos de mídias sociais e documentos falsos, vê-se hoje em dia a dificuldade de se manter seguro e com toda essa lista organizada e em ordem. Além dos problemas pessoais claros, também existem os problemas administrativos de possuir diversos documentos com diversos propósitos diferentes. Atualmente, para ser considerado cidadão brasileiro, são necessários diferentes documentos, que além de serem incumbidos ao porte pessoal, também possuem seus equivalentes dentro das diferentes secretarias e portarias. Tais documentos, seus processos e protocolos resultam em uma necessidade de armazenamento enorme. Estes arquivos também estão expostos a qualquer problema ambiental possível, podendo ser danificados pelas condições climáticas ou desastres naturais, como enchentes e até mesmo temporais fortes.

Com este problema em mente, este trabalho de conclusão tem como objetivo apresentar uma proposta de solução a partir do uso da tecnologia do blockchain criada pelo pseudônimo Satoshi Nakamoto [36]. De maneira mais específica, a distribuição *open source* Ethereum, como plataforma para criar, gerir e compartilhar as documentações do que chamaremos de Blockchain Governamental. Devido ao tempo limitado para o desenvolvimento de uma proposta de solução, foi realizada uma implementação inicialmente da Certidão de Nascimento e do Registro Geral. De forma resumida, a implementação usa das transações normais da *blockchain*, usando em modo privado, para garantir sua integridade e reduzir o custo das transações, mais conhecido como gás, à zero. Além disso, foram utilizadas de maneira extensiva às contas de *smart contracts* para serem a representação dentro da tecnologia dos documentos que foram implementados

Desta forma, pretende-se resolver o problema da veracidade dos documentos apresentados pelo portador, tendo em vista que ele terá apenas um número de sua conta na *blockchain* e seus documentos estarão dentro da tecnologia, sendo validados por ela. O problema do armazenamento foi resolvido já que todos registros estavam na nuvem em uma estrutura escalável. Do problema da necessidade de diversas portarias e secretarias foi solucionado uma vez que todos os processos poderão ser feitos de maneira online e diretos na *blockchain*. E por final do problema de lentidão dos

processos atuais deve-se solucionar uma vez que parte do processo pode ser automatizado dentro da tecnologia.

Com esta solução, espera-se poder mostrar que existe espaço para aprimoramento dentro das entidades governamentais, especialmente dentro da parte tecnológica destes setores e unificar os processos atuais hoje no Brasil. Desta forma, também pretende-se manter os pilares que motivaram a criação da *blockchain* intactos, mantendo-o imutável (na ordem das contas dos cidadãos), seguro e distribuído. Entregando, assim, uma solução capaz de ser implementada em estruturas diferentes a qual foi aplicada neste trabalho.

Este documento está organizado da seguinte maneira: o Capítulo 2 apresenta a fundamentação teórica do problema proposto; o Capítulo 3 apresenta os objetivos a serem obtidos com a realização deste trabalho; no Capítulo 4 é descrita a proposta de solução do problema; o Capítulo 5 apresenta a arquitetura da aplicação desenvolvida e no Capítulo 6 é descrito as telas do *front-end*; no Capítulo 7 apresenta os trabalhos acadêmicos e de mercado que são correlatos a este, no uso da tecnologia da *blockchain*; o Capítulo 8 são descritos os problemas encontrados durante a realização do trabalho e os meios utilizados para solucioná-los; no Capítulo 9 são descritos os benefícios e malefícios encontrados no desenvolvimento do trabalho; o Capítulo 10 aponta o planejamento realizado para conclusão das atividades propostas; no Capítulo 11 determina os recursos tecnológicos utilizados para conclusão do trabalho e o Capítulo 12 possui a conclusão obtida após o trabalho. Por fim, há a apresentação das referências bibliográficas e um glossário das palavras utilizadas neste trabalho.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será relatado o conhecimento histórico sobre gerenciamento de documentos, o processo atual para emissão dos documentos escopo do trabalho; da tecnologia e funcionalidades da *blockchain*; a apresentação da versão da *blockchain* utilizada pelo Ethereum e os problemas que são escopo da solução a ser desenvolvida.

2.1. Histórico da Documentação

Durante os cento e onze anos da invenção do RG até hoje, é possível observar uma pequena alteração nos padrões e tecnologias usados para criar estas documentações pessoais. Em 1907 foi criado o primeiro documento de registro para o povo brasileiro, dentro de um órgão governamental, chamado de Registro Civil [18]. Embora os cartórios tivessem sido criados no ano de 1887, grande parte dos registros, como certidões de nascimento, casamento e óbito eram emitidas pelas igrejas, não tendo, assim, um controle de todos os registros emitidos. Avançando para 2018, analisa-se que a tecnologia destes documentos começa a alcançar a tecnologia de mercado, com exemplos como o CNH digital [20]. Um grande passo para competir com as ondas de falsificação e fraudes de identidade que ocorrem no Brasil [25].

É necessário ter em mente que estes documentos mudam muito com o tempo, como por exemplo o primeiro RG criado o indivíduo era obrigado a registrar todas as suas digitais no documento, sua profissão, endereço e detalhes físicos que pudesse ajudar a identificar o portador do registro. Este registro não possuía validade, mas apresentava o problema da constante alteração devido as informações serem constantemente alteradas ou, até mesmo, pelo fato de ser uma regalia e não comum para todos os brasileiros. Observa-se hoje que a constante alteração de informações pode ser um problema, devido ao fato de, ultimamente, já ser reconhecida a possibilidade de troca de sexo e, até mesmo, a diferença de gênero e sexo, assim como o próprio nome do indivíduo. Devido a estas constantes alterações, algumas informações foram passadas para outros novos documentos, como a profissão atual e a moradia, enquanto outras foram adicionadas para uma maior ligação

entre todos estes registros, como o número do CPF. Um prazo de validade também foi adicionado para cada RG, assim como uma tentativa de um certificado digital, porém o custo era muito alto para que a medida fosse amplamente implementada [37].

Com todos estes desafios em mente foi possível imaginar o que poderia ser implementado em termos tecnológicos para realizar um controle centralizado, combater fraudes e ajudar na busca da validação dos documentos, que também tivesse como alterar suas informações e aprimorar suas validações. Com isso em mente, a tecnologia da *blockchain* foi escolhida.

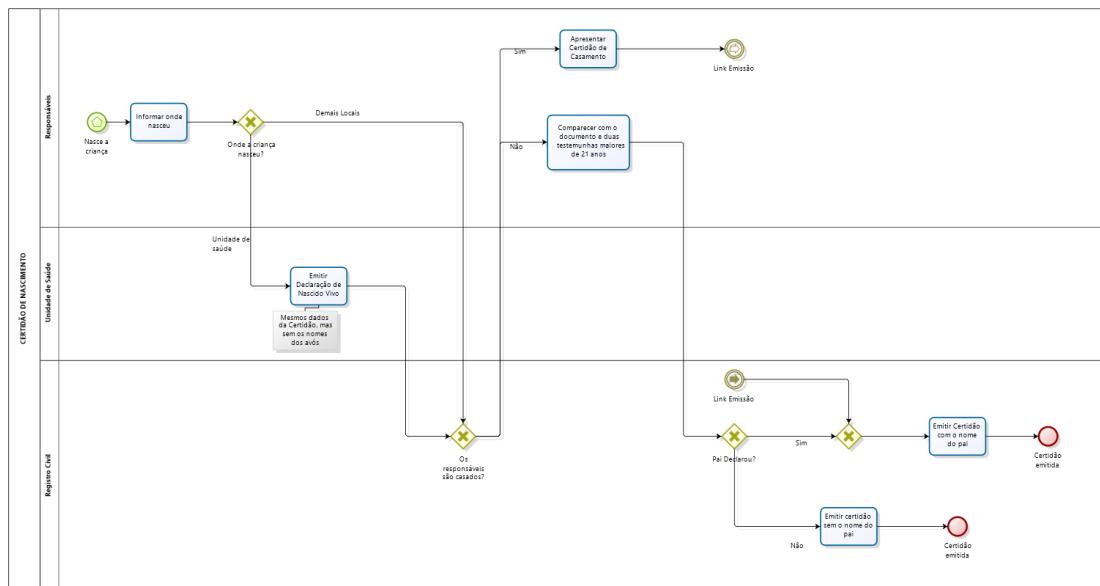
2.2. Certidão de Nascimento

A emissão de uma certidão de nascimento para uma criança é o ato de reconhecer sua existência e torná-la um cidadão brasileiro. Sendo um direito que todos os cidadãos brasileiros possuem, a emissão de sua primeira via é gratuita e geralmente é o primeiro documento que alguém possui [43]. A falta deste documento, implica no impedimento de acesso a alguns serviços sociais como a emissão de outros documentos, entre eles o CPF. Por este motivo, é um dos documentos mais importantes dos cidadãos brasileiros. O seu processo de emissão possui o envolvimento das seguintes partes: os responsáveis pela criança, a Unidade de Saúde e o Governo, representado através do Registro Civil.

O processo inicia-se com os responsáveis dirigindo-se ao Registro Civil e informando onde a criança nasceu, pois se foi em uma Unidade de Saúde, deve ser levado o documento chamado “Declaração de Nascido Vivo”, em que o nome dos pais e da criança já é informado. Este documento é emitido pela própria Unidade de Saúde, contudo o mesmo não garante o reconhecimento de paternidade da criança, se a mãe for iniciar esse processo. Como esse documento não é obrigatório para que o processo ocorra, ele não foi escopo deste trabalho e foi abstraído na solução desenvolvida. Depois que os pais informaram onde a criança nasceu, verifica-se se os pais da criança são casados e, caso sejam, é solicitada a Certidão de Casamento. Caso não forem casados, é necessária a presença de duas testemunhas maiores de 21 anos de idade. Sendo os pais casados o reconhecimento de paternidade ocorre

automaticamente e, diante disso, a Certidão de Nascimento já é emitida contendo o nome do pai da criança. Para os pais que não são casados e estejam acompanhados das testemunhas, é preciso que o pai reconheça a paternidade - pessoalmente ou por escrito em um documento oficial - e assim a certidão conterá o nome do pai. Caso contrário, será emitida somente com o nome da mãe. A seguir, o fluxograma realizado na notação BPMN - *Business Process Model and Notation*, do processo de Emissão da Certidão de Nascimento.

Imagen 1: Fluxograma de Emissão da Certidão de Nascimento



Presented by
bizagi
Illustrated

FONTE: Autoral

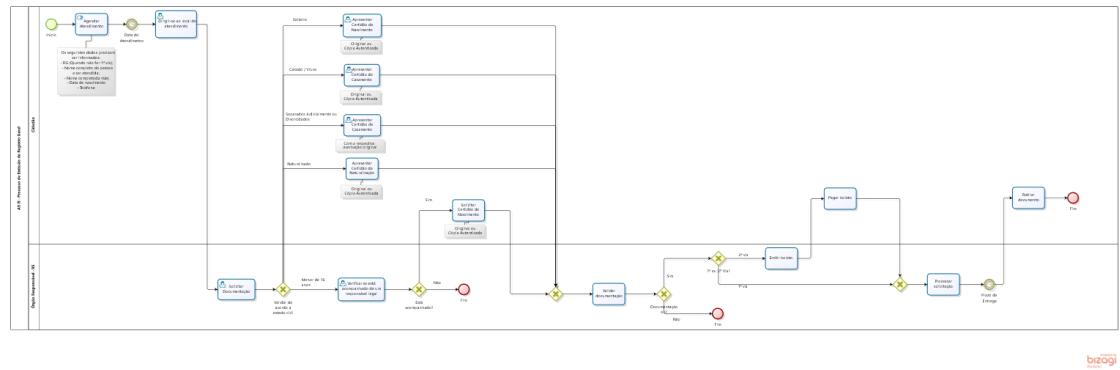
2.3. Registro Geral

Atualmente, no Brasil, o RG pode ser emitido por diferentes Órgãos Governamentais de acordo com o estado em que seja solicitado. No estado do Rio Grande do Sul (RS), por exemplo, esse documento pode ser solicitado através do Departamento de Identificação no Instituto Geral de Perícias do estado ou pelas unidades do Tudo Fácil, sendo este último uma central de serviços do estado criada em 1998, com a finalidade de centralizar os serviços públicos mais demandados pela população gaúcha [46].

O direito ao RG está previsto na Lei Nº 7116 do ano de 1983, em que regulamenta a aceitação do documento em todo o território brasileiro, além de informar que, para a emissão desse documento, é necessária a identificação das impressões digitais do cidadão, por isso sua importância para o Governo [13]. Um direito concedido por esse documento é a possibilidade de viajar para países pertencentes ao Mercado Comum do Sul (MERCOSUL) sem a necessidade de apresentar o passaporte ao ingressar, o que diminui a quantidade de documentos necessários para se realizar uma viagem pela América do Sul [50].

A emissão do RG, diferente da Certidão de Nascimento, pode ser realizada em qualquer momento da vida, por isso o seu processo inicia-se com o agendamento do atendimento no Órgão Expedidor e com o cidadão apresentando o documento necessário conforme suas características. Para menores de idade: certidão de nascimento e acompanhamento do responsável; solteiros: certidão de nascimento; casados, viúvos ou divorciados: apresentar a certidão de casamento; para naturalizados: certidão de naturalização. Após isso, são validados os documentos apresentados e, caso seja a primeira via, a emissão é gratuita e deve-se aguardar o documento ser emitido, e seja a segunda via, é gerado um boleto que, após a confirmação do pagamento, é necessário aguardar a emissão do documento. A imagem 2, representa o fluxograma em BPMN, do processo de emissão do RG.

Imagen 2: Fluxograma de Emissão do Registro Geral



FONTE: Autoral

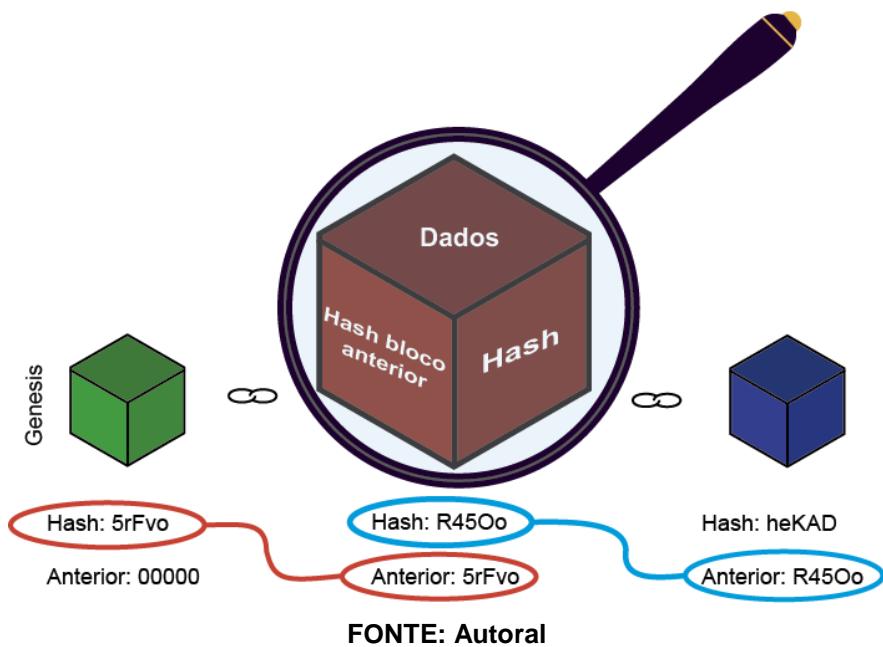
2.4. A Blockchain

Criada em 1991 pelo grupo de cientistas Stuart Haber e W. Scott Stornetta como uma forma de armazenar suas anotações laboratoriais sem correr o risco de serem alteradas [28], a *blockchain* não teve muito uso fora dos laboratórios até que, em 2008, a pessoa ou entidade de pessoas conhecida como Satoshi Nakamoto alterou sua estrutura para comportar sua nova tecnologia: a bitcoin. Desde então, a *blockchain* vem sendo alterado para comportar novas tecnologias e novas formas de usá-lo fora do mercado da criptomoeda. Sendo utilizado hoje por outros países, como a Estônia [53], para cuidar da documentação, finanças, sistema de saúde, eleições, entre outros. Além desta situação, muitas outras soluções vêm sendo estudadas, como é possível observar pelo caso da BRF e Carrefour, que pretendem utilizar a tecnologia da *blockchain* para tentar ter um maior controle do fluxo de seus alimentos e da qualidade acerca deles [8].

Agora que foi possível entender um pouco sobre a versatilidade e as possibilidades de aplicação desta solução, pode-se começar a compreender melhor como ela funciona. Cada *blockchain* é formado por blocos, como o nome sugere. O primeiro deles de cada *blockchain* é denominada *genesis* ou "bloco 0", que é responsável por ser o criador de toda esta corrente. Os demais são formados por três tipos de informação: dados referentes aquele bloco específico, como a origem dos dados, para quem foi enviado e a quantidade de moedas enviadas, o hash, que é formado a partir de toda a informação contida dentro dele mesmo e o hash do bloco anterior, para

garantir que a corrente não seja quebrada, e funciona como identificador único daquela unidade (uma impressão digital). A junção de todos nodos é responsável por gerar a corrente inteira da *blockchain*. A arquitetura desta tecnologia foi definida assim para que seja possível detectar qualquer alteração feita em um bloco. Tendo em mente que o identificador dele é gerado com base nas informações contidas dentro do mesmo, qualquer alteração realizada pode ser a partir do momento que o hash do bloco a frente não irá mais coincidir com o seu antecessor.

Imagen 3: Ilustração de blocos de uma *blockchain*



Para que um novo bloco seja adicionado dentro dessa estrutura de dados, uma prova de que certo poder computacional foi investido deve ser gerada pelo usuário. Depois o bloco, com a prova, é propagado para todos os membros da corrente, para que seja possível de se validar a legitimidade desse novo bloco. Esta tática é chamada de "*proof of work*" (POW) e serve para que não seja possível adicionar diversos blocos em um segundo, já que, hoje em dia, os computadores são capazes de calcular funções *hash* em milésimos de segundos. Este é um dos processos possíveis para que também exista uma coerência, ou consenso, dentro da rede. Além do algoritmo POW outras operações de consenso existem como por exemplo: "*Proof of Authority*" e "*Proof of Elapsed Time*". Atualmente, o tempo de resolução para esta

fórmula dentro da tecnologia do bitcoin é de 10 minutos [44], enquanto para o Ethereum, explicado na Seção 2.6, o tempo é de cerca de 15 segundos. Quando a equação é resolvida, o novo bloco gerado é passado para os demais participantes da *blockchain*, para que a resolução seja validada e não haja fraudes, e quando é comprovada verídica o bloco enviado é adicionado a todas as instâncias da *blockchain*.

Esta forma de funcionamento tem êxito devido ao fato de a *blockchain* ser baseado em uma arquitetura de rede *peer-to-peer* (P2P) - em que a premissa é que cada participante da rede possa desempenhar tanto o papel de Cliente quanto o de Servidor. Desde que não sejam desempenhados os dois papéis ao mesmo tempo, essa arquitetura possibilita a transmissão de dados imutáveis de uma forma descentralizada, na qual cada novo participante, ao entrar dentro da corrente de blocos, ganha uma cópia do estado atual dela mesma [45]. Desta mesma forma, toda vez que algo é alterado dentro da *blockchain*, todos os nodos recebem uma atualização do que foi alterado, mantendo o comportamento descrito anteriormente. Desta forma, a *blockchain* é capaz de garantir integridade em todas suas instâncias, não ficando vulnerável a falhas, sejam elas de rede ou técnicas, ou a fraudes, devido ao consenso, explicado a seguir, gerado entre todos os participantes.

Como anteriormente explicado, quando um novo nodo é adicionado a corrente de nodos, todos os participantes desta corrente recebem uma notificação e uma cópia do novo bloco gerado para garantir que o cálculo tenha sido resolvido de maneira correta. Esta técnica é chamada de consenso dentro da *blockchain*. Apenas quando um consenso é atingido dentro da *blockchain* um nodo deve ser adicionado dentro da corrente. Esta técnica é empregada dentro desta tecnologia para que a possibilidade de, caso haja uma alteração maliciosa dentre as alterações realizadas, o dono daquela alteração seja obrigado a enganar todos os usuários daquela corrente. O que se é aplicado em pequena escala pode ser relativamente fácil, mas quando promovido para larga escala se torna perto de impossível.

Devido a arquitetura de dados ser descentralizada e o sistema ser baseado em uma aplicação distribuída, há, portanto, uma cópia do estado atual em cada nodo. Sendo assim, nenhum bloco da corrente se torna "confiável" e, para que uma transação ocorra naturalmente dentro da

blockchain, o que está sendo realizado deve ser enviado via *broadcast* para os demais usuários que irão atuar na validação e na realização do consenso. Desta forma, todas as transações são vistas por todos os participantes do sistema e permite uma maior segurança, mantendo, também, uma imagem unificada do estado atual da plataforma, o que torna a inserção de blocos ilegítimos deveras difícil.

2.5. Smart Contracts

Smart Contracts, segundo Szabo (1997 citado por MILLER e outros, 2013, p. 1), são arranjos do tipo contrato expressos em código de programação, em que o comportamento do programa impõe os termos do “contrato” [34]. Ou seja, são contratos que possuem seus termos definidos de forma automática através de regras preestabelecidas que são estruturadas de forma lógica e desenvolvidas em uma linguagem de programação. Sendo que os mesmos são equivalentes aos contratos que hoje utilizamos para diversas situações do cotidiano, como por exemplo: confissões de dívidas, aluguéis, venda ou compra de imóveis, ações judiciais, etc. que são escritos em linguagem jurídica, sendo muitas vezes incompreendidas por cidadãos que não possuem este nível de conhecimento.

Um dos principais objetivos de utilizar *smart contracts* está relacionado a agregar confiança na relação direta entre fornecedor e o cliente, sendo possível retirar a necessidade de incluir uma terceira ou quarta parte nestes processos, apenas com a funcionalidade de garantir a veracidade. Como exemplo, utilizamos o seguinte cenário: o cliente de um estabelecimento deve confiar que receberá seu produto após uma compra realizada via internet, enquanto que, por sua vez, o fornecedor deve confiar que o crédito informado pelo cliente é suficiente para quitar a transação realizada. Neste cenário descrito, atualmente não há confiança direta entre os envolvidos, sendo necessária a ação de uma terceira ou quarta parte a fim de atestar a confiança entre cliente e fornecedor, sendo que estas partes cobram taxas preestabelecidas para realizar essa validação. Com os *smart contracts*, não há necessidade de intermediadores para garantir a confiança da transação, pois sabemos que as regras preestabelecidas foram cumpridas e validadas

automaticamente, reduzindo a ação de outras partes no processo e consequentemente por reduzir o custo agregado na transação a ser realizada. O desenvolvimento desses contratos, hoje, pode ser aplicado para diversos fins, como contratos por eleição, *crowdfunding*, leilões, votação, entre outros.

