ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
POLITÉCNICO
DO PORTO

P.PORTO

M

MESTRADO
ENGENHARIA INFORMÁTICA

Sistema de Consentimento Informado e Reputação Persistido em *Blockchain*

Hélder Miguel Ribeiro de Sousa
Doutor António Alberto dos Santos Pinto

Sistema de Consentimento Informado e Reputação Persistido em *Blockchain*

Politécnico do Porto
Escola Superior de Tecnologia e Gestão
Mestrado em Engenharia Informática

Autor
Hélder Ribeiro de Sousa

Orientador
Professor Doutor António Alberto dos Santos Pinto

Data de Publicação
11 de Fevereiro de 2019

Agradecimentos

Aproveito esta oportunidade para prestar a devida homenagem às pessoas e instituições que contribuíram para que este trabalho fosse possível.

Ao Professor António Pinto por, ao longo do último ano, ter emprestado todo o seu conhecimento e experiência ao desenvolvimento deste trabalho e por se ter mostrado, sempre, genuinamente entusiasmado pelo trabalho.

À Escola Superior de Tecnologia e Gestão do Politécnico do Porto por ter reconhecido, na edição de 2018 do ESTG Masters, o artigo "*On the Feasibility of Blockchain for Online Surveys with Reputation and Informed Consent Support*" que escrevi em parceria com o Professor António Pinto.

Ao "*9th International Symposium on Ambient Intelligence*" por ter publicado, no livro "*Advances in Intelligent Systems and Computing Volume 806*", o artigo "*On the Feasibility of Blockchain for Online Surveys with Reputation and Informed Consent Support*" que escrevi em parceria com o Professor António Pinto.

Ao Ricardo e ao Pedro por, no papel de responsáveis da minha entidade patronal (Loqr), terem ajudado a criar condições para que este trabalho se tivesse tornado realidade.

À malta da Loqr que sempre demonstrou interesse no trabalho que eu estava a desenvolver e me incentivou a continuar, especialmente ao Machado (que me apresentou 1001 coisas que eu desconhecia sobre o universo das criptomoedas e das *blockchains*) e ao Sérgio (que se disponibilizou para me dar um *refresh* em desenvolvimento *Android*).

Ao Nelson (sem acento) e ao Rui que estiveram comigo ao longo destes anos em que decidi aprender o "bê-á-bá" da computação.

Ao Joaquim e ao Francisco que sempre incentivaram o amigo "desaparecido em combate".

À minha namorada Andreia.

Aos meus pais.

Resumo

Na economia digital dos tempos que correm o crescimento económico depende significativamente da partilha de dados a nível global. O novo Regulamento Geral de Proteção de Dados (RGPD), que entrou em vigor a 25 de Maio de 2018, representou uma drástica mudança de paradigma neste tipo de legislação e um claro endurecimento das regras de proteção de dados.

Enquanto todas as organizações que lidam com dados pessoais de cidadãos europeus se debatem com necessidades tecnológico-legais prementes resultantes da entrada em vigor do RGPD, a tecnologia *blockchain* floresce e afirma-se como a solução *de facto* para cenários em que é fulcral manter um livro de razão distribuído e descentralizado.

Este trabalho propõe uma solução inovadora que permite obter o consentimento de um cidadão europeu para o processamento dos seus dados pessoais de acordo com o que está previsto no RGPD, recorrendo à tecnologia *blockchain* para persistir provas desse mesmo consentimento. O sistema proposto diminui significativamente a fricção resultante da adaptação às exigências da nova realidade regulatória, prevendo que sujeito dos dados seja capaz avaliar o desempenho do controlador de dados e de gerir os seus consentimentos comodamente através seu *smartphone*; que o controlador de dados tenha acesso a um sistema que lhe permite obter o consentimento do sujeito dos dados em conformidade com o RGPD; e dota a entidade reguladora de uma ferramenta que lhe permite consultar as provas de um consentimento e aferir qual o estado da relação entre o sujeito dos dados e o controlador de dados.

Palavras-chave: RGPD, Consentimento Informado, Sistemas Reputacionais, *Blockchain*, Plataforma *Ethereum*

Abstract

Today's digital economy relies heavily on global data exchange flows. On May 25th 2018, the European Union's General Data Protection Regulation (GDPR) came into force, representing a major paradigm shift in data protection legislation and a clear tightening of data protection rules.

While organizations strive to adapt their operations to a post-GDPR reality, blockchain technology flourishes and presents itself as the *de facto* solution for distributed and decentralized systems that require an immutable ledger.

This work introduces a innovative solution that aims to diminish the burden resulting from new regulatory demands on all stakeholders. The presented solution allows the data controller to collect a European citizen's consent in accordance to the GDPR and persist proof of said consent on the blockchain. On the other hand, the data subject will be able to express his consent conveniently through his smart-phone and evaluate the data controller's performance. The regulator's role was also contemplated, meaning that he could leverage certain system capabilities specifically designed to gauge the status of the relationships between data subjects and data controllers.

Keywords: GDPR, Informed Consent, Reputation Systems, Blockchain, Ethereum Platform

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Listagens

Lista de Acrónimos

ABI	*Application Binary Interface*
AES	*Advanced Encryption Standard*
API	*Application Programming Interface*
APN	*Android Push Notification*
BIN	Binário
CBC	*Cipher Block Chaining*
CNPD	Comissão Nacional de Proteção de Dados
CTO	*Chief Technology Officer*
Dapp	Aplicação Descentralizada
DEPD	Diretiva Europeia de Proteção de Dados
DSA	*Digital Signature Algorithm*
ECC	*Elliptic Curve Cryptography*
ECDSA	*Elliptic Curve Digital Signature Algorithm*
EUR	Euro
EVM	*Ethereum Virtual Machine*
GCHQ	*Government Communications Headquarters*
Geth	*Go-Ethereum*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hypertext Transfer Protocol Secure*
KDF	*Key Derivation Function*
IBM	*International Business Machines*
IETF	*Internet Engineering Task Force*
IoT	*Internet of things*
IPC	*Inter-Process Communication*
JDK	*Java Development Kit*

JSON	*Javascript Object Notation*
JSON-RPC	*Javascript Object Notation - Remote Procedure Call*
JVM	*Java Virtual Machine*
NEC	*Nippon Electric Company*
NTT	*Nippon Telegraph and Telephone*
SAP	*Systeme Andwendungen und Produkte in der Datenverarbeitung*
SSH	*Secure Shell*
REST	*Representational State Transfer*
RFC	*Request For Commments*
RGPD	Regulamento Geral de Proteção de Dados
RPC	*Remote Procedure Call*
RSA	Rivest-Shamir-Adleman
SHA	*Secure Hash Algorithm*
SHA-3	*Secure Hash Algorithm Version 3*
TLS	*Transport Layer Security*
UE	União Europeia
UML	*Unified Modeling Language*
UTF-8	*8-Bit Unicode Transformation Format*
USD	*United States Dollars*
WAR	*Web Application Resource*

Capítulo 1

Introdução

Na economia digital dos tempos que correm o crescimento económico depende significativamente da partilha de dados a nível global. Esta realidade leva a que organizações com múltiplas operações a este nível pretendam capturar, processar e armazenar dados pessoais fruto da relação digital com os seus clientes tendo em vista a simplificação de processos e obtenção de vantagem competitiva. Ao fazê-lo vêm-se obrigadas a lidar com legislação complexa que varia de acordo com a jurisdição e que, por vezes, é pouco clara [1]. Escândalos recentes como o *Cambridge Analytica*, empresa que entre 2014 e 2018 utilizou dados capturados de pelo menos 87 milhões de utilizadores da rede social *Facebook* para manipular o resultado de eleições nos Estados Unidos e no Reino Unido [2]; ou incidentes como a quebra de dados do gigante do crédito bancário *Equifax*, que em Setembro de 2017 viu dados pessoais de 143 milhões de cidadãos norte-americanos acedidos indevidamente [3]; são apontados pela sociedade civil como evidências inequívocas do uso abusivo e da gestão deficiente dos nossos dados pessoais por parte das organizações a quem os confiámos.

Este tipo de fenómenos, que se têm tornado recorrentes nos últimos anos, não passaram despercebidos à União Europeia (UE) que em Abril de 2016 promulgou o novo Regulamento Geral de Proteção de Dados (RGPD) que veio substituir a agora obsoleta Diretiva Europeia de Proteção de Dados (DEPD) datada de 1995. O RGPD entrou em vigor a 25 de Maio de 2018 e representou uma drástica mudança de paradigma neste tipo de legislação e um claro endurecimento das regras de proteção de dados, isto porque, ao contrário do que acontece com outra legislação semelhante, a sua aplicabilidade é extraterritorial [4]. Deste modo, toda e qualquer entidade, independentemente da sua localização geográfica, deverá atuar em conformidade com o RGPD sempre que em causa esteja a captura, o processamento ou o armazenamento de dados pessoais de cidadãos europeus [5]. Outro fator que contribuiu para a mediatização do RGPD foi o facto das multas por inconformidade poderem atingir os quatro milhões de Euros ou então quatro por cento da faturação anual da organização transgressora, sendo que será sempre aplicado o valor mais elevado. A fiscalização da aplicação do RGPD fica a cargo de uma entidade reguladora[1] que tem plena autonomia para sancionar as organizações incumpridoras consoante previsto no regulamento [5].

No mesmo momento em que todas as organizações, que lidam com dados pessoais de cidadãos europeus, se debatem com necessidades tecnológico-legais prementes impostas pela

[1] Em Portugal a entidade reguladora é a Comissão Nacional de Proteção de Dados [6].

entrada em vigor do RGPD, a tecnologia *blockchain* floresce e afirma-se como a solução *de facto* para cenários em que é fulcral manter uma "fonte de verdade" de um sistema distribuído e descentralizado, de modo a que deixe de ser necessário depositar confiança em intermediários ou autoridades centrais. Esta tecnologia disruptiva, que será dissecada no decorrer deste trabalho, assenta em diferentes algoritmos e princípios criptográficos sólidos que garantem a integridade e rastreabilidade das transações ocorridas numa rede que implemente um protocolo deste tipo [7]. A discussão gerada pelos protocolos *blockchain* rapidamente chamou a atenção de diferentes áreas de atividade que prontamente identificaram oportunidades de negócio proeminentes daí advindas [8].

Indústrias como os serviços financeiros, o imobiliário, os seguros, *a Internet of Things* (IoT) e muitas outras têm vindo a fomentar a investigação e o desenvolvimento da tecnologia *blockchain* para os mais diferentes fins. O intuito destas partes interessadas é incorporar tecnologia *blockchain* nos seus sistemas centrais tendo em vista a satisfação de diversas necessidades organizacionais internas, de negócio e até regulatórias [9]. Prova disso é o projeto *Hyperledger* lançado pela *Linux Foundation* em Dezembro de 2015 com o objetivo de unir esforços em torno de soluções *blockchain* altamente fiáveis e de alto desempenho que sirvam como suporte para transações a nível global entre diferentes organizações do sector tecnológico, financeiro e logístico. Entre os diferentes membros que integram o projeto podemos encontrar reputadas organizações dos sectores referidos como Cisco, *International Business Machines* (IBM), Fujitsu, Hitachi, Intel, *Nippon Electric Company* (NEC), *Nippon Telegraph and Telephone* (NTT), Red Hat, VMWare, Banco ABN AMRO, Banco ANZ, JP Morgan, Wells Fargo, *Systeme Andwendungen und Produkte in der Datenverarbeitung* (SAP), Accenture, Wipro, entre outras [10].

É neste contexto oportuno que este trabalho que se propõe a explorar os mais recentes avanços da tecnologia *blockchain* para conceptualizar um sistema que permita dar resposta ao desafio do consentimento informado levantado pelo RGPD, sistema esse que será concebido desde a sua génese para incluir as entidades reguladoras e tirar o máximo proveito da relação do sujeito dos dados com o seu *smartphone* para gerir o seu consentimento nas diferentes operações em que esteja envolvido.

1.1 Objetivos do Trabalho

O principal objetivo deste trabalho passa pelo estudo e pela aplicação da tecnologia *blockchain* para criar um sistema que permita resolver desafios impostos pelo consentimento informado segundo o RGPD a todas as partes envolvidas (sujeitos dos dados, entidades controladoras de dados e entidades reguladoras). Pretende-se que o sistema a ser criado utilize uma *blockchain* pública para persistir as provas resultantes do consentimento de um sujeito dos dados para com um processamento a ser realizado por um entidade controladora de dados.

Outros dos objetivos passa pela definição de um sistema reputacional que permita ao sujeito dos dados avaliar qualitativamente o comportamento de todas as entidades controladoras de dados a quem deu o seu consentimento para diferentes operações e consultar avaliações idênticas provenientes dos seus pares. O propósito deste sistema reputacional é dar voz e poder ao sujeito dos que é, tradicionalmente, o elo mais fraco da cadeia de processamento de dados e, adicionalmente, servir como barómetro para a intervenção da entidade reguladora.

1.2 Cenário Proposto

Com o intuito de demonstrar as potencialidades do sistema conceptualizado, é proposto um cenário de referência que evidencia as interações dos diferentes atores com os principais componentes do mesmo. Como representado na figura 1.1, o sujeito dos dados, o controlador de dados e a entidade reguladora comunicam com o sistema através de uma REST API que serve como ponto de entrada no *back end* do sistema. Por sua vez, esse *back end*, é responsável por assegurar o cumprimento das regras de negócio e recorrer à *blockchain* de uma rede para persistir os dados referentes ao sistema.

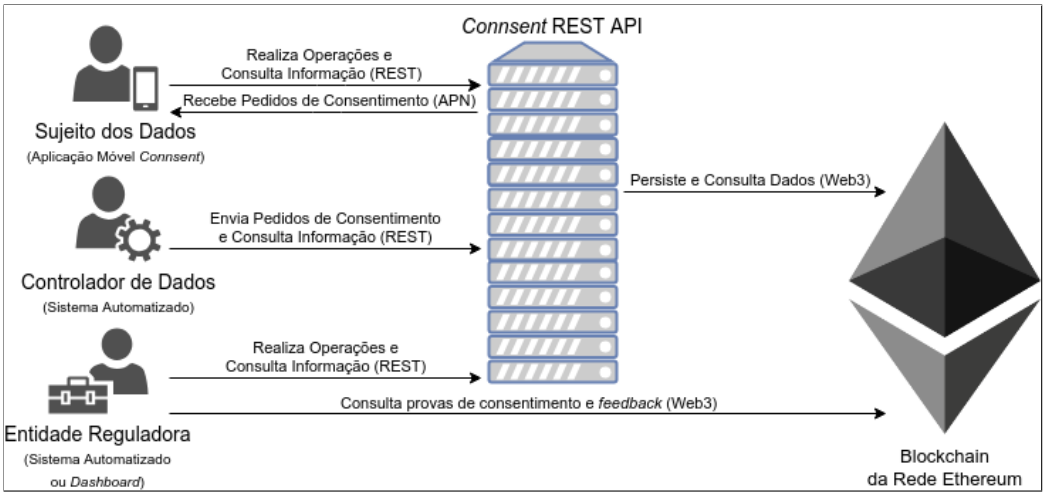

Figura 1.1: Representação do Cenário Proposto

Por oposição, em contextos onde se pretenda obter os dados diretamente da fonte sem qualquer sistema intermediário, está prevista a possibilidade de estabelecer uma ligação direta à *blockchain* de uma rede. Esta circunstância está representada na figura 1.1 pela consulta de provas de consentimento e *feedback* na *blockchain* da rede por parte da entidade reguladora. Embora esta arquitetura seja completamente agnóstica ao método de captura dos dados do sujeito, este cenário prevê este passo seja efetuado através de um inquérito *web*.

1.3 Resultados e Contribuições Relevantes

A combinação de uma tecnologia emergente como a *blockchain* para resolver necessidades levantadas por um regulamento tão recente como o RGPD resulta numa solução inovadora que endereça as necessidades de todas as partes envolvidas. O sistema proposto prevê que sujeito dos dados seja capaz avaliar o desempenho do controlador de dados e de gerir os seus consentimentos de um modo cómodo e imediato; que o controlador de dados tenha acesso a um sistema que lhe permite obter o consentimento do sujeito dos dados persistindo o mesmo em conformidade com o RGPD; e dota a entidade reguladora com uma ferramenta que lhe permite, a qualquer momento, consultar provas de consentimento e aferir qual o estado da relação entre o sujeito dos dados e o controlador de dados.

1.4 Estrutura do Relatório

O presente documento está organizado de modo a que o leitor possa ser gradualmente introduzido aos fundamentos e conceitos considerados essenciais ao entendimento dos capítulos seguintes. No Capítulo 2 aborda-se a temática principal deste trabalho, a tecnologia *blockchain*. No Capítulo 3 o leitor é introduzido ao conceito de consentimento segundo o RGPD e à temática da reputação e dos sistemas reputacionais. No Capítulo 4 apresenta-se o sistema desenvolvido e são detalhados os diferentes componentes que o perfazem. No Capítulo 5 são validados os resultados obtidos da implementação do sistema proposto. Por último, no Capítulo 6, são enunciadas as conclusões retiradas do trabalho efetuado e são propostas melhorias futuras.

Capítulo 2

Blockchain

É impossível abordar o tema *blockchain* sem mencionar o fenómeno das criptomoedas em geral e do *Bitcoin* em particular, tendo em conta que este último é alicerçado pela primeira. Desde que no final do ano de 2017 o valor das criptomoedas atingiu o seu pico máximo até à data, que se estabeleceu um interesse generalizado no tema [11]. As origens desta verdadeira "febre do ouro" [12] remontam ao final do ano 2008 em plena crise dos mercados financeiros, a mesma que resultou na falência do gigante de investimento Lehman Brothers e no escândalo Madoff. Foi neste período economicamente conturbado que Satoshi Nakamoto[1] publicou um artigo com o título *Bitcoin: A Peer-to-Peer Electronic Cash System*. Este artigo especificava um sistema descentralizado (*peer-to-peer*) de dinheiro eletrónico. Poucos meses depois, em Janeiro de 2009, Satoshi Nakamoto criou o primeiro registo na base de dados da rede *Bitcoin* e tornou pública a primeira versão da aplicação cliente que tinha vindo a desenvolver em colaboração com a comunidade *open source* que se gerou em torno do projeto. Estavam criadas as condições para começarem a ser mineradas as primeiras *bitcoins*. Desde então, a plataforma *Bitcoin* tem vindo a gozar de uma visibilidade e importância crescente, tendo também servido como estímulo para a criação de outras criptomoedas com características idênticas como, por exemplo, o *Ethereum*, o *Dash*, o IOTA, o *Ripple* e o *Litecoin* [14]. A figura 2.1 ilustra a valorização das criptomoedas referidas, fruto da procura crescente, até ao final do ano de 2017. Enquanto investidores particulares de diferentes *backgrounds* académicos e profissionais exploraram eventuais oportunidades de investimento proporcionadas por estas criptomoedas, a indústria da tecnologia ficou fascinada pelas possíveis aplicações práticas da tecnologia *blockchain* [9].

Blockchain é a designação genérica dada aos protocolos de persistência de transações que suportam esta recente vaga de criptomoedas. As diferentes implementações deste tipo de protocolo fundamentam-se em diferentes algoritmos e princípios criptográficos para assegurar a integridade e a rastreabilidade de todas as transações ocorridas no sistema, sem necessidade de depositar confiança numa entidade central, mantendo-o, portanto, descentralizado e distribuído [15].

O feito mais notável da primeira implementação de blockchain, na rede *Bitcoin*, é ter sido capaz de resolver o problema do *double spending*, problema esse que até à emergência

[1] À data de escrita deste trabalho todo o processo de conceptualização da plataforma *Bitcoin* encontra-se envolto em mistério, dado que a identidade da pessoa, ou grupo de pessoas, por trás do pseudónimo Satoshi Nakamoto permanece incógnita [13].

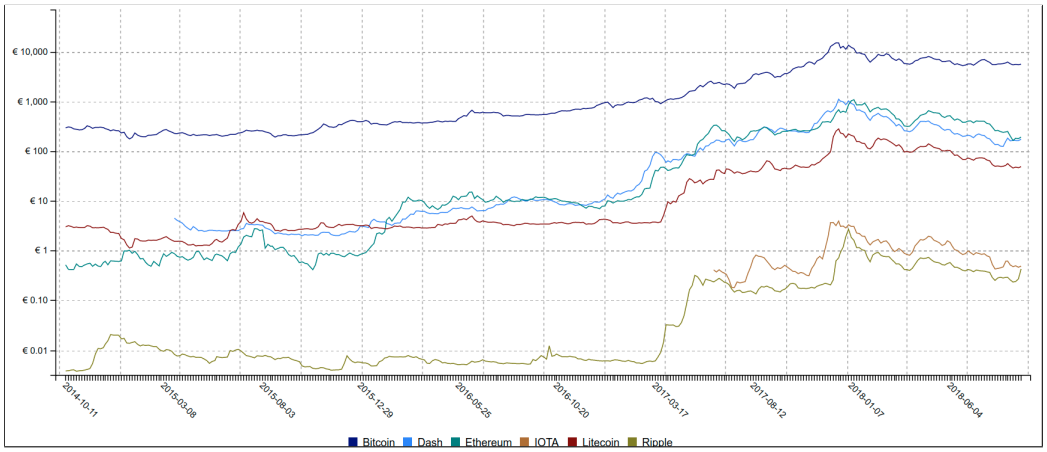

Figura 2.1: Evolução do valor da unidade (em Euro) das criptomoedas referidas (escala logarítmica)

tecnologia *blockchain* ainda não havia sido ainda resolvido sem recurso a qualquer forma centralização. Como em sistemas com uma arquitetura completamente descentralizada não existe nenhuma base de dados centralizada, é particularmente crítico o recurso a mecanismos que garantam a sincronização de todas as bases de dados distribuídas, evitando deste modo que versões desatualizadas ou membros maliciosos deixem a rede num estado inconsistente [16].

Este problema é conhecido como "O Problema dos Generais Bizantinos" e pode ser formulado do seguinte modo: imagine-se que inúmeras divisões do exército bizantino, cada uma delas comandada pelo seu general, estão estacionadas à porta de uma cidade inimiga; os generais apenas podem comunicar entre si através de um mensageiro. Após uma observação detalhada do inimigo os generais têm que chegar a um consenso sobre qual o plano de ação a ser posto em prática. Contudo, deverão ser cautelosos porque não está descartada a possibilidade de alguns dos generais serem traidores e tentarem impedir que os generais leais alcancem um entendimento favorável à causa bizantina. Para evitar que tal aconteça, os generais deverão encontrar um algoritmo que garanta, em primeiro lugar, que todos os generais leais decidam tomar o mesmo plano de ação e que, em segundo lugar, os generais traidores não sejam capazes de ludibriar os generais leais levando-os a optar por uma plano de ação desfavorável. À semelhança do que acontece com os generais bizantinos leais, os membros honestos da rede têm que chegar a um consenso sobre qual é o estado válido da *blockchain*, impedindo assim que os membros maliciosos lhes imponham uma versão ilegítima da mesma [17].

2.1 Princípios Criptográficos e de Codificação de Dados

A par de algoritmos criptográficos que permitem garantir a integridade, confidencialidade e autenticidade na comunicação entre entidades distintas, os esquemas de codificação de dados permitem a representação de um conjunto de dados no formato binário de uma forma mais compacta e comum.

2.1.1 Esquemas de Codificação de Dados

Esquemas de codificação de dados são utilizados frequentemente para divulgar informação binária através de meios de transmissão que lidam exclusivamente com texto. Os dois esquemas mais populares são o Hexadecimal e o Base64. O esquema de codificação de dados Hexadecimal permite codificar a informação contida em 4 *bits* (dígitos binários) num único símbolo do sistema de numeração, de raiz 16, hexadecimal (composto pelos símbolos 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E e F). O esquema Base64; assim designado por ser constituído por 64 valores representados pelos símbolos [A-Z], [a-z], [0-9], "/" e "+"; possibilita a codificação da informação contida em 6 *bits* num único símbolo. Observemos então, a título de exemplo, a tabela 2.1 onde a representação binária da mensagem "Criptografia é fixe!" em UTF-8 é codificada para ambos os esquemas de codificação de dados referidos.

Mensagem	Criptografia é fixe!
Binário (UTF-8)	01000011 01110010 01101001 01110000 01110100 01101111 01100111 01110010 01100001 01100110 01101001 01100001 00100000 11000011 10101001 00100000 01100110 01101001 01111000 01100101 00100001
Hexadecimal	43726970746f677261666 96120c3a9206669786521
Base64	Q3JpcHRvZ3JhZmlhIMOpIGZpeGUh

Tabela 2.1: Exemplo da aplicação de codificação em Hexadecimal e Base64

2.1.2 Funções de Resumo Criptográfico

Uma função de resumo criptográfico permite mapear conjuntos dados de diferentes tamanhos num conjunto de dados de saída de tamanho fixo. Este conjunto de dados de saída é designado por resumo criptográfico, soma criptográfica, valor de *hash*, código de *hash*, ou simplesmente *hash*. A origem do termo remonta à década de 50 do século passado e é atribuída a Hans Peter Luhn que à altura era funcionário da IBM [18]. As funções que implementam este tipo de algoritmos recebem como *input* um conjunto de dados que, após processados, retornam como *output* o resumo criptográfico dos dados de *input*. São especialmente úteis quando se pretende identificar diferentes conjuntos de dados rapidamente ou garantir a integridade de dados transmitidos por rede, tendo em conta que o mesmo conjunto de dados depois de processado pela mesma função resultará sempre no mesmo resumo criptográfico. Quando diferentes conjuntos de dados de *input* retornam o mesmo resumo criptográfico diz-se que ocorre uma colisão, algo que é inevitável tendo em conta que se está a mapear um domínio infinito para uma contradomínio finito. As colisões nos algoritmos de *hashing* mais avançadas são raras e causadas por conjuntos de dados extremamente distintos. Ao contrário de outras técnicas de criptográficas, o *hashing*

não tem como objetivo proteger informação de terceiros, pelo que não requer a utilização de qualquer tipo de chave. O cálculo de um resumo criptográfico requer que o sistema que realiza o cálculo despenda do poder computacional necessário para que todo o conteúdo binário a ser resumido seja processado pelo algoritmo. A rede *Bitcoin*, por exemplo, recorre ao algoritmo SHA-256, onde SHA é o nome do algoritmo (*Secure Hash Algorithm*) e o número 256 representa o tamanho em *bits* do resumo criptográfico resultante da sua aplicação [16]. A rede *Ethereum*, por sua vez, recorre ao algoritmo SHA3-256 (KECCAK), em parte por ser uma implementação mais recente que a rede *Bitcoin* [19]. Os algoritmos utilizados por ambas as redes permitem mapear qualquer conjunto de dados binário para um universo de 2^{256} combinações possíveis.

2.1.3 Criptografia de Chave Pública

O conceito de criptografia de chave pública, também conhecida por criptografia assimétrica, é relativamente recente, ao contrário do que acontece com o conceito de criptografia simétrica que já existe há pelos menos 4000 anos [20]. Apesar de ter sido proposto pela primeira vez por Whitfield Diffie, Martin Hellman e Ralph Merkle em 1976, documentos desclassificados pelo governo britânico em 1997 revelaram que os investigadores do GCHQ James Ellis, Cliffords Cocks e Graham Williamson já aplicavam os mesmo conceitos desde, pelo menos, 1973 [21]. Enquanto no conceito de criptografia simétrica existe uma única chave secreta que é utilizada para cifrar e decifrar a informação, em criptografia assimétrica existe um par de chaves, composto por uma chave pública e uma chave privada. Este par de chaves é gerado em conjunto, sendo que é computacionalmente inviável deduzir a chave privada através da chave pública em tempo útil. Esta abordagem permite que a troca de chaves públicas possa ser realizada através de canais inseguros sem que isso comprometa a autenticidade, confidencialidade e integridade da informação a ser protegida por sistemas criptográficos deste tipo, desde que se mantenha a confidencialidade das chaves privadas. Existem dois tipos de aplicações principais para este tipo de criptografia: o de cifra e o de assinatura digital. O de cifra garante que informação cifrada com recurso a uma chave pública é apenas decifrável pela respetiva chave privada, permitindo assim o transporte seguro de informação confidencial. Já o de assinatura digital permite assegurar, detendo a chave pública, que determinada informação provem, de facto, do detentor da chave privada (não repúdio) e que não foi adulterada, tendo em conta que foi assinada com a chave privada.

2.1.4 Criptografia de Curvas Elípticas

A introdução da criptografia de curvas elípticas (ECC) no ano de 1985 revolucionou o modo como a ciência da computação abordava a criptografia de chave pública, tendo em conta que este tipo de criptografia é mais poderoso e eficiente que os esquemas tradicionais propostos por Diffie-Hellman e pelo trio Rivest-Shamir-Adleman (RSA) [22]. Uma boa prova da sua eficiência é o facto de uma chave ECC de 256 *bits* garantir um nível de proteção superior ao proporcionado por uma chave RSA de 4096 *bits* [22]. Esta diferença considerável fica a dever-se ao facto do conceito matemático de curvas elípticas, que suporta este tipo de criptografia, ser diferente do conceito matemático de fatorizarão do produto de dois números primos grandes, que serve como suporte ao sistema criptográfico RSA [22]. Apesar destas características, a padronização dos sistemas criptográficos de curvas elípticas ocorreu apenas por volta do ano 2000 [22]. Sistemas e aplicações desenvolvidas de raiz mais recentemente tem optado por este tipo de criptografia que relaciona

dois pontos numa curva elíptica matemática [22]. Exemplo disto são as plataformas *Bitcoin* e *Ethereum* que recorrem a este tipo de criptografia para satisfazer todas as suas necessidades de criptografia de chave pública [19].

O algoritmo de assinatura padrão quando se recorre à criptografia de curvas elípticas, é o *Elliptic Curve Digital Signature Algorithm* (ECDSA). Este algoritmo, desde que começou a ser suportado por múltiplas implementações dos protocolos TLS e SSH, tem vindo a substituir o algoritmo de assinatura RSA e o clássico *Digital Signature Algorithm* (DSA) [22]. À semelhança do que acontece com os demais esquemas de assinatura digital, para assinar uma determinada mensagem com o algoritmo ECDSA é necessário, em primeiro lugar, computar o resumo de criptográfico da mesma. Para tal recorre-se a uma função de resumo criptográfico como, por exemplo, o SHA-256. Procede-se, de seguida, à assinatura digital do resumo criptográfico da mensagem com a chave privada que dará origem a um valor que representa a assinatura. Por oposição, sempre que se pretenda verificar a assinatura leva-se a cabo o processo inverso, que consiste em calcular novamente o resumo criptográfico da mensagem. Este resumo criptográfico, a par da chave pública e do valor da assinatura, servirá de *input* para a função de verificação de assinatura. Garantindo que a integridade da mensagem não foi afetada, algo que é assegurado pelo resumo criptográfico, e que o valor da assinatura digital proveio da assinatura do resumo criptográfico com a chave privada associada à chave pública utilizada para a verificação; a função de verificação retornará um estado de sucesso.

2.2 Protocolo *Blockchain*

Apesar de ser uma das tecnologias mais disruptivas dos últimos tempos, o conceito de cadeia de blocos tem vindo a ser explorado desde meados da década de 70 do século passado pela ciência computacional [23]. Em 1976 William F. Ehrsam, Carl H. W. Meyer, John L. Smith e Walter L. Tuchman registaram a patente US4074066A que especifica, entre outros, um sistema criptográfico de cifra simétrica designado por "*Cipher Block Chaining*" (CBC) onde o output de uma operação sob um bloco serve como *input* para uma operação do mesmo tipo sob o bloco seguinte [24]. Em 1991 Stuart Haber e W. Scott Stornetta implementaram um sistema criptográfico para assegurar que a assinatura temporal dos ficheiros de um determinado sistema não seria adulterada. Nesse sistema cada bloco referente a um único documento relaciona-se com o que o antecede. Em 1992 os mesmos autores incluíram suporte para árvores de Merkle, permitindo assim que mais que um documento seja referido em cada bloco [25]. O conceito de *blockchain*, tal como o conhecemos hoje, foi tornado público em 2008 por Satoshi Nakamoto e a sua primeira implementação foi incluída no núcleo da versão inicial da plataforma *Bitcoin*, onde serve como livro de razão de todas as transações ocorridas na rede [26].

Para garantir a integridade da informação e, consequentemente, de todas transações efetuadas na rede, os protocolos *blockchain* recorrem ao conceito de blocos e a um protocolo de consenso. Na sua essência, um bloco é composto por um valor numérico aleatório (*nonce*), pelo resumo criptográfico do bloco que o antecede e pela informação das transações contidas no novo bloco [13]. O resumo criptográfico do próprio bloco não é mais que um mero resumo criptográfico do conjunto de dados que perfaz o cabeçalho do bloco. Deste modo, o bloco n irá incluir o resumo criptográfico do bloco que o antecede (o bloco *n-1*). Por sua vez, o bloco *n+1* referirá o bloco que o antecede, que neste exemplo é o bloco n. É deste *design*, representado na figura 2.2, que surge a designação *blockchain* (corrente de blocos) e que fica garantida a imutabilidade de um estado anterior

da *blockchain*.

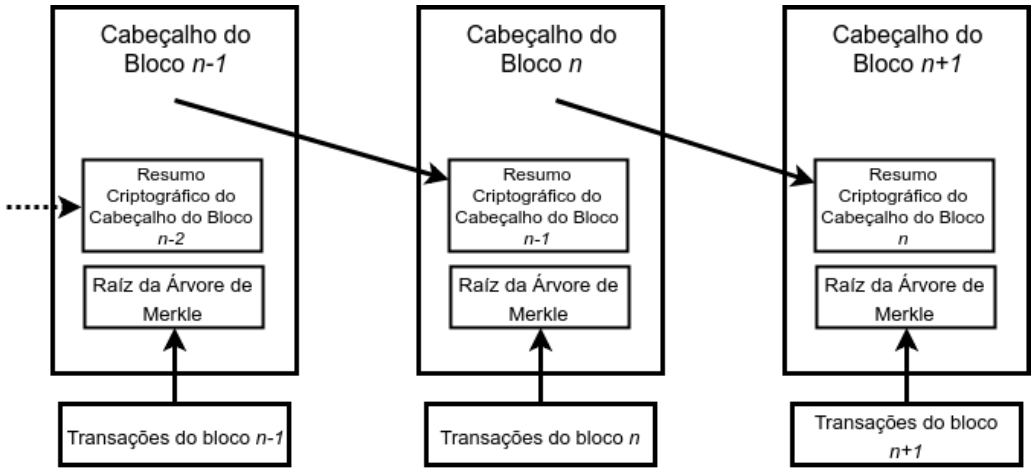

Figura 2.2: Exemplo simplificado do encadeamento de blocos no protocolo *blockchain*

Assim que surge a necessidade de persistir uma nova transação numa rede que implemente o protocolo *blockchain* é utilizado o próximo bloco disponível para albergar a informação referente à mesma. Numa *blockchain* com um protocolo de consenso do tipo *proof-of-work*, de que é exemplo o *Bitcoin*, para computar novos blocos recorre-se a um processo designado por mineração. Ao longo deste processo, os diferentes elementos da rede competem por ser o primeiro a minerar um novo bloco, obtendo algum tipo de recompensa em troca. No caso particular do *Bitcoin* o membro da rede que minerou o novo bloco recebe uma comissão de cada transação que lá será armazenada. Sempre que um membro da rede anuncia que foi o primeiro a minerar o próximo bloco, tira partido de um sistema criptográfico de *proof-of-work* para provar perante os demais membros dessa mesma rede que efetuou os cálculos computacionais necessário para tal [13]. Apesar de existirem diferentes implementações deste tipo de protocolo para as mais diversas finalidades, Satoshi Nakamoto recorreu, na implementação de *blockchain* da plataforma *Bitcoin*, ao sistema *Hashcash*. Este sistema foi inicialmente proposto por Adam Back, em 1997, como uma medida contra o uso abusivo de recursos não controlados existentes na Internet, como por exemplo o *e-mail*. O sistema *Hashcash* permite, através da invocação de uma função computacionalmente dispendiosa, calcular um *token* que permite ao seu portador fazer prova perante os seus pares que efetuou um cálculo computacional de determinada complexidade. Uma das principais características deste tipo de protocolos é a sua assimetria, ou seja, enquanto o algoritmo de cálculo de um *token* é necessariamente dispendioso computacionalmente, o processo de verificação de um *token* do mesmo tipo é pouco dispendioso computacionalmente. Estas características, que por sinal são bastante idênticas às de uma infraestrutura de chave pública, são particularmente relevantes para a integridade e o desempenho da rede tendo em conta que um *token* será computado uma única vez (pelo membro da rede que o originou) mas, em contrapartida, será validado pelo menos uma vez por cada um dos restantes membros da rede [27]. A mineração de cada novo bloco é um fenómeno favorável para a integridade da rede, tendo em conta que o protocolo de consenso de uma rede *blockchain proof-of-work* especifica que a versão válida da *blockchain* é sempre a mais longa [28].

Para adulterar o estado de uma *blockchain* que implemente um protocolo de consenso *proof-of-work*, um membro malicioso teria que computar o *token* de *proof-of-work* do bloco adulterado, os novos *tokens* de *proof-of-work* de todos os blocos subsequentes e, ainda, um novo bloco num intervalo de tempo inferior ao que levaria aos elementos honestos a computar um novo bloco. Neste tipo de ataque, conhecido como "o ataque dos 50%+1", um membro malicioso (ou grupo de membros maliciosos) com mais capacidade de processamento que todos os restantes membros da rede combinados seria capaz de impor uma versão privada da *blockchain* como válida [29]. Neste cenário o membro malicioso da rede começaria a minerar um *fork* privado da *blockchain* que não iria ser emitido para a rede enquanto, ao mesmo tempo, realizava múltiplas transações na versão tida como sendo a válida pelos demais membros da rede. Assim que tivesse levado a cabo todas as transações desejadas, o membro malicioso começaria então a emitir na rede a sua versão privada da *blockchain* que é mais longa que a conhecida pelos restantes membros honestos. Como seria de esperar, na versão privada do membro malicioso não constariam as suas transações realizadas na versão conhecida pelos membros honestos. Como esta versão seria mais longa do que a versão legítima, seria tida como válida pelos membros honestos da rede. Tal permitiria que o membro malicioso fosse capaz de fazer *double spending*, ou seja, realizar transações que futuramente não seriam consideradas como válidas. Entretanto, o membro malicioso teria usufruído das contrapartidas resultantes das transações realizadas enquanto a versão da *blockchain* que as continha fosse a versão consensual da rede [30]. À data de escrita deste artigo, criptomoedas com menos visibilidade como o *Krypton* e o *Feathercoin* já foram alvo de ataques deste tipo bem sucedidos [31, 32]. Ainda assim, este tipo de ataque é extremamente difícil de comportar computacionalmente, especialmente se a dificuldade de mineração de novos blocos da *blockchain* for ajustada à capacidade computacional contemporânea. Na rede *Bitcoin*, por exemplo, a dificuldade da mineração de um novo bloco é definida pela quantidade de zeros à esquerda do resumo criptográfico (em representação hexadecimal) resultante do *nonce* do novo bloco e da restante informação referente ao bloco conhecida *a priori* [16]. A evolução positiva

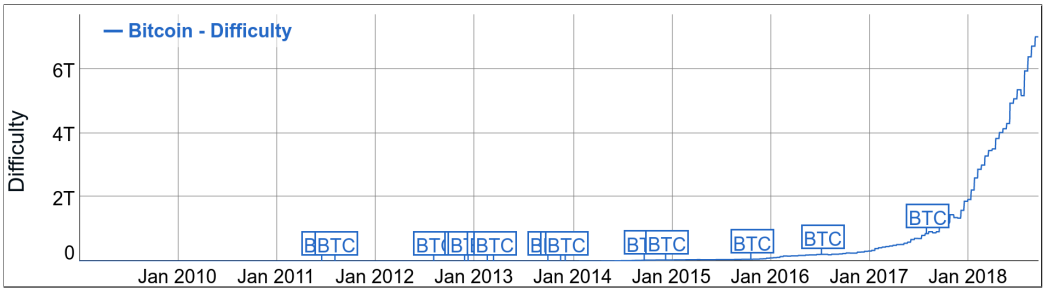

Figura 2.3: Evolução do poder computacional necessário (em *Terahash* para minerar um novo bloco na rede *Bitcoin* ao longo do tempo)

da dificuldade computacional necessária à mineração de *bitcoins*, apresentada na figura 2.3, foi conseguida através da incrementação gradual do número de zeros à esquerda que constam no resumo criptográfico dos novos blocos. Deste modo, a quantidade de zeros iniciais do resumo criptográfico de bloco está diretamente relacionada com a dificuldade computacional necessária à sua computação [16]. A título de exemplo, o bloco número 542298 da *blockchain* da rede *Bitcoin* minerado a 20 de Setembro de 2018 tem o resumo criptográfico "000000000000000000cacc22909e6b3ed521360818e2bbd1a73270c3badaf2",

ou seja contém vinte zeros à esquerda. Assim, e tendo em conta que a mineração de cada bloco é altamente dispendiosa computacionalmente e que, segundo o protocolo, perante diferentes versões existentes na rede, a *blockchain* válida é sempre a mais longa; rapidamente se conclui que a adulteração de um bloco numa *blockchain* com um protocolo de consenso *proof-of-work* é computacionalmente inviável. Deste modo, e garantindo que os membros honestos da rede detêm pelo menos 50%+1 do poder computacional da mesma, ficam salvaguardadas situações de *double spending* [16].

2.2.1 *Blockchain* 2.0

Blockchain 2.0 é a designação atribuída à nova geração de implementações de protocolos *blockchain* concebidos desde a sua génese para suportar a definição de regras de negócio e validações personalizadas por meio de *smart contracts* [33]. Sucede à linhagem anterior de protocolos *blockchain* designada por *blockchain* 1.0, cuja implementação se restringia a garantir que um conjunto pré-definido de validações eram respeitadas [33]. Esta estirpe mais recente de implementações do protocolo *blockchain* surgiu em resposta à crescente procura da indústria, que ansiava por uma *framework* que lhes permitisse explorar em pleno todas as potencialidades desta tecnologia para os mais diferentes fins. Nos protocolos *blockchain* 1.0, o protocolo de consenso utilizado é o do *proof-of-work* onde, como referido anteriormente, o consenso entre os membros da rede é estabelecido através da mineração de *tokens* que servem como prova da realização de um determinado esforço computacional a ser recompensado pela rede [26]. Já nas diferentes implementações de protocolos *blockchain* 2.0, foi incluído suporte para novos protocolos de consenso de que são exemplo o de *proof-of-stake* e o de *proof-of-authority* [7].

O protocolo de consenso *proof-of-stake* é particularmente utilizado em *blockchains* que servem como sustentação para criptomoedas, uma vez que tem como premissa que quanto mais um *stakeholder* estiver investido na rede menor será o seu interesse em corromper o processo de validação de transações ocorridas na mesma [34]. Neste protocolo de consenso os membros da rede incumbidos de validar as transações ocorridas na rede são conhecidos por *minters*, e antes que o possam fazer têm que proceder, necessariamente, ao depósito de uma espécie de caução como garantia. Deste modo, o algoritmo de escalonamento de *minters* atribuir-lhes-á validações de acordo com a proporção da caução [35]. A título de exemplo, a um *minter* que tinha uma *stake* de 10% será atribuída a validação de, aproximadamente, 10% das transações ocorridas na rede. Quando comparado com o protocolo de consenso *proof-of-work*, o protocolo *proof-of-stake* tem como principais vantagens o facto de não consumir grandes quantidades de energia elétrica[2], o de desincentivar práticas de cartel na rede, o de reduzir riscos de centralização do poder computacional da rede, e o de tornar ataques de maioria (50%+1) bastante mais dispendiosos [37].

Com o protocolo de consenso *proof-of-authority*, também conhecido por *identity at stake*, as transações ocorridas são validadas automaticamente, por *software*, por membros da rede com autoridade para tal que são conhecidos como validadores ou autoridades. Para ser tornar um validador, um membro da rede deverá ter a sua identidade verificada *on-chain*[3] e, de preferência, verificável também através de um domínio de acesso público. Deste modo, na eventualidade de um validador tentar comprometer a integridade da *blockchain* da rede, a sua identidade é conhecida e a sua reputação sairá afetada. Neste

[2]Estima-se que, em Agosto de 2018, a mineração na rede *Bitcoin* já representa 1% do gasto energético a nível mundial [36].

[3]Persistida dentro da própria *blockchain*.

protocolo de consenso é também muito importante a uniformização dos procedimentos de verificação para a atribuição deste estatuto. O facto do estatuto de validador ser difícil de obter faz com que o direito de validar transações seja merecido e estimado. Em *blockchains* que implementam este protocolo de consenso não é necessário recorrer à mineração de novos blocos, como acontece no protocolo *proof-of-work*, uma vez que todas as transações serão incluídas nos novos blocos pelos validadores. A esse tipo de poder computacional foi atribuída a designação de poder de manutenção.

2.2.2 *Smart Contracts*

Apesar do protocolo *blockchain* garantir a integridade das transações efetuadas, não garante que as mesmas obedeçam a um conjunto de regras de negócio *out of the box*. Recorrendo mais uma vez ao exemplo da plataforma *Bitcoin*, de modo a que um valor seja transferido de uma carteira para outra, foi necessário implementar lógica que permitisse assegurar que a carteira de origem tem fundos suficientes antes proceder à creditação da carteira de destino, salvaguardando deste modo situações de *double spending* [16]. Esta lógica está implementada no núcleo da plataforma e não é passível de ser alterada pelos membros da rede. Posteriormente, fruto da popularidade da plataforma *Bitcoin*, surgiu a necessidade de implementar uma segunda camada de regras de negócio que permitisse aos membros da rede definir regras personalizadas para a execução de uma transação. Para o efeito a *blockchain* da rede *Bitcoin* implementou a *Script*, uma linguagem de *scripting* que é, por motivos de segurança, bastante limitada pelo facto de não ser computacionalmente universal e de não suportar ciclos de execução [26].

```
1 2 3 OP_ADD 2 OP_MUL 1 OP_ADD 11 OP_EQUAL
```

Listagem 2.1: Exemplo da definição em linguagem *Script* de um conjunto de operações aritméticas sob um conjunto valores

A título de exemplo, o código fonte *Script* apresentado na listagem 2.1 define um conjunto de instruções matemáticas e valores que, aquando da execução, são interpretados da esquerda para a direita e o seu estado é persistido numa pilha (estrutura de dados). Na rede *Bitcoin* a linguagem *Script* é maioritariamente utilizada para definir um conjunto de circunstâncias perante as quais uma transação será concluída com sucesso [26]. A este tipo de regras de negócio e validações executadas por meio de uma transação digital foi atribuída a designação de *smart contract*. Embora o conceito tenha sido inicialmente proposto por Nick Szabo em 1994, como sendo "uma nova forma de formalizar os relacionamentos digitais institucionais de um modo mais inteligente e funcional que os tradicionais contratos em papel", não foi até ao interesse da indústria da tecnologia na *blockchain* que o mesmo ganhou particular notoriedade [38, 39]. Enquanto implementações do protocolo *blockchain* de âmbito mais restrito, como a da rede *Bitcoin*, recorrem a mecanismos de *smart contracts* bastante limitativos, outras, como a da *blockchain* da plataforma *Ethereum*, foram concebidas para suportar mecanismos genéricos que permitem a programadores externos explorar outras oportunidades proporcionadas por esta tecnologia através da definição de regras de negócio e validações ajustadas às suas necessidades por meio de *smart contracts* [15].