2.6. Ethereum

O Ethereum foi apresentado ao mundo por Vitalik Buterin, em 2014, como uma solução de *blockchain* que possibilitasse a criação de soluções descentralizadas, assim como a *blockchain* apresentado por Nakamoto em 2009 - o Bitcoin. Entretanto, diferente do Bitcoin, o Ethereum possibilita uma adaptabilidade e flexibilidade para o desenvolvimento de novas soluções baseadas em sua plataforma, o que o torna tão revolucionário quanto o Bitcoin, em relação ao seu lançamento em 2009 [23] [56].

Um dos principais diferenciais do Ethereum, para outras soluções que utilizam *blockchain*, é que o seu projeto foi desenvolvido não somente para criptomoedas, possibilitando ao desenvolvedor, desde o início de seu projeto, a elaboração de *smart contracts*. O próprio Buterin (2014, p.1) descreve esses contratos inteligentes utilizados pelo Ethereum como:

O Ethereum pega essa ideia e a leva um passo adiante. Em vez de os contratos serem acordos entre duas partes que começam e terminam, os contratos na Ethereum são como uma espécie de agente autônomo simulado pela blockchain. Cada contrato Ethereum possui seu próprio código de script interno, e o código de script é ativado toda vez que uma transação é enviada para ele. A linguagem de script tem acesso ao valor da transação, ao remetente e aos campos de dados opcionais, bem como alguns dados de bloqueio e sua própria memória interna, como entradas, e podem enviar transações.

Entretanto, foi Dr. Gavin Wood que apresentou a definição formal da Ethereum Virtual Machine (EVM) como uma solução baseada em transações de estados de máquinas, com a finalidade de construir uma generalização no conceito tecnológico. A EVM é capaz de fornecer uma incorruptibilidade de julgamento, pois na forma de um intérprete algorítmico desinteressado no processo, a mesma também permite a transparência e o registro de todas as

interações que ocorreram de forma a deixar claro como o julgamento foi realizado [17] [23] [58]. Atualmente, não é claro como os julgamentos são realizados, pois ocorrem através das decisões e interações humanas, sendo as mesmas descritas em linguagem natural, o que permite diferenças de interpretações, gerando dúvidas e desconfiança no processo. Por isso, a EVM automatiza essas decisões a partir de estados de máquinas com suas regras previamente definidas e que são seguidas continuamente, sempre que os requisitos forem atendidos.

Existem outras arquiteturas de contratos que são suportadas atualmente pelo Ethereum, permitindo uma configuração única e exclusiva de ações a serem realizadas de acordo a necessidade implementada. Como exemplo Wood (2017, p.14) apresenta em seu artigo o contrato denominado Feed de Dados como:

Feed de Dados. Um contrato de feed de dados é aquele que fornece um único serviço: dá acesso a informações do mundo externo dentro do Ethereum. A precisão e a pontualidade desta informação não é garantida e é a tarefa do autor do contrato secundário - o contrato que utiliza o feed de dados - que determina quanta confiança deve ser colocada em um único feed de dados.

2.7. Fundamentação do Problema: Veracidade dos Documentos

Atualmente se enfrenta grande dificuldade para a validação da veracidade de todos os tipos de documentos. Já houve casos, como o do próprio imperador norte-coreano, que tentou entrar com um passaporte brasileiro dentro da Europa [38]. Claro que, neste caso, houve sorte, já que o portador do passaporte era mundialmente conhecido e de fácil percepção, mas como fazer quando diversos documentos são criados com pouca tecnologia? Para que tal problema fosse resolvido, pode-se tentar analisá-lo de duas formas:

1. Os diferentes sistemas atuais deveriam possuir maior interface para com as outras aplicações, permitindo maior comunicação e maior agilidade na verificação dos documentos.

2. Maior emprego de tecnologia na criação dos documentos, dificultando assim a criação de registros falsificados facilmente.

Começando pelo primeiro ponto, é natural imaginar que tal solução poderia ser deveras fácil, apesar disso, com tantas secretarias e estados no Brasil, seria quase impossível conseguir colocar todos em comum acordo para investir em um projeto de tal magnitude, assim como garantir que toda a documentação dos sistemas estivesse em dia com o estado atual da aplicação. Neste caso, iria ser necessário conduzir uma ampla análise de tudo que há hoje e todas as diferenças que existem, para garantir, primeiro, uma maior permeabilidade entre a mesma aplicação dos diferentes estados e, depois, uma visão unificada dos tipos de dados existentes em cada base de dados. O trabalho poderia ser comparado com o de uma criação de *um Data Warehouse*, que necessita de um grande tempo de análise antes de sua implementação, apresentando diversas dificuldades de interpretação de dado e visão unificada.

Já na segunda possível solução seria difícil garantir quanto tempo a tecnologia presente nos documentos iria durar antes de ser quebrada. Como sabe-se, hoje, nada na área de Tecnologia da Informação (TI) fica seguro por muito tempo. De vídeo games [27] até bancos de dados [4], toda tecnologia que fica desatualizada por muito tempo corre o perigo de sofrer uma quebra em sua segurança. Atualmente outras plataformas resolvem este problema com sistemas de validação online, mas os documentos governamentais não poderiam usar tal solução, já que devem ser impressos em grande massa para a população toda vez que uma alteração é realizada e seus padrões devem ser mantidos por vários anos. Outro fator que poderia ser atribuído ao fracasso dessa segunda opção, seria o custo, como podemos observar no caso do Registro de Identificação Civil (RIC) que teve seu andamento suspenso em 2015, devido ao alto custo do mesmo, gerando uma demanda impossível de cobrir para tal tecnologia pelas secretarias e a casa da moeda [21].

3. OBJETIVOS

Neste capítulo estão descritos os objetivos a serem alcançados com a realização deste trabalho.

3.1. Objetivo Geral

O objetivo geral deste trabalho é propor uma solução para a criação, gerenciamento e compartilhamento de documentos, utilizando o conceito de *blockchain*, com ênfase na garantia da veracidade destes e unificando os diferentes sistemas em uma plataforma.

3.2. Objetivo Específico

Este trabalho tem como objetivo específico propor uma solução usando a tecnologia da *blockchain*, através da distribuição *open source* do Ethereum, para os problemas citados anteriormente dentro da introdução e da fundamentação teórica. Outro objetivo neste trabalho é praticar aquilo que foi aprendido durante o percorrer do curso de Sistema de Informação (SI) e ver em prática os conceitos passados em sala de aula em uma situação real de desenvolvimento e implementação. Há também o interesse em aprender mais sobre a tecnologia da *blockchain*, que tem entrado em voga, tanto no mercado profissional, quanto na área científica [35], não só por sua excelente utilização nas atuais criptomoedas, mas também por sua capacidade arquitetural em manter aplicações confiáveis, imutáveis, transparentes, íntegras, entre outras qualidades inerentes a esta tecnologia [55].

Acredita-se fortemente que, através da aplicação proposta, explicada no Capítulo 4, foi plausível demonstrar como novas tecnologias podem influenciar positivamente no setor público, não só nas áreas aqui trabalhadas (criação, organização e unificação dos diferentes documentos), mas também em outras áreas, como nas áreas da saúde, financeira, empresarial, entre outras. Reforçando, então, a necessidade de um olhar mais digital para com as soluções nos setores privados e públicos.

4. PROPOSTA DE SOLUÇÃO

Para este trabalho, como solução, pensa-se em uma implementação do *open source Ethereum*, utilizando-se de suas funcionalidades como *Smart Contracts*, para desenvolver uma plataforma capaz de realizar as funções de criar, organizar, gerir e validar documentos, listados durante a introdução e a fundamentação teórica, para o governo, com o intuito de trazer as qualidades inerente da tecnologia da *blockchain*, sendo elas integridade, confiabilidade, durabilidade, sustentabilidade e segurança. Estas qualidades, que são prerrogativas aos sistemas hoje de documentação dos setores públicos, devem ser traduzidas de forma intrínseca na aplicação, devido ao fato de lidar com dados sensíveis e de cunho legal e pessoal.

Referente, ainda, às qualidades do sistema, é importante ressaltar como as mesmas foram traduzidas do mundo real para o sistema. A integrabilidade, nos dias atuais, é vital para que documentos sejam criados da forma que foram projetados. Se existir certa disparidade entre eles, seja ela visual ou das informações presentes dentro deles, é possível que tais documentos sejam invalidados ou, até mesmo, não aceitos em outros estados. Desta forma, pretende-se traduzir esta integridade com regras definidas dentro de *Smart Contracts* divididos por documentos, mantendo, assim, a criação de cada um da mesma forma, evitando incoerências e disparidades.

A confiabilidade é, também, de suma importância, tendo em vista que tais documentações não possam ter sequer uma sombra de dúvida referente a veracidade deles. Seus usuários devem acreditar que sua origem e sua validade perante o território nacional e internacional é totalmente confiável, pois, caso contrário, o documento perde seu propósito e valor. Dessa forma, planejamos implementar que a criação destes seja realizada apenas por contas específicas da *blockchain*, com caráter governamental e que tenham acesso a essa funcionalidade. Desta forma, quando alguém realizar o pedido da criação de um documento, sua validade não poderá ser questionada, assim como sua origem, que será sempre realizada com base em um número seletivo de contas.

Em relação a durabilidade, ultimamente alguns documentos têm sua validade específica, como o RG que é válido em todo território nacional e no

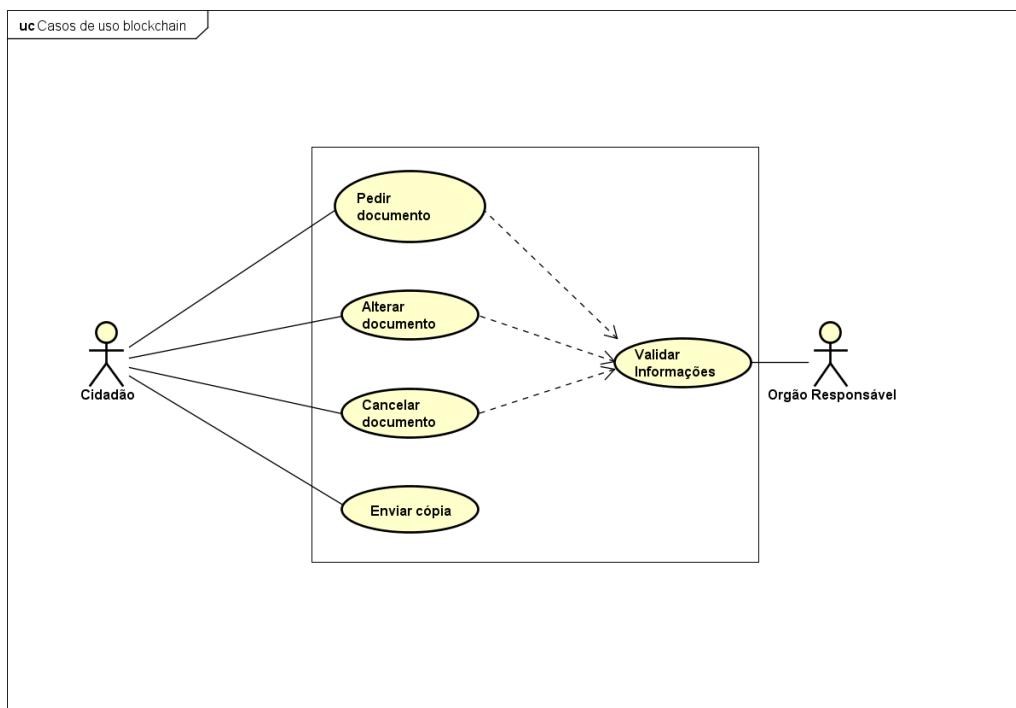
Mercosul durante dez anos, e outros não têm data de expiração explícita. Apesar da durabilidade, assim como a persistência dos dados não ser fortemente questionada hoje no mundo digital, é um fator vantajoso da *blockchain* que cada transação sua é gravada de maneira imutável, mantendo, assim, o dado ali gravado durante o tempo de funcionamento da plataforma.

Ainda referente a sustentabilidade dos documentos, o Brasil, hoje em dia, apresenta dificuldade em atualizar seus registros já existentes devido ao alto custo de impressão em massa que é necessária para o novo documento, assim como o fato de que, para muitos brasileiros, por falta de recursos, a impressão deveria ser entregue de forma gratuita. Além disso, o custo de manutenção e armazenamento referente às décadas de processos realizados acumulam a tal problema. Pensando desta forma, é que foi decidido realizar a solução para este problema utilizando a tecnologia de *blockchain*, que apresenta um custo mais baixo [2], assim como amenizar o custo, já que não existe a impressão do documento, apenas o de manutenção e o da transação para com a plataforma.

Por final, em relação a segurança, é de conhecimento geral e já aqui citado neste trabalho que, devido à falta de atualização tecnológica e da falta de um sistema unificado, tais documentos podem ser falsificados e usados por outros que não deveriam ter permissão para usá-los. Para este fim, além da ideia citada anteriormente, ou seja, de criação destes registros com base em contas governamentais, foi desenvolvida a ideia do empréstimo ou da cópia autenticada de tais documentos dentro da plataforma de *blockchain*. Nesta funcionalidade, um usuário poderia escolher enviar uma cópia de seu registro para outro. Assim esta pessoa teria como usar deste documento para assuntos legais, como no caso de procurações, das quais um advogado ou outro indivíduo usa de uma permissão prévia, para agir em nome de outra pessoa, e também com o intuito da criação de uma cópia autenticada pelo próprio sistema.

Nas funções que devem ser implementadas na solução aqui proposta seriam as de criar, gerir, alterar ou cancelar documentos já existentes, e compartilhá-los também. Para isto, no processo de ideação deste trabalho, é seguido o seguinte modelo:

Imagen 4: Casos de uso para a blockchain



FONTE: Autoral

Pedir documento: Neste caso de uso, o usuário realiza um pedido de criação de um documento, já implementado dentro da plataforma, através de um *front-end*. Nesta ação seria requerido dele preencher um certo formulário, no qual ele colocaria as informações hoje necessárias para criar o mesmo documento em questão.

Alterar documento: Processo responsável por alterar quaisquer informações do registro pessoal desejado pelo usuário. Neste caso de uso, o cidadão escolhe um documento, que o mesmo já possui, e altera suas informações. As informações editáveis são alteradas em cada registro e são decididas conforme a implementação mais à frente.

Cancelar documento: Neste caso de uso, o usuário realiza um pedido de cancelamento ao seu documento. Nele também é necessário informar um motivo para tal processo. Os motivos, assim como a forma de cancelamento, são decididos durante a fase de desenvolvimento da plataforma.

Enviar cópia: Muitas vezes, é preciso entregar cópias autenticadas para outra pessoa para a realização de alguma atividade civil. Neste caso de uso, o usuário envia uma cópia autenticada do seu documento na *blockchain*.

para outro usuário. Este deve ser marcado como cópia para evitar que seja utilizado por outros, como se fossem os donos reais deste registro.

Validar informações: Por último, neste caso de uso, todas as ações que exigirem análise, seja ela do sistema ou de uma entidade governamental, passam por este processo. Esta validação existe para garantir que mais de um documento do mesmo tipo não seja gerado para a mesma pessoa e também para validar informações, como foto, endereço, entre outras que podem ser necessárias maior análise.

Entende-se que tal solução possui espaço hoje para ser desenvolvida no atual cenário social e tecnológico, em âmbito global, devido a duas premissas básicas levantadas durante a fase de ideação deste trabalho:

1. A vivência pessoal referente aos problemas relacionados a gerar, gerir e manter os documentos aqui citados;
2. A atual existência de uma solução parecida com a deste trabalho;

Referente à premissa número um, é natural que todo cidadão passe pela frustração que todos passam ao longo da vida, referente ao processo longo e burocrático que é criar uma documentação ou tentar alterar um dado da mesma. Sabe-se também que alterar tais documentos têm alto custo para o estado, tendo em vista a dificuldade do RIC para as diferentes partes do país. E, por último, sabe-se por vivência dos problemas encontrados por todos os brasileiros, o que é ter que ir durante o horário de trabalho à uma secretaria ou portaria do estado realizar tais processos, devido às rotinas diárias que coincidem com o horário de funcionamento destes locais.

Sobre o segundo ponto, foram referenciadas, no Capítulo 7, outras empresas e países que estão movendo seus processos e seus sistemas para uma abordagem semelhante a proposta aqui relatada.

5. ARQUITETURA DA SOLUÇÃO

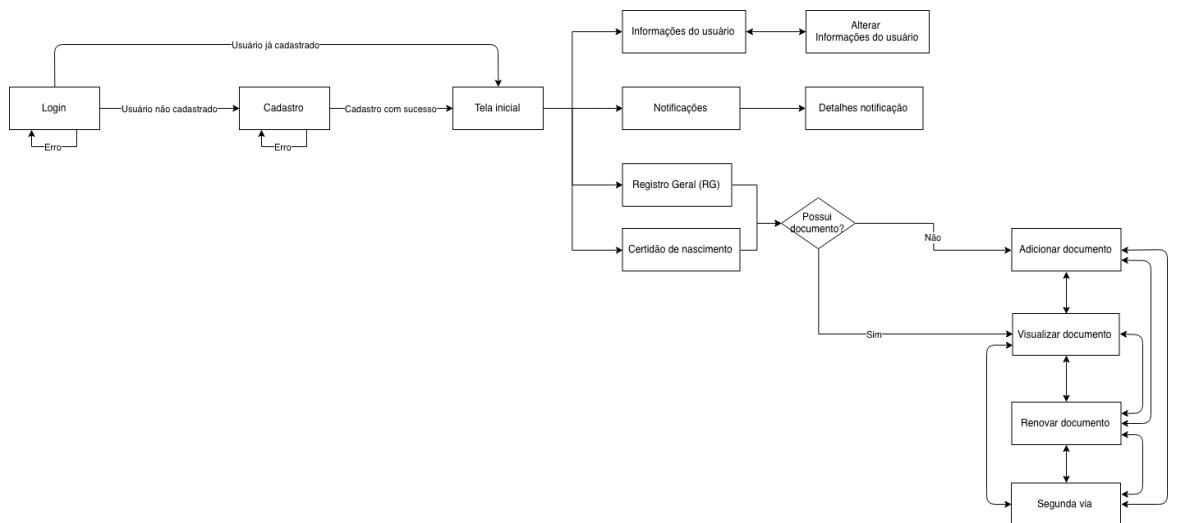
Neste capítulo foi descrita a arquitetura da solução, descrevendo as tecnologias utilizadas no *Front-end* e no *Back-end* da aplicação desenvolvida.

5.1. Front-End

Para o *front-end* foi utilizada a tecnologia Angular, em sua versão *major* seis [5]. Desta forma foi possível utilizar validadores quando criados os formulários, adicionado apenas nesta versão e outras funcionalidades de *releases* passadas, como *http clients* para chamadas de serviços e compatibilidade com *APIs* externas.

Além disso, no *front-end*, as boas práticas de desenvolvimento Angular foram usadas, procurando criar componentes diferentes para a reutilização de código e baixa manutenibilidade ao longo do desenvolvimento. Também houve o uso da navegação através de rotas, permitindo que componentes filhos da tela de login tenham, também, seus fluxos internos. Para uma melhor representação, foi criado o fluxo na imagem 5:

Imagen 5: Fluxograma do front-end



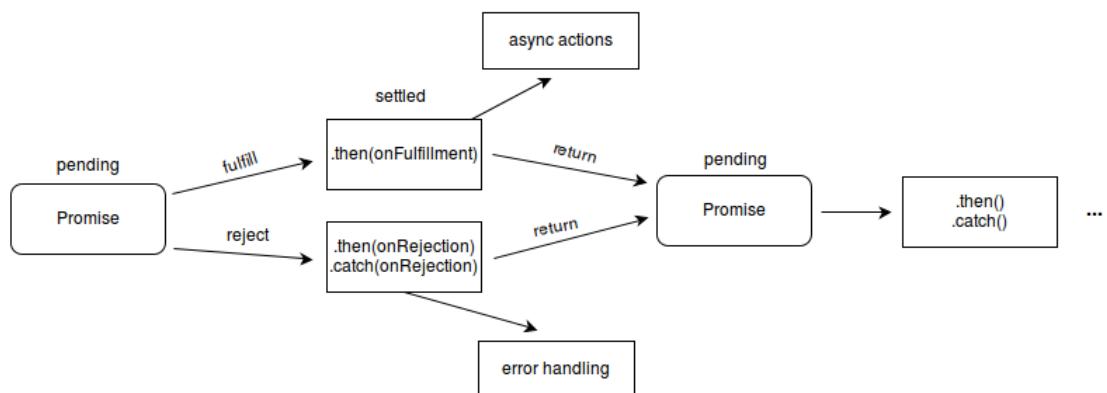
FONTE: Autoral

Esta tecnologia foi escolhida pela familiaridade com a mesma, assim como se manter no ecossistema das tecnologias Google, tendo em vista a

escolha por modelo de *Backend as a Service* (BaaS). Conseguindo, assim, uma velocidade maior de desenvolvimento na hora de integrar as soluções.

Além do Angular, foram utilizadas duas *Applications Programming Interface* (API) que possibilitaram o desenvolvimento deste trabalho e devem ser citadas. Vale ressaltar também que todos os serviços criados para a aplicação trabalham com promessas, é um objeto usado para processamento assíncrono, e garante maior agilidade na construção das páginas e segurança nos tratamentos de erros [40]. Toda requisição para a visualização ou criação de um documento retornam este objeto que apresenta sempre um estado (pendente, realizada, rejeitado, estabelecida) e ao concluir a função como esperado retornam a resolução, ou ao concluir de maneira não esperada retornam uma rejeição. O ciclo de vida de tal objeto se encontra na imagem 6.

Imagen 6: Ciclo de vida de uma promessa.



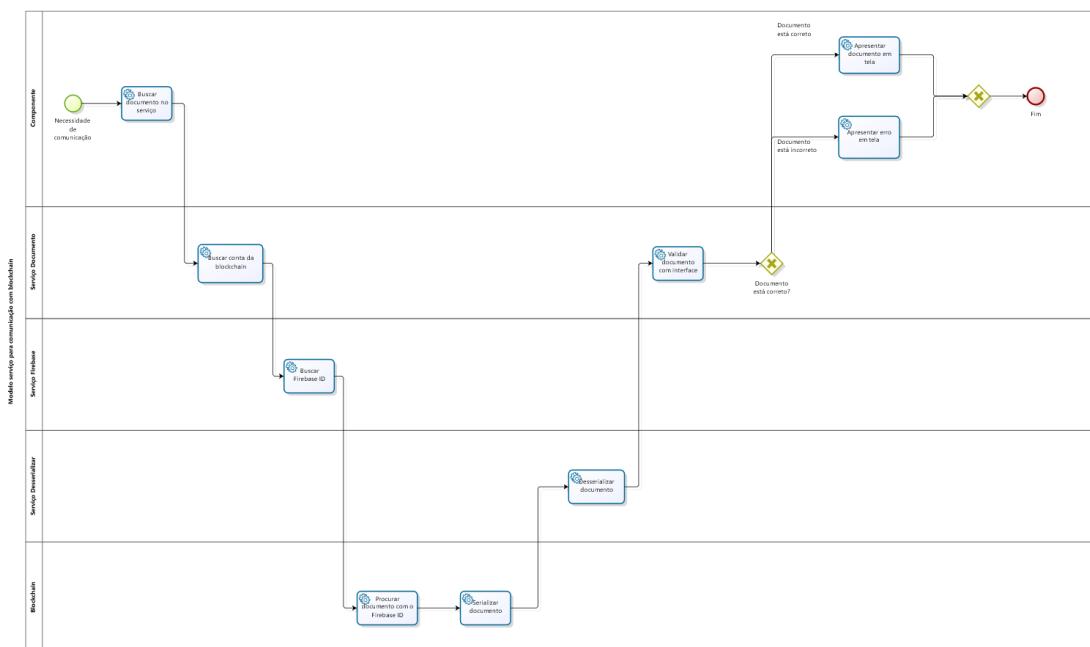
Fonte: Promise (2018).

5.1.1. Web3:

Esta dependência foi utilizada para realizar a comunicação entre o *front-end* e os contratos, citados na Seção 5.3, sem a necessidade da criação de um serviço *REST* para a troca de dados e a criação de novos documentos/blocos na blockchain [22]. Esta *API* permite instanciar contratos com endereços específicos e acionar suas funções através de *Application Binary Interface* (ABI), que é responsável por expor os métodos e atributos dos contratos para o desenvolvimento no *front-end*, assim como permite calcular o custo das transações. A *ABI* é salva em um arquivo JSON dentro do projeto

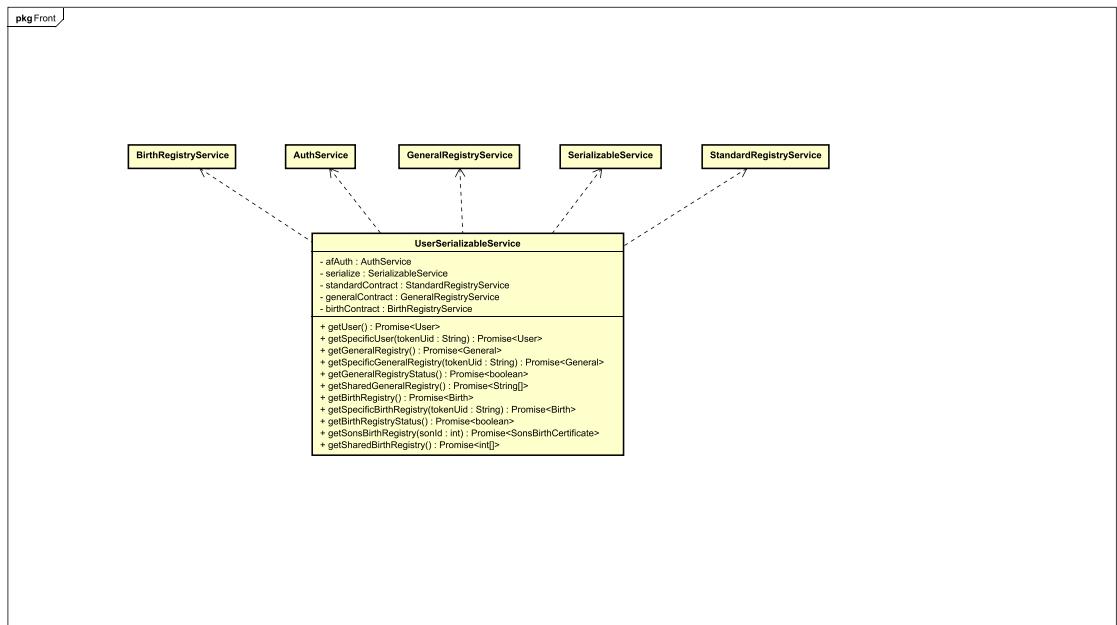
para que possa ser passada durante o tempo de execução para o Web3. Ao utilizar esta *API* nesta aplicação foi possível realizar a criação de serviços Angular (classe que não possui template ou estilo e tem apenas um objetivo único, comumente uma ação, garantindo assim maior modularidade e reusabilidade [5]. Desta forma, o modelo na imagem 7 foi possível de ser executado. Em conjunto com esta API fora criado um serviço Angular para cada documento. Desta forma é possível separar as funções por contexto e não ter uma classe grande que é responsável por todos os documentos. Apesar disso para garantir certa praticidade e diminuir o número de dependências entre os componentes, e uma forma de centralizar a lógica apresentada na imagem 7, foi criado um serviço de orquestração de todos os documentos que podem vir a ser apresentados em tela e executar a chamada da desserialização de tudo que é recebido do *blockchain*. Este serviço porém só possui métodos de busca de informação e não alteração e se encontra na imagem 8.

Imagen 7: Modelo serviço para comunicação com *blockchain*



FONTE: Autoral

Imagen 8: Diagrama de classe para serviço de serialização do usuário



FONTE: Autoral

5.1.2. Firebase / AngularFire2:

Dependência utilizada para realizar a orquestração das sessões dos usuários “logados”, assim como garantir a solução de login e validação das informações durante o processo de identificação do usuário [6]. Esta API permite trabalhar com a estrutura de observables, de forma assíncrona e, também, “ouvir” por notificações por usuário. Além de garantir uma gestão de identidade para a aplicação, foi utilizado nesta solução o Firebase ID como uma forma de gerar um identificador único entre o *front-end*, *blockchain* e *BaaS*. Devido ao fato de a linguagem de programação para contratos ainda não possuir funcionalidades como a geração de números randômicos seria muito custoso gerar estes identificadores dentro do contrato e realizar a comunicação entre as múltiplas camadas de sistema. Desta forma ao criar uma conta na Firebase (primeiro passo do registro de qualquer usuário) é passado o identificador para o contrato e armazenado ele lá também, como apresentado no fluxo na imagem 29.

5.2. Back-end

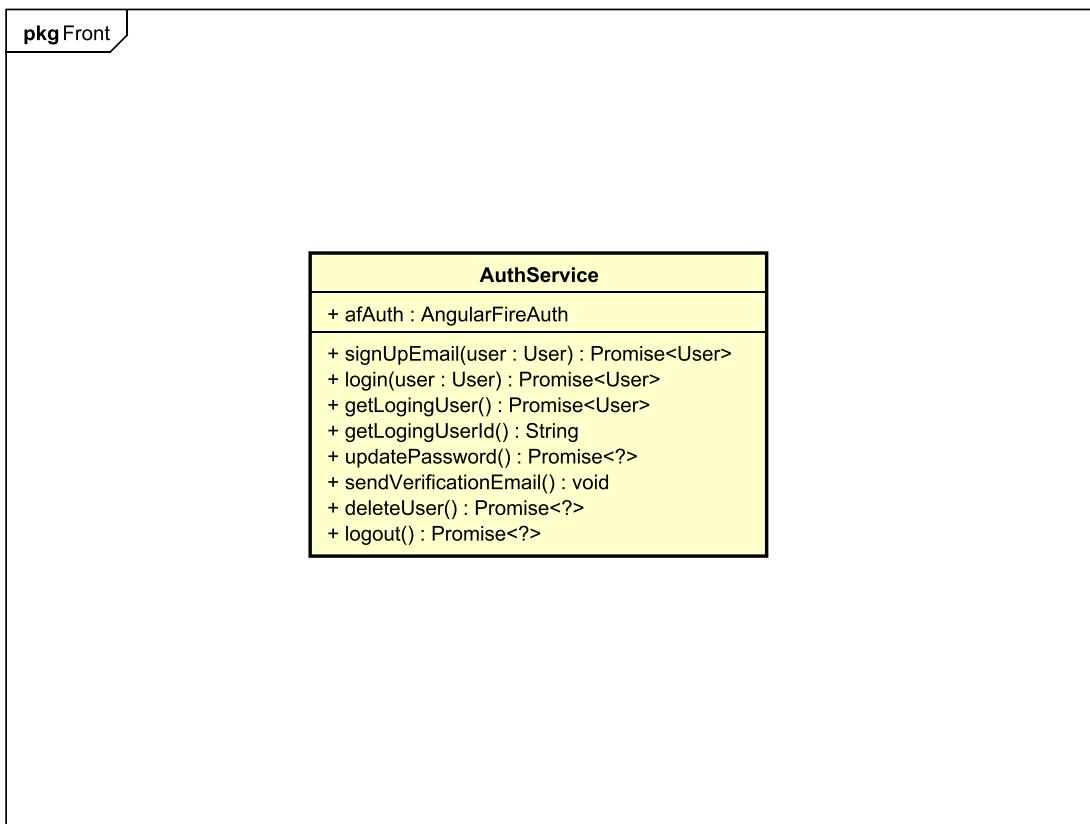
Com o intuito de diminuir o tamanho do *back-end* da aplicação, assim como facilitar o desenvolvimento, foi utilizado um modelo BaaS. Desta forma, a identificação dos usuários, foram delegadas para a solução descrita a seguir.

5.2.1. Firebase:

É uma plataforma que, através de diferentes APIs, oferta diferentes serviços para diversas aplicações. Para a solução deste trabalho, esta plataforma foi utilizada para o controle de autenticação e cadastro dos usuários [42], assim como a sua geração de um identificador único [42]. Esta solução do Google eliminou praticamente todo o esforço necessário para o desenvolvimento destas duas funcionalidades, garantindo controle, segurança e uma solução sem custo para este trabalho.

A comunicação com esta solução da Google foi feita através da API Angularfire2 explicada na Seção 5.1.2 e do serviço criado para a orquestração e centralização das funções necessárias para múltiplos componentes, diminuindo assim código repetido e mantendo a lógica concisa em apenas um lugar.

Imagen 9: Serviço de orquestração de chamadas Firebase



FONTE: Autoral

5.2.2. Ethereum

Para a criação do blockchain, assim como a dos Smart Contracts (Seção 5.3), foi utilizado o framework Ethereum que foi descrito na Seção 2.6. Nele foram implementados os contratos e criado as contas para a fase de testes. Na arquitetura desta aplicação a *blockchain* foi utilizado para armazenar os documentos do usuário, assim como armazenar as notificações do sistema. Assim sendo foram implementados cinco contratos. Parse, responsável por transformar String em milissegundos, comparar Strings e realizar cortes em Strings (operações que não existem hoje na linguagem); Standard Registry, responsável por armazenar o documento padrão explicado na Seção 5.3.2; General Registry, onde são armazenados os RGs; Birth Registry, no qual estão os dados referentes a certidão de nascimento e Notification, que é responsável por enviar as mensagens aos usuários, assim como armazenar as mensagens já enviadas pela aplicação.

Além disso por ser uma *blockchain* privada todas as transações realizadas que tem custo (mineração de blocos) é realizada de maneira gratuita. Apesar de haver consumo de Ether, assim como o Ethereum realizar o cálculo de custo, nada é descontado dos usuários em tempo de transação. Uma medida de cobrança poderia ter sido adotada, mas foge do escopo proposto aqui.

Para mais detalhes todos contratos serão explicados a seguir.

5.3. Smart Contracts

Como citado na Seção 5.2.2 deste documento, utilizou-se para o desenvolvimento da parte referente a *blockchain* da solução a plataforma descentralizada *open source* Ethereum. A partir dela foi possível construir os contratos citados nesta Seção, assim como aplicar todas as regras de negócio levantadas durante a etapa de entendimento dos processos atuais para os documentos de certidão de nascimento e RG.

Ethereum permitiu abstrair toda a complexidade da criação formal de uma plataforma descentralizada e possibilitou aplicar esforços em cima do que realmente era preciso, os contratos. Apesar disso, ainda foi utilizada a ferramenta Truffle / Ganache para o desenvolvimento que será explicada a seguir.

5.3.1. Truffle / Ganache:

É uma *suite* que provê ferramentas de desenvolvimento com o objetivo de facilitar a vida dos desenvolvedores, assim como Angular CLI para aplicações Angular ou Create React App para aplicações React. Truffle permitiu criar um ambiente de desenvolvimento *blockchain* muito próximo ao ambiente real de produção. Ainda dentro desta ferramenta, foi possível criar um processo automatizado de *deploy* de *blockchain* e manter o último estado válido da aplicação entre um dia e outro sem ser realizados deploys em um ambiente com servidores.

Esta *suite* também permitiu a configuração do ambiente para melhor adequamento a realidade desta *blockchain* privada permitindo aumentar o

custo máximo de cada transação, assim como o custo do Gas para cada transação.

5.3.2. Contrato de Registro Padrão:

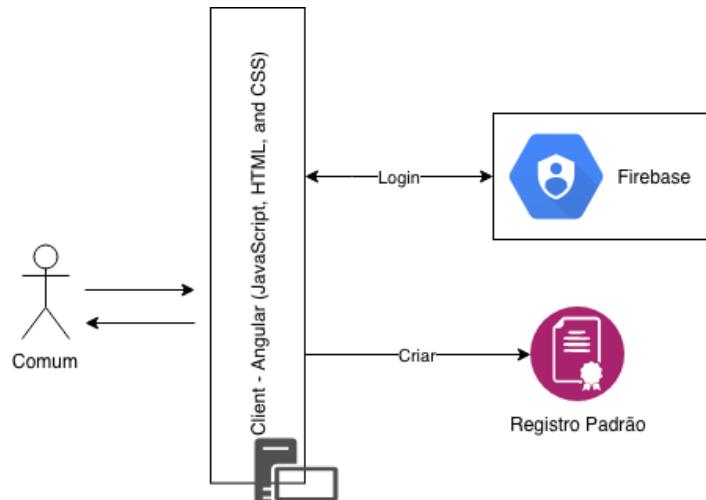
Para que seja possível validar a existência do usuário dentro da aplicação deste trabalho, foi gerado um contrato genérico responsável por manter as informações comuns do usuário, assim como aquelas que hoje já há a dificuldade de manter atualizadas, como endereço, telefone e e-mail. Desta forma ao tentar realizar a criação de um novo usuário a aplicação sempre irá procurar primeiro um usuário que compartilhe as mesmas informações (este fluxo foi exemplificado na imagem 29). Este contrato possui as seguintes informações:

- Primeiro nome
- Nome de família
- Telefone
- Data de nascimento
- Endereço
- Complemento de Endereço
- Cidade
- Estado
- CEP
- E-mail

Apesar de existir um contrato responsável pela certidão de nascimento, foi escolhido perguntar aqui, também, a data de nascimento, para obter controle se o usuário é menor de idade ou não. Além disso, este contrato serve para compartilhar as informações mais comuns a todos os outros. Desta forma, a replicação que existe hoje de informações, como por exemplo o nome, não acontece no sistema e mantém os outros contratos apenas com informações pertinentes ao documento. Por final, todos os documentos futuros irão validar se existe um registro neste contrato, como uma forma de validar

que o usuário existe no sistema. Para o contrato do registro padrão a arquitetura ficou de acordo com o diagrama contido na imagem 10:

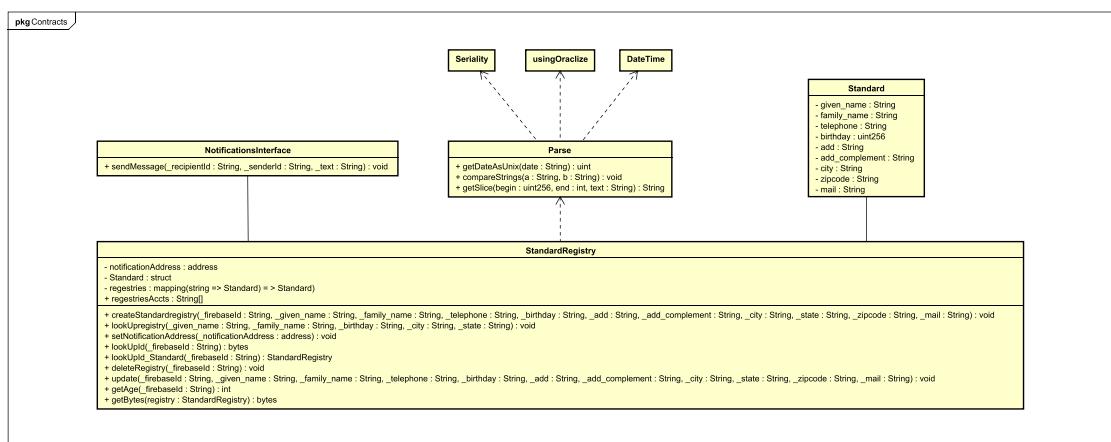
Imagen 10: Diagrama do Registro Padrão



FONTE: Autoral

Este contrato também possui uma interface para o contrato de notificações para que assim consiga realizar o envio de mensagens, quando um documento é alterado ou criado com sucesso. Desta forma a implementação deste contrato seguiu o diagrama de classe da imagem 11.

Imagen 11: Diagrama de classe para o contrato Registro Padrão



FONTE: Autoral

5.3.3. Contrato de Certidão de Nascimento:

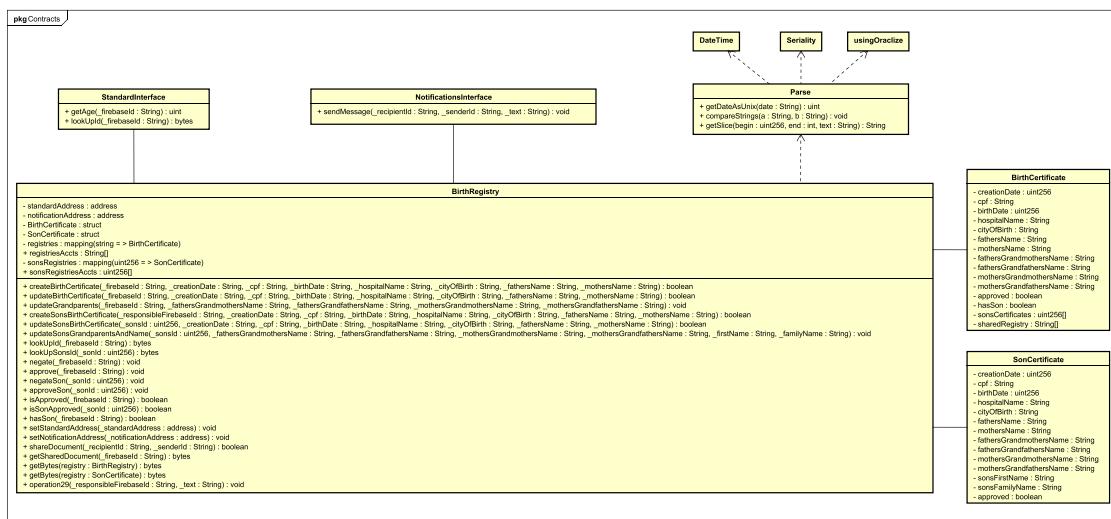
Tratando-se de um dos primeiros documentos que os cidadãos brasileiros possuem por direito, a solicitação de sua emissão para um recém-nascido é de responsabilidade de seus pais, contudo para que ocorra o reconhecimento de paternidade dessa criança, ou o processo é iniciado pelo pai - para que assim o ato voluntário de registrar seja reconhecido formalmente como reconhecimento de paternidade - ou por sua mãe, se a mesma possuir um documento oficial em que o pai autoriza o reconhecimento de paternidade. Sendo essas as principais regras para que a certidão de nascimento seja emitida, elas acabam por não mais se atenderem as necessidades de uma sociedade que busca alcançar uma igualdade de gêneros e uma pluralidade social no conceito familiar nos últimos anos. E com esse objetivo, o registro de uma criança nesta aplicação ocorre pelo seu responsável, desta forma, é plausível realizar um vínculo do documento da criança com o do seu responsável, até que ela atinja a maioridade penal e, assim, gerar um novo registro.

Dado que para a criação de um documento é preciso ter algum tipo de informação de que o usuário, cujo qual a transação se originou, existe e faz parte do sistema, foi decidido que, para a certidão de nascimento, todas as pessoas com menos de dezoito anos devem ser vinculadas a um usuário já existente na *blockchain*. Este vínculo foi realizado criando uma lista de filhos ao responsável que o está registrando. Se a implementação de um registro de casamento tivesse sido realizada, também seria necessário encontrar o outro usuário de registro e cadastrar o filho nele. Como o escopo deste trabalho era apenas a criação dos documentos de RG e certidão de nascimento, foi criada a função de compartilhamento para que o outro responsável tivesse como receber o mesmo documento de seu filho.

Nos casos de pessoas com mais de dezoito anos ou que passaram por um processo de emancipação, é possível adicionar um documento já existente. Para eliminar a possibilidade de existir dois documentos para a mesma pessoa, o identificador de cada documento é o mesmo identificador (ID) gerado pela solução Firebase. Desta forma, toda vez que o usuário acessa a página do documento é requisitado a *blockchain* os documentos

atrelados aquele identificador único. Se nenhum documento é encontrado, os menus de visualização e renovação são desabilitados e o usuário é direcionado ao menu para adição de um documento, seja ele já existente ou um novo, este fluxo se encontra presente na imagem 34. A implementação deste contrato ficou de acordo com o diagrama de classe exemplificado na imagem 12.

Imagen 12: Diagrama de classe para certidão de nascimento.



FONTE: Autoral

Caso o usuário já possua algum documento, ele consegue ver os seus registros e tem a opção de adicionar um novo documento, em caso do nascimento de um filho, ou renovar, atualizando suas informações, alterando sua data de vencimento e trocando o documento para um estado de pendente. Documentos pendentes precisam ser aprovados por um órgão governamental que, devido ao curto espaço de tempo para a implementação, não foi desenvolvido. Para validar o perfil de cada usuário foi disponibilizada na tela uma forma de chamar a funcionalidade para aprovar e desaprovar cada documento.