2.3 Plataforma *Ethereum*

A plataforma *Ethereum* foi fundada pelo principal precursor da segunda geração de *blockchains*, o jovem programador russo-canadiano Vitalik Buterin. No inicio do ano de 2014, quando tinha apenas 19 anos, Buterin tornou pública a sua intenção de criar uma nova criptomoeda chamada *Ethereum* que seria suportada por uma *blockchain* homónima [40]. O mesmo referiu, no fórum do projeto, que o nome *Ethereum* foi escolhido por si depois de ter consultado uma lista de elementos químicos ficcionais [41]. A sua visão foi financiada com sucesso através de uma campanha de *crowdfunding*, que decorreu durante o verão de 2014, onde foi feita a pré-venda das primeiras 11.900.000 moedas. A primeira versão do sistema foi lançada a 30 de Julho de 2015 com os *ethers* vendidos durante o processo de *crowdfunding* pré-minerados [42]. Desde o seu lançamento que a plataforma *Ethereum* se tornou referência para programadores que pretendem explorar a tecnologia *blockchain* para o desenvolvimento de aplicações descentralizadas (Dapps) [43]. À data de escrita desde trabalho o *ether*, cujas diferentes unidade de quantificação são apresentadas na tabela 2.2, tem-se mantido consistentemente como a segunda criptomoeda mais valiosa, apenas atrás do *bitcoin*, tendo a sua cotação máxima desde no inicio do ano de 2018 atingido os 633,95 EUR a 1 de Janeiro e a mínima de 148,22 EUR a 19 de Novembro.

Unidade	**Valor de uma unidade *ether***
Tether	0.000000000001
Gether	0.000000001
Mether	0.000001
Kether, Grand, Einstein	0.001
Ether	1
Finney, Milliether, Milli	1000
Szabo, Microether, Micro	1000000
Gwei, Shannon, Nanoether, Nano	1000000000
Mwei, Babbage, Picoether	1000000000000
Kwei, Ada, Femtoether	1000000000000000
Wei	1000000000000000000

Tabela 2.2: Unidades de quantificação da moeda *ether* (por ordem decrescente de grandeza)

2.3.1 *Gas*

Na plataforma *Ethereum* o conceito de *gas* (combustível em português) é utilizado para quantificar o esforço computacional necessário para realizar um determinado conjunto de instruções computacionais no sistema de um membro da rede. É com base nesta unidade que os membros de uma rede *Ethereum* são recompensados pelo esforço computacional realizado pelo seu sistema para computar uma transação. Este modelo de recompensa é apenas possível graças à existência da *Ethereum Virtual Machine* (EVM) e à natureza

determinística[4] de todos os programas em si executados. Estas características garantem que todos os cálculos computacionais executados na EVM gastam a mesma quantidade de recursos em todas as plataformas de *hardware* [19].

Sempre que um membro de rede pretende levar a cabo uma transação deverá especificar, à partida, qual é o limite máximo de *gas* que está disposto a pagar à rede pelo seu processamento e o valor que está disposto a pagar por unidade de gás. O membro deverá assegurar-se que o valor que definiu é superior ao que é necessário para concluir o processamento da transação, sob pena de ver a mesma ser revertida por falta de *gas*. Neste cenário eventual, o membro que ordenou a transação não reaveria parte do valor que pagou à rede pelo *gas*, porque apesar da transação ter sido revertida a rede despendeu, ainda assim, do esforço computacional necessário para computar a transação até ao ponto de falha. Por oposição, ser-lhe-ia devolvida a diferença na eventualidade do valor máximo de *gas* definido para a computação da sua transação ser superior ao valor que de facto foi necessário. Cada transação na rede *Ethereum* requer, no mínimo, 21.000 unidades de *gas*. Já o limite máximo de *gas* por transação está associado à capacidade máxima de um bloco no momento de processamento da transação. Juntamente com o limite máximo de *gas*, o membro da rede deverá também especificar qual o valor que está disposto a pagar à rede por unidade de *gas*. Quanto maior for o valor oferecido pelo membro da rede por unidade de *gas* mais rapidamente a sua transação será confirmada, uma vez que, como a contrapartida oferecida é maior, a rede dar-lhe-á prioridade. A relação entre o custo da unidade de *gas* e o tempo de espera até a transação estar persistida varia de acordo com o estado da rede no momento da sua execução. Face a esta imprevisibilidade existem serviços, como o *ETH Gas Station*, que através da análise de dados recentes da rede conseguem indicar com alguma precisão qual o valor de *gas* que deverá ser especificado para que a transação seja confirmada num determinado intervalo de tempo. Assim, o custo total de uma transação numa rede *Ethereum* pode ser determinado pela fórmula em baixo[19].

$$\text{Custo Total da Transação} = \text{Gas Necessário} \times \text{Custo da Unidade de Gas}$$

2.3.2 *Ethereum Smart Contracts*

Ao contrário do que acontece na plataforma *Bitcoin*, todas as computações na rede *Ethereum* ocorrem numa instância da máquina virtual *Ethereum* que é computacionalmente universal (*Turing-complete*). Isto significa que a rede *Ethereum* é capaz de executar o *bytecode* necessário para resolver qualquer tipo de problema [19]. Esta arquitetura de execução serve como fundação para a definição de *smart contracts* em diferentes linguagens de alto nível que, posteriormente, serão compilados para *bytecode* e executados em todos os nós membros da rede [44]. A linguagem de alto nível mais popular para a definição de *smart contracts* para a rede *Ethereum* é a *Solidity*. Esta *framework* de desenvolvimento para a plataforma *Ethereum* abriu portas a um universo de Dapps que recorrem à *blockchain* de diferentes redes *Ethereum* como livro de razão para a sua operação [44].

Na rede *Ethereum* o *deploy* de um *smart contract* dá origem a um tipo especial de conta, designada por *contract account*. Este tipo especial de estrutura de dados, representada na figura 2.4, armazena um contador de transações efetuadas no contexto do

[4]Diz-se que um algoritmo é determinístico quando da sua execução com os mesmos *inputs* resultam sempre nos mesmos *outputs* e a máquina responsável pelo processamento percorre sempre a mesma sequência de estados.

smart contract, o saldo da *contract account*, o *bytecode* que especifica toda lógica do *smart contract* e ainda o espaço de armazenamento associado ao *smart contract*.

Figura 2.4: Representação simplificada da estrutura de uma *contract account* na plataforma *Ethereum*

Daí em diante todas as transações associadas a esse *smart contract* ficarão armazenadas nessa conta e serão executadas por todos os nós membros da rede [40]. Para recompensar a rede pelo esforço computacional despendido para o *deploy* de um novo *smart contract*, e eventuais transações a ele associadas, será deduzida da carteira ordenante uma quantidade de créditos proporcional à dificuldade computacional da execução da transação. Deste modo, fica claro que as validações e regras de negócio implementadas na lógica dos *smart contracts* deverão restringir-se ao estritamente essencial, sob pena das transações se tornarem altamente dispendiosas ou até ultrapassaram o limite máximo de *gas* por transação imposto pela rede, impedindo que a mesma seja concluída com sucesso. Por outro lado, operações do tipo *view* não implicam uma recompensa à rede, uma vez que não alteram o seu estado, limitando-se a ler dados do espaço de armazenamento da *contract account*. Uma das características mais relevantes da implementação de *smart contracts* da rede *Ethereum* é o suporte para emissão de eventos na rede que, como veremos em detalhe posteriormente, são particularmente úteis para observar mudanças de estado no contexto da conta de um *smart contract*.

2.3.3 Linguagem *Solidity*

A linguagem *Solidity* é, neste momento, a mais popular linguagem de definição de *smart contracts* [45]. Foi proposta em Agosto de 2014 pelo co-fundador, e atual CTO do projeto *Ethereum*, Gavin Wood, e trata-se de uma linguagem computacionalmente universal orientada a contractos, estaticamente tipada e cuja sintaxe tem inúmeras semelhanças com as linguagens C e *Javascript* [46, 45]. A *Solidity* oferece suporte para variáveis de estado, funções, modificadores de funções, definição de eventos e enumerações. No que concerne às estruturas de dados, a linguagem disponibiliza estruturas tipadas, vetores de dados e mapeamentos. As estruturas tipadas permitem definir novos tipos de dados através da combinação de tipos já existentes. Os vetores, por sua vez, permitem armazenar múltiplas variáveis do mesmo tipo [47]. Para último ficou a estrutura de dados mais complexa e poderosa da linguagem, os mapeamentos. Esta estrutura de dados apresenta algumas semelhanças com o conceito tradicional de *hashtable*, uma vez que permite obter

um valor de um determinado tipo previamente armazenado e identificado por uma chave que é alvo de um resumo criptográfico (SHA3-256 neste caso). As semelhanças terminam aqui, uma vez que nos mapeamentos da linguagem *Solidity* o universo possível de chaves é virtualmente inicializado, no momento da sua criação, com o valor pré-definido do tipo de dados a armazenar. Ou seja, após a inicialização de um mapeamento em que as chaves são do tipo *string* e os valores do tipo *uint*, a consulta por qualquer chave retornaria sempre o valor 0 (valor pré-definido do tipo de dados *uint*). Esta característica muito especial faz com que um mapeamento não implemente o conceito de comprimento e não possa ser iterado sem recurso a uma implementação de uma estrutura de dados mais complexa que armazene e faça a gestão do mapeamento, das chaves alteradas depois da inicialização e da respetiva contagem [48]. Um mapeamento é particularmente útil no contexto dos *smart contracts*, isto porque, ao contrário do que acontece com os vetores, não precisa de ser iterada para retornar um valor nela armazenado e não existe a necessidade de a estar constantemente a redimensionar para acomodar novos valores. Isto é particularmente crítico no contexto de execução das transações de um *smart contract*, uma vez que todos os membros da rede teriam que ser ressarcidos pela computação necessária para levar a cabo essas tarefas. De notar, por fim, que os diferentes tipos de estruturas referidas podem ser combinados de modo a satisfazer as necessidades de armazenamento do *smart contract*. Exemplo disso é a definição do armazenamento do *smart contract* "Escola" apresentado na listagem 2.2.

```
1  contract Escola {
2
3      struct Livro {
4          string titulo;
5          string autor;
6          string genero;
7          string editora;
8          uint ano;
9          string isbn;
10     }
11
12     struct Estudante{
13         string nome;
14         string id;
15         string dataDeNascimento;
16         Livro[] livrosFavoritos;
17     }
18
19     mapping( string => Estudante[]) turmas;
20 }
```

Listagem 2.2: Exemplo da especificação em *Solidity* de um *smart contract* que combina vários tipos de dados

2.3.4 *Software* Cliente

De modo a permitir que qualquer interessado se pudesse tornar membro de uma rede *Ethereum*, foi necessário implementar e disponibilizar *software* cliente que implemente

o protocolo *blockchain*. Este *software* cliente levará a cabo, de modo automatizado no sistema onde for executado, todo o tipo de operações que se espera de um membro da rede, como por exemplo a validação de novas transações ou a mineração de novos blocos, entre outras. Apesar da sua implementação mais popular ser a *go-ethereum* (*geth*), que foi implementada com recurso à linguagem *Go*; existem outras como a *cpp-ethereum* que foi implementada na linguagem C++; a *py-ethereum* implementada em *Python*; e a *Parity* que é implementada em *Rust*. Particularmente importante para os programadores de aplicações descentralizadas são as APIs de gestão disponibilizadas por estas implementações cliente. Particularmente porque é através destas interfaces que este tipo de aplicações é capaz de interagir com a rede e, por consequência, com a *blockchain* da mesma. A título de exemplo, a implementação *geth* permite que as suas APIs sejam consumidas através *Javascript Object Notation - Remote Procedure Call* (JSON-RPC) 2.0, *batch-requests*, *Hypertext Transfer Protocol* (HTTP), Inter-Process Communication (IPC) e *websockets* [49].

2.3.5 Protocolo JSON-RPC e Biblioteca Web3j

JSON-RPC é um protocolo de invocação remota de procedimentos com uma filosofia minimalista. Tira partido do formato de troca de dados *Javascript Object Notation* (JSON), inicialmente definido no *Request For Comments* (RFC) 4627 da *Internet Engineering Task Force* (IETF) [50], que permite a representação de números, sequências de caracteres, listas ordenadas e coleções de pares chave-valor. Este protocolo limita-se a definir as estruturas de dados e regras necessárias à invocação de um determinado procedimento remoto. Outra das principais características deste protocolo é a de ser agnóstico a qualquer tipo de protocolo de transporte de rede, podendo ser transmitido entre processos do sistema operativo, por *sockets*, sob o protocolo HTTP ou através de outros protocolos de envio de mensagens [51].

A biblioteca Web3j disponibiliza métodos que facilitam a interação entre redes descentralizadas e aplicações descentralizadas implementadas em linguagens que têm como base a *Java Virtual Machine* (plataforma *Android* e linguagens *Java*, *Kotlin* e *Scala*). Adicionalmente, disponibiliza também um conjunto de utilitários que agiliza a utilização de *smart contracts* no processo de desenvolvimento dessas mesmas aplicações [52]. Existem também implementações Web3 idênticas para outras linguagens, como por exemplo o Web3js para *Javascript* [53].

2.3.6 Redes de Teste

Como acontece em todo o tipo de plataformas, surgiu a necessidade de criar diferentes redes para satisfazer diferentes propósitos. Cada rede *Ethereum* tem os seus próprios membros, a sua própria *blockchain* e opta pelo protocolo de consenso que considera mais adequado às suas necessidades. É como se cada uma destas redes fosse uma pequena galáxia do universo *Ethereum*. Durante o processo de desenvolvimento da plataforma *Ethereum*, a equipa que levou a cabo o mesmo precisou de uma rede para testar as funcionalidades que vinham sendo implementadas, nascendo assim a rede teste *Olympic* [54]. Assim que a versão inicial foi tornada pública, foi também disponibilizada a rede principal *Frontier* onde ocorreriam as transações reais, a maioria proveniente de transações da criptomoeda *ether*. A rede principal *Frontier* viria a ser alvo de diversas atualizações e, em Março de 2016, foi rebatizada para *Homestead* [55].

O crescente interesse dos programadores em aplicações descentralizadas fez com que a comunidade tivesse que criar alternativas à rede principal para que existisse um ambiente *sandbox* que permitisse a todos os interessados testar as funcionalidades disponibilizadas pela plataforma *Ethereum* sem incorrer em avultadas despesas. Recorde-se que as transações executadas na rede principal pressupõem o pagamento de uma recompensa aos membros da rede, pelo poder computacional despendido, em *ether* real. Surgiram então algumas redes de teste que permitem a qualquer um tirar partido das funcionalidades da plataforma *Ethereum* recorrendo a *ethers* ficcionais, à semelhança do que acontece com o dinheiro no famoso jogo de tabuleiro *Monopoly*. Estes créditos ficcionais podem ser obtidos através de *faucets* (torneiras em português) em troca de uma pequena tarefa que serve como recompensa ao grupo que mantém a rede de testes e, ao mesmo tempo, garante que estes pedidos são levadas a cabo por humanos e não por sistemas de automação [44]. Atualmente as redes de teste mais populares são a *Ropsten*, a *Kovan*, a *Rinkeby* e a *Sokol*. Foi convencionado que as redes de teste públicas devem ser batizadas com nomes de estações de comboio/metro [56]. A tabela 2.3 compara as características e comportamento de várias redes de teste.

Rede	*Ropsten*	*Kovan*	*Rinkeby*	*Sokol*
Id da Rede	3	42	4	77
Protocolo de Consenso	*Proof-of-Work*	*Proof-of-Authority*	*Proof-of-Authority*	*Proof-of-Authority*
Endereço de *faucet*	faucet.ropsten.be	faucet.ropsten.be	faucet.rinkeby.io	faucet-sokol.herokuapp.com
Explorador de Blocos da Rede	ropsten.etherscan.io	kovan.etherscan.io	rinkeby.etherscan.io	sokol.etherscan.io
Frequência Média de Novo Bloco	30 segundos	4 segundos	15 segundos	5 segundos
***Software* Cliente Suportado**	*Geth* e *Parity*	*Parity*	*Geth*	*Parity*
Data de Lançamento	Novembro de 2016	Março de 2017	Abril de 2017	Dezembro de 2017
Ocorre Mineração de Blocos	Sim	Não	Não	Não
Maintainers	Equipa *Ethereum*	Equipa *Parity*	Equipa *Ethereum*	Equipa *PoA.Network*

Tabela 2.3: Comparação das características das principais redes de teste para a plataforma *Ethereum*

Capítulo 3

Consentimento e Sistemas Reputacionais

Tradicionalmente, o elo mais fraco das plataformas *web* é o utilizador final. Nesse tipo de arquitetura, a atividade (e os dados resultantes) deste ator fica exposta aos administradores do sistema e a potenciais atacantes; e, em plataformas onde ocorrem transações entre membros, o utilizador pode tornar-se alvo de outros utilizadores mal intencionados. Para mitigar estes problemas tem sido criada legislação, um pouco por todo o mundo, que tem como objetivo defender a privacidade individual e, em cenários críticos, foram adotados sistemas reputacionais que disponibilizam informação a um utilizador sobre os seus pares [1, 57].

3.1 Regulamento Geral de Proteção de Dados

Escândalos recentes como o *Cambridge Analytica*, empresa que entre 2014 e 2018 utilizou dados capturados de 87 milhões de utilizadores da rede social *Facebook* para manipular o resultado de eleições nos Estados Unidos e no Reino Unido [2], ou incidentes como a quebra de dados do gigante do crédito bancário *Equifax*, que em Setembro de 2017 viu dados pessoais de 143 milhões de cidadãos norte-americanos acedidos indevidamente[3], são apontados pela sociedade civil como evidências inequívocas do uso abusivo e da gestão deficiente dos nossos dados pessoais por parte das organizações a quem os confiámos. Como seria de esperar, estes incidentes que se têm tornado recorrentes nos últimos anos não passaram despercebidos às entidades que regulam o setor e resultaram em duras reações por parte das mesmas [1]. Exemplo disso é o Regulamento Geral de Proteção de Dados da União Europeia[1] (RGPD) que vem substituir a Diretiva Europeia de Proteção de dados de 1995. Este regulamento, que entrou em vigor a 25 de Maio deste ano, apresenta-se como um conjunto de regras para a harmonização dos fluxos de dados a nível internacional. O facto de se aplicar aos dados de todos os cidadãos europeus, e não a atividades levadas a cabo no espaço territorial europeu, confere-lhe uma aplicabilidade extraterritorial que obriga toda e qualquer entidade, independentemente da sua origem e da localização da

[1] Regulamento EU 2016/679.

operação, a agir em conformidade sempre que esteja em questão a captura e/ou processamento de dados pessoais de cidadãos europeus [5]. Em síntese, este regulamento representa um claro endurecimento das regras de proteção de dados e das respetivas penalizações por incumprimento. Apresenta-se como um desafio para as organizações que se vêm obrigadas a rever, atempadamente, todos os seus processos que envolvam a captura e/ou processamento de dados pessoais de cidadãos europeus e a pôr em prática medidas que lhes permitam estar em conformidade com o RGPD recorrendo à tecnologia atual [1].

3.1.1 Princípios Fundamentais do RGPD

O regulamento assenta em seis princípios fundamentais que visam assegurar a privacidade dos dados dos cidadãos, sendo eles [5]:

Transparência, clareza e conformidade no processamento

O sujeito dos dados deve ser informado sobre o tipo de processamento que os seus dados vão sofrer. O processamento dos dados deverá estar de acordo com a descrição apresentada ao sujeito os dados aquando do pedido de consentimento e deverá, também, passar nos testes descritos na cláusula 1(a) do artigo 5º.

Limitação do propósito de utilização

De acordo com a cláusula 1(b) do artigo 5º do RGPD, os dados pessoais do sujeito podem apenas ser capturados para fins específicos, explícitos e legítimos. Ou seja, o processamento dos dados capturados deverá apenas ocorrer para um propósito específico e sempre mediante prova do consentimento do sujeito dos dados. A utilização dos dados para outro propósito que não o especificado originalmente requer uma nova prova de consentimento do sujeito dos dados.

Minimização da quantidade dos dados

A quantidade e a qualidade dos dados recolhidos sobre um determinado sujeito deverá ser adequada, relevante e limitada em relação ao propósito do seu processamento. Deste modo, deverá ser mantida a quantidade de dados estritamente necessária para o processamento em questão.

Exatidão dos dados

Tendo em conta que o estabelecimento de políticas base para a gestão de dados garante uma boa proteção dos dados e dificulta roubos de identidade, os dados pessoais deverão ser exatos e, quando necessário, mantidos atualizados. As entidades detentoras de dados de cidadãos europeus deverão elaborar mecanismos de gestão e arquivo de dados que permitam retificar os mesmos.

Limitação de armazenamento dos dados

Atentando à cláusula 1(e) do artigo 5º, fica claro que o regulador espera que os dados pessoais que possam identificar o sujeito dos mesmos sejam armazenados por um período de tempo limitado. Deste modo, dados que já não sejam necessários deverão ser removidos. O principal objetivo desta medida é minimizar o intervalo temporal em que este tipo de dados particularmente sensível é suscetível a fenómenos de quebra de dados e acesso indevido.

Integridade e confidencialidade dos dados

A cláusula 1(f) do artigo 5º obriga a entidade que processa os dados a tomar medidas que garantam a segurança dos mesmos, protegendo-os do processamento ilícito, de perdas acidentais, da destruição, da deterioração e de outros fenómenos similares.

3.1.2 Consentimento informado

No contexto deste estudo assume particular importância o conceito de consentimento informado que, segundo o regulamento em discussão, deve ser dado pelo sujeito dos dados de modo [5]:

Livre

O sujeito dos dados deverá dar o seu consentimento de um modo livre e tendo sempre como alternativa a recusa de consentimento sem que qualquer espécie de penalização possa daí resultar. Este deverá também ser informado sobre o direito de retirada do consentimento, sendo que o exercício desse direito deverá ter um nível de complexidade idêntico ao processo de consentimento original. Contudo, esta retirada não invalida processamentos levados a cabo no período em que o consentimento vigorou. O regulamento vai ainda mais longe, especificando que o consentimento obtido através de contratos que dependam da atribuição de consentimento do sujeito não é válido.

Específico e informado

De modo a que o consentimento cumpra estes dois critérios, o sujeito deverá ser informado sobre qual é a entidade responsável pelo tratamento dos seus dados; sobre terceiros que, eventualmente, possam estar envolvidos no processamento; e sobre qual o propósito específico do processamento dos dados, tendo em conta que se ocorrerem n processamentos deverão igualmente ocorrer n pedidos de consentimento.

Inambíguo

Um dos pontos-chave do regulamento é o processo de captura de consentimento, definindo que este deverá ser inambíguo e requerendo a captura de uma prova do consentimento do sujeito através de uma ação positiva e explícita por parte do mesmo. Desta forma, não serão tidas como válidas provas de consentimento obtidas através de sistemas que utilizem, por exemplo, *checkboxes* pré-preenchidas ou que impossibilitem o sujeito dos dados de manifestar o seu não consentimento.

O RGPD deixa ainda bem claro que a entidade responsável pelos dados deverá ser capaz de, a qualquer momento, fazer prova do consentimento do sujeito dos dados para com a captura e processamento dos seus dados. Em cenários onde o sujeito ainda não atingiu a maioridade legal, o regulamento específica que a entidade responsável pelo processamento dos dados deverá obter o consentimento parental ou, em alternativa, tutelar.

3.2 Confiança e Reputação

No nosso quotidiano facilmente identificamos, inconscientemente, traços de confiabilidade em alguém e agimos de acordo tais instintos. Apesar de ser algo intrínseco ao ser humano, é difícil especificar quais são essas características e como se manifestam. No entanto, pode-se definir o conceito de confiança como a expectativa na fiabilidade do comportamento de alguém ou algo. Um exemplo disso é quando um dado individuo X espera que outro alguém tome uma determinada ação ou posição da qual o seu bem estar depende [58]. Confiança pode também ser definida pelo ponto até ao qual o individuo X se sente confortável em depender de alguém ou de algo perante uma situação da qual podem advir consequências negativas [59]. De notar que, em ambas as definições, fica patente uma situação de dependência do individuo X para com alguém ou algo.