Para adicionar um novo documento o usuário apenas precisa preencher um formulário padrão com as informações hoje já necessárias. Estes dados compreendem-se por:

- CPF (opcional);

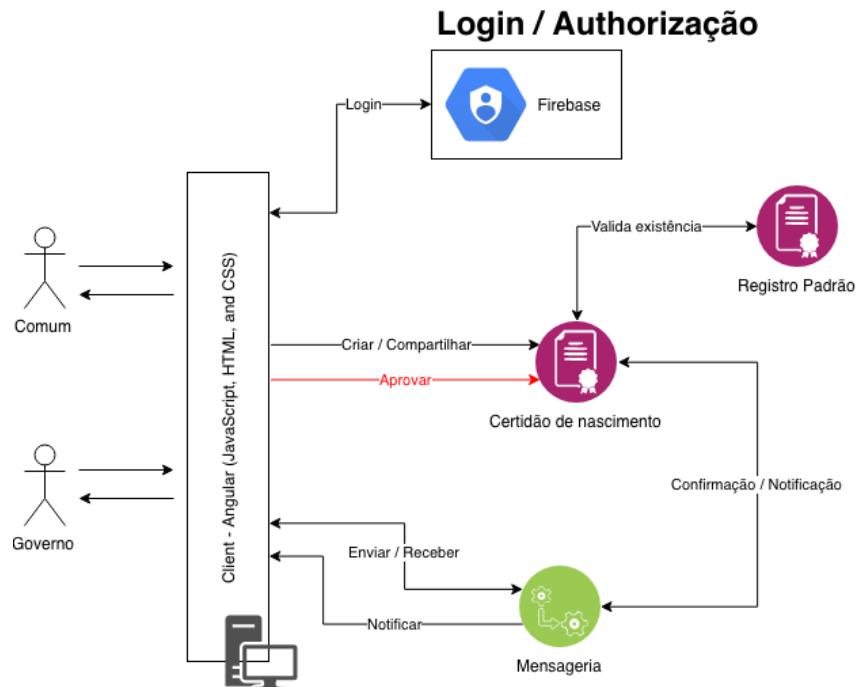
- Data de nascimento;
- Nome do hospital;
- Cidade do nascimento;
- Nome do pai;
- Nome da mãe;
- Nome do avô paterno;
- Nome da avó paterna;
- Nome do avô materno;
- Nome da avó materna.

Após o preenchimento destas informações, algumas validações são realizadas na camada *front-end*, como o preenchimento de todas as informações obrigatórias no formulário, assim como se as regras de negócio do documento foram atendidas.

No caso de a certidão de nascimento ser preenchida por um responsável legal para o seu filho, seja ele recém-nascido ou de um menor, mais um campo existirá para que seja apontado qual será o nome do menor, já que o registro do mesmo não existe ainda dentro do sistema.

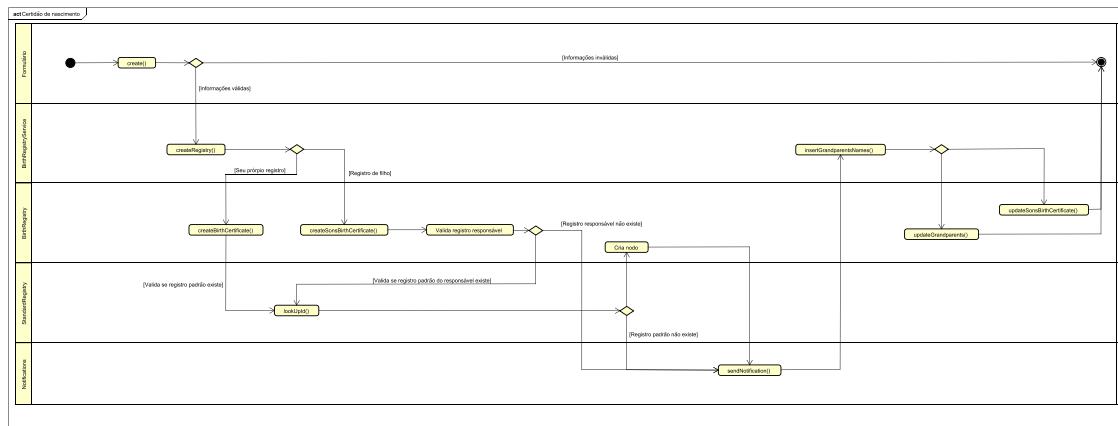
Caso todos as validações que foram citadas anteriormente estejam de acordo, os dados serão enviados para a blockchain, onde é validado que o responsável declarante já existe no sistema e tem um registro ativo. Na hipótese de um contrato de casamento ter sido criado, dever-se-ia também validar se o responsável é legalmente casado e se o registro de casamento existe. Todos integrantes do fluxo de criação e compartilhamento da certidão de nascimento, com o que foi deixado fora do escopo, pode ser visto no diagrama da imagem 13 e o fluxo de atividades executados durante a criação podem ser visto na imagem 14.

Imagen 13: Diagrama da Certidão de Nascimento



FONTE: Autoral

Imagen 14: Diagrama de atividade da Certidão de Nascimento



FONTE: Autoral

5.3.4. Contrato de Registro Geral:

Tendo em vista que o RG não é criado em conjunto com o nascimento de um cidadão, este documento não apresenta o mesmo problema original da certidão de nascimento para com o sistema. Para que um usuário consiga criar este documento, é preciso que, primeiramente, ele tenha se cadastrado

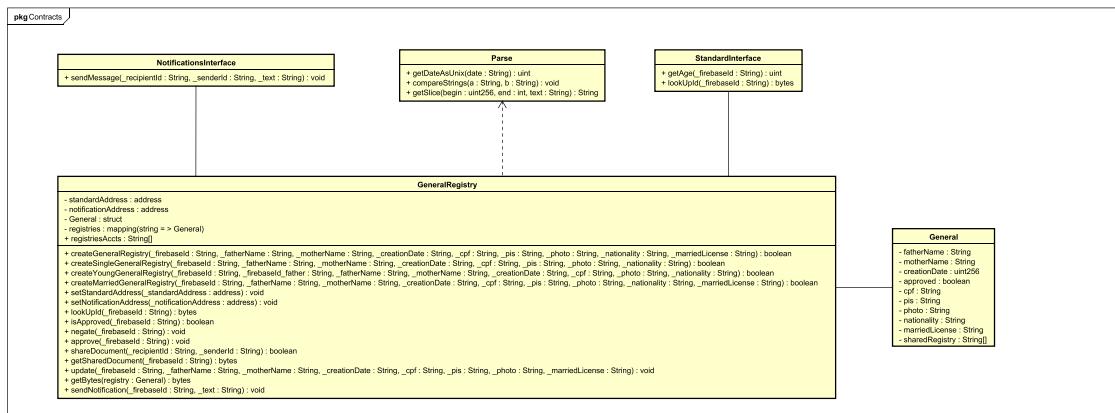
na aplicação e, consequentemente, tenha um registro padrão válido, conseguindo realizar, assim, a solicitação do registro geral. Da mesma forma que o documento anterior, o identificador único para este documento também é o Firebase ID, gerado pela aplicação durante o cadastro do usuário.

Os dados registrados dentro deste contrato são:

- Nome do pai
- Nome da mãe
- Data da criação
- CPF (opcional)
- PIS (opcional)
- Foto
- Nacionalidade
- Licença de casamento (opcional)
- Status atual do registro (ativo ou inativo)

Os seguintes dados foram escolhidos para serem armazenados devido sua necessidade de constarem no documento físico original e foram armazenados em um *Struct*, como nos contratos anteriores. No entanto, informações já existentes em outros contatos foram excluídas desse registro, já que o sistema pode ver os outros contratos para usarem essa informação. É possível ver a implementação destes dados no diagrama de classe na Imagem 15:

Imagen 15: Diagrama de classe Registro Geral

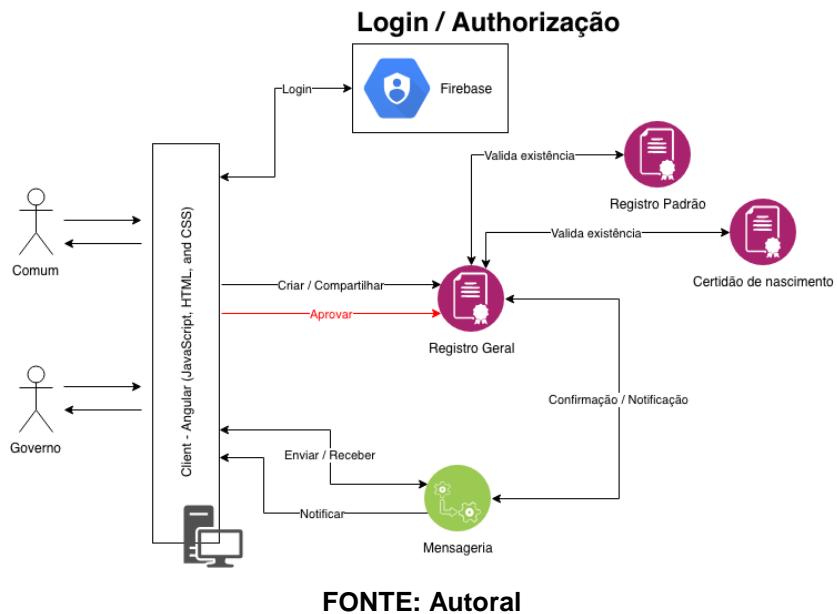


FONTE: Autoral

O fluxo interno para a criação é similar ao fluxo de validação da certidão de nascimento. Ao tentar criar um novo registro dentro do contrato do documento de registro geral, uma chamada de validação para o registro padrão será realizada. Caso esta chamada devolva um valor vazio, o RG não será criado. Além disto, uma outra chamada será realizada para o contrato da certidão de nascimento para garantir que o usuário já possui este documento e que ele está de acordo com o padrão definido pelo governo. No início do desenvolvimento, foi debatido entre nós se o registro da certidão de nascimento deveria ser obrigatório, afinal, alguns usuários poderiam ser adultos e apenas estariam registrando seu documento já existente, mas como a certidão de nascimento é um documento obrigatório para todos os brasileiros, achou-se necessário realizar tal validação. Ainda foi debatido entre os integrantes se as duas chamadas deveriam ser realizadas, afinal, para a criação de uma certidão de nascimento em seu nome, é preciso já ter sido criado um registro dentro da aplicação, mas devido ao fato de chamadas de consulta interna na *blockchain* não serem taxadas, não havia motivo para não criar esta validação extra, mantendo a consistência do sistema independente da de outros contratos.

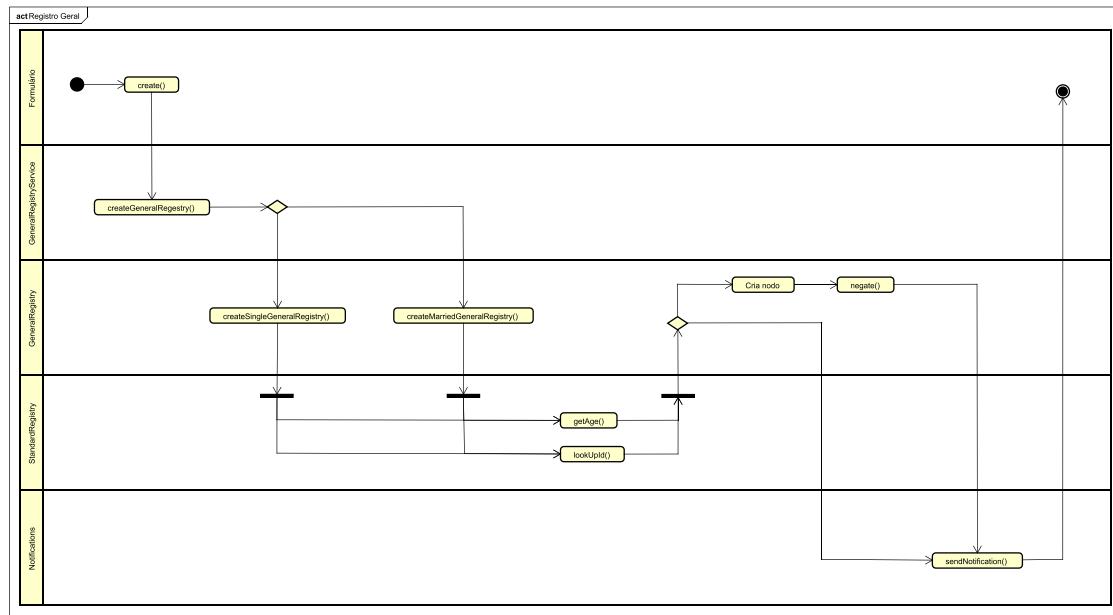
Assim como na certidão de nascimento a lógica de aprovação não foi implementada, mas uma chamada foi disponibilizada em nível de *front-end* para o teste desta funcionalidade. Desta forma a arquitetura da do contrato de Registro Geral ficou de acordo com a Imagem 16 e o fluxo de criação de acordo com a Imagem 17.

Imagen 16: Diagrama do Registro Geral



FONTE: Autoral

Imagen 17: Diagrama de atividade do Registro Geral



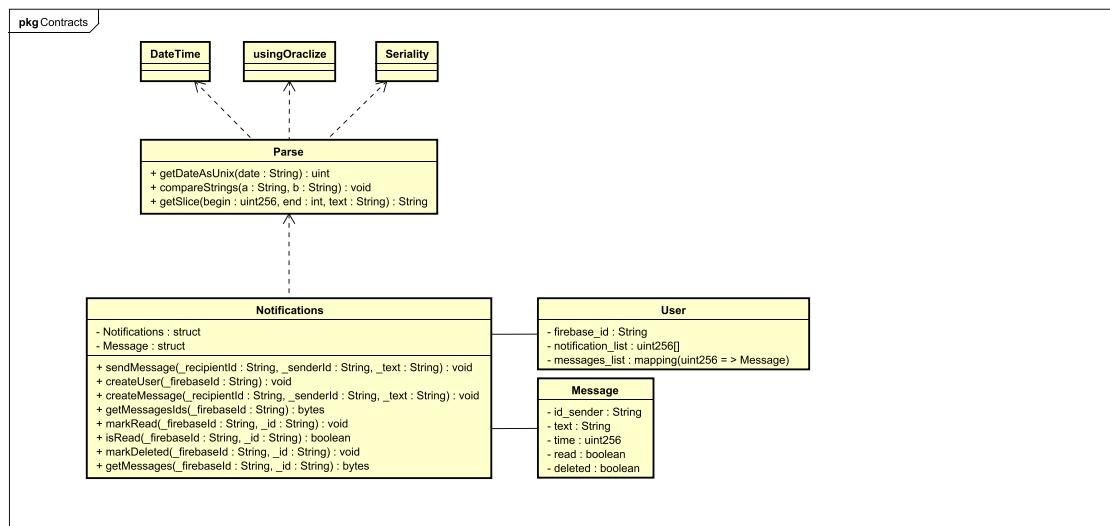
FONTE: Autoral

5.3.5. Notification

Este contrato foi criado para armazenar as notificações e prover uma interface de envio de notificações para os demais contratos. Nele foram criadas duas *Structs* para padronizar a criação de usuários e de mensagens. A

estrutura de usuário possui os parâmetros de Firebase ID, uma lista dos ids das notificações e um mapa de todas as mensagens recebidas por aquele usuário. Foi preciso quebrar o *Map* em uma lista de índices e de estruturas devido ao fato de o framework Ethereum não suportar iterações sobre mapas. Desta forma uma lista de índices pode ser usada como guia para gerar uma iteração. Na segunda estrutura os parâmetros são os seguintes: identificador de quem enviou a notificação, podendo ser o sistema ou outro usuário; texto da notificação; a hora do envio convertido para *Epoch Time* (medida de tempo dos segundos passados desde 1º de Janeiro de 1970); *boolean* para sinalizar se a mensagem já foi lida; e *boolean* para sinalizar se a mensagem já foi apagada. As mensagens mesmo quando apagadas são mantidas para geração de um log extraoficial, permitindo que durante a etapa de desenvolvimento fosse possível validar mensagens passadas mesmo após sua exclusão. Desta forma o contrato de notificações foi implementado de acordo com o diagrama de classe presente na imagem 18:

Imagen 18: Diagrama de classe do contrato de Notificação



FONTE: Autoral

Durante o tempo de desenvolvimento foi preciso alterar o texto armazenado pelo contrato devido ao custo de serialização de mensagens muito longas. Desta forma foram criados códigos que a camada de *front-end* poderia implementar. Esta forma de desenvolvimento pode ser visto comumente em aplicações que possuem mais de um idioma suportado e

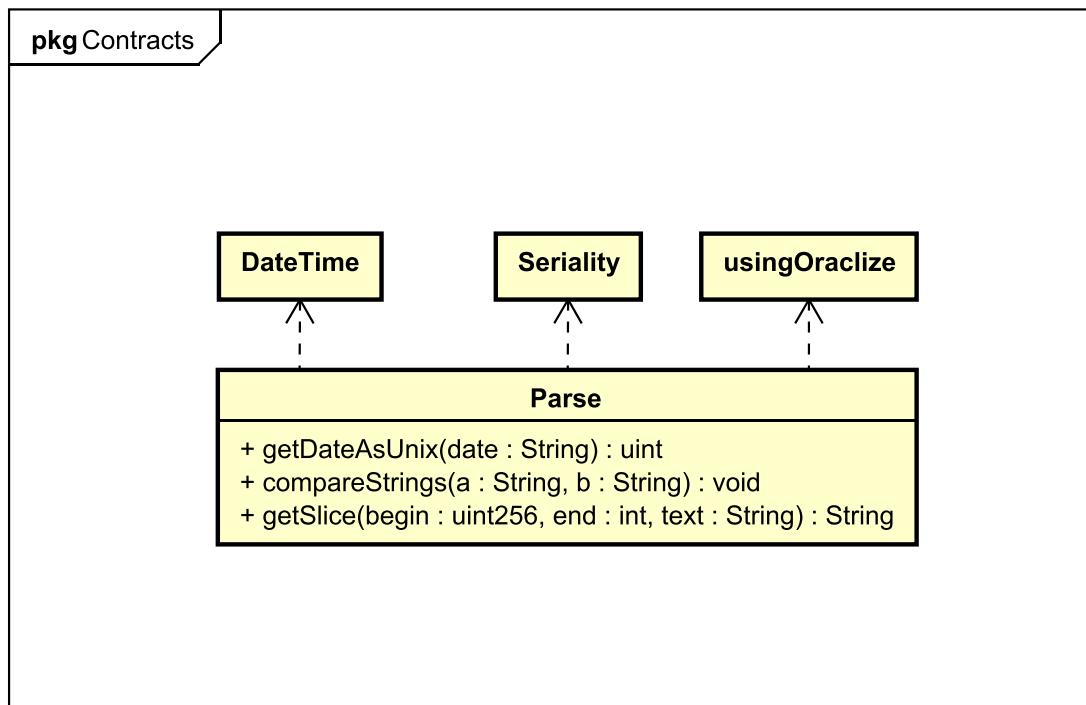
normalmente diversas traduções são armazenadas em *bundles* diferentes. Um exemplo de uma TAG seria: STANDARD_CREATE_SUCCESS teria o significado em português de "Seu Registro Padrão foi criado com sucesso!".

5.3.6. Parse

Devido a linguagem Solidity e o framework Ethereum não possuírem algumas funções básicas para outras linguagens, foi preciso criar um contrato com métodos para converter dias em *Epoch Time*, comparar textos e também para partir partes de textos quando necessário. Este contrato não possui atributos e seu único objetivo é prover para os outros contratos estas operações comuns sem que seja necessário repetir as mesmas funções. A implementação deste contrato pode ser vista na imagem 19. Este tipo de abordagem permitiu que quando um dos métodos estivesse com defeito, apenas um local precisasse ser corrigido.

Este contrato também possui três dependências de outros contratos: Seriality, responsável por fazer a conversão de tipos para *bytes*; usingOraclize, responsável por realizar a concatenação e outras interpolações de textos; e DateTime, que permitiu a realização da conversão para *Epoch Time*.

Imagen 19: Diagrama de classe do contrato Parse

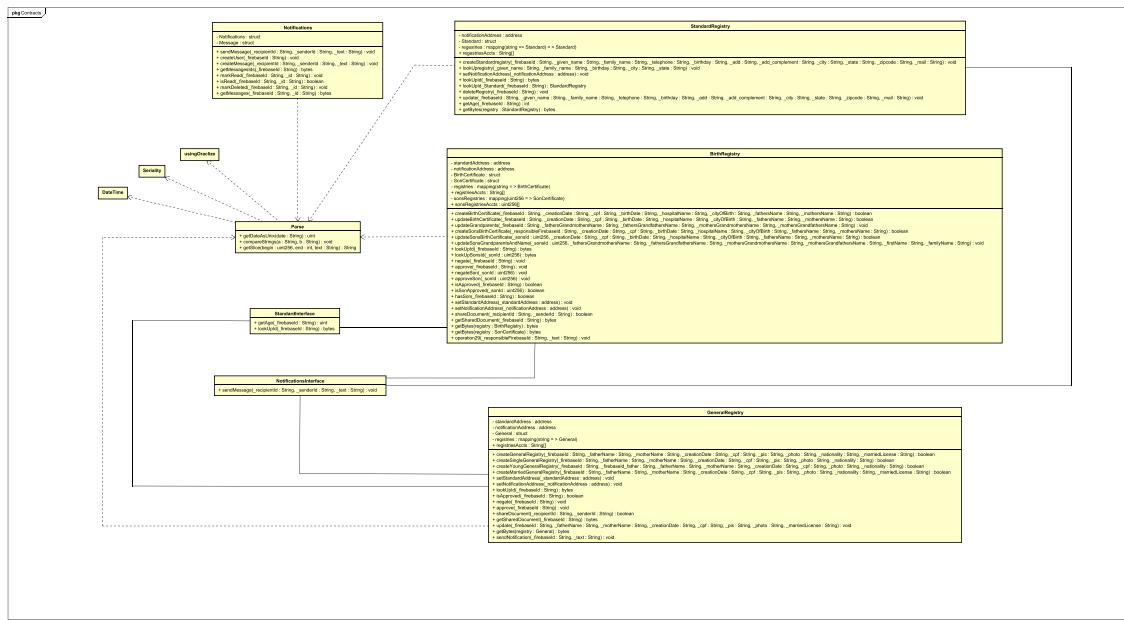


FONTE: Autoral

5.3.7. Estrutura final

Desta forma a estrutura final dos contratos na *blockchain* ficou de acordo com a imagem 20:

Imagen 20: Diagrama de classe de todos os contratos presentes



FONTE: Autoral

5.4. Testes

Na parte de testes desta aplicação foi estruturado uma metodologia comum para sistemas com modelos multicamadas, adotando-se testes para o *front-end*, *back-end* e integrados.

5.4.1. Camada front-end

Nesta camada os testes foram separados em três metodologias diferentes, hoje todas consideradas padrões de mercado. Além destas poderiam ter sido criados também testes automatizados usando frameworks como Selenium [16], mas já que tal aplicação não será mantida em longo prazo, foi optado por uma sistemática apenas funcional.

5.4.1.1. Testes unitários (Karma e Jasmine)

Os testes unitários foram criados com base nos padrões atualmente estabelecidos pela Google para plataformas desenvolvidas no framework Angular [5]. Cada componente e serviço possuem sua própria classe de testes

unitários, mantendo, assim, o contexto de cada execução único e agnóstico a chamadas externas. As regras para decisão se um teste unitário deveria ser criado ou não foram as seguintes:

- O componente possui lógica de template condicional (*NgIf*) ou de repetição (*NgFor*)?
- As funções do componente possuem chamadas de serviços externos ou internos dentro de suas funções?
- As funções aplicam alguma validação dos dados inseridos na camada de *front-end*?
- As funções possuem alguma lógica, seja ela de negócio ou de programação, complexa que precisa ser testada?