Reputação é aquilo que é geralmente dito ou tido como verdade relativamente ao carácter ou comportamento de algo ou alguém [60]. Para além de ser aplicado a um indivíduo, o conceito de reputação poderá também ser aplicado a um grupo de indivíduos, sendo que neste cenário a reputação do grupo pode ser medida, por exemplo, pela reputação média de todos os seus membros. Deste modo, um novo membro de um grupo herda a reputação do grupo, passando a gozar *a priori* desse estatuto no seio da comunidade [61]. Este conceito está intimamente ligado ao de confiança, uma vez que um é derivado a partir do outro. Partindo deste principio, o modo mais rápido e fiável de obter informação sobre uma pessoa tendo em vista o estabelecimento de um grau de confiança é fazer jus na reputação dessa pessoa no meio social em que se encontra inserida [62]. Tem-se, portanto, que a reputação de alguém não é nada mais nada menos do que o somatório da confiança (ou falta dela) que os membros de uma comunidade têm num dos seus membros. Ainda assim, o sentimento de confiança em alguém não pressupõe a aceitação da sua reputação, exemplo disso são afirmações plausíveis como "eu não confio em ti apesar da tua boa reputação" ou "eu confio em ti apesar da tua má reputação" [63].

3.2.1 Sistemas Reputacionais

Hoje em dia as maiores plataformas de comércio eletrónico, quando comparadas com o comércio tradicional, operam a larga escala e permitem transações pseudo-anónimas[2] [65]. Boa parte do sucesso deste tipo de plataformas fica a dever-se a sistemas reputacionais que capturam, agregam e distribuem informação sobre o comportamento de um membro da comunidade em transações anteriores. A análise dessa informação por um potencial comprador ajuda-o a inferir o risco da potencial transação e a ponderar a possibilidade de adquirir bens ou serviços a um desconhecido quando não os consegue inspecionar fisicamente. Como a esmagadora maioria dos membros não se conhecem, o *feedback* gerado pela comunidade é o principal instrumento ao dispor dos seus membros para possam decidir em quem confiar, para fomentar o bom comportamento no seio da comunidade e para desencorajar a participação de membros desonestos ou pouco habilitados [66].

Um exemplo é o *eBay*, a maior plataforma de leilões online que este ano atingiu os 175 milhões de utilizadores [67]. Esta plataforma não oferece qualquer tipo de garantia sobre os produtos que são transacionados no seu sistema, o que faz com que todos os riscos associados a uma transação sejam assumidos na totalidade pelos utilizadores, sejam eles o vendedor ou o comprador. Apesar de, à partida, esta abordagem parecer a receita perfeita para cenários de fraude e dolo, a taxa de transações concluídas com sucesso permanece surpreendentemente alta [66]. A plataforma atribui os louros deste sucesso ao seu sistema reputacional que permite, após a conclusão de uma transação, que o vendedor e o comprador se avaliem mutuamente. Esta avaliação é feita sob a forma de uma breve mensagem e de um indicador simbólico que indica se a transação ocorreu de um modo positivo, neutro ou negativo. Esta informação será depois compilada e apresentada no perfil de cada membro da comunidade. Outras plataformas de sucesso, como por exemplo a *Amazon*, replicaram este sistema substituindo o indicador simbólico por um indicador numérico (como por exemplo a classificação por estrelas) e adicionando múltiplos parâmetros de avaliação (simpatia, tempo de resposta, qualidade do produto, etc) [68].

Segundo um ponto de vista mais atento, os sistemas reputacionais têm como principal

[2]Diz-se que alguém goza de pseudo-anonimato quando não se consegue identificar a identidade dessa pessoa *online* sem recurso a técnicas de geolocalização e outro tipo de informação privilegiada [64].

objetivo a reposição da perspetiva de negócios futuros gerados pelo desempenho em cada transação, uma vez que no futuro os potenciais clientes terão ao seu dispor informação relativa ao desempenho anterior do vendedor (e vice-versa) [65]. Ainda que, no contexto do comércio eletrónico, a interação e relação entre o vendedor e o comprador seja bastante menos significativa do que no comércio tradicional, a sua importância acaba por ser a mesma, uma vez que a reputação do vendedor será de igual modo construída com base na experiência partilhada pelo comprador. Mesmo que apenas ocorra uma única transação entre ambos, os potenciais futuros compradores tê-la-ão em conta na tomada da sua decisão em transacionar ou não com o vendedor em questão. Partindo do principio que todos os compradores agem desta maneira e que o comportamento de alguém no passado é um bom indicador do seu comportamento no futuro, fica claro que a reputação de um vendedor terá impacto no resultado de negócios futuros. Deste modo, e tendo em vista o sucesso do seu negócio, os vendedores procurarão acumular *feedback* positivo e evitarão *feedback* negativo [66]. Vendedores com uma excelente reputação poderão até tirar partido desse fator para cobrar um valor extra pelos seus produtos, já que muitos compradores estão dispostos a pagar um valor ligeiramente superior perante fortes indicadores de que o serviço prestado pelo vendedor é de qualidade superior [69]. Como neste tipo de sistemas *online* a distribuição da informação é feita por terceiros (pela plataforma que suporta a transação), não ocorrem situações que são comuns em cenários *offline* onde as criticas negativas tendem a ser ocultadas e as positivas enaltecidas.

Para operar de um modo eficiente, um sistema reputacional tem que deter as seguintes características [66]:

Entidades de longa duração

Como demonstrado pela teoria dos jogos, um ramo da matemática aplicada, existem limitações inerentes à eficácia de um sistema reputacional sempre que os membros de uma comunidade possam facilmente criar um novo perfil e recomeçar a sua atividade sob outra identidade. Assim sendo, essa comunidade deverá estabelecer mecanismos que sirvam como barreira inicial aos novos membros, como por exemplo o pagamento de uma joia. Outra abordagem possível passaria pela validação obrigatória da identidade real dos membros, garantindo assim que cada membro seria apenas capaz de criar um único perfil.

Distribuição futura de informação relativa a transações atuais

É importante criar mecanismos que permitam transferir o *feedback* adquirido numa plataforma para outra. Inicialmente, a plataforma *Amazon* permitia que os seus utilizadores importassem a sua classificação e o seu *feedback* da plataforma *eBay*, tendo removido esta funcionalidade depois do *eBay* ter protestado veemente e afirmado que o *feedback* e a classificação de todos os seus utilizadores eram propriedade da sua plataforma. Restrições na distribuição desta informação limitam fortemente a eficácia deste tipo de sistemas, uma vez que restringem o escopo do desempenho de uma entidade a uma única plataforma.

***Feedback* como métrica de confiança na comunidade**

É necessário assegurar, tendo em vista a tomada de decisão sobre a confiabilidade dos membros da comunidade, que todo o *feedback* obtido é agregado e apresentado do modo mais eficiente possível. Deste modo, para além do sistema reputacional apresentar o índice de performance do membro da comunidade também é possível encontrar respostas para questões pertinentes neste contexto, como qual o valor das

transações que originaram o *feedback* ou qual a reputação dos membros que deixaram o *feedback*.

Como não poderia deixar de ser, os sistemas reputacionais também apresentam as suas fragilidades. Depois de uma análise aos sistemas reputacionais de grandes plataformas *online* foram identificadas as diversas fragilidades [66]. Estes sistemas têm em conta apenas os indicadores positivos e negativos no cálculo da reputação de um membro. Esta abordagem não reflete com exatidão nem o comportamento nem as experiências de um membro da comunidade. Assumem que todo *feedback* é honesto e imparcial, não valorizando o *feedback* proveniente de membros mais bem reputados em detrimento do deixado por membros menos reputados. Não permitem que um membro que esteja a analisar a reputação de outro possa definir filtros de acordo com contextos específicos. Esta funcionalidade seria particularmente importante em perfis de vendedores que disponibilizam produtos e serviços de áreas completamente distintas [70]. Várias implementações deste tipo de sistema não se adaptam à evolução da performance dos membros da comunidade, uma vez que valorizam o *feedback* relativo a transações recentes e o *feedback* relativo a transações que ocorreram num passado mais longínquo de igual modo. Adicionalmente, não oferecem incentivos suficientes para que um membro da comunidade despenda do seu tempo para avaliar o desempenho de outrem.

Apesar das fragilidades teóricas e práticas apontadas, os sistemas reputacionais têm tido um desempenho bastante razoável e mantêm-se como um pilar fundamental das comunidades online graças à confiança acumulada entre os seus utilizadores com o passar do tempo. Prova dessa observação empírica coletiva são as milhares de transações que ocorrem diariamente entre partes que estabeleceram a relação de confiança necessária à realização do negócio tendo como fundamento a informação disponibilizada por um sistema reputacional [66].

3.2.2 Comparação de Sistemas Reputacionais

Existem inúmeros sistemas reputacionais disseminados por comunidades *online* de diferentes áreas. Contudo, olhando à dimensão e ao tipo dos riscos associados, é no contexto do comércio e das transações eletrónicas que estes sistemas desempenham um papel mais crítico [65]. Neste mercado, dois dos principais *players* a nível mundial são o *eBay* e a *Amazon*[3], estando ambas as plataformas equipadas com um sistema reputacional que serve como principal ferramenta para o estabelecimento de relações de confiança entre os seus utilizadores [71, 72]. É, portanto, de particular relevância a análise das principais características destes dois sistemas reputacionais.

No *eBay* uma avaliação positiva corresponde à atribuição de um ponto, uma avaliação neutra não corresponde à atribuição de nenhum ponto e uma avaliação negativa corresponde à perda de um ponto. O índice reputacional de um membro é derivado da média ponderada de cada tipo de ocorrência face ao número total de ocorrências. Outros aspetos como o rigor da descrição do produto, a comunicação do vendedor, o tempo de envio e o custo de envio são avaliados através de um sistema de pontuação por estrelas. A escala varia entre uma e cinco estrelas, estando o número de estrelas proporcional à qualidade do desempenho do membro avaliado. Como demonstra a Figura 3.1, a informação relativa a cada utilizador é apresentada no topo da sua página perfil e pode ser vista com maior

[3]No ano de 2017 50% das vendas *online* no mercado dos Estados Unidos da América ocorreram através das plataformas *eBay* e *Amazon* [71].

detalhe com um simples clique na opção *See all feedback*. Nessa página está discriminada a sua pontuação e o *feedback* proveniente das transações em que este participou enquanto vendedor e enquanto comprador, as avaliações que fez a terceiros, entre outras informações mais detalhadas. Caso se encontre descontente com o *feedback* recebido ou tenha tomado providências para remediar as ações que levaram ao mesmo, um membro da comunidade *eBay* pode enviar um pedido ao autor para alterar o *feedback* previamente deixado. Ficará a cargo do autor a decisão de alterar ou não o mesmo [73].

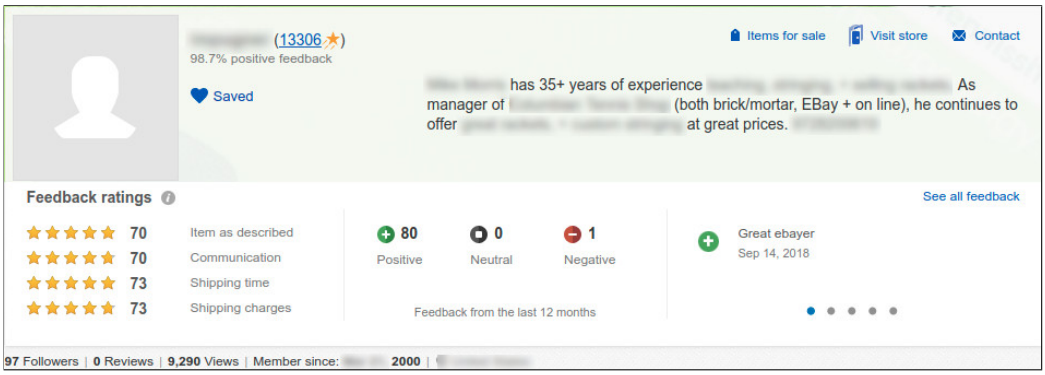

Figura 3.1: Reputação de um membro na comunidade *eBay*

Por sua vez, na plataforma *Amazon*, os membros da comunidade também recorrem a um sistema de estrelas para avaliar a sua satisfação com o desempenho de outro membro da comunidade. Uma avaliação de uma ou duas estrelas é negativa, uma avaliação de três estrelas é neutra e avaliações de 4 ou mais estrelas são tidas como positivas. Para o cálculo da reputação de um membro, é feita a média ponderada de todas as avaliações. Ao contrário do que acontece no *eBay*, onde o vendedor e o comprador se podem avaliar mutuamente, na *Amazon* apenas o comprador pode avaliar o vendedor. Apenas em circunstancias bastante circunscritas um vendedor será capaz de remover *feedback* deixado pelo comprador, como por exemplo o uso de impropérios e linguagem abusiva ou a partilha de dados pessoais. Como demonstra a Figura 3.2, esta informação é apresentada na totalidade no topo página de perfil do membro [74].

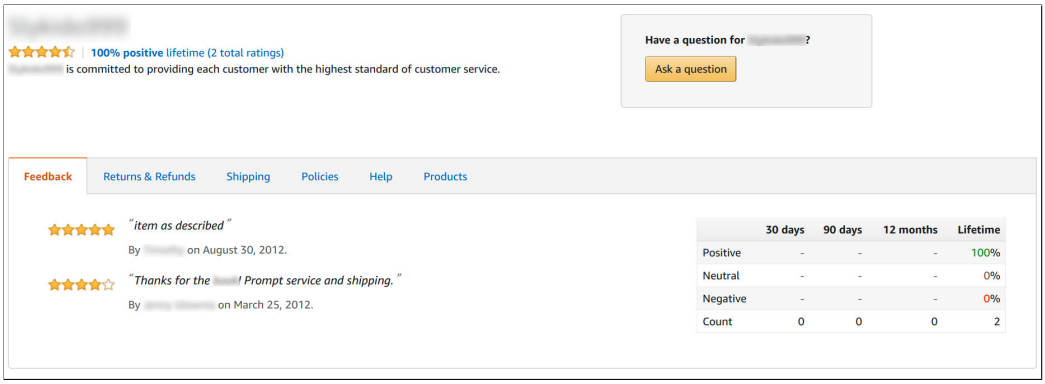

Figura 3.2: Reputação de um membro da comunidade *Amazon*

Depois desta análise são inegáveis as semelhanças entre os sistemas reputacionais de ambas as plataformas. Não obstante do facto dos modelos de negócio destas plataformas serem vincadamente diferentes, este fenómeno pode ser explicado pelo facto do *eBay* e da *Amazon* concorrerem pelo mesmo mercado. Retira-se também, de um modo indireto, que este tipo de sistema reputacional tem provas dadas na indústria do comércio eletrónico.

Capítulo 4

Descrição da Solução

A solução *Connsent* tem como objetivos principais a gestão de todas as etapas do consentimento informado segundo o RGPD e a avaliação do desempenho das entidades controladoras de dados tendo por base *feedback* emitido pelos sujeitos dos dados. Em virtude do que está previsto no RGPD[1], toda a informação relativa a este sistema será persistida numa rede de *blockchain* pública. Deste modo, as características da tecnologia *blockchain* são alavancadas para garantir a integridade de toda a informação proveniente do sistema, seja ela referente a operações de consentimento ou a avaliações dos sujeitos dos dados ao desempenho das entidades controladoras de dados.

Antes que seja possível ocorrer qualquer tipo de operação de consentimento, é necessário que todas as entidades (controlador de dados, sujeito dos dados e entidade reguladora) intervenientes nessa operação se registem no sistema. O registo de uma entidade pressupõe a geração de um par de chaves "ECDSA" com uma força de 224 *bits*, a escolha de identificador e uma assinatura digital[2]. Estes três artefactos serão então transmitidos, através de uma REST API, ao sistema *Connsent* que levará a cabo as validações de negócio necessárias e tratará da sua persistência na *blockchain*. Em conjunto com estes três artefactos, será persistido na *blockchain* o *timestamp* do momento da criação do bloco que vai conter estes dados. É neste conjunto de dados que se fundamentarão todas as provas de operações futuras, uma vez que, por exemplo, a chave pública da entidade armazenada na *blockchain* será utilizada para garantir a autenticidade e integridade das suas operações.

Partindo do pressuposto que todos os intervenientes já se encontram registados no sistema, está previsto que as interações com o sistema *Connsent*, que tenham como finalidade operações de consentimento, se iniciem no momento em que a entidade controladora de dados capturou dados relativos a um determinado sujeito e pretende levar a cabo algum tipo de operação com os mesmos. Nesse instante, a entidade processadora de dados deverá comunicar ao sistema *Connsent*, através da sua REST API, que pretende obter o consentimento de um determinado sujeito dos dados apresentando: o identificador da operação de consentimento, o identificador do sujeito dos dados, o identificador dos termos de consentimento, o identificador da entidade reguladora que supervisionará o processo, o resumo criptográfico dos dados capturados sobre o sujeito dos dados e uma assinatura

[1] O RGPD prevê que o controlador dos dados seja obrigado a disponibilizar à entidade reguladora provas inequívocas do consentimento do sujeito dos dados.

[2] A estrutura de todas as assinaturas digitais previstas pelo sistema *Connsent* são detalhadas no Ponto 4.2.

digital deste conjunto informação. Munido desta informação, o sistema é capaz, após levar a cabo as devidas validações, de proceder ao envio de uma notificação *push* para o dispositivo móvel do sujeito dos dados. A ação do sujeito dos dados sob esta notificação, que será apresentada no seu dispositivo móvel, remetê-lo-á para a aplicação móvel *Connsent* que lhe apresentará toda a informação relativa ao pedido de consentimento, sendo lhe dada a opção de consentir (ou não) a utilização dos seus dados para os fins propostos pelo controlador de dados. Na eventualidade do sujeito dos dados pretender dar o seu consentimento, e como o registo do sujeito do sujeito dos dados no sistema aconteceu através da aplicação móvel, será calculada uma assinatura digital que servirá como prova do seu consentimento. Essa prova será transmitida pela aplicação móvel ao *back end* do sistema, novamente através da REST API do mesmo, que por sua vez se encarregará de efetuar as validações necessárias para assegurar sua a validade e a sua persistência na *blockchain* da rede que suporta o sistema. A partir do momento em que todos os dados referentes a esta operação de consentimento estejam persistidos na *blockchain* passará a vigorar o consentimento do sujeito dos dados com os termos que lhe foram propostos pela controlador de dados.

Sempre que se pretenda remover um consentimento previamente dado, o sujeito dos dados poderá fazê-lo por intermédio da aplicação móvel. Nesse instante a aplicação móvel gerará as provas criptográficas que atestam a vontade do sujeito dos dados e, de seguida, comunicará as mesmas ao *back end* do sistema por meio da REST API. O *back end* do sistema, por sua vez, realizará as validações de negócio necessárias e persistirá a remoção de consentimento na rede *blockchain*.

Como referido no Ponto 1.1, um dos principais objetivos do sistema passa por disponibilizar ao sujeito dos dados um subsistema reputacional que lhe permita avaliar o comportamento do controlador dos dados, equilibrando assim a distribuição de forças entre ambas as partes envolvidas neste relacionamento. Assim, o sujeito dos dados poderá recorrer a este subsistema para avaliar o comportamento do controlador de dados a quem consentiu a utilização dos seus dados pessoais através da aplicação móvel *Connsent*. A um consentimento poderão ser associadas várias (uma ou mais) avaliações ao desempenho do controlador de dados. Estas avaliações, também designadas como entradas de *feedback* no âmbito deste trabalho, são compostas por uma nota e por uma mensagem. Os notas previstas são: negativa, neutra e positiva. Espera-se que o sujeito dos dados recorra à nota negativa sempre que considera que o desempenho do controlador de dados foi insatisfatório, à neutra sempre que considera que o desempenho não foi nem positivo nem negativo e à nota positiva quando considera que o controlador dos dados teve um bom desempenho. A mensagem deverá informar, de um modo sintético, os restantes membros da comunidade sobre o desempenho do controlador de dados no âmbito daquele consentimento em particular. Da junção da nota e da mensagem com outros valores que identificam univocamente os entidades envolvidas e a operação de consentimento em questão resulta uma prova criptográfica que testifica a opinião do sujeito dos dados, prova essa que é gerada no dispositivo móvel deste com recurso a uma assinatura digital computada com a chave privada associada à chave pública que o identifica no sistema. Essa prova criptográfica será então transmitida pela aplicação móvel ao *back end* do sistema que, por sua vez, a persistirá no armazenamento da rede *blockchain* que suporta o sistema. Mesmo que o sujeito dos dados remova o seu consentimento poderá continuar a deixar novas avaliações ao comportamento do controlador dos dados, uma vez que à retirada do consentimento poderá estar a associado um mau desempenho do controlador de dados. Este *design* garante que um sujeito dos dados seja apenas capaz de avaliar um controlador de dados a quem deu, de facto, o seu consentimento. Todo o *feedback* referente a um controlador de

dados será agregado e apresentado no seu perfil público, juntamente com pontuações e dados estatísticos deduzidos a partir dessa mesma informação.

Nos próximos pontos são detalhadas as técnicas e fundamentos utilizados para implementar uma solução com o comportamento que aqui foi descrito.

4.1 Modelação do Sistema

De modo a de garantir que a solução apresentada cumpre os objetivos pré-definidos e suporta as funcionalidades pretendidas, procedeu-se ao levantamento dos casos de usos e das entidades que deverão ser previstas pelo sistema. Neste ponto serão detalhados os resultados deste levantamento recorrendo à *Unified Modeling Language* (UML).

4.1.1 Casos de Uso

O diagrama de casos de uso UML apresentado na Figura 4.1 representa as funcionalidades que o sistema *Connsent* irá apresentar e com quais os diferentes atores irão interagir. O sistema "Plataforma de Inquéritos" representa uma plataforma de inquéritos genérica detida por um controlador de dados que tem como finalidade a captura de dados de múltiplos sujeitos dos dados. Embora se encontre fora do âmbito do sistema *Connsent*, este componente é apresentado neste diagrama porque é critico à representação das interações entre os diversos atores. De seguida serão detalhados e descritos todos os casos de usos identificados.

Criar Inquérito

Inclui caso(s) de uso: Não Aplicável
Incluído pelos caso(s) de uso: Não Aplicável
Usado pelos atores: Controlador de Dados
Descrição: Um controlador de dados cria o inquérito na plataforma de inquéritos que detém com o intuito de capturar dados de sujeitos de dados que se disponham a fornecê-los.

Preencher Inquérito

Inclui caso(s) de uso: Não Aplicável
Incluído pelos caso(s) de uso: Submeter Inquérito
Usado pelos atores: Não Aplicável
Descrição: Um sujeito dos dados preenche o inquérito com intuito de facultar um conjunto de dados ao controlador de dados que detêm a plataforma de inquéritos.

Submeter Inquérito

Inclui caso(s) de uso: Preencher Inquérito, Pedir Consentimento do Sujeito dos Dados, Enviar Notificação *Push*
Incluído pelos caso(s) de uso: Não Aplicável
Usado pelos atores: Não Aplicável
Descrição: Um sujeito dos dados submete o inquérito após o seu preenchimento, despoletando assim um pedido de consentimento através de uma notificação *push*.