Caso tais condições estejam presentes no componente analisado, um ou mais cenários devem ser criados até que todas as regras estejam satisfeitas. Estes testes são responsáveis por validar as regras de negócio estabelecidas e descritas aqui, assim como garantir que ao alterar algum componente ou serviço comum a outras partes do sistema nenhum defeito seja criado. Estas validações foram configuradas para rodarem junto com o *build* do sistema, não permitindo uma nova liberação, caso falhem.

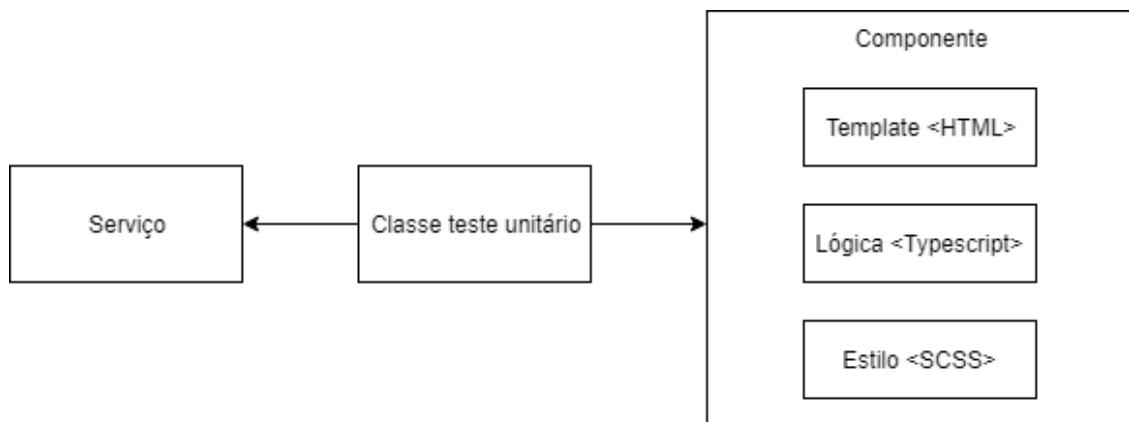
A lista de todos os testes unitários se encontra no Apêndice C.

Imagen 21: Resultado dos testes unitários

```
C:\Users\Nathan\Desktop\TCC\front-end (master -> origin)
λ npm test
  - test/specificationComponent
    - should create
      > personal-blockchain@0.0.0 test C:\Users\Nathan\Desktop\TCC\front-end
      > ng test
        - should route to activate
        - should route to view
        Chrome 70.0.3538 (Windows 10 0.0.0): Executed 70 of 70 SUCCESS (3.692 secs / 3.639 secs)
        TOTAL: 70 SUCCESS
        TOTAL: 70 SUCCESS
```

FONTE: Autoral

Imagen 22: Diagrama de testes unitários para aplicações Angular



FONTE: Autoral

5.4.1.2. Lint

Lint ou Linter é uma ferramenta de avaliação de código que busca erros, desvios de boas práticas e possíveis *bugs* [32]. Ao criar um projeto com a ferramenta Angular-CLI, normalmente é criado também as configurações para a análise do software. Além das configurações iniciais, foram utilizados os padrões compartilhados pela empresa Airbnb, que é referência de mercado.

Apesar de *linting* não ser categorizado como um teste formal, foi decidido que seria adotado, já que também é tido como um padrão recomendado pelo Google para todas as plataformas desenvolvidas em Angular [5] e por contribuir na qualidade do código em geral.

Todas as regras usadas pela ferramenta para avaliação do código se encontram no Apêndice D.

Imagen 23: Resultados lint do projeto

```
C:\Users\Nathan\Desktop\TCC\front-end>npm run lint
> personal-blockchain@0.0.0 lint C:\Users\Nathan\Desktop\TCC\front-end
> ng lint

All files pass linting.

All files pass linting.
```

Fonte: Autoral

5.4.1.3. Testes funcionais

Para os testes funcionais foi adotada uma estratégia, na qual cada tela deveria responder as regras de negócio citadas nas Seções 2.2, 2.3 e 5.3. Os integrantes do grupo acessaram a aplicação e a usaram normalmente, como um usuário padrão usaria, testando todas as ações possíveis.

A lista completa de cenários de testes funcionais está presente no Apêndice E.

5.4.2. Camada *back-end*

Nesta seção é descrito os testes realizados nessa camada durante o desenvolvimento deste trabalho.

5.4.2.1. Testes funcionais

Testes unitários e automatizados ainda são tema de debate entre a comunidade de desenvolvimento em tecnologias *blockchain*. Para a estratégia de testes deste trabalho foi decidido que seria usado apenas uma validação funcional, seguindo os mesmos padrões adotados no *front-end* citados na Seção 5.4.1.3.

Para estes testes foi utilizado a plataforma de desenvolvimento Remix e suas funções de *debug* para a validação dos resultados obtidos entre cada

A seguir exemplos deste processo:

Imagen 25: Tela defeitos abertos no repositório de Smart Contracts

Issues (1–4 of 4)

Title

-
- #5: Ao alterar o nome no front durante o processo de criação do documento RG o contrato de registro padrão também deve ser alterado.
 - #3: Arrumar referência de filho
 - #2: Criar função de compartilhamento de filho
 - #1: Criar micro serviço que autoriza para menores a criação do RG
-

Fonte: Autoral

Imagen 26: Tela defeitos e tarefas do repositório front-end

Issues (1–15 of 15)

Title

-
- #15: Ao clicar em Registro Geral e Certidão de nascimento na Navbar elas levam ao componente pai e não redirecionam
 - #14: Telas depois de deslogar não estão levando usuário para tela de login
 - #13: [Navbar] Logout icon não tem pointer de cursor
 - #12: [HOME] Title de RG e CN ficam mesmo sem documento
 - #3: [Cadastro] Criar contrato para cadastro do usuário no blockchain
 - #2: [Cadastro] Não está validando informações em campo
 - #8: [Cadastro - Update] Implementar atualização do cadastro no blockchain.
 - #6: [Cadastro] Não está validando email
 - #11: Funcionalidade de filtro não foi implementada
 - #10: Menu lateral está repaginando
 - #9: [Cadastro - Delete] Criar forma de deletar cadastro do blockchain
 - #7: [Firebase] Criar local de signout dentro da aplicação
 - #5: [Notificações] Criar um serviço para enviar e receber notificações
 - #4: [Cadastro] Dropdowns não estão mostrando input selecionado, quando o usuário não selecionou nada ainda.
 - #1: [Cadastro] Não está enviando email de verificação ao cadastrar o usuário
-

Fonte: Autoral

6. DESCRIÇÃO TELAS DA SOLUÇÃO

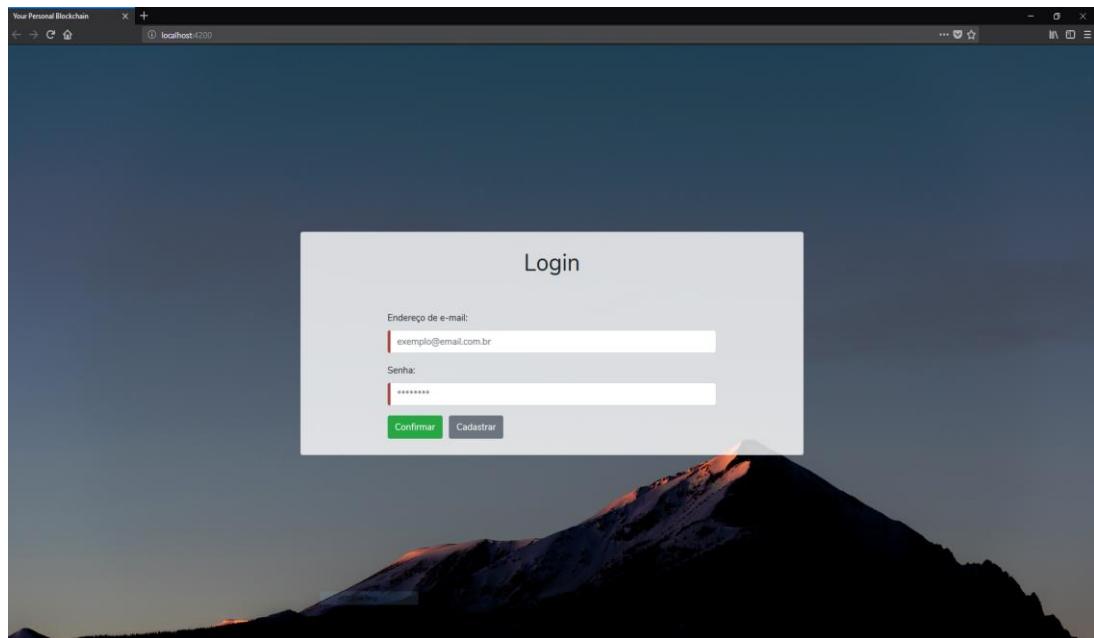
Neste capítulo é descrito o fluxo do usuário na aplicação desenvolvida, assim como a funcionalidades de *front-end* e *back-end* em cada menu do sistema desenvolvi

6.1. Tela Inicial

A tela inicial desta aplicação permite que sejam realizadas duas ações: Logar-se ou Cadastrar-se.

Ao realizar o Login, a aplicação realiza uma chamada na API de autenticação da Firebase para que seja validado as informações de login (e-mail e senha) do usuário e retorne um objeto de usuário, do qual mapeamos para uma interface no *front-end* chamada de *user*.

Imagen 27: Tela de login



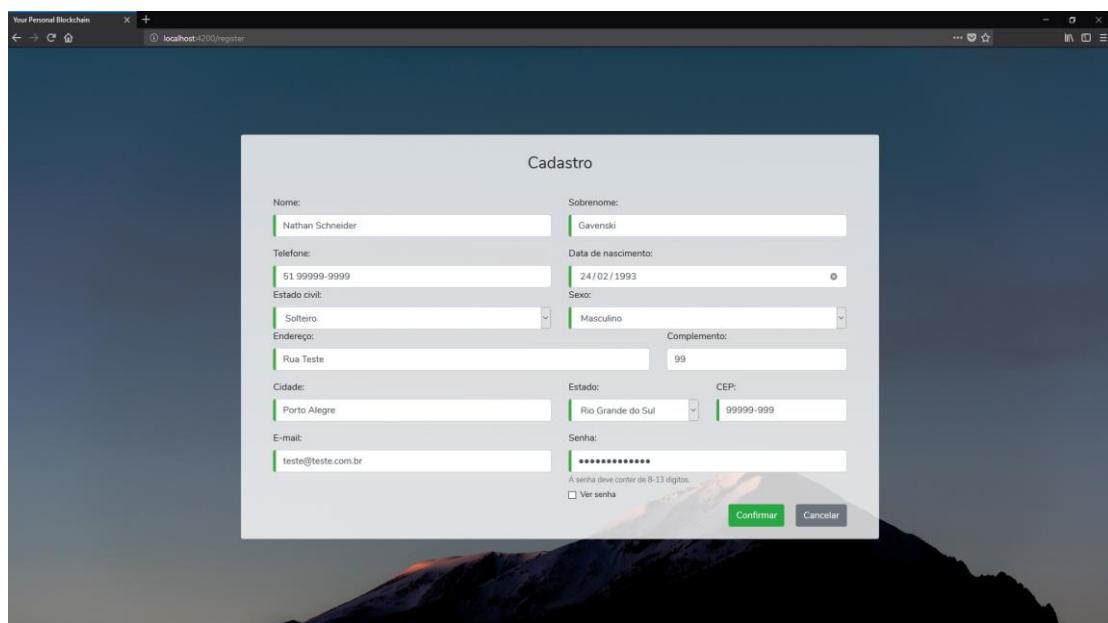
Fonte: Autoral

6.2. Registro

Na tela de registro, o usuário deve preencher as informações do formulário que são encaminhadas para a Firebase e para o *blockchain*. Nesta

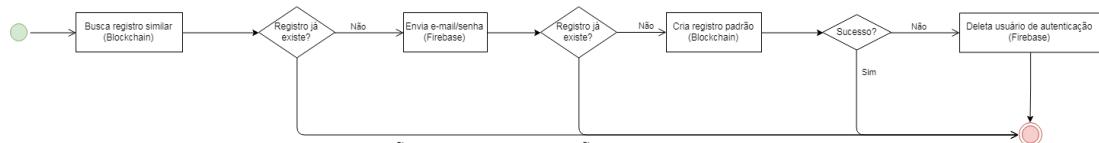
parte, as informações são encaminhadas primeiramente para o *blockchain* a fim de validar se já existe um documento padrão criado com aquelas informações. Caso não seja encontrado nenhum registro, o e-mail e senha são enviados para a Firebase que irá tentar criar um usuário. Neste momento é criado o *token* do usuário descrito no Capítulo 5. Com o retorno positivo do console e com o *token* todas as informações, exceto a senha por questões de segurança, são encaminhadas para a camada do *blockchain* e o usuário é redirecionado para a *home page*.

Imagen 28: Tela do formulário



Fonte: Autoral

Imagen 29: Fluxo de criação de registro padrão



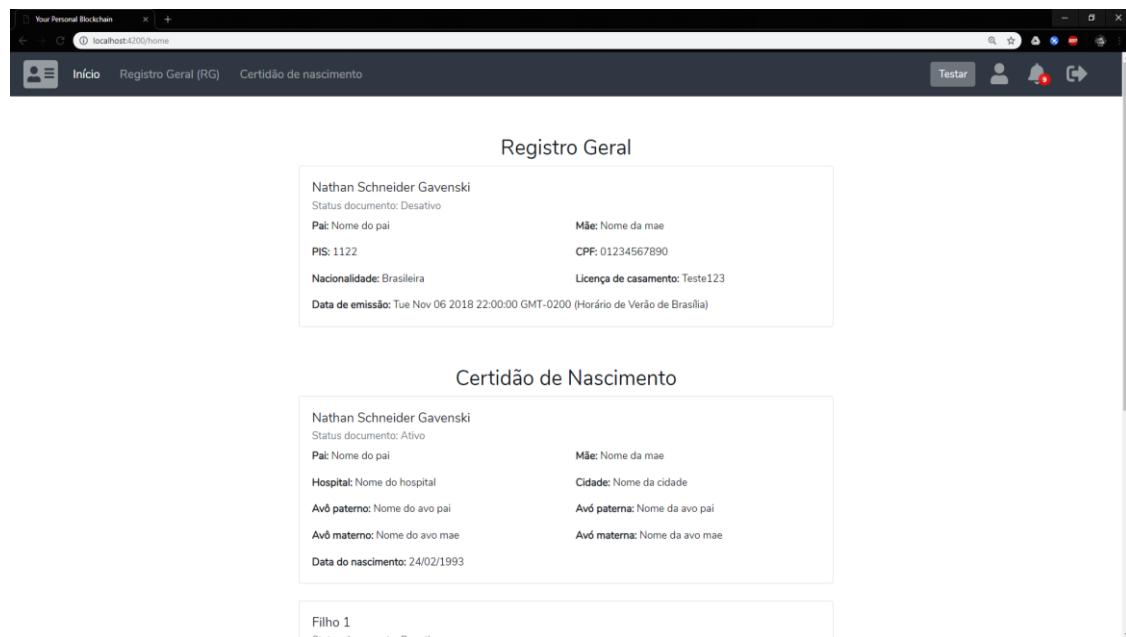
Fonte: Autoral

6.3. Home

Nesta tela apresentam todos os documentos que o usuário possui. Sempre que a aplicação for acessada e o usuário estiver “logado” na aplicação da Firebase, ele também será redirecionado para esta página.

Se o usuário também for o responsável, estes documentos também serão exibidos aqui.

Imagen 30: Tela da Home Page

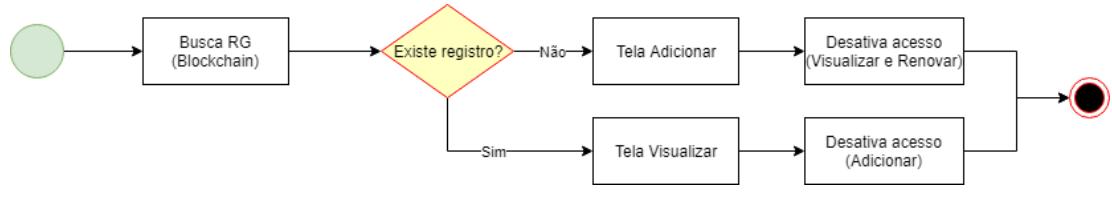


Fonte: Autoral

6.4. Registro Geral

O menu de Registro Geral foi criado de forma que todas as ações pertinentes a este documento sejam realizadas do mesmo local. Ao tentar acessar o menu “pai” a aplicação irá realizar uma chamada para o sistema, buscando documentos que tenham o mesmo Firebase ID do usuário logado. Se nenhum registro for encontrado, o usuário é automaticamente direcionado para o submenu “Adicionar”, caso algum registro seja encontrado ele será encaminhado para o submenu “Visualizar”. No primeiro caso, os menus “Visualizar” e “Renovar” se encontram desativados e para o segundo caso o contrário.

Imagen 31: Fluxo telas RG



Fonte: Autoral

6.4.1. Adicionar

Neste submenu, o usuário é apresentado com um formulário, requisitando os mesmos dados hoje já usados para criar o documento. O nome e sobrenome já vêm preenchidos, mas podem ser alterados. Este comportamento foi implementado tendo em vista que o cidadão brasileiro tem a possibilidade de alterar seu nome, portanto, ao serem alterados, uma atualização será realizada no registro padrão.

Todas as outras informações deste formulário são preenchidas e enviadas para o contrato de RG. A imagem é convertida para base64 antes de ser enviada.

A criação de um registro geral depende diretamente do registro padrão, como citado anteriormente neste documento. Devido ao contrato precisar do endereço do contrato padrão, o botão teste é habilitado em ambiente de teste para que a referência seja realizada ou atualizada, caso necessário.

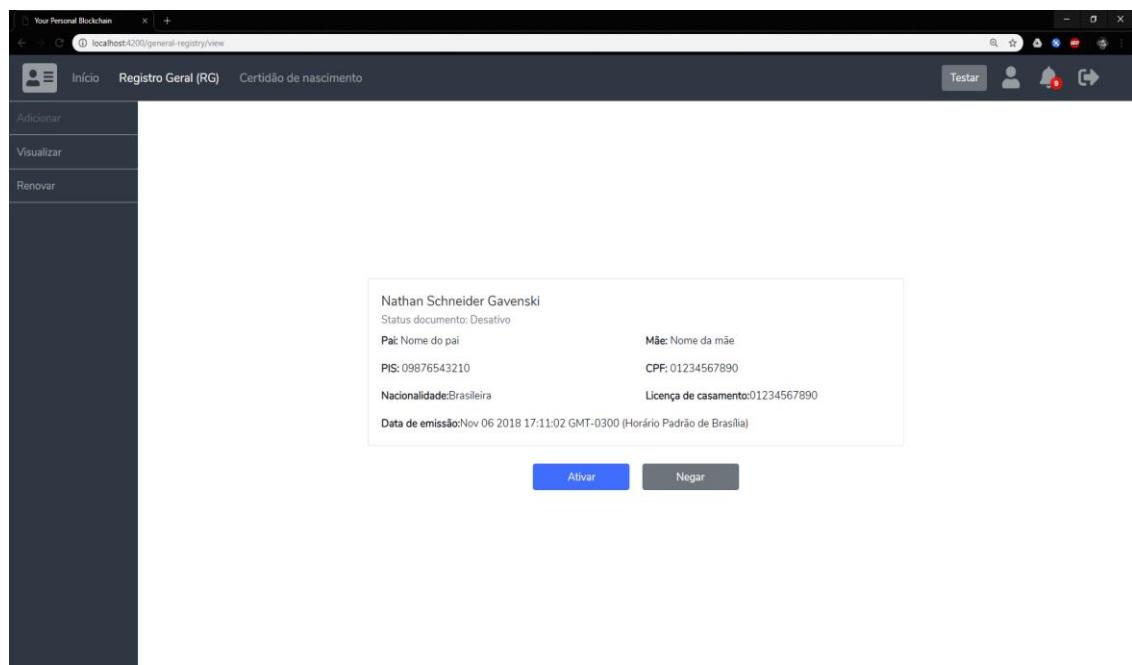
Imagen 32: Tela de Adicionar RG

Fonte: Autoral

6.4.2. Visualizar

O submenu “Visualizar” é responsável por mostrar ao usuário os dados do seu registro geral e o status atual de seu documento. Os dados são requisitados ao *blockchain* ao entrar no menu “Registro Geral (RG)” e são mantidos em seção, sendo perdidos apenas em *refresh*, o que diminui as chamadas de visualização, que não possuem custo para o usuário, para aplicação.

Os botões implementados nesta tela são apenas para realização dos testes de ativação e inativação dos documentos. Ao serem acionados, realizam uma chamada para a aplicação trocando o status do registro.

Imagen 33: Tela de Visualizar**Fonte: Autoral**

6.4.3. Renovar

Em “Renovar”, o usuário pode atualizar seus dados cadastrais, assim como renovar a validade de seu RG. Já que o documento possui validade de dez anos, é necessário que o usuário renove seu documento e que a data de vencimento seja atualizada.

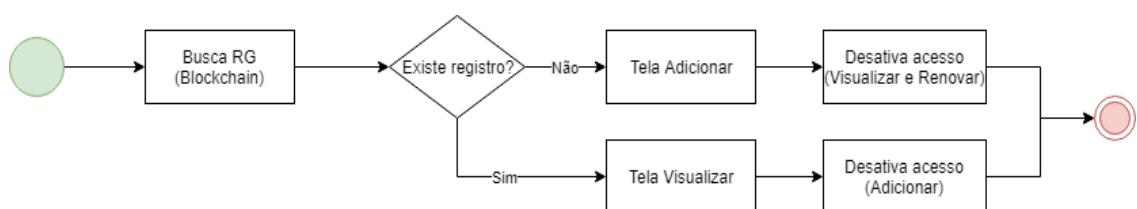
Imagen 34: Tela de Renovar

Fonte: Autoral

6.5. Certidão de nascimento

O design desta funcionalidade seguiu os mesmos princípios do Registro Geral. Ao acessar a funcionalidade de Certidão de Nascimento, o sistema irá realizar uma chamada a aplicação para que seja validado se já existe um documento cadastrado para o Firebase ID do usuário “logado”. A diferença que existe para o menu do RG é que, caso já exista um registro, o menu adicionar passa a tornar-se ativo para a adição de novos registros de possíveis dependentes. Esta diferenciação foi desenvolvida devido às necessidades de negócio vistas no Capítulo 5.