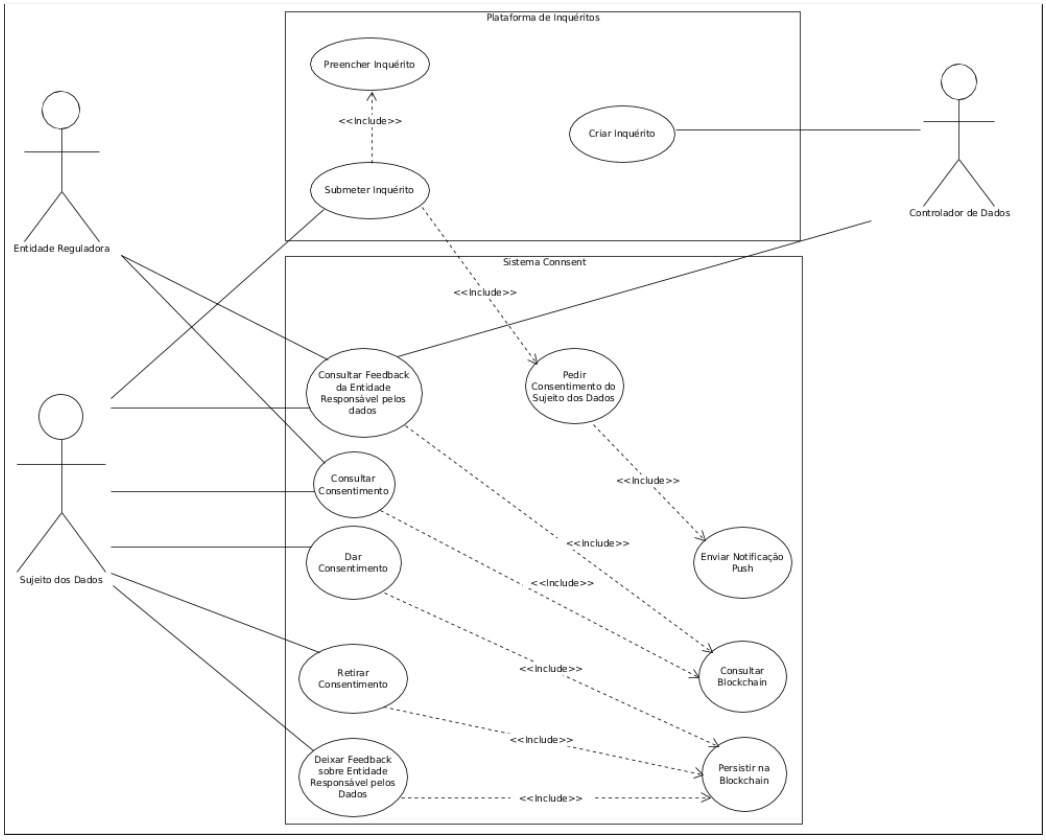

Figura 4.1: Diagrama de casos de uso

Pedir Consentimento do Sujeito dos Dados

Inclui caso(s) de uso: Enviar Notificação *Push*
Incluído pelos caso(s) de uso: Submeter Inquérito
Usado pelos atores: Não Aplicável
Descrição: É criado um novo pedido de consentimento no sistema *Connsent*.

Enviar Notificação *Push*

Inclui caso(s) de uso: Não Aplicável
Incluído pelos caso(s) de uso: Pedir Consentimento do Sujeito dos Dados
Usado pelos atores: Não Aplicável
Descrição: Um dispositivo móvel é notificado através de notificação *push*.

Consultar *Feedback* do Controlador de Dados

Inclui caso(s) de uso: Consultar *Blockchain*
Incluído pelos caso(s) de uso: Não Aplicável
Usado pelos atores: Entidade Reguladora, Sujeito dos Dados e Controlador de Dados
Descrição: Um utilizador do sistema *Connsent*, independentemente do seu papel, consulta o *feedback* de um controlador de dados.

Consultar Consentimento

Inclui caso(s) de uso: Consultar *Blockchain*
Incluído pelos caso(s) de uso: Não Aplicável
Usado pelos atores: Entidade Reguladora, Sujeito dos Dados e Controlador de Dados
Descrição: Um utilizador do sistema *Connsent*, independentemente do seu papel, consulta um determinado consentimento.

Consultar *Blockchain*

Inclui caso(s) de uso: Não Aplicável
Incluído pelos caso(s) de uso: Consultar Consentimento e Dar Consentimento
Usado pelos atores: Não Aplicável
Descrição: É efetuada uma leitura de dados persistido numa rede *blockchain*.

Dar Consentimento

Inclui caso(s) de uso: Persistir na *Blockchain*
Incluído pelos caso(s) de uso: Não Aplicável
Usado pelos atores: Sujeito dos Dados
Descrição: Um sujeito dos dados, a pedido, consente que um conjunto dos seus dados pessoais seja processado por um controlador de dados.

Retirar Consentimento

Inclui caso(s) de uso: Persistir na *Blockchain*

Incluído pelos caso(s) de uso: Não Aplicável

Usado pelos atores: Sujeito dos Dados

Descrição: Um sujeito dos dados remove um consentimento previamente dado a um controlador de dados para processar um conjunto dos seus dados pessoais.

Deixar *Feedback* Sobre Controlador de Dados

Inclui caso(s) de uso: Persistir na *Blockchain*

Incluído pelos caso(s) de uso: Não Aplicável

Usado pelos atores: Sujeito dos Dados

Descrição: Um sujeito dos dados partilha com os restantes membros do sistema *feedback* relativo à sua experiência com o controlador de dados a quem consentiu o processamento de um conjunto dos seus dados pessoais.

Persistir na *Blockchain*

Inclui caso(s) de uso: Não Aplicável

Incluído pelos caso(s) de uso: Dar Consentimento, Retirar Consentimento e Deixar *Feedback* Sobre Controlador de Dados

Usado pelos atores: Não Aplicável

Descrição: É efetuada uma operação de escrita na *blockchain* de uma rede com o objetivo de persistir um conjunto de dados referentes ao sistema *Connsent*.

4.1.2 Entidades e Relacionamentos

Perante a necessidade de representar dados de uma forma estruturada no sistema, foi feito um levantamento das entidades físicas e virtuais e dos múltiplos relacionamentos entre si. Posto isto, foram identificadas as seguintes entidades:

Consentimento – Representa um consentimento dado por um sujeito dos dados;

Controlador de Dados – Representa um controlador de dados no sistema;

Entidade Reguladora – Representa uma entidade reguladora no sistema;

Feedback – Representa uma entrada de *feedback* relativa a um consentimento;

RemConsentimento – Representa a remoção de um consentimento previamente dado por um sujeito dos dados;

Sujeito dos Dados – Representa um sujeito dos dados no sistema;

Termo de Consentimento – Representa um termo de consentimento criado por um

controlador de dados no sistema.

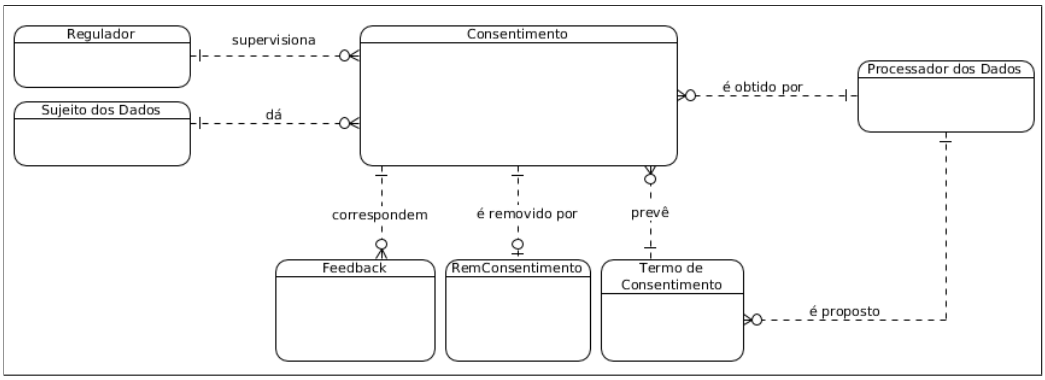

Figura 4.2: Diagrama de entidades-relacionamento

O digrama UML apresentado na Figura 4.2, para além de representar as entidades enunciadas, representa ainda os seguintes relacionamentos entre as mesmas:

Entidade Reguladora <=> Consentimento

Cardinalidade: 1 para 0..*
Descrição: Uma entidade reguladora supervisiona zero ou mais consentimentos.

Sujeito dos Dados <=> Consentimento

Cardinalidade: 1 para 0..*
Descrição: Um sujeito dos dados dá zero ou mais consentimentos.

Consentimento <=> RemConsentimento

Cardinalidade: 1 para 0..1
Descrição: A um consentimento corresponde uma ou nenhuma remoção de consentimento.

Consentimento <=> *Feedback*

Cardinalidade: 1 para 0..*
Descrição: A um consentimento correspondem zero ou muitas entradas de *feedback*.

Consentimento <=> Termo de Consentimento

Cardinalidade: 0..n para 1
Descrição: Múltiplos consentimentos podem estar associados a um único termo de consentimento.

Consentimento <=> Controlador de Dados

Cardinalidade: 0..n para 1
Descrição: Múltiplos consentimentos podem estar associados a um único controlador de dados.

Termo de Consentimento <=> Controlador de Dados

Cardinalidade: 0..n para 1
Descrição: Múltiplos termos de consentimento podem ser registados por um único controlador de dados.

4.2 Formatos de Assinatura Digital

De acordo com o que foi referido no inicio deste Capítulo, a concordância de uma entidade no sistema com as operações que lhe são imputadas no âmbito do sistema será provada pela assinatura digital de um conjunto de dados com a chave privada persistida no *smartphone* por ela detida. Esta assinatura poderá ser validada publicamente; já que a chave pública, o valor da assinatura e os dados assinados serão persistidos numa rede *blockchain* aberta e acessíveis publicamente. Para cada uma das operações que implicam uma assinatura digital, como prova da concordância da entidade interveniente, foi definido um formato de assinatura digital que tem como objetivo agregar os diferentes dados que perfazem o valor a ser assinado digitalmente, com a respetiva chave privada, de um modo estruturado.

O algoritmo de assinatura a utilizar será o ECDSA (referido no Ponto 2.1.4) que assinará o valor resultante do resumo criptográfico SHA-512 (consultar Ponto 2.1.2) da representação UTF-8 (consultar Ponto 2.1) do valor da assinatura definido pelo formato de assinatura em questão. De seguida são detalhados os formatos de assinatura previstos.

Assinatura Digital de Registo de Entidade

Autor: Sujeito dos dados, controlador de dados ou entidade reguladora.
Objetivo: Servir como prova de registo de um sujeito dos dados, de um controlador de dados ou de uma entidade reguladora. A assinatura digital do identificador ou designação e da chave pública demonstra o conhecimento prévio do identificador ou designação pela qual a entidade é conhecida no sistema e prova, também, que a entidade está na posse da chave privada associada à chave pública registada.
Formato: "<id_entidade>/<chave_publica>"
Componentes da assinatura:
id_entidade – Identificador ou designação da entidade.
chave_publica – Chave pública da entidade codificada em Base64.
Exemplo: ''Lusocom/ME4wEAYHKoZIzj0CAQYFK4EEACEDOgAEUCEvt3YEjyccZTkaVDpvK7EVwEXttovwAL8d289eS1OcqT4+ke0Qdpdp3+i7x64wikS619zlcs4=''

Assinatura Digital de Registo de Termo de Consentimento

Autor: Controlador de dados

Objetivo: Servir como prova de registo de um termo de consentimento por parte de um controlador de dados. A assinatura digital do identificador do termo de consentimento e do seu texto prova que este foi criado pelo controlador de dados referido.
Formato: ''<id_termo>/<txt_termo>/<desig_controlador>''
Componentes da assinatura:
id_termo – Identificador do termo de consentimento.
txt_termo – Texto do termo de consentimento.
desig_controlador – Designação do controlador de dados.
Exemplo: "e8ef400a-b2da-460b-a9a5-7f80dd8026a1/Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum./Lusocom"

Assinatura Digital de Pedido de Consentimento

Autor: Controlador de dados
Objetivo: Servir como prova que o pedido de consentimento teve origem no controlador de dados referido. A assinatura digital do identificador do consentimento, do identificador do termo de consentimento, do identificador do sujeito dos dados, da designação do controlador de dados, da designação da entidade reguladora, e do resumo criptográfico dos dados associados ao consentimento baliza o pedido consentimento a uma única ocorrência que jamais se repetirá na mesma instância do sistema.
Formato: ''<id_consentimento>/<id_termo>/<id_sujeito>/<desig_controlador>/<desig_ent_reguladora>/<resumo_cripto>''
Componentes da assinatura:
id_consentimento – Identificador do consentimento.
id_termo – Identificador do termo de consentimento.
id_sujeito – Identificador do sujeito dos dados.
desig_controlador – Designação do controlador de dados.
desig_ent_reguladora – Designação da entidade reguladora.
resumo_cripto – Resumo criptográfico dos dados codificado em hexadecimal.
Exemplo: ''47c4c510-bf6e-47ff-a5ab-039d1f111cce/ad307559-4406-4ebc-bc21-7bbfd4d8c0a3/8879f13c-27ba-480d-88f8-9a37d1f4c49d/Lusocom/CNPD/68aaf0cdbbd7853a1167db5efcbe8fcc018e3b765b82e8b9f52f1dea71fcc821b35ab259b2c343b1305096be9bc7f656675e5a9be8954212042c000dc1d26a43''

Assinatura Digital de Consentimento

Autor: Sujeito dos dados
Objetivo: Servir como prova que o pedido de consentimento teve origem no controlador de dados referido. A assinatura digital do identificador do consentimento, do identificador do termo de consentimento, do identificador do sujeito dos dados, da designação do controlador de dados, da designação da entidade reguladora, e do resumo criptográfico dos dados associados ao consentimento baliza o pedido consentimento a uma única ocorrência que jamais se repetirá na mesma instância do sistema.

Formato: ‘‘<id_consentimento>/<id_termo>/<id_sujeito>/<desig_controlador>/<desig_ent_reguladora>/<resumo_cripto>/<val_assinatura_pedido>’’
Componentes da assinatura:
id_consentimento – Identificador do consentimento.
id_termo – Identificador do termo de consentimento.
id_sujeito – Identificador do sujeito dos dados.
desig_controlador – Designação do controlador de dados.
desig_ent_reguladora – Designação da entidade reguladora.
resumo_cripto – Resumo criptográfico dos dados codificado em hexadecimal.
val_assinatura_pedido – Valor da assinatura do pedido de consentimento codificado em Base64.
Exemplo: ‘‘47c4c510-bf6e-47ff-a5ab-039d1f111cce/ad307559-4406-4ebc-bc21-7bbfd4d8c0a3/8879f13c-27ba-480d-88f8-9a37d1f4c49d/Lusocom/CNPD/68aaf0cdbbd7853a1167db5efcbe8fcc018e3b765b82e8b9f52f1dea71fcc821b35ab259b2c343b1305096be9bc7f656675e5a9be8954212042c000dc1d26a43/MD0CHFfrnKmHQjsiwyavOvlyWjGLoc5amNslWw7qC9kCHQD8a3H9uIr8R4jebh/j/rOqBcFTlLG03wH06unJ’’

Assinatura Digital de Remoção de Consentimento

Autor: Sujeito dos dados
Objetivo: Servir como prova da vontade do sujeito dos dados em remover um consentimento previamente dado. A assinatura digital do identificador do consentimento e do selo temporal de criação do bloco que armazena os dados referentes ao consentimento a ser removido baliza a remoção do consentimento a uma única ocorrência que jamais se repetirá na mesma instância do sistema.
Formato: ‘‘<id_consentimento>/<timestamp_criacao>’’
Componentes da assinatura:
id_consentimento – Identificador do consentimento.
timestamp_criacao – Selo temporal do momento de criação do bloco que contém os dados do consentimento a ser removido.
Exemplo: “47c4c510-bf6e-47ff-a5ab-039d1f111cce/1537129333”

Assinatura Digital Entrada de *Feedback*

Autor: Sujeito dos dados
Objetivo: Servir como prova que a entrada de *feedback* foi criada pelo sujeito dos dados associado ao consentimento referido.
Formato: ‘‘<id_consentimento>/<nota>/<mensagem>’’
Componentes da assinatura:
id_consentimento – Identificador do consentimento.
nota – Representação numérica da nota atribuída pelo sujeito dos dados ao controlador de dados.
mensagem – Mensagem do sujeito dos dados relativamente ao desempenho do controlador de dados.
Exemplo: “47c4c510-bf6e-47ff-a5ab-039d1f111cce/2/O controlador de dados cumpriu o termo de consentimento escrupulosamente!”

4.3 Subsistema Reputacional

O sistema *Connsent* engloba um subsistema reputacional, semelhante aos analisados no Ponto 3.2.2, que permite que um sujeito dos dados possa avaliar o desempenho de um controlador de dados enquanto entidade a quem confiou os seus dados pessoais. Um sujeito dos dados pode, no âmbito do mesmo consentimento, deixar uma ou mais avaliações ao desempenho do controlador de dados, podendo continuar a fazê-lo mesmo depois de retirar o seu consentimento. Todo o *feedback* deixado pelos sujeitos dos dados neste subsistema será agregado por controlador de dados e consultável publicamente no perfil do controlador de dados no sistema. O principal objetivo desta funcionalidade é dar poder ao sujeito dos dados que, tradicionalmente, é o elo mais fraco da cadeia de processamento de dados. A entidade reguladora poderá também recorrer ao subsistema para monitorizar a satisfação dos sujeitos dos dados com os controladores de dados, podendo até fazê-lo em tempo real e de um modo automatizado através da integração de uma *dashboard* de monitorização com os eventos emitidos na rede *blockchain* pela *contract account* que suporta o subsistema. Um cenário de referência para a integração dos eventos da rede *blockchain* emitidos pela *contract account* que suporta sistema *Connsent* na *dashboard* da entidade reguladora, tendo vista a monitorização da satisfação dos sujeitos dados, seria a emissão de um alerta sempre que fosse deixada uma entrada de *feedback* com nota negativa a um controlador de dados no âmbito de um consentimento sob a supervisão dessa mesma entidade reguladora. Esse alerta despoletaria uma análise ao *feedback* negativo com o intuito de aferir se existiria fundamento para a atuação da entidade reguladora nesse caso particular. No Ponto 4.4.4 está disponível informação relativa aos eventos de rede *blockchain* previstos para o sistema *Connsent*.

Uma entrada de *feedback*, também designada por avaliação, consiste numa nota e numa mensagem até quinhentos caracteres. A nota deverá representar a experiência do sujeito dos dados com o controlador de dados, enquanto a mensagem deverá fundamentar a nota atribuída. A nota poderá ser de um dos três seguintes tipos:

Negativa – Sempre que o sujeito dos dados considerar que o desempenho do controlador de dados não foi adequado;
Neutra – Quando o sujeito dos dados considerar que o comportamento do controlador de dados foi adequado;
Positiva – Sempre que o sujeito dos dados considerar que o desempenho do controlador de dados foi adequado e excedeu as suas expectativas.

A uma avaliação positiva corresponderá a atribuição de um ponto, a uma avaliação negativa corresponderá a subtração de um ponto e a uma avaliação neutra não corresponderá a atribuição nem a subtração de nenhum ponto. De notar que em consentimentos onde foram deixadas múltiplas avaliações apenas será considerada a entrada mais recente para o cálculo da pontuação.

A reputação de um controlador de dados pode ser aferida pela pontuação resultante da diferença entre a contagem das avaliações positivas e as avaliações negativas que lhe foram conferidas pelos sujeitos dos dados com que interagiu e pela percentagem de avaliações positivas, neutras e negativas. O valor das percentagens será arredondado à primeira casa decimal e apresentado desse modo. A título de exemplo, vejamos o caso do controlador de dados "Lusocom":

Total de avaliações: 193
Avaliações Positivas: 160
Avaliações Neutras: 30

Avaliações Negativas: 3
Pontuação: 157 pontos
Percentagem de avaliações positivas: 82,9 %
Percentagem de avaliações neutras: 1,6 %
Percentagem de avaliações negativas: 15,5 %

Tendo em conta que a chave privada utilizada para gerar provas das operações dos sujeitos dos dados reside na aplicação móvel *Connsent*, está previsto que as avaliações sejam criadas com recurso à mesma sendo, depois, transmitidas ao *back end* do sistema através da REST API. A reputação de um controlador de dados pode ser consultada na aplicação móvel *Connsent* ou através de qualquer sistema que integre com a REST API. Os cálculos necessários à apresentação da reputação do controlador de dados serão computados *on demand* pelo *back end* do sistema sob a informação persistida na *blockchain* da rede utilizada pelo sistema *Connsent*. Adicionalmente, seria possível a qualquer membro da rede *blockchain* computar a reputação de qualquer controlador de dados através da informação persistida pelo sistema na rede *blockchain*.

4.4 *Ethereum Smart Contract*

De maneira a que o sistema possa utilizar a *blockchain* de uma rede *Ethereum* pública como livro de razão, foi necessário definir um *smart contract* que especificasse:

- A definição das enumerações necessárias à criação de um tipo de dados através da um conjunto de valores pré-definidos;
- A definição das estruturas de dados complexas criadas a partir da combinação de variáveis de outros tipos de dados;
- As variáveis necessárias para persistir os dados no armazenamento da *contract account*;
- Eventos que definissem a estrutura da informação a ser emitida na rede em momentos considerados críticos;
- Transações que permitissem persistir novos dados na *contract account* através da alteração do estado da mesma, e em consequência, da rede;
- *Views* que possibilitassem a leitura de dados do armazenamento da *contract account*.

O compilador de linguagem *Solidity* "solc" estima que a criação da *contract account* através do *smart contract* especificado ao longo deste ponto terá um custo de 5968976 unidades de *gas*. O código fonte está disponível no Apêndice 8.1.

4.4.1 Enumerações

As enumerações apresentadas neste ponto permitiram criar tipos de dados a partir de um conjunto de valor pré-definidos que representam determinados estados no sistema. Esta funcionalidade da linguagem *Solidity* foi particularmente útil para representar as notas das entradas de *feedback* no subsistema reputacional descrito no Ponto 4.3.

FeedbackType

Objetivo: Definir os valores válidos para a nota a ser deixada pelo sujeito dos dados ao controlador de dados no contexto do subsistema reputacional.
Valores previstos: "NEGATIVE", "NEUTRAL" e "POSITIVE".

4.4.2 Estruturas de Dados

O objetivo das estruturas de dados aqui enumeradas é representar as entidades identificadas no diagrama de casos de uso enunciado no Ponto 4.1.2 na *contract account* que suporta o sistema *Connsent*. Tal é conseguido através da combinação de um conjunto de variáveis, de tipos primitivos da linguagem *Solidity* e tipos definidos pelas enumerações definidas no Ponto 4.4.1 que, combinados, permitissem representar uma entidade complexa.

DataController

Objetivo: Suportar os dados necessários ao armazenamento de um controlador de dados.
Variáveis:
string name – Armazena a designação do controlador de dados.
string signature – Armazena o valor da assinatura digital de registo do controlador de dados.
string publicKey – Armazena a chave pública do controlador de dados codificada em Base64.
uint timestamp – Armazena o selo temporal do momento em que o bloco que contém a informação de registo do controlador de dados foi criado.

DataSubject

Objetivo: Suportar os dados necessários ao armazenamento de um sujeito dos dados.
Variáveis:
string uid – Armazena o identificador do sujeito dos dados.
string signature – Armazena o valor da assinatura digital de registo do sujeito dos dados.
string publicKey – Armazena a chave pública do sujeito dos dados codificada em Base64.
uint timestamp – Armazena o selo temporal do momento em que o bloco que contém a informação de registo do sujeito dos dados foi criado.

Regulator

Objetivo: Suportar os dados necessários ao armazenamento de uma entidade reguladora.
Variáveis:
string name – Armazena a designação da entidade reguladora.
string signature – Armazena o valor da assinatura digital de registo da entidade reguladora.
string publicKey – Armazena a chave pública da entidade reguladora codificada em Base64.
uint timestamp – Armazena o selo temporal do momento em que o bloco que contém a informação de registo da entidade reguladora foi criado.

Terms

Objetivo: Suportar os dados necessários ao armazenamento de um termo de consentimento.
Variáveis:
string uid – Armazena o identificador do termo de consentimento.
string terms – Armazena o texto do termo de consentimento.
string dataControllerName – Armazena a designação do controlador de dados que criou o termo de consentimento.

string termsSig – Armazena o valor da assinatura digital (codificada em Base64) do termo de consentimento por parte do controlador de dados que o registou.
uint timestamp – Armazena o selo temporal do momento em que o bloco que contém a informação do termo de consentimento foi criado.

Consent

Objetivo: Suportar os dados necessários ao armazenamento de consentimento.
Variáveis:
string uid – Armazena o identificador da operação de consentimento.
string termsUid – Armazena o identificador do termo de consentimento associado à operação de consentimento.
string dataSubjectUid – Armazena o identificador do sujeito dos dados associado à operação de consentimento.
string dataControllerName – Armazena a designação do controlador de dados associado à operação de consentimento.
string regulatorName – Armazena a designação da entidade reguladora associada à operação de consentimento.
string dataDigest – Armazena o resumo criptográfico (em codificação hexadecimal), segundo o algoritmo SHA-512, da informação que o sujeito dos dados consentiu que o controlador de dados processasse.
string requestSig – Armazena o valor da assinatura digital (codificado em Base64) que serve como prova do pedido de consentimento efetuado pelo controlador de dados.
string consentSig – Armazena o valor da assinatura digital (codificado em Base64) que serve como prova o consentimento do sujeito dos dados.
uint timestamp – Armazena o selo temporal do momento em que o bloco que contém a informação do consentimento foi criado.

RemovedConsent

Objetivo: Suportar os dados necessários ao armazenamento de um consentimento removido.
Variáveis:
string consentUid – Armazena o identificador do consentimento removido.
string removalSig – Armazena o valor da assinatura digital (codificado em Base64) que serve como prova da vontade do sujeito dos dados em remover o seu consentimento.
uint timestamp – Armazena o selo temporal do momento em que o bloco que contém a informação da remoção de consentimento foi criado.

Feedback

Objetivo: Suportar os dados necessários ao armazenamento de todo o *feedback* relativo a um consentimento.
Variáveis:
string consentUid – Armazena o identificador do consentimento a que o *feedback* se refere.
FeedbackEntry[ ] entries – Armazena todas as entradas de *feedback* referentes ao consentimento em questão.

FeedbackEntry

Objetivo: Suportar os dados necessários ao armazenamento de uma entrada de *feedback*.
Variáveis:
FeedbackType feedbackType – Armazena a nota com o sujeito dos dados avaliou o desempenho do controlador de dados.
string message – Armazena a mensagem deixada pelo sujeito dos dados relativamente ao desempenho do controlador de dados.
string feedbackSig – Armazena o valor da assinatura digital (codificado em Base64) que prova que a entrada de *feedback* foi criada pelo sujeito dos dados.
uint timestamp – Armazena o selo temporal do momento em que o bloco que contém a entrada de *feedback* foi criado.

4.4.3 Variáveis

As variáveis de um *smart contract* permitem especificar a estrutura do armazenamento da *contract account* originada pelo mesmo. No caso do *smart contract* em questão, as variáveis, à exceção da variável "owner", são mapeamentos de valores do tipo primitivo *string* para estruturas de dados complexas dos tipos referidos é no Ponto 4.4.2. Esta abordagem permite operações de leitura e de criação altamente eficientes utilizando como chave o identificador ou a designação das entidades definidas no Ponto 4.1.2. Esta eficiência é particularmente crítica no contexto de uma *blockchain* distribuída onde qualquer operação computacional, por mais ínfima que seja, implica o pagamento de uma taxa à rede. A modificação das variáveis desta *contract account* será levada a cabo pelas transações definidas no Ponto 4.4.5 e o seu conteúdo poderá ser acedido através das *views* definidas no Ponto 4.4.6.

owner

Tipo de dados: address
Visibilidade: private
Objetivo: Armazenar o endereço da carteira que criou a *contract account*. O valor armazenado nesta variável será utilizado por um conjunto de validações que visa garantir que apenas o criador da *contract account* pode alterar o armazenamento da mesma.

_dataControllers

Tipo de dados: mapping (string => DataController)
Visibilidade: private
Objetivo: Armazenar todos os controladores de dados.

_dataSubjects

Tipo de dados: mapping (string => DataSubject)
Visibilidade: private
Objetivo: Armazenar todos os controladores de dados.

_regulators

Tipo de dados: mapping (string => Regulator)
Visibilidade: private
Objetivo: Armazenar todas entidades reguladoras.

_consents

Tipo de dados: mapping (string => Consent)
Visibilidade: private
Objetivo: Armazenar todas as operações de consentimento.

_terms

Tipo de dados: mapping (string => Terms)
Visibilidade: private
Objetivo: Armazenar todos os termos de consentimento.

_removedConsents

Tipo de dados: mapping (string => RemovedConsent)
Visibilidade: private
Objetivo: Armazenar todos os consentimentos removidos.

_feedbacks

Tipo de dados: mapping (string => Feedback)
Visibilidade: private
Objetivo: Armazenar todo o *feedback* relativo a operações de consentimento.

4.4.4 Eventos

O principal objetivo dos eventos definidos neste ponto é o de solicitar que a rede informe quando determinados eventos ocorrem. Esta funcionalidade permite que membros da rede, que subscreveram um ou mais destes eventos, sejam informados em tempo real de determinadas ocorrências no seio da *contract account*.

NewConsent

Objetivo: Informar todos os membros da rede que subscreveram eventos deste tipo que foi persistido um novo consentimento.
Valores emitidos:
string consentUid – Identificador do novo consentimento.
string dataSubjectUid – Identificador do sujeito dos dados associado ao novo consentimento.
string dataControllerName – Designação do controlador de dados associado ao novo consentimento.
string regulatorName – Designação da entidade reguladora associada ao novo consentimento.

NewRemovedConsent

Objetivo: Informar todos os membros da rede que subscreveram eventos deste tipo que foi removido um consentimento.
Valores emitidos:
string consentUid – Identificador do consentimento removido.
string dataSubjectUid – Identificador do sujeito dos dados associado ao consentimento removido.
string dataControllerName – Designação do controlador de dados associado ao consentimento removido.
string regulatorName – Designação da entidade reguladora associada ao consentimento removido.

NewFeedbackEntry

Objetivo: Informar todos os membros da rede que subscreveram eventos deste tipo que existe uma nova entrada de *feedback*.
Valores emitidos:
string consentUid – Identificador do consentimento associado à entrada de *feedback*.
string dataSubjectUid – Identificador do sujeito dos dados associado à entrada de *feedback*.
string dataControllerName – Designação do controlador de dados associado à entrada de *feedback*.
string regulatorName – Designação da entidade reguladora associada à entrada de *feedback*.

4.4.5 Transações

O objetivo das transações descritas neste ponto é o de persistir novos dados na *contract account* do sistema. Tal implica a alteração das variáveis da mesma. Apenas um membro de rede na posse da carteira que deu origem à *contract account* será capaz de levar a cabo estas transações com sucesso. Este comportamento é assegurado pela primeira validação efetuada aquando da execução de cada uma delas. A invocação destas transações implica o pagamento de uma taxa à rede referente ao custo computacional necessário à execução da lógica da mesma e ao armazenamento adicional despendido.

addDataSubject

Objetivo: Registar um novo sujeito dos dados na *contract account* do sistema.
Visibilidade: public
***Inputs*:**
string uid – Identificador do sujeito dos dados.
string publicKey – Chave pública do sujeito dos dados codificada em Base64.
string signature – Valor da assinatura digital de registo do sujeito dos dados.
***Outputs*:**
string result – Mensagem de retorno.
Validações:
– Apenas a carteira que criou a *contract account* pode executar com sucesso a transação.
– O identificador do sujeito dos dados tem pelo menos um carácter.
– O valor da assinatura digital do sujeito dos dados tem pelo menos um carácter.
– A chave pública do sujeito dos dados tem pelo menos um carácter.

– O identificador do sujeito dos dados está disponível.
Eventos emitidos: N/A

addDataController

Objetivo: Registar um novo controlador de dados na *contract account* do sistema.
Visibilidade: public
***Inputs*:**
string name – Designação do controlador de dados.
string publicKey – Chave pública do controlador de dados codificada em Base64.
string signature – Valor da assinatura digital de registo do controlador de dados.
***Outputs*:**
string result – Mensagem de retorno.
Validações:
– Apenas a carteira que criou a *contract account* pode executar com sucesso a transação.
– A designação do controlador de dados tem pelo menos um carácter.
– O valor da assinatura digital do controlador de dados tem pelo menos um carácter.
– A chave pública do controlador de dados tem pelo menos um carácter.
– A designação do controlador de dados está disponível.
Eventos emitidos: N/A

addRegulator

Objetivo: Registar uma nova entidade reguladora na *contract account* do sistema.
Visibilidade: public
***Inputs*:**
string name – Designação da entidade reguladora.
string publicKey – Chave pública da entidade reguladora codificada em Base64.
string signature – Valor da assinatura digital de registo da entidade reguladora.
***Outputs*:**
string result – Mensagem de retorno.
Validações:
– Apenas a carteira que criou a *contract account* pode executar com sucesso a transação.
– A designação da entidade reguladora tem pelo menos um carácter.
– O valor da assinatura digital da entidade reguladora tem pelo menos um carácter.
– A chave pública da entidade reguladora tem pelo menos um carácter.
– A designação da entidade reguladora está disponível.
Eventos emitidos: N/A

addTerms

Objetivo: Registar um novo termo de consentimento na *contract account* do sistema.
Visibilidade: public
***Inputs*:**
string uid – Identificador do termo de consentimento.
string termsText – Texto do termo de consentimento.
string dataControllerName – Designação do controlador de dados que pretende registar o termo de consentimento.

string termsSig – Valor da assinatura digital que prova que o termo de consentimento foi criado pelo controlador de dados.
***Outputs*:**
string result – Mensagem de retorno.
Validações:
– Apenas a carteira que criou a *contract account* pode executar com sucesso a transação.
– O identificador do termo de consentimento tem pelo menos um carácter.
– O texto do termo de consentimento tem pelo menos um carácter.
– A designação do controlador de dados tem pelo menos um carácter.
– O valor da assinatura digital tem pelo menos um carácter.
– O controlador de dados referido existe.
– O identificador do termo de utilização está disponível.
Eventos emitidos: N/A

addConsent

Objetivo: Registar um novo consentimento na *contract account* do sistema.
Visibilidade: public
***Inputs*:**
string uid – Identificador do consentimento.
string termsUid – Identificador do termo de consentimento associado ao consentimento.
string dataSubjectUid – Identificador do sujeito dos dados associado ao consentimento.
string dataControllerName – Designação do controlador de dados associado ao consentimento.
string regulatorName – Designação da entidade reguladora associada ao consentimento.
string dataDigest – Resumo criptográfico (codificado em hexadecimal), segundo o algoritmo SHA-512, da informação associada ao consentimento.
string requestSig – Valor da assinatura digital que atesta que o pedido de consentimento teve origem no controlador de dados referido.
string consentSig – Valor da assinatura digital que prova o consentimento do sujeito dos dados.
***Outputs*:**
string result – Mensagem de retorno.
Validações:
– Apenas a carteira que criou a *contract account* pode executar com sucesso a transação.
– O identificador do consentimento tem pelo menos um carácter.
– O identificador do termo de consentimento tem pelo menos um carácter.
– O identificador do sujeito dos dados tem pelo menos um carácter.
– A designação do controlador de dados tem pelo menos um carácter.
– A designação da entidade reguladora tem pelo menos um carácter.
– O texto do termo de consentimento tem pelo menos um carácter.
– O valor do resumo criptográfico dos dados tem pelo menos um carácter.
– O valor da assinatura digital de consentimento tem pelo menos um carácter.
– O valor da assinatura digital do pedido tem pelo menos um carácter.
– O termo de consentimento referido existe no sistema.
– O sujeito dos dados referido existe no sistema.
– O controlador de dados referido existe no sistema.
– O termo de consentimento foi criado pelo controlador de dados referido.
– A entidade reguladora referida existe no sistema.

– O identificador do consentimento está disponível.
Eventos emitidos: NewConsent

addRemovedConsent

Objetivo: Registar uma nova remoção de consentimento na *contract account* do sistema.
Visibilidade: public
***Inputs*:**
string consentUid – Identificador do consentimento.
string removalSig – Valor da assinatura digital que atesta a vontade do sujeito dos dados em remover o consentimento.
***Outputs*:**
string result – Mensagem de retorno.
Validações:
– Apenas a carteira que criou a *contract account* pode executar com sucesso a transação.
– O identificador do consentimento tem pelo menos um carácter.
– O valor da assinatura digital de remoção do consentimento tem pelo menos um carácter.
– O consentimento existe.
– O consentimento ainda não foi removido.
Eventos emitidos: NewRemovedConsent

addFeedbackEntry

Objetivo: Registar uma nova entrada de *feedback* na *contract account* do sistema.
Visibilidade: public
***Inputs*:**
string consentUid – Identificador do consentimento associado ao *feedback*.
uint feedbackType – Nota atribuída pelo sujeito dos dados ao desempenho do controlador de dados.
string message – Mensagem deixada pelo sujeito dos dados relativamente ao desempenho do controlador de dados.
string feedbackSig – Valor da assinatura digital que prova que a entrada de *feedback* teve origem no sujeito dos dados associado ao consentimento referido.
***Outputs*:**
string result – Mensagem de retorno.
uint entryIndex – Índice da nova entrada na lista de *feedback* do consentimento.
Validações:
– Apenas a carteira que criou a *contract account* pode executar com sucesso a transação.
– A mensagem tem pelo menos um carácter.
– O valor da assinatura digital tem pelo menos um carácter.
– O valor da nota é valido.
– O consentimento associado à entrada de *feedback* existe.
Eventos emitidos: NewFeedbackEntry

4.4.6 *Views*

As *views* especificadas para a *contract account* têm como objetivo o acesso a um conjunto de dados armazenados na mesma. A invocação de uma *view* não implica o

pagamento de uma taxa à rede, uma vez que a leitura do armazenamento de uma *contract account* não altera o seu estado.

getDataSubject

Objetivo: Ler um sujeito dos dados persistido na *contract account* do sistema.
Visibilidade: public
***Inputs*:**
string dataSubjectUid – Identificador do sujeito dos dados a ler.
***Outputs*:**
string uid – Identificador do sujeito dos dados lido.
string signature – Valor da assinatura digital de registo do sujeito dos dados lido.
string publicKey – Chave pública (codificada em Base64) associada ao sujeito dos dados lido.
uint timestamp – Selo temporal do momento de criação do bloco que armazena os dados do sujeito dos dados lido.
Validações:
- O identificador do sujeito dos dados tem pelo menos um carácter.
- Existe um sujeito dos dados com o identificador referido.

getDataController

Objetivo: Ler um controlador de dados persistido na *contract account* do sistema.
Visibilidade: public
***Inputs*:**
string dataControllerName – Designação do controlador de dados a ler.
***Outputs*:**
string name – Designação do controlador de dados lido.
string publicKey – Chave pública (codificada em Base64) associada ao controlador de dados lido.
string signature – Valor da assinatura digital de registo do controlador de dados lido.
uint timestamp – Selo temporal do momento de criação do bloco que armazena os dados do controlador de dados lido.
Validações:
- A designação do controlador de dados tem pelo menos um carácter.
- Existe um controlador de dados com o identificador referido.

getRegulator

Objetivo: Ler uma entidade reguladora persistida na *contract account* do sistema.
Visibilidade: public
***Inputs*:**
string regulatorName – Designação da entidade reguladora a ler.
***Outputs*:**
string name – Designação da entidade reguladora lida.
string publicKey – Chave pública (codificada em Base64) associada à entidade reguladora lida.
string signature – Valor da assinatura digital de registo da entidade reguladora lida.

uint timestamp – Selo temporal do momento de criação do bloco que armazena os dados da entidade reguladora lida.
Validações:
– A designação da entidade reguladora tem pelo menos um carácter.
– Existe uma entidade reguladora com o identificador referido.

getTerms

Objetivo: Ler um termo de consentimento persistido na *contract account* do sistema.
Visibilidade: public
***Inputs*:**
string termsUid – Identificador do termo de consentimento a ler.
***Outputs*:**
string uid – Identificador do termo de consentimento lido.
string consentTerms – Texto do termo de consentimento lido.
string dataControllerName – Designação do controlador de dados que registou o termo de consentimento lido.
string termsSig – Valor da assinatura digital que prova que o termo de consentimento lido foi criado pelo controlador de dados associado.
uint timestamp – Selo temporal do momento de criação do bloco que armazena os dados do termo de consentimento lido.
Validações:
– O identificador do termo de consentimento tem pelo menos um carácter.
– Existe um termo de consentimento com o identificador referido.

getConsentParts

Objetivo: Obter as designações e identificadores de todas partes envolvidas num consentimento persistido *contract account* do sistema.
Visibilidade: public
***Inputs*:**
string consentUid – Identificador do consentimento a ler.
***Outputs*:**
string dataSubjectUid – Identificador do sujeito dos dados envolvido no consentimento.
string dataControllerName – Designação do controlador de dados envolvido no consentimento.
string termsUid – Identificador do termo de consentimento associado ao consentimento.
string regulatorName – Designação da entidade reguladora associada ao consentimento.
Validações:
– O identificador do consentimento tem pelo menos um carácter.
– Existe um consentimento com o identificador referido.

getConsentValidationData

Objetivo: Obter os dados necessários à validação de um consentimento persistido na *contract account* do sistema.
Visibilidade: public
***Inputs*:**
string consentUid – Identificador do consentimento.

***Outputs*:**
string dataSubjectUid – Identificador do sujeito dos dados envolvido no consentimento.
string termsUid – Identificador do termo de consentimento associado ao consentimento.
string dataDigest – Resumo criptográfico (codificado em hexadecimal) dos dados associados ao consentimento.
string requestSig – Valor da assinatura digital que atesta que o pedido de consentimento lido teve origem no controlador de dados associado.
string consentSig – Valor da assinatura digital que prova o consentimento lido foi dado pelo sujeito dos dados associado.
uint timestamp – Selo temporal do momento de criação do bloco que armazena os dados do consentimento lido.
Validações:
- O identificador do consentimento tem pelo menos um carácter.
- Existe um consentimento com o identificador referido.

getRemovedConsent

Objetivo: Ler um consentimento removido persistido na *contract account* do sistema.
Visibilidade: public
***Inputs*:**
string consentUid – Identificador do consentimento removido a ler.
***Outputs*:**
string removalSig – Valor da assinatura digital que atesta a vontade do sujeito dos dados em que o consentimento fosse removido.
uint timestamp – Selo temporal do momento de criação do bloco que armazena os dados da remoção de consentimento lida.
Validações:
- O identificador do consentimento tem pelo menos um carácter.
- Existe uma remoção associada ao consentimento referido.

getFeedbackEntry

Objetivo: Ler uma entrada de *feedback* persistida na *contract account* do sistema.
Visibilidade: public
***Inputs*:**
string consentUid – Identificador do consentimento ao qual a entrada de *feedback* está associada.
uint entryIndex – Índice da entrada de *feedback* a ser lida na lista de de *feedbacks* asociados ao consentimento referido.
***Outputs*:**
uint feedbackType – Nota associada à entrada de *feedback* lida.
string message – Mensagem deixada pelo sujeito dos dados na entrada de *feedback* lida.
string feedbackSig – Valor da assinatura digital que prova que a entrada de *feedback* lida foi criada pelo sujeito dos dados associado.
uint timestamp – Selo temporal do momento de criação do bloco que armazena os dados da entrada de *feedback* lida.
Validações:
- O identificador do consentimento tem pelo menos um carácter.
- Existe pelo menos uma entrada de *feedback* associada ao consentimento referido.

– Existe uma entrada de *feedback* com o índice referido.

getFeedbackEntryCount

Objetivo: Contar as entradas de *feedback* associadas a um consentimento persistido na *contract account* do sistema.
Visibilidade: public
***Inputs*:**
string consentUid – Identificador do consentimento.
***Outputs*:**
uint count – Contagem de entradas de *feedback* associadas ao consentimento referido.
Validações:
– O identificador do consentimento tem pelo menos um carácter.
– Existe pelo menos uma entrada de *feedback* associada ao consentimento referido.

4.5 *Connsent* REST API

O cenário proposto para o sistema *Connsent* no Ponto 1.2 prevê que este seja capaz de suportar o subsistema reputacional e abstrair as operações com a rede *blockchain* de todos os atores do sistema, devendo esta abstração acontecer independentemente do ator recorrer à aplicação móvel *Connsent* ou a qualquer outro sistema que integre com o sistema *Connsent*. Para tal, a solução prevê a implementação de uma REST API que deverá definir os *endpoints* necessários à execução de todas as operações previstas no âmbito do sistema, sejam estas relacionadas com a persistência na rede *blockchain* ou com o subsistema reputacional. Esta abordagem permitirá que, posteriormente, qualquer sistema externo que pretenda integrar com a solução o possa fazer através de uma REST API que poderá ser consumida em qualquer plataforma de desenvolvimento. A própria aplicação móvel *Connsent* recorrerá à mesma para comunicar com o *back end* do sistema. Todas as operações de escrita do sistema na rede *blockchain* só serão possíveis se forem levadas a cabo por meio desta REST API, uma vez que é este componente do sistema *Connsent* que se encontra configurado com a carteira que originou *contract account* do sistema. Os *endpoints* que estão previstos para a REST API, e que são detalhadas de seguida, têm por finalidade servir como interfaces para a entrada e saída de dados do *back end* do sistema *Connsent*, dando assim resposta às necessidades de integração de componentes da própria plataforma *Connsent* e de sistemas sob o controlo de terceiros (controladores de dados, entidades reguladoras, *et cetera*) que sejam utilizados pelos atores do sistema identificados no Ponto 4.1.1.

/subject (POST)

Objetivo: *Endpoint* que deverá ser invocado sempre que um novo sujeito dos dados pretenda registar-se no sistema.
Operações no *back end*:
– Persistência do novo sujeito dos dados na *contract account* associada ao sistema.
***Inputs*:**
uid – Identificador do novo sujeito dos dados.
publicKey – Chave pública do novo sujeito dos dados.
signature – Assinatura de registo de entidade.

***Outputs*:**
transactionHash – Resumo criptográfico da transação de persistência.
link – Hiperligação para a página da transação de persistência.
contractAddress – Endereço da *contract account* onde foi persistido o novo sujeito dos dados.
Resposta HTTP:
Código HTTP 200 – Sujeito dos dados registado com sucesso.
Código HTTP 400 – *Input(s)* obrigatório(s) em falta.
Código HTTP 400 – Assinatura digital de registo de entidade inválida.
Código HTTP 500 – Erro interno no servidor.

/subject (GET)

Objetivo: *Endpoint* que deverá ser invocado sempre que se pretenda obter a informação relativa a um sujeito dos dados.
Operações no *back end*:
– Leitura da informação relativa ao sujeito dos dados.
***Inputs*:**
uid – Identificador do sujeito dos dados.
***Outputs*:**
uid – Identificador do sujeito dos dados.
publicKey – Chave pública do sujeito dos dados.
signature – Assinatura de registo de entidade do sujeito dos dados.
created – Selo temporal de criação do bloco que armazena os dados do sujeito dos dados.
Resposta HTTP:
Código HTTP 200 – Sujeito dos dados obtido com sucesso.
Código HTTP 400 – Identificador do sujeito dos dados em falta.
Código HTTP 404 – Sujeito dos dados não encontrado.
Código HTTP 500 – Erro interno no servidor.

/subject/consents (GET)

Objetivo: *Endpoint* que deverá ser invocado sempre que se pretenda obter todos os consentimentos dados por um determinado sujeito dos dados.
Operações no *back end*:
– Leitura da informação relativa ao sujeito dos dados.
– Leitura de todos eventos passados do tipo "*NewConsentEvent*" que refiram o sujeito dos dados em questão.
– Leitura de todos os consentimentos que referem o sujeito dos dados.
***Inputs*:**
uid – Identificador do sujeito dos dados.
***Outputs*:**
– Lista de todos os consentimentos dados pelo sujeito dos dados.
Resposta HTTP:
Código HTTP 200 – Consentimentos do sujeito dos dados lidos com sucesso.
Código HTTP 400 – Identificador do sujeito dos dados em falta.
Código HTTP 404 – Sujeito dos dados não encontrado.
Código HTTP 500 – Erro interno no servidor.

/subject/consents/removed (GET)

Objetivo: *Endpoint* que deverá ser invocado sempre que se pretenda obter todos os consentimentos removidos por um determinado sujeito dos dados.
Operações no *back end*:
– Leitura da informação relativa ao sujeito dos dados.
– Leitura de todos eventos passados do tipo "*NewRemovedConsentEvent*" que refiram o sujeito dos dados em questão.
– Leitura de todas as remoções de consentimento que referem o sujeito dos dados.
***Inputs*:**
uid – Identificador do sujeito dos dados.
***Outputs*:**
– Lista de todos as remoções de consentimento efetuadas pelo sujeito dos dados.
Resposta HTTP:
Código HTTP 200 – Remoções de consentimento do sujeito dos dados lidas com sucesso.
Código HTTP 400 – Identificador do sujeito dos dados em falta.
Código HTTP 404 – Sujeito dos dados não encontrado.
Código HTTP 500 – Erro interno no servidor.