Imagen 35: Fluxo telas Certidão de Nascimento



Fonte: Autoral

6.5.1. Adicionar

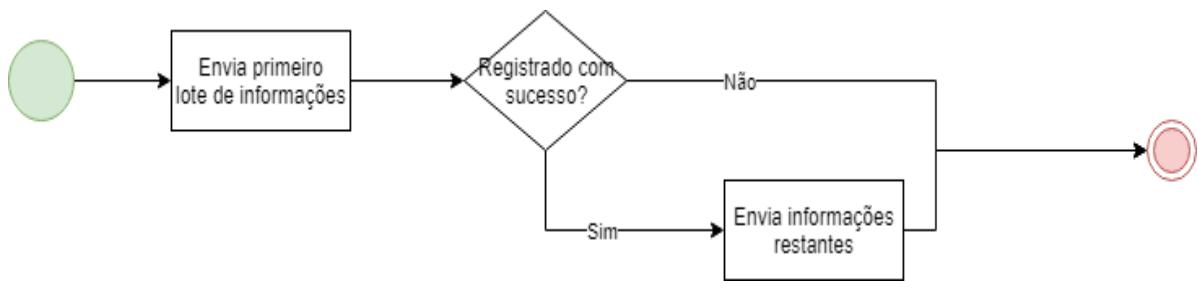
A tela adicionar apresenta ao usuário um formulário com todos os dados necessários para a criação de uma certidão de nascimento atual. Este submenu apresenta a mesma funcionalidade de enviar o endereço do contrato do registro padrão para o contrato de certidão de nascimento.

Devido ao grande número de parâmetros que devem ser enviados para a criação deste documento, foi necessário que fossem realizadas duas chamadas síncronas para o contrato dentro do *blockchain*. Primeiramente, são enviados os valores dos campos: CPF, data de nascimento, nome do hospital, cidade do nascimento, nome do pai e nome da mãe. Após estes dados serem enviados e criado o registro corretamente dentro do contrato, os demais dados, nome do avô paterno, nome da avó paterna, nome do avô materno, nome da avó materna e nome do filho (no caso da adição de um dependente) são enviados para complementar o registro já existente.

Imagen 36: Tela adicionar Certidão de Nascimento

Fonte: Autoral

Imagen 37: Fluxo de criação de Certidão de Nascimento

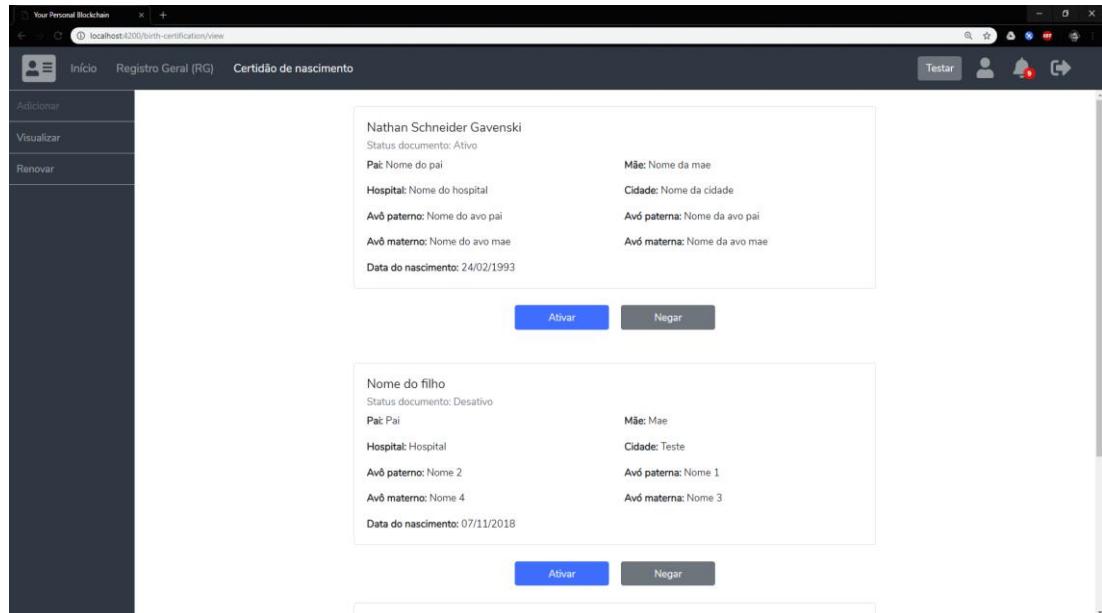


Fonte: Autoral

6.5.2. Visualizar

A tela “Visualizar” irá apresentar todos os documentos de certidão de nascimento ligados ao Firebase ID já “logado” na plataforma. Cada documento terá seus próprios botões para ativar e negar o documento e seguem a seguinte ordem: Documento do dono da conta, demais documentos ordenados por ordem de adição.

Imagen 38: Tela de visualização de Certidão de Nascimento



Fonte: Autoral

6.5.3. Renovar

A tela “Renovar” existe apenas para os casos, dos quais um erro foi cometido ao cadastrar um documento. Uma certidão de nascimento tem validade eterna e, por isso, é apenas usada para correções.

Ao entrar neste submenu, o usuário deve escolher qual registro deseja alterar e depois preencher arrumar o formulário de acordo.

Imagen 39: Tela de renovação de Certidão de Nascimento

The screenshot shows a web application interface for renewing a birth certificate. The URL in the address bar is `localhost:4200/birth-certification/renew`. The main content area has a heading "Atenção" with the instruction: "Ao atualizar o seu documento ele ficará pendente até que as informações sejam validadas." Below this are several input fields arranged in pairs:

- Nome do hospital / Cidade do nascimento
- Nome do pai / Nome da mãe
- Nome do avô paterno / Nome da avó paterna
- Nome do avô mae / Nome da avo mae
- Nome do avo mae / Nome da avo mae
- Data do nascimento / CPF

At the bottom of the form are two buttons: "Criar" (Create) and "Teste" (Test).

Fonte: Autoral

6.6. Perfil

Na funcionalidade de “Perfil”, o usuário consegue ver todas as suas informações referentes ao seu registro padrão no blockchain. Ao clicar em habilitar campos, ele pode alterar suas informações e reenviá-las para atualização.

Como a senha deve permanecer secreta, ao habilitar os campos nada deve aparecer. Se alguma informação for preenchida no campo, ela será enviada para o console da Firebase para alteração da senha do usuário. O comportamento de um *checkbox* para visualização da nova senha, assim como no primeiro formulário de registro da plataforma, foi mantido.

Imagen 40: Tela de informações de perfil

The screenshot shows a web application interface for profile management. At the top, there's a header bar with tabs for 'Início', 'Registro Geral (RG)', and 'Certidão de nascimento'. On the right side of the header are icons for 'Testar', a user profile, notifications, and a refresh arrow. Below the header, the main content area contains several input fields for personal information:

- Name: Nathan Schneider
- Sobrenome: Gavenski
- Telefone: 51-93134034
- Data de nascimento: 730512000
- Endereço: Tv. Universina Araújo Nunes 72
- Complemento: 307
- Cidade: Porto Alegre
- Estado: Rio Grande do Sul
- CEP: 90450150
- E-mail: nathan@gavenski@gmail.com
- Senha: (empty field)

At the bottom right of the form area is a green button labeled 'habilitar campos' (Enable fields).

Fonte: Autoral

6.7. Notificações

Na tela de “Notificações” são apresentadas todas as notificações do sistema para o usuário, em ordem da mais nova para a mais velha, possibilitando, também, um filtro para visualização apenas das mensagens não lidas.

Imagen 41: Tela de Notificações

The screenshot shows a web application interface for notification management. At the top, there's a header bar with tabs for 'Início', 'Registro Geral (RG)', and 'Certidão de nascimento'. On the right side of the header are icons for 'Testar', a user profile, notifications, and a refresh arrow. Below the header, the main content area has a title 'Lista de notificações' (List of notifications). The list displays three notifications:

De:	Mensagem:	Ações
De: teste_from	teste_msg	[Marcar como lida] [Excluir]
De: teste2_from	Long message content (Lorem ipsum placeholder)	[Marcar como não lida] [Excluir]
De: teste_from	teste_msg	[Progress bar: blue]

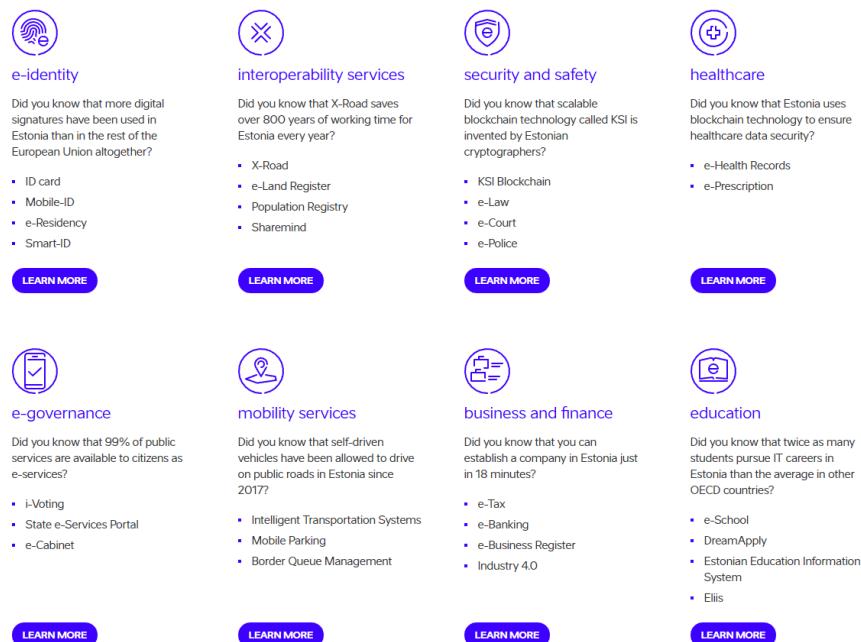
Fonte: Autoral

7. TRABALHOS SEMELHANTES

Ao realizar a pesquisa para este trabalho e a possibilidade de implementar um *blockchain* capaz de realizar todas as atividades que já existem hoje nos sistemas do governo, deparou-se com diferentes soluções que provam que a ideia desta monografia não se encontra longe da realidade do mundo atual.

7.1. E-stonia - ESTÓNIA

Imagen 42: Soluções criadas pela E-stonia.



FONTE: E-stonia (2018)

A primeira solução encontrada ao procurar por países, onde a tecnologia de *blockchain* ganhou espaço dentro dos processos, foi a Estônia. Este movimento apoiado pelo governo desde sua independência como estado soberano, em 1991, vem ganhando cada vez mais espaço dentro do próprio país [53]. De computadores dentro de salas de aula, até as próprias implementações, como podemos ver na imagem 41, a Estônia vem abrindo cada vez mais espaço para um governo, no qual toda e qualquer interação tem como base uma solução tecnológica.

De votos, educação e planos de saúde até serviços pessoais e financeiros, o movimento E-stonia tem seus dados armazenados em uma plataforma chamada de X-road, que possui sua implementação baseada em um *blockchain* chamado Keyless Signature Infrastructure (KSI). Nos dias atuais, este sistema se encontra fora do país, em Luxemburgo, e garante que, caso uma possível invasão da Rússia tenha efeito, o estado poderá continuar funcionando sem suas fronteiras [47] [49].

Além disso, o movimento em 2004 conseguiu implementar o primeiro sistema de cidadania eletrônica, permitindo que os não residentes tenham um cartão de identidade estoniano. Este programa se chama e-residency e ganha força, já que permite, uma vez concedido o título, que qualquer cidadão tenha acesso aos mesmos serviços, assim como qualquer estoniano. A única restrição, hoje, dá-se ao documento não permitir residência física e nem acesso ao país [9].

7.2. BaaS - REINO UNIDO

Imagen 43: Documento oficial sobre blockchain do Reino Unido



FONTE: UNITED KINGDOM. Office for Science (2016, p.1)

Outro país que vem procurando introduzir a tecnologia da *blockchain* em seu sistema é o Reino Unido. Em um documento lançado pelo país, o

mesmo admite já estar trabalhando com uma plataforma, que foi nomeada de *Blockchain-as-a-Service* (BaaS). O Reino Unido pretende implementar tal tecnologia em diferentes processos do seu estado, procurando combater fraudes, corrupção, entre outros crimes legais.

No mesmo, os autores descrevem a tecnologia e reconhecem seu poder disruptivo no mundo atual, assim como o potencial que ela traz para dentro do governo e a forma como os atuais processos podem ser alterados e melhorados [51]. Atualmente, o governo está investindo em startups que usam a tecnologia da *blockchain* como base para seu produto para tentar incentivar o mercado, como já vimos os Estados Unidos fazendo no passado, com as startups de energia renovável [31].

7.3. Distributed Access Control Evaluation on IoT Ledger-based Architecture

Este trabalho, organizado pelos doutorandos Roben Castagna Lunardi, Regio Antonio Michelin, Charles Varlei Neu e pelo professor Avelino Francisco Zorzo, apesar de não tratar do mesmo tema que este trabalho, aborda uma solução necessária para o desenvolvimento do trabalho. Ao estudar uma forma de realizar o acesso distribuído e controlado para os dispositivos diversos presentes na internet das coisas (IoT), o trabalho encontrou uma forma de produzir um *ledger* com informações mutáveis. Desta forma, todos os dispositivos conectados à rede poderiam receber, sempre, uma nova informação para sua chave de criptografia.

Este trabalho mostra que a possibilidade de alteração de informações dentro de um *ledger* é possível e sendo debatido no Capítulo 8, onde são descritos os problemas encontrados durante o planejamento deste projeto.

O trabalho também mostra a performance necessária para a utilização de *blockchains* em dispositivos menores, algo que não é visto neste trabalho, mas precisa ser considerado durante o desenvolvimento do sistema aqui proposto, em caso de implementação em larga escala [33].

8. PROBLEMAS ENCONTRADOS

Este capítulo está reservado para a descrição dos problemas encontrados durante a elaboração da proposta e sistema, a definição de o que não será abordado neste trabalho e os meios utilizados para contorná-los.

8.1. Imutabilidade

Apesar de tal característica ser um dos pontos mais desejados desta tecnologia para implementar este sistema, os dados presentes dentro do mesmo não escapam da mutabilidade do mundo real. Além das informações que hoje se podem pensar como editáveis (gênero, nome, endereço, idade), outros dados podem vir a tornar-se mutáveis. Com este pensamento, encontra-se a primeira dificuldade na implementação deste sistema de gestão de documentos governamentais e de seus documentos, como os aqui previamente citados. Embora as informações aqui listadas não sejam tão comumente alteradas, um cidadão tem o direito de alterá-las, através de processos hoje já existentes. Para a solução deste problema, foram pesquisadas as soluções implementadas atualmente com o mesmo objetivo (mencionadas no Capítulo 5), assim como trabalhos dentro do mundo acadêmico que tentam resolver o mesmo problema para outros sistemas reais que possuem esta característica de suas informações não poderem persistir para sempre as mesmas dentro da *blockchain*.

8.1.1. Mutabilidade de parte dos dados de um bloco

Para esta solução foi estudado um dos trabalhos acadêmicos citados no Capítulo 7, no qual a solução proposta é a de existir parte do bloco não utilizada no *hash* de identificação. Deixando, assim, uma parte onde as informações podem ser alteradas livremente, sem que o identificador do bloco seja alterado e quebre as referências lógicas da *blockchain*. O problema desta solução é a perda de rastreabilidade do histórico destas informações, já que ao alterar uma destas informações, o *hash* do bloco não está sendo alterado e

nada está sendo gravado no sistema, perdendo, assim, o objetivo de controle dos documentos e transações entre os usuários e suas documentações.

8.1.2. Capacidade de alterar os dados e recriar o hash

Esta possibilidade foi vista durante a etapa de revisão bibliográfica em um documento de análise da possibilidade de criar um *hash* camaleão [3]. Tal *hash* poderia ser mimetizado com outras informações, fazendo, assim, ser possível a alteração do *hash* principal e mantendo a informação dentro do *hash*, onde apenas o sistema seria capaz de recriar este identificador. Apesar desta possibilidade parecer tentadora, apresentava, ainda, o mesmo problema da primeira solução. Ao alterar as informações e recriar o *hash* do bloco, ainda assim, se estaria perdendo a historização destes blocos e o sistema não seria capaz de analisar qualquer histórico de alterações criadas por ele e por terceiros. O segundo problema seria a falha de segurança que a possibilidade de duplicar o primeiro *hash* criado implica. Se tal possibilidade acontece, é um sinal de que o algoritmo utilizado pelo sistema para a criação deste identificador é fraco e mais cedo ou mais tarde algum usuário poderia descobrir e quebrar a segurança e alterar seus dados.

8.1.3. Dados mutáveis serem alocados dentro de uma estrutura dinâmica

Tendo em vista que qualquer alteração no bloco não seria uma possibilidade, visto pelos exemplos citados nas Seções 8.1.1 e 8.1.2, e que foi decidido usar a plataforma *Ethereum* para o desenvolvimento deste trabalho como uma maneira de escapar de algumas características do Bitcoin, por exemplo o UTXO (função utilizada para quebrar transações de quantidade maior que o desejado em duas transações, uma na quantia desejada de envio e outra com a "sobra" da transação original, que é enviada para o usuário original) [1][52]. A última possibilidade encontrada foi a de usar a estrutura de *Smart Contracts* para armazenar tais informações de maneira dinâmica, já que parte do código desta estrutura pode ser alterada dentro do própria *blockchain*. Esta solução também irá resolver o problema de historização dos dados, já que cada transação para com o contrato é gravada, também, dentro da

estrutura em blocos. Desta forma, quaisquer dados alterados estariam dentro da *blockchain* e nada seria perdido durante uma alteração.

8.2. Permissão de acesso aos dados

Tendo em vista que tal sistema deve herdar todas as leis e comportamentos atuais da sociedade atual, existe, também, o problema de que para um cidadão menor de dezoito anos deve ter um responsável legalmente (podendo este ser um de seus pais, familiares ou tutores).

No problema anterior foi escolhido criar *Smart Contracts* para o armazenamento e a gestão dos documentos deste sistema, tal estrutura deve ser capaz de armazenar quem são os responsáveis legais pelos usuários que tiverem menos de 18 anos ou não passaram por um processo de emancipação. Além disso, o sistema deve conter a inteligência de que quando este se torna legalmente responsável por si, os que antes tinham acesso a sua conta não o devem ter mais. O problema para tal estrutura se torna muito importante, quando se conjectura os cenários reais que implicam tais medidas, como o fato de um menor não poder alterar seu nome sem aprovação ou no caso de emancipação devido à violência domiciliar.

Para este problema, a solução encontrada seria a de atualizar a chave privada da conta, quando um processo de atribuição de maioridade penal fosse iniciado. Esta solução teria como vantagem nenhum dado precisaria ser replicado dentro do sistema quando o processo fosse iniciado e manteria, também, o histórico. Este processo poderia ser inicializado pelo sistema ou pelo próprio usuário e deve atualizar todos os contratos, nos quais constam documentos daquela conta, assim como a conta. Este processo também poderia ser resolvido mantendo um local de armazenamento, onde a atualização ocorreria e, nos próximos acessos, os *Smart Contracts* já iriam validar a chave primária correta.

8.3. Versionamento

Devido ao tempo de implementação deste trabalho, este foi um dos problemas encontrados que não foi resolvido. O problema consiste, de modo

geral, nos apontamentos dos contratos criados dentro da *blockchain* para outros contratos na corrente. Tendo em vista que cada contrato tem como implementação um documento do governo ou a automatização de algum processo, como a aprovação de um passo, quando o documento for alterado uma nova versão deve ser criada dentro da *blockchain* e os novos apontamentos devem ser realizados. Este problema precisa ser tratado de forma que todos os apontamentos sejam alterados, tanto com clientes, quanto com outros contratos, assim como também deve ser feito de uma forma que os dados existentes dentro de um contrato não sejam perdidos.

8.4. Emissão de Certidão de Nascimento e Registro Geral

Ao realizar a pesquisa histórica da Certidão de Nascimento e do RG, foram encontrados documentos federais escritos no século XIX [12] que, além de abordarem temas considerados atualmente como crime contra a humanidade, possuem uma versão ortográfica da Língua Portuguesa que torna a leitura e o entendimento complexo.

Ainda assim, atualmente a emissão de uma certidão de nascimento possui algumas regras que já podem ser consideradas ultrapassadas, pois o reconhecimento da paternidade só ocorre se o registro for realizado pelo pai ou o mesmo assinar um documento que declara a paternidade para que a mãe inicialize a emissão da certidão de nascimento. Essa situação acaba por causar constrangimentos para famílias que não são tradicionais que buscam em seu momento máximo de felicidade reconhecer seu filho(a), pois necessitam recorrer à justiça através de processos burocráticos e lentos para que, assim, seja realizado o reconhecimento da filiação da criança [26] [19].

9. Benefícios e malefícios encontrados durante o desenvolvimento

Durante o desenvolvimento deste trabalho foram constatados alguns pontos positivos e negativos. É natural que alguns benefícios e malefícios aqui descritos sejam inerentes a tal plataforma/paradigma e não sejam verdade para os demais, assim como futuras versões desta distribuição não apresentem mais os mesmos pontos.

9.1. Benefícios

Nesta seção são descritos os benefícios encontrados ao longo do desenvolvimento do trabalho.

9.1.1. Modelo SALT:

Um anagrama para *Sequential, Agreed, Ledgered, Temper-resistant*, comumente chamado de SALT foi o nome batizado por Stefan Tai, Jacob Eberhardt e Markus Klems sobre o modelo formal para as *blockchain* hoje de mercado [48]. Assim como seu irmão para bancos de dados relacionais que possui as seguintes propriedades: Atomicidade, Consistência, Isolamento e Durabilidade (ACID); este modelo defende alguns conceitos muito importantes para qualquer aplicação distribuída e para se ter um conceito transacional. *Sequential* refere-se a uma ordem transacional que deve ocorrer em alguma sequência, mesmo esta sendo determinada pelo minerador (usuário que consegue descobrir o *hash* do próximo bloco) vencedor. *Agreed* aborda o fato de existir a necessidade de haver consenso no que ocorreu dentro da *blockchain* para produzir um estado comum, impossibilitando que algo inválido seja adicionado ao sistema e fortificando a aplicação. *Ledgered* alude ao caso de todas as evidências serem mantidas em um registro histórico que permite a validação do estado atual como correta.

E, por fim, *Temper-resistant* que trata do fato de haver a necessidade de existir uma determinada pluralidade para ocorrer uma alteração na *blockchain*. Tal modelo permite que aplicações transacionais sejam criadas sem problema dentro desta tecnologia. Além disso, é possível traçar um

paralelo direto ao modelo comum aos bancos de dados permitindo uma arquitetura, embora oriunda de outro paradigma, similar.