/subject/feedbacks (GET)

Objetivo: *Endpoint* que deverá ser invocado sempre que se pretenda obter todas as entradas de *feedback* criadas por um determinado sujeito dos dados.
Operações no *back end*:
– Leitura da informação relativa ao sujeito dos dados.
– Leitura de todos eventos passados do tipo "*NewFeedbackEntryEvent*" que refiram o sujeito dos dados em questão.
– Leitura de todas as entradas de *feedback* que referem o sujeito dos dados.
***Inputs*:**
uid – Identificador do sujeito dos dados.
***Outputs*:**
– Lista de todas as entradas de *feedback* criadas pelo sujeito dos dados.
Resposta HTTP:
Código HTTP 200 – Entradas de *feedback* do sujeito dos dados lidas com sucesso.
Código HTTP 400 – Identificador do sujeito dos dados em falta.
Código HTTP 404 – Sujeito dos dados não encontrado.
Código HTTP 500 – Erro interno no servidor.

/controller (POST)

Objetivo: *Endpoint* que deverá ser invocado sempre que um novo controlador de dados pretenda registar-se no sistema.
Operações no *back end*:
– Persistência do novo controlador de dados na *contract account* associada ao sistema.
***Inputs*:**
name – Designação do novo controlador de dados.
publicKey – Chave pública do novo controlador de dados.
signature – Assinatura de registo de entidade.
***Outputs*:**
transactionHash – Resumo criptográfico da transação de persistência.

link – Hiperligação para a página da transação de persistência.
contractAddress – Endereço da *contract account* onde foi persistido o novo controlador de dados.
Resposta HTTP:
Código HTTP 200 – Controlador de dados registado com sucesso.
Código HTTP 400 – *Input(s)* obrigatório(s) em falta.
Código HTTP 400 – Assinatura digital de registo de entidade inválida.
Código HTTP 500 – Erro interno no servidor.

/controller (GET)

Objetivo: *Endpoint* que deverá ser invocado sempre que se pretenda obter a informação relativa a um controlador de dados.
Operações no *back end*:
– Leitura da informação relativa ao controlador de dados.
***Inputs*:**
name – Designação do controlador de dados.
***Outputs*:**
name – Designação do controlador de dados.
publicKey – Chave pública do controlador de dados.
signature – Assinatura de registo de entidade do controlador de dados.
created – Selo temporal de criação do bloco que armazena os dados do controlador de dados.
Resposta HTTP:
Código HTTP 200 – Controlador de dados obtido com sucesso.
Código HTTP 400 – Identificador do controlador de dados em falta.
Código HTTP 404 – Controlador de dados não encontrado.
Código HTTP 500 – Erro interno no servidor.

/controller/feedback (GET)

Objetivo: *Endpoint* que deverá ser invocado sempre que se pretenda obter a reputação de um determinado controlador de dados.
Operações no *back end*:
– Leitura da informação relativa ao controlador de dados.
– Iterar todos as entradas de *feedback*.
– Calcular reputação do controlador de dados.
***Inputs*:**
name – Designação do controlador de dados.
***Outputs*:**
totalEntries – Contagem do total de avaliações que se qualificam para o cálculo da reputação do controlador de dados.
positivePercentage – Percentagem positiva de entradas de *feedback* relativas ao controlador de dados.
neutralPercentage – Percentagem neutra de entradas de *feedback* relativas ao controlador de dados.
negativePercentage – Percentagem negativa de entradas de *feedback* relativas ao controlador de dados.

negativePercentage – Percentagem negativa de entradas de *feedback* relativas ao controlador de dados.
score – Pontuação do controlador de dados.
negative – Contagem de avaliações negativas que se qualificam para o cálculo da reputação do controlador de dados.
negative – Contagem de avaliações neutral válidas para o cálculo da reputação do controlador de dados.
negative – Contagem de avaliações positivas válidas para o cálculo da reputação do controlador de dados.
entries – Vetor com todas as entradas de *feedback* relativas ao controlador de dados. Cada índice representa uma avaliação e contem a nota, a mensagem e o selo temporal.
Resposta HTTP:
Código HTTP 200 – Reputação do controlador de dados controlador de dados obtida com sucesso.
Código HTTP 400 – Identificador do controlador de dados em falta.
Código HTTP 404 – Controlador de dados não encontrado.
Código HTTP 500 – Erro interno no servidor.

/controller/consents (GET)

Objetivo: *Endpoint* que deverá ser invocado sempre que se pretenda obter todos os consentimentos dados a um controlador de dados.
Operações no *back end*:
– Leitura da informação relativa ao controlador de dados.
– Leitura de todos eventos passados do tipo "*NewConsentEvent*" que refiram o controlador de dados em questão.
– Leitura de todos os consentimentos que referem o controlador de dados.
***Inputs*:**
name – Designação do controlador de dados.
***Outputs*:**
– Lista de todos os consentimentos dados ao controlador de dados.
Resposta HTTP:
Código HTTP 200 – Consentimentos dados ao controlador de dados lidos com sucesso.
Código HTTP 400 – Identificador do controlador de dados em falta.
Código HTTP 404 – Controlador de dados não encontrado.
Código HTTP 500 – Erro interno no servidor.

/controller/consents/removed (GET)

Objetivo: *Endpoint* que deverá ser invocado sempre que se pretenda obter todos os consentimentos removidos a um controlador de dados.
Operações no *back end*:
– Leitura da informação relativa ao controlador de dados.
– Leitura de todos eventos passados do tipo "*NewRemovedConsentEvent*" que refiram o controlador de dados em questão.
– Leitura de todas as remoções de consentimento que referem o controlador de dados.
***Inputs*:**
name – Designação do controlador de dados.
***Outputs*:**

– Lista de todos os consentimentos removidos ao controlador de dados.
Resposta HTTP:
Código HTTP 200 – Consentimentos removidos lidos com sucesso.
Código HTTP 400 – Identificador do controlador de dados em falta.
Código HTTP 404 – Controlador de dados não encontrado.
Código HTTP 500 – Erro interno no servidor.

/controller/feedbacks (GET)

Objetivo: *Endpoint* que deverá ser invocado sempre que se pretenda obter todas as entradas de *feedback* relativas a um determinado controlador de dados.
Operações no *back end*:
– Leitura da informação relativa ao controlador de dados.
– Leitura de todos eventos passados do tipo "*NewFeedbackEntryEvent*" que refiram o controlador de dados em questão.
– Leitura de todas as entradas de *feedback* que referem o controlador de dados.
***Inputs*:**
name – Designação do controlador de dados.
***Outputs*:**
– Lista de todas as entradas de *feedback* relativas ao controlador de dados.
Resposta HTTP:
Código HTTP 200 – Entradas de *feedback* relativas ao controlador de dados lidas com sucesso.
Código HTTP 400 – Identificador do controlador de dados em falta.
Código HTTP 404 – Controlador de dados não encontrado.
Código HTTP 500 – Erro interno no servidor.

/regulator (POST)

Objetivo: *Endpoint* que deverá ser invocado sempre que uma nova entidade reguladora pretenda registar-se no sistema.
Operações no *back end*:
– Persistência de uma entidade reguladora na *contract account* associada ao sistema.
***Inputs*:**
name – Designação da nova entidade reguladora.
publicKey – Chave pública do nova entidade reguladora.
signature – Assinatura de registo de entidade.
***Outputs*:**
transactionHash – Resumo criptográfico da transação de persistência.
link – Hiperligação para a página da transação de persistência.
contractAddress – Endereço da *contract account* onde foi persistido a nova entidade reguladora.
Resposta HTTP:
Código HTTP 200 – Entidade reguladora registada com sucesso.
Código HTTP 400 – *Input(s)* obrigatório(s) em falta.
Código HTTP 400 – Assinatura digital de registo de entidade inválida.
Código HTTP 500 – Erro interno no servidor.

/regulator (GET)

Objetivo: *Endpoint* que deverá ser invocado sempre que se pretenda obter a informação relativa a uma entidade reguladora.
Operações no *back end*:
– Leitura da informação relativa a uma entidade reguladora.
***Inputs*:**
name – Designação da entidade reguladora.
***Outputs*:**
name – Designação da entidade reguladora.
publicKey – Chave pública da entidade reguladora.
signature – Assinatura de registo de entidade da entidade reguladora
created – Selo temporal de criação do bloco que armazena os dados da entidade reguladora.
Resposta HTTP:
Código HTTP 200 – Entidade reguladora obtida com sucesso.
Código HTTP 400 – Identificador da entidade reguladora em falta.
Código HTTP 404 – Entidade reguladora não encontrada.
Código HTTP 500 – Erro interno no servidor.

/regulator/consents (GET)

Objetivo: *Endpoint* que deverá ser invocado sempre que se pretenda obter todos os consentimentos supervisionados por uma determinada entidade reguladora.
Operações no *back end*:
– Leitura da informação relativa à entidade reguladora.
– Leitura de todos eventos passados do tipo "*NewConsentEvent*" que refiram a entidade reguladora em questão.
– Leitura de todos os consentimentos que referem a entidade reguladora.
***Inputs*:**
name – Designação da entidade reguladora.
***Outputs*:**
– Lista de todos os consentimentos supervisionados pela entidade reguladora.
Resposta HTTP:
Código HTTP 200 – Consentimentos supervisionados pela entidade reguladora lidos com sucesso.
Código HTTP 400 – Identificador da entidade reguladora em falta.
Código HTTP 404 – Entidade reguladora não encontrada.
Código HTTP 500 – Erro interno no servidor.

/regulator/consents/removed (GET)

Objetivo: *Endpoint* que deverá ser invocado sempre que se pretenda obter todos os consentimentos removidos supervisionados por uma determinada entidade reguladora.
Operações no *back end*:
– Leitura da informação relativa à entidade reguladora.
– Leitura de todos eventos passados do tipo "*NewRemovedConsentEvent*" que refiram a entidade reguladora em questão.
– Leitura de todas as remoções de consentimento que referem a entidade reguladora.
***Inputs*:**
name – Designação da entidade reguladora.

***Outputs*:**
– Lista de todos os consentimentos removidos supervisionados pela entidade reguladora.
Resposta HTTP:
Código HTTP 200 – Consentimentos removidos lidos com sucesso.
Código HTTP 400 – Identificador da entidade reguladora em falta.
Código HTTP 404 – Entidade reguladora não encontrada.
Código HTTP 500 – Erro interno no servidor.

/regulator/feedbacks (GET)

Objetivo: *Endpoint* que deverá ser invocado sempre que se pretenda obter todas as entradas de *feedback* relativas a consentimentos supervisionados por uma determinada entidade reguladora.
Operações no *back end*:
– Leitura da informação relativa à entidade reguladora
– Leitura de todos eventos passados do tipo "*NewFeedbackEntryEvent*" que refiram a entidade reguladora em questão.
– Leitura de todas as entradas de *feedback* que referem a entidade reguladora.
***Inputs*:**
name – Designação da entidade reguladora.
***Outputs*:**
– Lista de todas as entradas de *feedback* relativas consentimentos supervisionados pela entidade reguladora.
Resposta HTTP:
Código HTTP 200 – Entradas de *feedback* relativas a consentimentos supervisionados pela entidade reguladora lidas com sucesso.
Código HTTP 400 – Identificador da entidade reguladora em falta.
Código HTTP 404 – Entidade reguladora não encontrada.
Código HTTP 500 – Erro interno no servidor.

/terms (POST)

Objetivo: *Endpoint* que deverá ser invocado sempre que um controlador de dados pretenda registar um novo termo de consentimento.
Operações no *back end*:
– Leitura do controlador de dados que pretende registar o novo termo de consentimento.
– Verificação da assinatura de registo do termo de consentimento.
– Persistência do novo termo de consentimento na *contract account* associada ao sistema.
***Inputs*:**
uid – Identificador do novo termo de consentimento.
termsText – Texto do termo de consentimento.
controllerName – Designação do controlador de dados.
signature – Assinatura de registo do termo de consentimento.
***Outputs*:**
transactionHash – Resumo criptográfico da transação de persistência.
link – Hiperligação para a página da transação de persistência.
contractAddress – Endereço da *contract account* onde foi persistido o novo controlador de dados.
Resposta HTTP:

Código HTTP 200 – Termo de consentimento registado com sucesso.
Código HTTP 400 – *Input(s)* obrigatório(s) em falta.
Código HTTP 400 – Assinatura digital de registo do termo de consentimento inválida.
Código HTTP 404 – Controlador de dados não encontrado.
Código HTTP 500 – Erro interno no servidor.

/terms (GET)

Objetivo: *Endpoint* que deverá ser invocado sempre que se pretenda obter a informação relativa a um termos de consentimento.
Operações no *back end*:
– Leitura da informação relativa ao termo de consentimento.
***Inputs*:**
uid – Identificador do termo de consentimento.
***Outputs*:**
uid – Identificador do termo de consentimento.
termsText – Texto do termo de consentimento.
signature – Assinatura de registo de entidade do controlador de dados.
controllerName – Designação do controlador de dados que registou o termo.
signature – Assinatura digital de registo do termo de consentimento.
created – Selo temporal do momento em que o bloco que armazena o termo de consentimento foi criado.
Resposta HTTP:
Código HTTP 200 – Termo de consentimento obtido com sucesso.
Código HTTP 400 – Identificador do termo de consentimento em falta.
Código HTTP 404 – Termo de consentimento não encontrado.
Código HTTP 500 – Erro interno no servidor.

/request (POST)

Objetivo: *Endpoint* que o sistema do controlador de dados deverá invocar sempre que pretender obter o consentimento de um sujeito dos dados para uma determinada operação que envolva os seus dados pessoais.
Operações no *back end*:
– Leitura do sujeito dos dados referido no pedido de consentimento.
– Leitura do termo de consentimento referido no pedido de consentimento.
– Leitura do controlador de dados referido no pedido de consentimento.
– Leitura da entidade reguladora referida no pedido de consentimento.
– Verificação da assinatura do pedido de consentimento.
– Envio de notificação *push* para o dispositivo móvel do sujeito dos dados.
***Inputs*:**
uid – Identificador do novo consentimento.
subjectUid – Identificador do sujeito dos dados a quem vai ser pedido o consentimento.
termsUid – Identificador do termo de consentimento proposto ao sujeito dos dados.
controllerName – Designação do controlador de dados.
regulatorName – Designação da entidade reguladora que supervisionará o consentimento.
dataDigest – Resumo criptográfico dos dados do sujeito dos dados que o controlador de dados capturou.
requestSig – Assinatura digital de pedido de consentimento.

***Outputs*:**
Não aplicável.
Resposta HTTP:
Código HTTP 200 – Pedido realizado com sucesso.
Código HTTP 400 – *Input(s)* obrigatório(s) em falta.
Código HTTP 400 – Assinatura digital do pedido inválida.
Código HTTP 404 – Sujeito dos dados referido não encontrado.
Código HTTP 404 – Termo de consentimento referido não encontrado.
Código HTTP 404 – Controlador de dados referido não encontrado.
Código HTTP 404 – Entidade reguladora referida não encontrada.
Código HTTP 500 – Erro interno no servidor.

/consent (POST)

Objetivo: *Endpoint* invocado sempre que se procede ao envio do consentimento de um sujeito dos dados para com um pedido efetuado por um controlador de dados.
Operações no *back end*:
– Leitura do sujeito dos dados referido no pedido de consentimento.
– Leitura do termo de consentimento referido no pedido de consentimento.
– Leitura do controlador de dados referido no pedido de consentimento.
– Leitura da entidade reguladora referida no pedido de consentimento.
– Verificação da assinatura do pedido de consentimento.
– Verificação da assinatura de consentimento.
– Persistência do consentimento na *contract account* associada ao sistema.
***Inputs*:**
uid – Identificador do novo consentimento.
subjectUid – Identificador do sujeito dos dados a quem vai ser pedido o consentimento.
termsUid – Identificador do termo de consentimento proposto ao sujeito dos dados.
controllerName – Designação do controlador de dados.
regulatorName – Designação da entidade reguladora que supervisionará o consentimento.
dataDigest – Resumo criptográfico dos dados do sujeito dos dados que o controlador de dados capturou.
requestSig – Assinatura digital de pedido de consentimento.
consentSig – Assinatura digital de consentimento.
***Outputs*:**
transactionHash – Resumo criptográfico da transação de persistência.
link – Hiperligação para a página da transação de persistência.
contractAddress – Endereço da *contract account* onde foi persistido o novo consentimento.
Resposta HTTP:
Código HTTP 200 – Consentimento persistido com sucesso.
Código HTTP 400 – *Input(s)* obrigatório(s) em falta.
Código HTTP 400 – Assinatura digital do pedido inválida.
Código HTTP 400 – Assinatura digital de consentimento inválida.
Código HTTP 404 – Sujeito dos dados referido não encontrado.
Código HTTP 404 – Termo de consentimento referido não encontrado.
Código HTTP 404 – Controlador de dados referido não encontrado.
Código HTTP 404 – Entidade reguladora referida não encontrada.
Código HTTP 500 – Erro interno no servidor.

/consent (GET)

Objetivo: *Endpoint* invocado para obter informação relativa a um determinado consentimento.
Operações no *back end*:
– Leitura das designações e identificadores das partes envolvidas no consentimento.
– Leitura dos dados de validação do consentimento.
***Inputs*:**
uid – Identificador do consentimento.
***Outputs*:**
uid – Identificador do consentimento.
termsUid – Identificador do termo de consentimento associado ao consentimento.
subjectUid – Identificador do sujeito dos dados associado ao consentimento.
controllerName – Designação do controlador de dados associado ao consentimento.
regulatorName – Designação da entidade reguladora associada ao consentimento.
dataDigest – Resumo criptográfico dos dados associados ao consentimento.
requestSig – Assinatura do pedido de consentimento.
consentSig – Assinatura de consentimento.
created – Selo temporal do momento em que o bloco que armazena o consentimento foi criado.
Resposta HTTP:
Código HTTP 200 – Obtido com sucesso.
Código HTTP 400 – Identificador do consentimento não enviado.
Código HTTP 404 – Consentimento não encontrado.
Código HTTP 500 – Erro interno no servidor.

/consent (DELETE)

Objetivo: *Endpoint* invocado sempre que se procede à remoção de um consentimento previamente dado por um sujeito dos dados.
Operações no *back end*:
– Leitura dos dados validação do consentimento a ser removido.
– Validação da assinatura de remoção do consentimento.
– Persistência da remoção de consentimento na *contract account* associada ao sistema.
***Inputs*:**
consentUid – Identificador do consentimento a ser removido.
removalSig – Assinatura de remoção do consentimento.
***Outputs*:**
transactionHash – Resumo criptográfico da transação de persistência.
link – Hiperligação para a página da transação de persistência.
contractAddress – Endereço da *contract account* onde foi persistida a remoção do consentimento.
Resposta HTTP:
Código HTTP 200 – Remoção efetuada com sucesso.
Código HTTP 400 – *Input(s)* obrigatório(s) em falta.
Código HTTP 400 – Assinatura digital de remoção inválida.
Código HTTP 404 – Consentimento a remover não encontrado.
Código HTTP 500 – Erro interno no servidor.

/consent/validate (GET)

Objetivo: *Endpoint* invocado para validar todas as provas e entidades relacionadas com um determinado consentimento.
Operações no *back end*:
– Leitura das partes envolvidas no consentimento.
– Leitura dos dados de validação do consentimento.
– Leitura do sujeito dos dados associado ao consentimento.
– Verificação da assinatura de registo do sujeito dos dados associado ao consentimento.
– Leitura da entidade reguladora associada ao consentimento.
– Verificação da assinatura de registo da entidade controladora associada ao consentimento.
– Leitura do controlador de dados associado ao consentimento.
– Verificação da assinatura de registo do controlador de dados associado ao consentimento.
– Leitura do termo de consentimento associado ao consentimento.
– Verificação da assinatura de criação do termo de consentimento associado ao consentimento.
– Verificação da assinatura de pedido de consentimento.
– Verificação da assinatura de consentimento.
***Inputs*:**
uid – Identificador do consentimento.
***Outputs*:**
validSubjectSignature – Valor booleano que indica a validade da assinatura de registo do sujeito dos dados.
validRegulatorSignature – Valor booleano que indica a validade da assinatura de registo da entidade reguladora.
validControllerSignature – Valor booleano que indica a validade da assinatura de registo do controlador de dados.
validTermsSignature – Valor booleano que indica a validade da assinatura de criação do termo de consentimento.
validConsentRequestSignature – Valor booleano que indica a validade da assinatura de pedido de consentimento.
validConsentSignature – Valor booleano que indica a validade da assinatura de consentimento.
valid – Valor booleano que indica a validade do consentimento.
Resposta HTTP:
Código HTTP 200 – Operações de verificação efetuadas com sucesso
Código HTTP 400 – Identificador do consentimento não enviado.
Código HTTP 404 – Consentimento não encontrado.
Código HTTP 404 – Sujeito dos dados não encontrado.
Código HTTP 404 – Entidade regulador não encontrada.
Código HTTP 404 – Controlador de dados não encontrado.
Código HTTP 404 – Termo de consentimento não encontrado.
Código HTTP 500 – Erro interno no servidor.

/consent/removal (GET)

Objetivo: *Endpoint* invocado para obter informação relativa à remoção de um determinado consentimento.
Operações no *back end*:

– Leitura das dados de remoção do consentimento.
***Inputs*:**
uid – Identificador do consentimento.
***Outputs*:**
uid – Identificador do consentimento.
removalSig – Assinatura digital de remoção de consentimento.
created – Selo temporal do momento em que o bloco que armazena a remoção de consentimento foi criado.
Resposta HTTP:
Código HTTP 200 – Obtido com sucesso.
Código HTTP 400 – Identificador do consentimento não enviado.
Código HTTP 404 – Consentimento não encontrado.
Código HTTP 500 – Erro interno no servidor.

/consent/removal/validate (GET)

Objetivo: *Endpoint* invocado para validar todas as provas e entidades relacionadas com a remoção de um determinado consentimento.
Operações no *back end*:
– Leitura da remoção do consentimento.
– Leitura das partes envolvidas no consentimento removido.
– Leitura dos dados de validação do consentimento removido.
– Leitura do sujeito dos dados associado ao consentimento removido.
– Verificação da assinatura de registo do sujeito dos dados associado ao consentimento removido.
– Leitura da entidade reguladora associada ao consentimento removido.
– Verificação da assinatura de registo da entidade controladora associada ao consentimento removido.
– Leitura do controlador de dados associado ao consentimento removido.
– Verificação da assinatura de registo do controlador de dados associado ao consentimento removido.
– Leitura do termo de consentimento associado ao consentimento removido.
– Verificação da assinatura de criação do termo de consentimento associado ao consentimento removido.
– Verificação da assinatura de pedido do consentimento removido.
– Verificação da assinatura do consentimento removido.
– Verificação da assinatura de remoção do consentimento removido.
***Inputs*:**
uid – Identificador do consentimento removido.
***Outputs*:**
validSubjectSignature – Valor booleano que indica a validade da assinatura de registo do sujeito dos dados.
validRegulatorSignature – Valor booleano que indica a validade da assinatura de registo da entidade reguladora.
validControllerSignature – Valor booleano que indica a validade da assinatura de registo do controlador de dados.
validTermsSignature – Valor booleano que indica a validade da assinatura de criação do termo de consentimento.

validConsentRequestSignature – Valor booleano que indica a validade da assinatura de pedido de consentimento.
validConsentSignature – Valor booleano que indica a validade da assinatura de consentimento.
validRemovalSignature – Valor booleano que indica a validade da assinatura de remoção consentimento.
valid – Valor booleano que indica a validade da remoção de consentimento.
Resposta HTTP:
Código HTTP 200 – Operações de verificação efetuadas com sucesso
Código HTTP 400 – Identificador do consentimento não enviado.
Código HTTP 404 – O consentimento não foi removido.
Código HTTP 404 – Consentimento não encontrado.
Código HTTP 404 – Sujeito dos dados não encontrado.
Código HTTP 404 – Entidade regulador não encontrada.
Código HTTP 404 – Controlador de dados não encontrado.
Código HTTP 404 – Termo de consentimento não encontrado.
Código HTTP 500 – Erro interno no servidor.