9.1.2. Log transacional:

Como citado anteriormente, todos os dados passados em uma *blockchain* devem ser armazenados em um registro histórico, os blocos da própria corrente. Isso permite que o estado atual seja validado a qualquer momento. Isso traz um benefício único para a aplicação que permite que transações passadas e bugs sejam analisados com dados imutáveis e com uma capacidade de armazenamento de log persistente. Embora o log formal da aplicação não esteja dentro do bloco, devido ao seu estado determinístico, é possível realizar uma regressão lógica e descobrir qual o motivo de tal transação ter sido adicionada no estado que ela fora, mesmo com ocorrência de bugs. Este ponto foi crucial ao longo do desenvolvimento deste trabalho, já que ao adicionar um documento e ele ter sido executado de maneira correta, mas os dados de maneira incorreta, foi possível fazer uma regressão lógica para encontrar o erro da implementação.

9.1.3. Capacidade de ter a lógica no contrato:

Com a invenção dos *Smart Contracts*, foi possível trazer para a tecnologia da *blockchain* a capacidade de executar código contendo a lógica de negócio dentro da própria máquina virtual. Isso transforma o registro em algo capaz de executar código e condicionar transações a uma determinada logística, mantendo a ordem dentro da *blockchain* e garantindo, assim, uma aplicação mais robusta. Quando se analisa o processo hoje existente na criação dos documentos implementados neste trabalho, é possível notar que vários condicionais e critérios devem ser aplicados para que a criação do registro seja realizada com sucesso. Caso esta lógica tivesse que estar em um serviço externo ou até mesmo no *front-end*, acabaria tendo um problema em encontrar onde cada regra de negócio foi aplicada. Isso acarretaria em perda de lógica e, até mesmo, em documentação extensa para manutenções posteriores. Desta forma, toda a lógica descrita hoje pelos órgãos

governamentais ficou dentro de seus respectivos contratos e toda lógica transacional na aplicação.

9.1.4. Simplicidade na comunicação:

Dado que a lógica de negócio, assim como chamadas para serviços externos, pode ser realizadas dentro da máquina virtual da *blockchain*, a necessidade de existir um barramento para realizar a comunicação entre o *front-end* e o *back-end* diminui. Apesar de isso já existir como padrão de mercado para diversas soluções, a forma criada pelas APIs atualmente se demonstra muito prática e simples de implementar, fazendo, assim, a curva de aprendizado não tão acentuada quanto outras tecnologias de mercado, como o Spring. Para o desenvolvimento deste trabalho, foram criadas classes de serviços no angular, das quais suas funções eram expor as chamadas para a blockchain e centralizar na aplicação este controle. Esta forma de implementação é incentivada pela Google, como boas práticas de desenvolvimento Angular e facilita muito na manutenção e na criação de novos componentes que devem usar as mesmas informações ou chamadas de outras telas.

9.1.5. Praticidade na criação dos contratos:

Apesar de tocar no assunto sobre a limitação da linguagem para a criação de *Smart Contracts*, a implementação dos mesmos é simples de se executar e de fácil compreensão. Para aqueles que já possuem conhecimento de outras linguagens como Javascript ou C#, aprender Solidity linguagem formal para a criação de contratos dentro da *blockchain Ethereum* é fácil. Durante o desenvolvimento deste trabalho, foi encontrada certa dificuldade em localizar um ambiente de desenvolvimento integrado do inglês Integrated Development Environment (IDE) que tivesse a praticidade de outras ferramentas de mercado, como refatoração de código e sinalizar erros de compilação. No entanto, dado o conhecimento prévio com linguagens de programação, como Typescript, Javascript e Python, a criação dos contratos não apresentou uma curva de aprendizado alta.

9.1.6. Visibilidade dos dados:

Todos os dados gravados no registro histórico da *blockchain* são de fácil acesso para aqueles com acesso ao *ledger*, precisando apenas realizar a desserialização de tais dados. Isso torna o acesso a informação muito simples em uma aplicação da *blockchain*, já que hoje há sites para a realização destas conversões [24]. Trazendo, assim, transparência de informação para os sistemas e seus usuários. Este fator contribuiu com os testes da aplicação, principalmente nos primeiros momentos. Ao realizar a inserção dos primeiros documentos, sem ainda ter gerado a serialização dos dados, mencionada na Seção 9.2.5, a capacidade de poder ver os dados inseridos facilitou a validação das lógicas e do comportamento esperado de cada função.

9.2. Malefícios

Nesta seção são descritos os malefícios percebidos ao longo do desenvolvimento do trabalho.

9.2.1. Número máximo de parâmetros:

Durante o desenvolvimento deste trabalho, deparou-se com esta limitação dentro da linguagem de programação Solidity. Para a criação de um documento é necessário enviar diversos dados ao contrato, como nome, sobrenome, nome do pai, nome da mãe, nome do avô paterno, nome da avó paterna, nome do avô materno, entre muitos outros. Tendo em vista que estes dados são obrigatórios para a criação de um documento de certidão de nascimento, a linguagem de programação não permite criar apenas um método com todos estes parâmetros, o que acabou sendo realizado fora a criação de diferentes funções, cada qual com seu objetivo que iriam atualizar o registro original. Apesar de este comportamento resolver o problema do número de dados a serem inseridos em uma estrutura de um registro, ele cria a necessidade de criar múltiplas transações e, assim, múltiplos blocos de transações dentro da *blockchain* para a criação de apenas um registro,

fazendo, assim, em uma *blockchain* não privada a criação ter um alto custo e a necessidade de implementação de um controle transacional. Se a criação do bloco falhar, as demais transações não podem ser iniciadas até que o registro seja finalmente concluído com sucesso.

9.2.2. Controle transacional:

Com o fim de não criar uma camada entre a interface e o *back-end* se fez necessário manter o controle transacional no *front-end*. Este tipo de abordagem traz alguns problemas para a aplicação, como o fato de toda a lógica transacional estar contida em uma camada que pode ser facilmente fraudada e que, sem a utilização de algo que dificulte a leitura do código, ele ficará exposto para qualquer um que inspecionar a aplicação ler o mesmo. Tendo em vista que uma aplicação distribuída deve conter o menor número possível de centralização, a necessidade de criar uma camada para este controle não pareceu ideal e dificultou o desenvolvimento, forçando a pensar em meios de a aplicação se manter resiliente sem os controles usualmente implementados, gerando funções nos contratos apenas para a autorização e até mesmo validações, se a ordem das transações, assim como contratos anteriores, ocorreram corretamente.

9.2.3. Linguagem limitada:

Dado que a linguagem de programação Solidity está ainda em construção, estando atualmente na sua distribuição 0.4.25, ela apresenta certas limitações, que hoje já são naturais para outras linguagens. Durante o desenvolvimento deste trabalho, surgiu a necessidade de criar um identificador único e, enquanto normalmente um número aleatório seria utilizado para este propósito, foi necessário encontrar uma nova solução tendo em vista que Solidity não apresenta esta funcionalidade de modo nativo. Normalmente, tal feito é recriado buscando o *hash* do bloco anterior e aplicando alguma forma matemática nele. Neste trabalho foi decidido utilizar o identificador criado pela solução Firebase. Outro problema encontrado foi a criação de estruturas. Apesar de a linguagem suportar objetos dentro de contratos, a passagem

delas para o *front-end* ainda não existe, entrando no problema de serialização descrito na Seção 9.2.5. Este problema foi solucionado criando métodos de serialização que passam números hexadecimais para a camada do *front* que, por sua vez, realizam a desserialização. Isto gera, obviamente, um problema de quantidade de código para algo tão natural em outras linguagens, apesar de tal dificuldade hoje se encontrar dentro, também, da API web3, mencionada no Capítulo 5. Além destes, ainda foram vistos problemas de tratamento de dados, como no caso da concatenação de *strings* ou, até mesmo, a comparação dos mesmos. Para tal, foi preciso transformar cada *string* em um *hash* e comprá-los, que apesar de ser uma solução válida dificulta a entrada para programadores menos experientes e aumenta a quantidade de código a ser escrito significativamente. Alguns contratos externos foram utilizados para tais ações, mas também cria a necessidade do controle de versionamento de contratos externos ao desenvolvimento para soluções simples, como a conversão de textos para inteiros.

9.2.4. Determinístico versus não determinismo:

Devido a tecnologia *blockchain* ser desenhada para ser inteiramente determinística, existe um grande debate dentro da comunidade, em que se discute se chamadas de serviços externos, como SOAP e REST, devem ou não ser feitas dentro das aplicações [57]. Como a internet é não determinística, existe um problema claro que, cada vez que for realizada uma chamada para o serviço, o resultado do mesmo pode alterar e não garantir o determinismo da rede. Isso também interfere na maneira que tais chamadas devem ser feitas para minimizar a não replicabilidade dentro da *blockchain*. Para fazer as chamadas são utilizados os contratos da Oraclize [29], que salva todos os dados utilizados dentro de uma transação externa e depois realiza as chamadas desejadas para fora do contrato. Isso permite que chamadas sejam realizadas e seus parâmetros sejam armazenadas para fins de replicabilidade. Ainda assim, foi necessário alterar a linha de pensamento para somente realizar chamadas de notificação e chamadas que não alterariam o status da *blockchain* para não afetar o determinismo desta aplicação, já que chamadas que originassem alterações no estado da rede poderiam não ser replicáveis.

9.2.5. Serialização e desserialização:

Devido às limitações da linguagem previamente citadas, hoje não existe a possibilidade de passar estruturas complexas de contratos para o *front-end* e a funcionalidade de passar estes objetos para outros contratos. Com isso existe a necessidade de implementar funções de serializações, para passar todos os dados para um formato hexadecimal, e funções de desserialização, para passar os dados do formato de base hexadecimal para uma estrutura previamente conhecida de novo. Embora este processo não seja complexo devido a contratos como Seriality [39], que provém funções de conversão de textos, booleanos e inteiros para bytes, tal implementação é custosa e verbosa, criando a necessidade de diversas implementações tanto *no back-end*, quanto no *front-end*, que outras linguagens não necessitam.

9.2.6. Verbosidade:

Embora a linguagem Solidity tenha sido criada com alguns princípios de Javascript, ela ainda requer muitas linhas de código e palavras reservadas para a codificação. Métodos acabam ficando grandes e com linhas muitas vezes desnecessárias em outras linguagens. Tal malefício não apresenta dificuldade na hora de codificar os contratos, mas sim ao realizar manutenções. Durante o desenvolvimento do trabalho de conclusão, alguns bugs foram encontrados depois de muito tempo e, devido a quantidade excessiva de código gerado, algumas vezes ficava difícil de se realizar uma análise assertiva e rápida. Embora tenham sido realizadas funções usando as melhores práticas, sendo elas com objetivo e métodos claros, os contratos acabaram ficando com um número grande de linhas. Cerca de 180 linhas por contrato.

9.2.7. Versionamento:

Devido a linguagem Solidity estar ainda em estado de criação, assim como seu ecossistema (Web3, Oraclize, Seriality, etc), o desenvolvimento

acaba tornando-se muito mais complexo do que deveria ser. Com dificuldades em usar versões *major* atuais, forçando-nos a usar cinco versões passadas de algumas APIs e versões experimentais de outras, o desenvolvimento acabou sendo mais lento e necessitando de diversas soluções alternativas. Este fato gera certa angústia para aqueles que pensam na manutenção de um sistema de longo prazo. Já que ao utilizarmos algumas APIs, como Web3, foi necessário regredir as versões destas até encontrar uma estável para o desenvolvimento. Isto resulta em atualizações de versões muito custosas, devido a grandes alterações e, até mesmo, versões *minors* terem "*breaking changes*" (às vezes devido a imaturidade da comunidade em separar suas alterações, ou talvez um baixo controle do que está entrando em cada versão). Apesar disso, este malefício apresenta certo benefício, pois já que tais APIs são de código aberto, existe a possibilidade para a resolução de defeitos. Assim, dando maior controle ao suporte à aplicação.

10. CONCLUSÃO

Neste projeto, foi implementada uma aplicação distribuída de cadastro e gestão de documentos governamentais, utilizando um framework de *blockchain* no lugar de uma ferramenta padrão de banco de dados, como hoje é utilizado no mercado. Do ponto de vista de implementação e arquitetura desta solução, este projeto se demonstrou muito interessante, dado que, determinados paradigmas se demonstraram menos eficientes e outros modelos precisaram ser revistos, quando se fala de aplicações de armazenamento massivo de dados.

Por um lado, foi possível validar todos os pontos positivos aqui citados, assim como sobre tais tecnologias aqui utilizadas. A possibilidade de ter transações, cujos resultados são uma forma imutável dos dados gravados na própria aplicação, oferece uma garantia não presente hoje em outras aplicações de bancos de dados formais. Tal implicação pode se tornar vital no combate à fraude, não só dos documentos aqui desenvolvidos e exemplificados, mas igualmente para outras esferas públicas. Registros poderiam ser armazenados com a certeza de que nenhuma alteração seria feita aos mesmos, além de prover, também, um histórico muito mais preciso com o passar dos anos e do desenvolvimento do país. Este fato está diretamente ligado à questão de gerar um país de informações abertas, permitindo a organização de pedidos para a visibilidade de dados, podendo ser totalmente online. Juntamente se torna um caso de comodidade, permitindo que o cidadão não precise sair de casa para criar e renovar seus documentos, fato este que contribui com os modelos de negócios mais comuns, como Spotify, NuBank e Uber, que apresentam uma forma mais simples e prática de realizar tudo digitalmente. Finalmente, também permitindo que a impressão de milhares de documentos não seja necessária [21] e garantindo cadastros mais completos para todos.

Claro que tais benefícios não viriam desacompanhados. Movimentos como a *General Data Protection Regulation* (GDPR), na Europa, ou a Lei Geral de Proteção de Dados (LGPD), no Brasil, que entrará em vigor em fevereiro de 2020 [14], conjecturam sobre a proteção de dados dos cidadãos no âmbito público ou no privado. A definição de dados pessoais difere um

pouco entre cada uma das leis, entretanto, o conceito geral refere-se a dados que, de forma isolada ou em conjunto, a outros identificam um indivíduo ou seu comportamento [7]. Outro direito, garantido por essas legislações, está relacionado ao indivíduo poder solicitar a exclusão de todos os seus dados pelo portador. Esse fato entra em conflito com soluções que utilizam a tecnologia da *blockchain*, devido a implementação da mesma. Ao criar um bloco dentro da corrente, é impossível realizar sua remoção. Mesmo que este trabalho use de uma estrutura dinâmica de dados capaz de alterar a informação contida neles para o sistema, apenas mais um bloco foi criado. Esta característica inerente a todas as soluções que utilizam *blockchain* deverá ser alterada para casos de países com legislações parecidas. Além disto, é necessário que, para uma larga implementação de um sistema como este, as leis e processos sejam semelhantes ou iguais e que os locais de implementação utilizem de infraestrutura igual. Atualmente, no Brasil, este sistema também apresentaria um problema de implementação, tendo em vista que existe independência na adoção de sistemas e em adquirir infraestrutura em cada município e estado [41]. Este ponto em questão se torna muito importante quando se cogita a implementação de um sistema distribuído, já que, para o seu funcionamento, como o nome já sugere, esta aplicação deve ser distribuída. Para tal funcionamento, acredita-se que os municípios e estados devem optar de maneira nacional a adoção de um sistema único e uma estrutura semelhante para a correta implementação. Vale frisar, também, que uma vez que tal aplicação esteja em vigor, será necessário que uma equipe de suporte e implementação de novos documentos/contratos existam. Sendo assim, se faz necessário o treinamento de diversos times qualificados para lidar com uma tecnologia e linguagem que está em constante desenvolvimento por sua comunidade e possui problemas históricos de versionamento.

Desta forma, conclui-se que, apesar de uma *blockchain* privada apresentar inúmeros benefícios desejáveis para um sistema de controle de documentação, atualmente, tal implementação não seja viável. Para que esta solução seja implantada em pequena escala, seria necessária a cooperação de grande parte dos municípios na adoção de um novo sistema e na adequação de uma única infraestrutura para que diversos *full nodes* sejam

utilizados e a aplicação se torne distribuída. Este percalço na implementação de sistemas de forma uniforme já vem sendo visto em iniciativas como na publicação de Estratégia de Governança Digital (EGD), que procuram criar a cultura nos setores públicos [15]. Porém, além da cultura, existe a questão do custo de tal implementação. Como comentado anteriormente, tal tecnologia ainda apresenta uma linguagem nova ao mercado e treinamentos precisam ser ministrados para que os menores municípios consigam suportar a própria aplicação. Tais cursos não necessitam ser de desenvolvimento, mas ainda sim entender como a aplicação funciona e como relatar erros e resolver menores defeitos pela aplicação seria necessário para não deixá-los esperando por suas capitais. Um modelo como este, precisa permanecer escalável fisicamente, tanto quanto digitalmente. Por fim, apesar da infraestrutura de cidades menores não precisar ser tão robusta quanto das maiores, devido a aplicação ter a característica da disponibilidade de um sistema distribuído, ainda assim, se faz necessário mais investimento neste quesito. Também é essencial a adequação desta solução às novas leis provenientes da LGPD, já que tais normas entram em vigor e exigem que as informações sejam deletadas de sistemas como o deste trabalho. Sendo assim, concorda-se que existe espaço para a implementação de um sistema de difícil fraude, distribuído e digital no futuro no Brasil, mas a conclusão tomada é a de que esta aplicação não possui ainda base no quesito de infraestrutura e organização entre estados e municípios para tal.

REFERÊNCIAS

- [1] **A Guide to Smart Contracts and Their Implementation.** Disponível em: <https://rubygarage.org/blog/guide-to-smart-contracts>. Acesso em: jun. 2018.
- [2] ACCENTURE PLC. Accenture Consulting. **Banking on Blockchain: a value analysis for investment banks.** Dublin, 2017. 10 p. Disponível em: <<https://www.accenture.com/us-en/insight-banking-on-blockchain/>> Acesso em: nov. 2018.
- [3] ACCENTURE PLC. Accenture Consulting. **Editing the Uneditable Blockchain: Why distributed ledger technology must adapt to an imperfect world.** Dublin, 2016. 8p. Disponível em: <<https://www.accenture.com/br-pt/company-news-release-editable-blockchain-prototype/>> Acesso em: nov. 2018.
- [4] **Analyzing Oracle Security – Oracle Critical Patch Update January 2018.** ERPScan. Disponível em: <<https://erpscan.com/press-center/blog/analyzing-oracle-security-oracle-critical-patch-update-january-2018/>>. Acesso em: abr. 2018.
- [5] **Angular Docs.** Disponível em: <<https://angular.io/>>. Acesso em: nov. 2018.
- [6] ANGULAR. **angular/angularfire2.** Disponível em: <<https://github.com/angular/angularfire2>>. Acesso em: nov. 2018.
- [7] **Arquivos Agenda da Privacidade e da Proteção de Dados.** Disponível em: <<https://www.jota.info/opiniao-e-analise/colunas/agenda-da-privacidade-e-da-protecao-de-dados/>>. Acesso em: nov. 2018.
- [8] **BRF e Carrefour: projeto de blockchain com IBM.** Baguete. Disponível em: <<https://www.baguete.com.br/noticias/09/11/2017/baguete-brf-e-carrefour-projeto-de-blockchain-com-ibm>>. Acesso em: abr. 2018.
- [9] **Become an e-resident.** e-Residency. Disponível em: <<https://e-resident.gov.ee/become-an-e-resident>>. Acesso em: abr. 2018.
- [10] BITTENCOURT, Wastony Aguiar. **A Constituição de 1988: Democracia e Política.** Jus.com.br. Disponível em: <<https://jus.com.br/artigos/56301/a-constituicao-de-1988-democracia-e-politica-art-14>>. Acesso em: abr. 2018.
- [11] BRASIL. Constituição (1988). **Constituição da República Federativa do Brasil:** promulgada em 5 de outubro de 1988. Organização do texto: Juarez de Oliveira. 4. ed. São Paulo: Saraiva, 1990. 168 p. (Série Legislação Brasileira).
- [12] BRASIL, Decreto n. 5.604, de 25 de março de 1874. **Manda observar o Regulamento desta data para execução do art. 2º da Lei nº 1829 de 9 de**

- Setembro de 1870, na parte em que estabelece o registro civil dos nascimentos, casamentos e óbitos.** Disponível em:
<<http://www2.camara.leg.br/legin/fed/decret/1824-1899/decreto-5604-25-marco-1874-550211-publicacaooriginal-65873-pe.html>> Acesso em: nov. 2018.
- [13] BRASIL. Lei n. 7.116, de 29 de agosto de 1983. **Assegura validade nacional às Carteiras de Identidades regula sua expedição e dá outras providências.** Disponível em: <http://www.planalto.gov.br/ccivil_03/leis/1980-1988/l7116.htm>. Acesso em: nov. 2018.
- [14] Brasil. Lei n. 13.709, de 14 de agosto de 2018. **Dispõe sobre a proteção de dados pessoais e altera a Lei nº 12.965, de 23 de abril de 2014 (Marco Civil da Internet).** Disponível em:
<http://www.planalto.gov.br/ccivil_03/_Ato2015-2018/2018/Lei/L13709.htm>. Acesso em: nov. 2018.
- [15] BRASIL. Ministério do Planejamento, Orçamento e Gestão. **Estratégia de Governança Digital da Administração Pública Federal 2016- 19 / Ministério do Planejamento, Orçamento e Gestão, Secretaria de Tecnologia da Informação.** -- Brasília: MP, 2016. 36 p.: il. Disponível em:<<https://www.governodigital.gov.br/documentos-e-arquivos/egd-estrategia-de-governanca-digital-da-administracao-federal-2016-2019.pdf>>. Acesso em: nov. 2018.
- [16] **Browser Automation.** Disponível em: <<https://www.seleniumhq.org/>>. Acesso em: nov. 2018.
- [17] BUTERIN, Vitalik. **Ethereum: A Next-Generation Cryptocurrency and Decentralized Application Platform.** Bitcoin Magazine. Disponível em:<<https://bitcoinmagazine.com/articles/ethereum-next-generation-cryptocurrency-decentralized-application-platform-1390528211/>>. Acesso em: abr. 2018.
- [18] CADARI, Luciano. **Primeiro RG do Brasil, RankBrasil - Recordes Brasileiros.** , RankBrasil - Recordes Brasileiros. Disponível em:<http://www.rankbrasil.com.br/Recordes/Materias/06Wu/Primeiro_Rg_Do_Brasil>. Acesso em: Abril. 2018.
- [19] **Casal homoafetivo registra criança com duas mães diretamente no cartório, no Pará.** Disponível em: <<https://g1.globo.com/pa/para/noticia/casal-homoafetivo-registra-crianca-com-duas-maes-diretamente-no-cartorio-no-pará.ghtml>>

homoafetivo-do-para-consegue-na-justica-o-direito-de-registrar-crianca-com-duas-maes.ghtml>. Acesso em: nov. 2018.

[20] **CNH Digital.** Disponível em:
 <<http://www.detran.rs.gov.br/conteudo/48734/cnh-digital>>. Acesso em: abr. 2018.

[21] **Em que fase está o projeto de criar um documento único para os brasileiros.** Nexo Jornal. Disponível em:
 <<https://www.nexojornal.com.br/expresso/2017/03/20/Em-que-fase-está-o-projeto-de-criar-um-documento-único-para-os-brasileiros>>. Acesso em: abr. 2018.

[22] **ETHEREUM. ethereum/web3.js.** Disponível em:
 <<https://github.com/ethereum/web3.js/>> . Acesso em: nov. 2018.

[23] **ETHEREUM. ethereum/wiki.** GitHub. Disponível em:
 <<https://github.com/ethereum/wiki/wiki/White-Paper>>. Acesso em: jun. 2018.

[24] **ETHERSCAN.IO. Ethereum (ETH) Blockchain Explorer.** Disponível em:
 <<https://etherscan.io/>>. Acesso em: nov. 2018.

[25] **Falsificação de documento representa 75% das fraudes registradas em MG.** Jornal Hoje. Disponível em: <g1.globo.com/jornal-hoje/noticia/2014/02/falsificacao-de-documento-representa-75-das-fraudes-registradas-em-mg.html>. Acesso em: abr. 2018.

[26] **FOLHA. Mãe trans é impedida de registrar filho biológico em cartório no RS.** Disponível em: <<https://www1.folha.uol.com.br/cotidiano/2018/08/mae-trans-e-impedida-de-regularizar-filho-biologico-em-cartorio-no-rs.shtml>>. Acesso em: nov. 2018.

[27] GARTENBERG, Chaim. **The PS4 gets hacked for homebrew software and PS2 emulation, but there's a catch.** The Verge. Disponível em:
 <<https://www.theverge.com/circuitbreaker/2018/1/23/16922798/playstation-4-homebrew-hack-software-firmware-2016-ps2-emulation-linux>>. Acesso em: abr. 2018.

[28] HABER, Stuart and STORNETTA, W. Scott. **How to time-stamp a digital document.** [s.l.]: DIMACS, Center for Discrete Mathematics and Theoretical Computer Science, 1990. 11 p.

[29] **Home.** Disponível em: <<http://docs.oraclize.it/>>. Acesso em: nov. 2018.

[30] **IBGE.** IBGE - Instituto Brasileiro de Geografia e Estatística. Disponível em: <<https://ww2.ibge.gov.br/home/presidencia/noticias/11122001onu.shtml>>. Acesso em: abr. 2018.

- [31] ICO & CRYPTO NEWS. **The British government is ready to invest £ 31 million in Blockchain startups.** Medium. Disponível em: <<https://medium.com/ico-crypto-news/the-british-government-is-ready-to-invest-31-million-in-blockchain-startups-15b6a837e94f>>. Acesso em: abr. 2018.
- [32] JOHNSON, S. C. **Lint, a C program checker.** Tradução. [s.l.] Bell Telephone Laboratories, 1977.
- [33] LUNARDI, R. C. et al. Distributed access control on IoT ledger-based architecture. **NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium**, 2018.
- [34] MILLER, Mark S.; CUTSEM, Tom Van and TULLOH, Bill. Distributed Electronic Rights in JavaScript. **Programming Languages and Systems Lecture Notes in Computer Science**, p. 1–20, 2013.
- [35] **Muito além da criptomoeda.** Revista Amanhã. Disponível em: <<http://www.amanha.com.br/posts/view/5354/muito-alem-da-criptomoeda>>. Acesso em: abr. 2018.
- [36] NAKAMOTO, Satoshi. **Bitcoin: A Peer-to-Peer Electronic Cash System.** Japão, 2008.
- [37] **Novo RG com chip pode custar até R\$ 40 para o governo.** Brasil. Disponível em: <globo.com/brasil/noticia/2011/05/novo-rg-com-chip-pode-custar-ate-r-40-para-o-governo.html>. Acesso em: abr. 2018.
- [38] **PF avalia investigar se outro passaporte brasileiro também foi usado por ditador norte-coreano.** G1. Disponível em: <globo.com/politica/blog/matheus-leitao/post/2018/03/06/pf-avalia-investigar-se-outro-passaporte-brasileiro-tambem-foi-usado-por-ditador-norte-coreano.ghtml>. Acesso em: abr. 2018.
- [39] POULADZADE. **pouladzade/Seriality.** Disponível em: <<https://github.com/pouladzade/Seriality>>. Acesso em: nov. 2018.
- [40] **Promise.** Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Promise>. Acesso em: dez. 2018.
- [41] PRZEYBILOVICZ et al. **O uso da tecnologia da informação e comunicação para caracterizar os municípios: quem são e o que precisam para desenvolver ações de governo eletrônico e smart city.**

Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0034-76122018000400630&lng=pt&nrm=iso>. Acesso em: nov. 2018.

[42] Recurso do Firebase de mensagens no aplicativo | Firebase.

Disponível em: <<https://firebase.google.com/products/in-app-messaging/?hl=pt-br>>. Acesso em: nov. 2018.

[43] Saiba como fazer a certidão de nascimento. Disponível em: <<http://www.brasil.gov.br/cidadania-e-justica/2009/10/saiba-como-fazer-a-certidao-de-nascimento>> . Acesso em: nov. 2018

[44] Saiba como funcionam as taxas de transação do Bitcoin. Blog Foxbit. Disponível em: .foxbit.com.br/saiba-como-funcionam-as-taxas-de-transacao-do-bitcoin/. Acesso em: jun. 2018.

[45] SCHOLLMEIER, R. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. **Proceedings First International Conference on Peer-to-Peer Computing**, p. 1-2, 2002.

[46] Sobre - TudoFácil RS. Disponível em: <<http://www.tudofacil.rs.gov.br/sobre>>. Acesso em: nov. 2018

[47] Solutions. E-stonia. Disponível em: <<https://e-estonia.com/solutions>>. Acesso em: abr. 2018.

[48] TAI, S.; EBERHARDT, J.; KLEMS, M. Not ACID, not BASE, but SALT - A Transaction Processing Perspective on Blockchains. **Proceedings of the 7th International Conference on Cloud Computing and Services Science**, 2017.

[49] Toolkit. E-stonia. Disponível em: <<https://e-estonia.com/toolkit>>. Acesso em: abr. 2018.

[50] Turismo. Disponível em: <<https://www.mercosur.int/pt-br/cidadaos/turismo>>. Acesso em: nov. 2018

[51] UNITED KINGDOM. Office for Science. **Distributed Ledger Technology: beyond block chain.** London, 2016. 88 p.

[52] Unspent Transaction Output, UTXO. Bitcoin - Open source P2P money. Disponível em: <<https://bitcoin.org/en/glossary/unspent-transaction-output>>. Acesso em: jul. 2018.

[53] We have built a digital society and so can you. E-stonia. Disponível em: <<https://e-estonia.com/>>. Acesso em: abr. 2018.

[54] **World Population Prospects – 2017 Revision: Global population | Multimedia Library - United Nations Department of Economic and Social Affairs.** United Nations. Disponível em: <<https://www.un.org/development/desa/publications/graphic/wpp2017-global-population>>. Acesso em: abr. 2018.

[55] **What is blockchain? - Definition from WhatIs.com.** SearchCIO. Disponível em: <<https://searchcio.techtarget.com/definition/blockchain>>. Acesso em: abr. 2018.

[56] **What is Ethereum?**. What is Ethereum? - Ethereum Homestead 0.1 documentation. Disponível em: <<http://www.ethdocs.org/en/latest/introduction/what-is-ethereum.html>>. Acesso em: jun. 2018

[57] **Why can't contracts make API calls?** Disponível em: <<https://ethereum.stackexchange.com/questions/301/why-cant-contracts-make-api-calls>>. Acesso em: nov. 2018.

[58] WOOD, Gavin. **Ethereum: A Secure Decentralised Generalised Transaction Ledger.** Inglaterra, 2017. Disponível em: <<http://yellowpaper.io/>>. Acesso em: abr. 2018.

GLOSSÁRIO

Back-End - Parte do sistema onde encontra-se a lógica, instruções e ações que o sistema realiza a partir das interações do usuário com o *front-end*.

Breaking Changes – Qualquer alteração em um componente de um sistema que possa levar outra parte a falhar.

Broadcast - Método de transferência de dados ou pacotes de dados para todos os componentes de uma rede.

Build - Em desenvolvimento de softwares este termo é utilizado para caracterizar partes compiladas, testadas e validadas do sistema para sua versão final.

Bugs - Erros ou falhas em softwares que podem provocar seu mau desempenho.

Cliente - Termo referente a um componente de uma rede que consome serviços ou recursos ofertados por outro componente da mesma rede.

Criptomoeda - Moeda digital que se utiliza da tecnologia de *blockchain* e da criptografia associada para assegurar a validade da transação.

Crowdfunding - Captação financeira por múltiplas fontes de forma coletiva para uma pesquisa de interesse de todos.

Debug - Processo de encontrar e reduzir *bugs*

Front-end - Parte do sistema onde ocorre a interação com o usuário e que envia instruções para interpretação do *back-end*.

Gas - Nome dado a taxa de pagamento necessário para realizar uma transação no Ethereum.

Genesis - Ponto inicial que faz algo existir, no caso aplicado ao trabalho, é o termo referente ao primeiro bloco da cadeia de blocos.

Homo Economicus - Ou Homem Econômico, é um ser humano fictício formulado por economistas.

Intrínseca - Que faz parte da essência de alguém; que é característico, próprio, essencial ou fundamental; inerente: qualidade intrínseca.

IoT - Nome atribuído a objetos que possuem sistemas embarcados, sensores ou conectores a uma rede.

Linguagens de Programação - Método padronizado de instruções e regras sintáticas ou semânticas utilizadas para definir um programa de computador.

Linting - Ato de realizar *lint* (revisão de código) em desenvolvimento de softwares.

Logado - Define-se o ato de realizar o acesso pessoal em um sistema ou meio eletrônico.

Minors – Versão compacta de um software.

Mercosul - Mercado Comum do Sul composto por alguns países da América do Sul.

Open Source - Modelo de desenvolvimento de programas de computador de forma que seu licenciamento seja livre.

Oraclize - Serviço da Oracle para transportes de dados para *blockchain*.

Observables – API que fornece serviço de mensageria.

Prerrogativas - Direito próprio de um ofício, cargo ou profissão; regalia: usufruía das prerrogativas que a política lhe trazia.

Responsável Legalmente - Representante de uma entidade, personificada ou não de forma jurídica.

Servidor - Termo associado ao componente de uma rede que oferta serviços ou recursos para outro componente da mesma rede.

Solidity - Linguagem de programação utilizada para desenvolvimento de contratos inteligentes.

Startup - Ato de começar algo, normalmente relacionado com companhias e empresas que estão no início de suas atividades e que buscam explorar atividades inovadoras no mercado.

TAG - Estrutura de linguagem de marcação que contém instruções pré-estabelecidas.

UTXO - Uma transação na *blockchain* que não foi utilizada como entrada e uma nova transação.

Verbosa - Em desenvolvimento de software, códigos verbosos são aqueles que necessitam de bastante instruções para que a aplicação execute o que é desejado.

APÊNDICE A - Definição do Cronograma TC I e TC II

Neste apêndice estão apresentadas as atividades previstas para a realização deste trabalho, tanto para o primeiro semestre de 2018 quanto para o segundo semestre.

Atividades Previstas TC I

As atividades descritas nesta seção determinam, de forma resumida, as principais atividades relacionadas à realização deste trabalho no primeiro semestre de 2018.

1. Definir o tema a ser realizado neste trabalho.
2. Revisão bibliográfica de materiais e informações que auxiliarão no desenvolvimento do trabalho.
3. Elaboração da fundamentação teórica relativa à proposta de trabalho de conclusão de curso.
4. Elaboração da proposta de trabalho.
5. Entrega da proposta.
6. Indicação do avaliador.
7. Elaborar volume final da proposta.
8. Pesquisar sobre trabalhos acadêmicos correlatos.
9. Revisão dos pontos citados na primeira avaliação.
10. Entrega do volume final.
11. Reuniões com o orientador.

Dependências e Cronograma de Atividades TC I

Neste tópico consta como este trabalho foi organizado e em quais meses ocorreram as entregas das principais atividades para a conclusão deste trabalho. A seguir, conforme a tabela 1, o cronograma definido para o primeiro semestre de 2018.

Tabela 1: Cronograma de Atividades TC I

Atividades	2018 /1				
	Março	Abril	Maio	Junho	Julho
Definir tema					
Revisão bibliográfica					
Elaboração da fundamentação teórica relativa à proposta					
Elaboração da proposta					
Entrega da proposta					
Indicação do avaliador					
Elaboração do volume final					
Pesquisa sobre trabalhos acadêmicos correlatos					
Revisão dos pontos citados na primeira avaliação					
Entrega do volume final					
Reuniões com o orientador					

Fonte: Autoral

Atividades Previstas TC II

As atividades descritas neste item determinam, de forma resumida, as principais atividades relacionadas à realização deste trabalho no segundo semestre de 2018.

1. Desenvolver a solução proposta.
2. Realizar documentação do trabalho.
3. Montar *banner* do trabalho.
4. Definir banca de avaliadores.
5. Defender este trabalho na banca de avaliadores.
6. Revisão dos pontos citados pela banca de avaliadores.
7. Entrega da solução.
8. Entrega da documentação.
9. Reuniões com o orientador.

Dependências e Cronograma de Atividades TC II

Neste tópico consta como este trabalho foi organizado e em quais meses ocorreram as entregas das principais atividades para sua conclusão. A seguir, conforme a tabela 2, o cronograma definido para o segundo semestre de 2018.

Tabela 2: Cronograma de atividades TC II

Atividades	2018 /2				
	Agosto	Setembro	Outubro	Novembro	Dezembro
Desenvolver a solução proposta					
Realizar documentação do trabalho					
Montar <i>banner</i> do trabalho					
Definir banca de avaliadores					
Defender este trabalho na banca de avaliadores					
Revisão dos pontos citados pela banca de avaliadores.					
Entrega da solução proposta					
Entrega da documentação					
Reuniões com orientador					

Fonte: Autoral

APÊNDICE B - RECURSOS NECESSÁRIOS

Neste apêndice, há a descrição dos recursos que foram necessários para a realização deste trabalho.

Recursos de Software

- Adobe Illustrator: Software para a criação das imagens.
- Angular: Framework para desenvolvimento do *front-end*.
- Astah: Ferramenta para modelagem dos casos de uso.
- Bizagi: Ferramenta para modelagem de processos em notação BPMN.
- Chrome ou Safari: navegadores utilizados.
- Draw.io: Software para criação de diagramas.
- EasyBib: Complemento do Google Docs para formatar citações.
- Ethereum: Framework de código aberto, que foi utilizado como base do projeto.
- Firebase: BaaS para mensageria e autenticação.
- Google Docs: processador de texto de acesso via internet.
- Jasmine: Framework para desenvolvimento orientado a comportamento.
- Karma: Executor de testes unitários para o framework Angular.
- Microsoft Word: processador de texto.
- Remix: Editor de código fonte para desenvolvimento de *smarts contracts*.
- Sistema Operacional: Windows 10 ou Mac OS Mojave.
- Tlint: ferramenta para análise de código em Typescript.
- Truffle: Framework para desenvolvimento de blockchain.
- Visual Studio Code: Editor de código fonte para múltiplas linguagens de programação.

Recursos de Hardware

- Computador com acesso à internet.
- Espaço em disco de 500Gb, para armazenamento dos dados relativos a solução.
- Memória Ram de 4Gb, necessária para suportar os softwares de desenvolvimento do sistema, sem que haja interrupções da máquina devido ao uso da memória Ram.
- Processador i5 de 1.8 Ghz, com capacidade para manter funcionando a aplicação.

APÊNDICE C - Testes Unitários

Neste apêndice serão listados todos os cenários de testes unitários realizados. Para identificação dos mesmos no código-fonte da aplicação ocorre através das nomenclaturas: <nome-componente>.component.spec.ts ou <nome-serviço>.service.spec.ts. Localizado nas pastas de cada componente ou serviço.

Cenários dos testes unitários:

- AppRoutingModule
 - should create an instance
- AppComponent
 - should create the app
 - should render the router-outlet tag
- BirthCertificationComponent
 - should create
 - should activate NavBar
 - should route to activate
 - should route to view
- ActivateBirthComponent
 - should create
 - should validate non optional camps
 - should navigate to view
 - should trigger service calls
- RenewBirthComponent
 - should create
 - should validate non optional camps
 - should navigate to view
 - should trigger service calls
- SonCertificateComponent
 - should create
- GuardianCertificateComponent
 - should create

- SharedCertificateComponent
 - should create
- ViewBirthComponent
 - should create
 - should trigger service calls
 - should show Birth Certificate
- GeneralRegistryComponent
 - should create
 - should activate NavBar
 - should route to activate
 - should route to view
- ActivateGeneralComponent
 - should create
 - should validate non optional camps
 - should navigate to view
 - should trigger service calls
- RenewGeneralComponent
 - should create
 - should validate non optional camps
 - should navigate to view
 - should trigger service calls
- ViewGeneralComponent
 - should create
 - should trigger service calls
 - should show General Registry
- HomeComponent
 - should create
 - should show title when there are General Regestries
 - should show title when there are Birth Certificates
- LoginComponent
 - should create
 - should direct to home if user is in session
 - should login when promise is resolved
 - should alert when promise is rejected

- should alert when user is undefined
- should alert when mail is undefined
- should alert when password is undefined
- should navigate to register and storage info into session
- NavbarComponent
 - should create
 - should update with the right amount of notification
 - should logout user
- NotificationsComponent
 - should create
 - should mark as readed
 - should mark as deleted
 - should appear no message text
 - should contain notifications
- RegisterComponent
 - should create
- ShareModalComponent
 - should create
 - should emit a cancel event
 - should share a document and call cancel
 - should not share a document and call setErrorMessage
 - should not share a document for undefined recipient
- SidebarNavComponent
 - should create
 - should not have normal-activate and have son-activate
 - should have normal-activate and not have son-activate
 - should deactivate the activate menu
 - should deactivate the view and renew menu
- StandardRegestryComponent
 - should create
 - should call lookUp
 - should call createStandardRegestry
 - should call lookUpId
 - should call update

- BirthRegistryService
 - should be created
 - should call createRegistry with transactionObject
 - should call createSon with transactionObject
 - should call createSonRegistry with transactionObject
 - should call setStandardAddress with transactionObject
 - should call isApproved
 - should call lookUpId
 - should call lookUpSonsId
 - should call negate with transactionObject
 - should call approve with transactionObject
 - should call share with transactionObject
 - should call getSharedDocument
 - should call update with transactionObject
 - should call updateAdditionalInfo with transactionObject
- GeneralRegistryService
 - should be created
 - should call createSingleGeneralRegistry with transactionObject
 - should call createMarriedGeneralRegistry with transactionObject
 - should call update with transactionObject
 - should call share with transactionObject
 - should call getSharedDocument
 - should call setStandardAddress with transactionObject
 - should call negate with transactionObject
 - should call approve with transactionObject
 - should call lookUpId
 - should call isApproved
- NotificationService
 - should be created
- NotificationSerializeService
 - should be created
 - should call sendMessage with transactionObject

- should call getMessagesIds
- should call getMessage
- should call isRead
- should call markRead with transactionObject
- should call isDeleted
- should call markDeleted with transactionObject
- SerializableService
 - should be created
 - should deserialize a message
 - should return empty
- UserSerializableService
 - should be created
- StandardRegistryService
 - should be created
- AuthServiceService
 - should be created
 - should try to create an user and login
 - should try to login
 - should try to get the login user
 - should try to delete the user
 - should try to logout the user
- NavHandlerService
 - should be created
 - should activate menu
- WindowRefService
 - should be created
 - should return the global window variable

APÊNDICE D - Avaliação de Código

Este apêndice será dedicado para as regras de avaliação de código através da ferramenta Lint. A documentação desses testes estão disponíveis em: https://bitbucket.org/nathans_projects/front-end/src/master/tslint.json.

APÊNDICE E - Testes Funcionais

Testes Funcionais *Front-End*

- FT001: Realizar cadastro - Positivo
- FT002: Realizar cadastro - email inválido - Negativo
- FT003: Realizar cadastro - usuário já existe - Negativo
- FT004: Realizar login - Positivo
- FT005: Realizar login - email inválido - Negativo.
- FT006: Realizar login - senha inválida - Negativo.
- FT007: Realizar cadastro RG - Positivo.
- FT008: Realizar cadastro RG - não preenchimento de informações não opcionais - Negativo.
- FT009: Realizar renovação RG - Positivo.
- FT010: Realizar compartilhamento RG - Positivo.
- FT011: Realizar cadastro CN - Positivo.
- FT012: Realizar cadastro CN - não preenchimento de informações não opcionais - Negativo.
- FT013: Realizar renovação CN - Positivo.
- FT014: Realizar compartilhamento CN - Positivo.
- FT015: Realizar criação de CN para filho - Positivo.
- FT016: Realizar compartilhamento de CN de filho - Positivo
- FT017: Realizar atualização RP - Positivo.
- FT018: Realizar compartilhamento inválido - Negativo
- FT019: Validar notificação criação RG - Positivo
- FT020: Validar notificação criação CN - Positivo
- FT021: Validar notificação atualização RG - Positivo
- FT022: Validar notificação atualização CN - Positivo
- FT023: Validar notificação atualização RG - Negativo
- FT024: Validar notificação atualização CN - Negativo
- FT025: Validar notificação compartilhamento RG - Postivio
- FT026 Validar notificação compartilhamento CN - Positivo
- FT027: Realizar logout - Positivo

Testes Funcionais *Back-End*

- FT028: Realizar cadastro RP - Positivo.
- FT029: Realizar cadastro RG - Positivo.
- FT030: Realizar cadastro CN - Positivo.
- FT031: Realizar cadastro RP - Firebase ID inválido - Negativo.
- FT032: Realizar cadastro RG - Firebase ID inválido - Negativo.
- FT033: Relazar cadastro CN - Firebase ID inválido - Negativo.
- FT034: Realizar busca RP - Firebase ID inválido - Negativo.
- FT035: Realizar busca RG - Firebase ID inválido - Negativo.
- FT036: Realizar busca CN - Firebase ID inválido - Negativo
- FT039: Realizar atualização RP - Positivo
- FT040: Realizar atualização RG - Positivo
- FT041: Realizar atualização CN - Positivo
- FT042: Compartilhar RG - Positivo
- FT043: Compartilhar CN - Positivo
- FT044: Realizar cadastro CN filho - Positivo
- FT045: Compartilhar CN filho - Positivo
- FT046: Enviar notificação - Positivo
- FT047: Criar usuário notificação - Positivo
- FT048: Criar mensagem notificação - Postivo
- FT049: Realizar busca notificação - Positivo
- FT050: Realizar conversão string para unix - Positivo
- FT051: Realizar comparação de string - Positivo
- FT052: Realizar comparação de string - Negativo
- FT053: Realizar slice de string - Positivo