/feedback (POST)

Objetivo: *Endpoint* invocado sempre que se procede ao envio de uma avaliação de um sujeito dos dados ao desempenho de um controlador de dados.
Operações no *back end*:
– Leitura dos dados de validação do consentimento associado à avaliação.
– Verificação da assinatura de entrada de *feedback*.
– Persistência da nova entrada de *feedback* na *contract account* do sistema.
***Inputs*:**
consentUid – Identificador do consentimento associado à entrada de *feedback*.
feedbackType – Nota associada à entrada de *feedback*.
message – Mensagem associada à entrada de *feedback*.
feedbackSig – Assinatura de entrada de *feedback*.
***Outputs*:**
transactionHash – Resumo criptográfico da transação de persistência.
link – Hiperligação para a página da transação de persistência.
contractAddress – Endereço da *contract account* onde foi persistido o novo consentimento.
Resposta HTTP:
Código HTTP 200 – Nova entrada de *feedback* persistida com sucesso.
Código HTTP 400 – *Input(s)* obrigatório(s) em falta.
Código HTTP 400 – Assinatura digital da entrada de *feedback* inválida.
Código HTTP 400 – Nota inválida.
Código HTTP 404 – Consentimento não encontrado.
Código HTTP 500 – Erro interno no servidor.

/feedback (GET)

Objetivo: *Endpoint* invocado para se obter todas a entras de *feedback* relativas a um determinado consentimento.
Operações no *back end*:
– Leitura das designações e identificadores das partes envolvidas no consentimento.

– Leitura da contagem de entradas de *feedback* associadas ao consentimento.
– Leitura de cada uma das entradas de *feedback* associadas ao consentimento.
***Inputs*:**
uid – Identificador do consentimento.
***Outputs*:**
consentUid – Identificador do consentimento.
termsUid – Identificador do termo de consentimento associado ao consentimento.
subjectUid – Identificador do sujeito dos dados associado ao consentimento.
controllerName – Designação do controlador de dados associado ao consentimento.
regulatorName – Designação da entidade reguladora associada ao consentimento.
entries – Lista de todas as entradas de *feedback* associadas ao consentimento.
Resposta HTTP:
Código HTTP 200 – Obtido com sucesso.
Código HTTP 400 – Identificador do consentimento não enviado.
Código HTTP 404 – Consentimento não encontrado.
Código HTTP 500 – Erro interno no servidor.

/feedback/validate (GET)

Objetivo: *Endpoint* invocado para validar todas as provas e entidades relacionadas com o *feedback* de um determinado consentimento.
Operações no *back end*:
– Leitura de todas as entradas de *feedback* do consentimento.
– Leitura das partes envolvidas no consentimento.
– Leitura dos dados de validação do consentimento.
– Leitura do sujeito dos dados associado ao consentimento.
– Verificação da assinatura de registo do sujeito dos dados associado ao consentimento.
– Leitura da entidade reguladora associada ao consentimento.
– Verificação da assinatura de registo da entidade controladora associada ao consentimento.
– Leitura do controlador de dados associado ao consentimento.
– Verificação da assinatura de registo do controlador de dados associado ao consentimento.
– Leitura do termo de consentimento associado ao consentimento.
– Verificação da assinatura de criação do termo de consentimento associado ao consentimento.
– Verificação da assinatura de pedido de consentimento.
– Verificação da assinatura de consentimento.
– Verificação da assinatura de todas as entradas de consentimento.
***Inputs*:**
uid – Identificador do consentimento.
***Outputs*:**
validSubjectSignature – Valor booleano que indica a validade da assinatura de registo do sujeito dos dados.
validRegulatorSignature – Valor booleano que indica a validade da assinatura de registo da entidade reguladora.
validControllerSignature – Valor booleano que indica a validade da assinatura de registo do controlador de dados.
validTermsSignature – Valor booleano que indica a validade da assinatura de criação do termo de consentimento.

validConsentRequestSignature – Valor booleano que indica a validade da assinatura de pedido de consentimento.

validConsentSignature – Valor booleano que indica a validade da assinatura de consentimento.

validFeedbackEntries – Valor booleano que indica a validade de todas as entradas de *feedback*.

valid – Valor booleano que indica a validade do consentimento.

Resposta HTTP:

Código HTTP 200 – Operações de verificação efetuadas com sucesso

Código HTTP 400 – Identificador do consentimento não enviado.

Código HTTP 404 – Consentimento não encontrado.

Código HTTP 404 – Sujeito dos dados não encontrado.

Código HTTP 404 – Entidade regulador não encontrada.

Código HTTP 404 – Controlador de dados não encontrado.

Código HTTP 404 – Termo de consentimento não encontrado.

Código HTTP 500 – Erro interno no servidor.

4.6 Aplicação Móvel *Connsent*

A aplicação móvel *Connsent* é o elo de ligação do sujeito dos dados ao sistema. Este componente tem um papel vital no sistema tendo em conta que é através dele que o sujeito dos dados interage com o sistema . Como tal, os requisitos do sistema para o sujeito dos dados incluem o acesso a um *smartphone* e a instalação da aplicação móvel no mesmo. As funcionalidades suportadas pela aplicação móvel são-no graças à comunicação com o *back end* do sistema por intermédio da REST API definida no Ponto 4.5. Quando executada pela primeira vez no dispositivo móvel, a aplicação móvel gerará um par de chaves ECDSA com uma força de 224 *bits* que servirá como base para operações de autorização e autenticação do sujeito dos dados no sistema através de assinaturas digitais que serão criadas e validadas com recurso a esse mesmo par de chaves . Finalizada a geração deste par de chaves, a lógica da aplicação móvel tratará de armazenar a chave privada no armazenamento da aplicação móvel e transmitir a chave pública ao *back end do sistema* juntamente com a assinatura de registo de entidade (consultar Ponto 4.2). Esta inicialização marca o registo do sujeito dos dados no sistema e permite-lhe começar a receber pedidos de consentimento enviados por controladores de dados. A atenção do sujeito dos dados para estes pedidos será captada através de notificações *push* que serão enviadas sempre que um controlador de dados pretenda o consentimento do sujeito dos dados para um determinado fim.

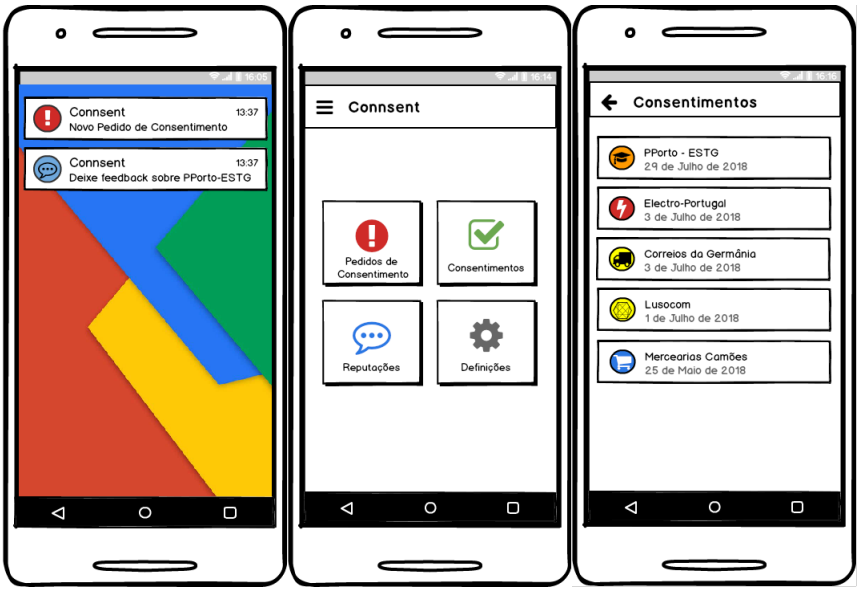

Figura 4.3: *Wireframes* da aplicação móvel *Connsent*

Pretende-se que a vista inicial da aplicação móvel *Connsent*, uma das vistas representadas na Figura 4.3, tenha um *design* simples e que destaque as suas três principais vistas: os pedidos de consentimento recebidos pelo sujeito dos dados, os consentimentos dados pelo sujeito dos dados e a reputação dos controladores de dados. Cada uma destas áreas focar-se-á especificamente numa funcionalidade crítica para o sujeito dos dados. A vista de pedidos de consentimento recebidos pelo sujeito dos dados permitir-lhe-á consultar pedidos de consentimento que recebeu anteriormente e sob os quais ainda não agiu; a vista de consentimentos dados possibilitar-lhe-á a consulta de consentimentos que deu no passado e a eventual remoção de um deles, ou deixar uma avaliação ao desempenho do controlador de dados a quem deu o seu consentimento (como previsto no Ponto 4.3); e a vista de consulta da reputação dos controladores de dados que servirá como ponto de entrada no subsistema reputacional embutido no sistema *Connsent* que lhe permitirá aferir a reputação de qualquer controlador de dados registado na plataforma. Está também prevista uma área da aplicação móvel onde o sujeito dos dados poderá alterar determinadas configurações da mesma. A implementação de referência da aplicação móvel *Connsent* destinar-se-á a *smartphones* equipados com o sistema operativo *Android*.

4.7 Implementações de Referência

Como referido anteriormente, está previsto que sistemas sob o controlo de terceiros (controladores de dados, entidades reguladoras, *et cetera*) interajam com componentes do sistema *Connsent*. Como a sua implementação varia de caso para caso e está fora âmbito da solução aqui apresentada, surge a necessidade de especificar implementações de referência para a integração de sistemas de terceiros com componentes do sistema *Connsent* como a REST API ou a *contract account*. Estas especificações são particularmente úteis enquanto exemplos de eventuais integrações.

4.7.1 Plataforma de Captura de Dados

Antes que o controlador de dados possa pedir o consentimento do sujeito dos dados terá que capturar os seus dados pessoais. Apesar da captura de dados de ocorrer num sistema fora do controlo do sistema *Connsent*, a solução contemplará um inquérito sob a forma de uma página *web* para demonstrar como um controlador de dados poderia integrar as plataformas sob o seu controlo com o sistema referido. Nesta integração de referência, representada na Figura 4.4, um sujeito dos dados daria a sua opinião sobre um evento ficcional disponibilizando, também, dados pessoais que o identificam. A submissão do inquérito resultaria no envio dos dados preenchidos ao controlador de dados que detém a plataforma de inquéritos, sendo que este, por sua vez, invocaria a REST API do sistema *Connsent* dando origem a um novo pedido de consentimento. O sistema *Connsent* terá apenas acesso ao resumo criptográfico dos dados capturados ao sujeito dos dados e jamais, em momento algum, terá acesso aos dados. Apesar da solução apresentada não o prever, seria possível estabelecer canais de comunicação cifrada e autenticada entre os utilizadores do sistema tirando partido de esquemas criptográficos de chave pública com suporte para os pares de chaves utilizados no sistema *Connsent*.

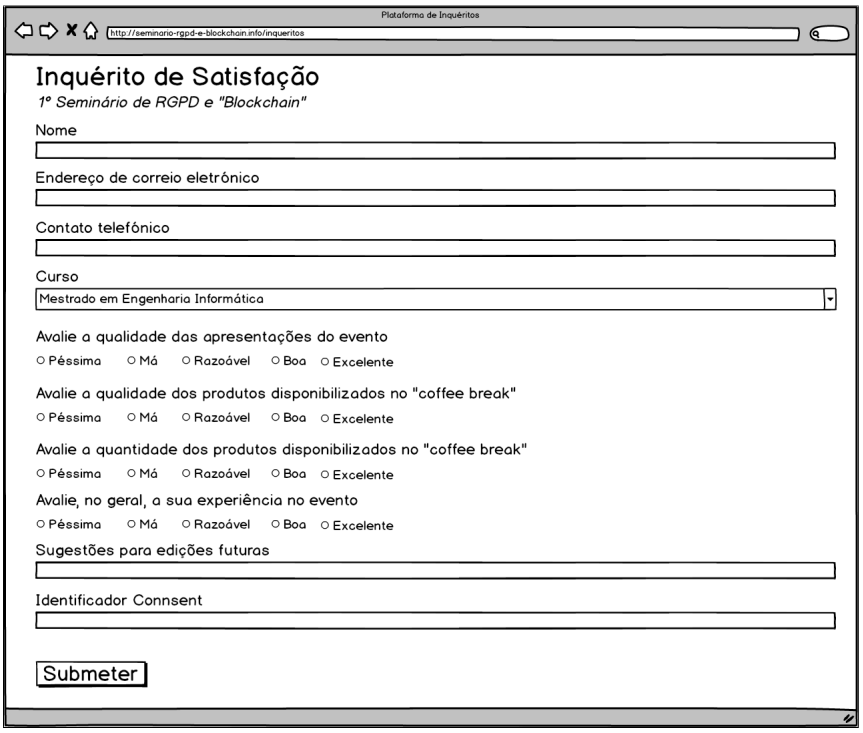

Figura 4.4: *Wireframe* para a implementação de referência para uma plataforma de captura de dados

4.7.2 *Dashboard* da Entidade Reguladora

Na definição do subsistema reputacional, no Ponto 4.3, foi referida a possibilidade de uma entidade reguladora tirar partido dos eventos emitidos pelo mesmo na rede *blockchain* para monitorizar em tempo real, através da sua *dashboard*, a satisfação dos sujeitos dos

dados para com os controladores de dados nas operações de consentimento sob a sua supervisão. É, portanto, pertinente uma implementação de referência da *dashboard* de uma entidade reguladora ficcional que apresenta informação sobre operações sob a sua supervisão em tempo real, como novos consentimentos, consentimentos removidos e novas avaliações de sujeitos dos dados a controladores de dados. A implementação de referência, representada na Figura 4.5, tira partido dos eventos da *contract account* definidos no Ponto 4.4.4 e dos *endpoints* da REST API definida no Ponto 4.5 para apresentar informação relevante sob a perspetiva de uma entidade reguladora.

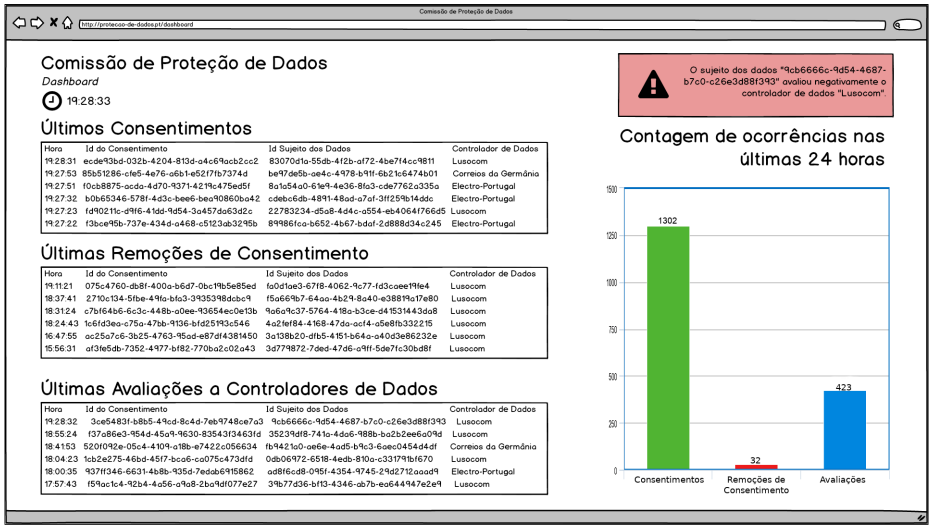

Figura 4.5: *Wireframe* para a implementação de referência da *dashboard* de uma entidade reguladora

Capítulo 5

Validação de Resultados

Neste capítulo serão descritos os desafios resultantes da implementação de um protótipo do sistema especificado no capítulo anterior e analisados os resultados daí obtidos. Sob um ponto de vista geral, o protótipo representará o resultado de múltiplas etapas que consistem na implementação dos vários componentes que o perfazem, mais concretamente a criação da *contract account* na rede *blockchain* que suporta o sistema através do *deploy* do *smart contract* (especificado no Ponto 4.4), a implementação da REST API que permite integrar com o *back end* do sistema via HTTPS, a implementação da aplicação móvel que permite ao sujeito dos dados interagir com o sistema, das implementações de referência para a plataforma de captura de dados do controlador de dados e da *dashboard* da entidade reguladora, e de outras tarefas como a configuração dos nós nas redes *Ethereum* utilizadas. No que às redes de *blockchain* concerne optou-se pela experimentação nas redes *Rinkeby* e *Ropsten*; tendo em consideração que estas redes, apesar de ambas serem mantidas pela equipa *Ethereum*, implementam protocolos de consenso diferentes. Como verificado no Ponto 2.3.6, a rede *Rinkeby* implementa um protocolo de consenso de *Proof-of-Authority* enquanto a rede *Ropsten* implementa um protocolo de consenso de *Proof-of-Work*. Esta abordagem permitirá analisar o desempenho do sistema *Connsent* quando suportado por redes *blockchain* com diferentes tipos de protocolo de consenso e comparar os resultados obtidos em ambos os cenários.

5.1 Criação de Nós nas Redes de Teste

Como descrito no capítulo 2, para que se possa aceder a uma rede *blockchain* é necessário tornar-se um membro da rede. Para se ser membro da rede é necessário configurar um nó da mesma, bastando para tal executar um dos *softwares* cliente suportados pela rede em questão (numa máquina com acesso à Internet e que cumpra os requisitos mínimos do mesmo) e sincronizar a *blockchain* da rede localmente. A duração deste processo de sincronização variará de acordo com o tamanho da *blockchain* da rede, da largura de banda do acesso à Internet, e das especificações do *hardware* da máquina. Terminado o processo de sincronização inicial, em que o *software* cliente verificará a validade de todos os blocos da rede desde o seu inicio, o novo membro da rede será capaz de realizar operações na rede enquanto, em paralelo, o *software cliente* continuará a manter a cópia local da *blockchain* sincronizada com a versão da rede. As aplicações que pretenderem interagir programaticamente com a rede através desse membro podem-no fazer por meio da API JSON-RPC

que o software cliente disponibiliza[1].

Como se optou por analisar o desempenho do *smart contract* (definido no Ponto 4.4) nas redes *Ropsten* e *Rinkeby*, foi necessário criar um nó em cada uma das redes que servisse como ponto de entrada das operações do sistema *Connsent* na respetiva rede *blockchain*. Apesar de já existirem serviços, como o *Infura*, que permitem aceder à rede *blockchain* através de APIs fornecidas por terceiros, optou-se utilização de uma versão própria para o efeito. Esta abordagem elimina fatores externos que poderiam afetar o desempenho do sistema e deturpar as leituras obtidas. A Listagem 5.1 apresenta o comando utilizado para criar um nó na rede *Rinkeby* para consumo do protótipo desenvolvido. A sua execução indicará ao cliente *go-ethereum* que se deverá conectar à rede *Ethereum* com o identificador 4 (identificador da rede *Rinkeby*), armazenar uma cópia completa da *blockchain* da rede no diretório "/blockchain/rinkeby", utilizar 1024 *megabytes* de memória volátil para *caching*, utilizar o nó especificado para se conectar durante a inicialização (opção *bootnodes*), e ativar o servidor HTTP-RPC deixando-o à escuta em todos os *interfaces* de rede no porto 8545.

```
$ geth --port=30303
       --networkid=4
       --datadir=/blockchain/rinkeby
       --cache=1024
       --syncmode=full
       --bootnodes="enode://a24ac7c5484ef4ed0c5eb2d36620ba4e4aa13b
                              8c84684e1b4aab0cebea2ae45cb4d375b77eab5
                              6516d34bfbd3c1a833fc51296ff084b770b94fb
                              9028c4d25ccf@52.169.42.101:30303"
       --rpc
       --rpcaddr 0.0.0.0
       --rpcport 8545
       --rpcapi "db,eth,net,web3"
```

Listagem 5.1: Comando de execução do cliente *go-ethereum* para o nó da rede *Rinkeby* utilizado pelo protótipo.

Por sua vez, a Listagem 5.2 apresenta o comando utilizado na criação do nó na rede *Ropsten* que o protótipo utiliza quando suportado por essa rede. A execução do cliente *go-ethereum* através deste comando indicar-lhe-á o diretório "/blockchain/ropsten" como destino da cópia local da *blockchain* da rede, a rede "testnet" (identificador da rede *Ropsten*) como rede a conectar-se, 1024 *megabytes* como a quantidade de memória volátil a utilizar para *caching*, a utilização dos dois nós especificados para se conectar durante a inicialização (opção *bootnodes*), e o porto 8546 para expor o servidor HTTP-RPC em todas as interfaces de rede.

```
$ geth --port=30304
       --testnet
       --bootnodes="enode://6332792c4a00e3e4ee0926ed89e0d27ef98542
                                4d97b6a45bf0f23e51f0dcb5e66b8757775064
                                58aea7af6f9e4ffb69f43f3778ee73c81ed9d3
                                4c51c4b16b0b0f@52.232.243.152:30303,
                    enode://94c15d1b9e2fe7ce56e458b9a3b672ef11894d
```

[1] Esta funcionalidade encontra-se desativada por predefinição.

```
                        dedd0c6f247e0f1d3487f52b66208fb4aeb817
                        9fce6e3a749ea93ed147c37976d67af557508d
                        199d9594c35f09@192.81.208.223:30303"
    --datadir=/blockchain/ropsten
    --cache=1024
    --rpc
    --rpcaddr 0.0.0.0
    --rpcport 8546
    --rpcapi "db,eth,net,web3"
```

Listagem 5.2: Comando de execução do cliente *go-ethereum* para o nó da rede *Ropsten* utilizado pelo protótipo.

Como demonstrado na Listagem 5.3, à data de 1 de Novembro de 2018 pelas 17:15 (UTC) as cópias locais das *blockchains* das redes *Rinkeby* e *Ropsten* ocupavam, respetivamente, 24.824 e 29.418 *megabytes* de armazenamento local.

```
$ date; du -sm /blockchain/*
Thu Nov 1 17:15:09 UTC 2018
24824 /blockchain/rinkeby
29418 /blockchain/ropsten
```

Listagem 5.3: Armazenamento ocupado pelas cópias locais das *blockchains* utilizadas (em MB).

5.2 Criação da Carteira e das *Contract Accounts*

Como referido no Ponto 4.4, o *smart contract* da solução (cujo código fonte está disponível no Apêndice 8.1) dará origem a uma *contract account* nas redes em que for *deployed*. Como se pretende avaliar o desempenho do *smart contract* da solução nas redes *Rinkeby* e *Ropsten* é necessário levar a cabo a operação de criação em ambas as redes. Como a criação de uma *contract account* numa rede *Ethereum* é uma operação que altera o estado da mesma, é necessário o pagamento de uma taxa aos membros da rede. Para que tal fosse possível surgiu a necessidade de criar uma carteira *Ethereum* e obter fundos suficientes para levar a cabo a operação de criação da *contract account* nas redes *Rinkeby* e *Ropsten*. Para o efeito recorreu-se à aplicação *web3j* para gerar uma carteira na rede *Ethereum* através da execução do comando apresentado na Listagem 5.4.

```
$ web3j wallet create
```

Listagem 5.4: Comando executado para criar uma wallet na rede *Ethereum* através da aplicação *web3j*.

À carteira criada foi atribuído o endereço de rede "0x02ef63504f08D5b3118793170985D8AF4E472727", podendo as suas transações na rede *Rinkeby* ser consultadas na hiperligação `https://rinkeby.etherscan.io/address/0x02ef63504f08d5b3118793170985d8af4e472727` e na rede *Ropsten* na hiperligação `https://ropsten.etherscan.io/address/0x02ef63504f08d5b3118793170985d8af4e472727`. A carteira gerada foi persistida

num ficheiro com uma estrutura JSON que armazena a chave privada que permite realizar operações nas redes *Ethereum* com a carteira. O seu conteúdo é apresentado na Listagem 5.5 e, como se pode entender através da sua análise, a chave privada (atributo "crypto.ciphertext") está cifrada com o algoritmo de cifra simétrica *Advanced Encryption Standard* (AES). A chave de cifra/decifra da chave privada é derivada a partir de uma *Key Derivation Function* (KDF) que recebe como *inputs* os valores persistidos no objeto "crypto.kdfparams" e uma *password* que não está persistida no ficheiro da carteira e que deverá ser disponibilizada sempre que se pretenda utilizar a carteira. Para que fosse possível recompensar as redes de teste pelas transações que pretendiamos fazer, recorreu-se a um *faucet* com o objetivo de obter créditos para a carteira criada. Um *faucet* é um serviço criado por um ou vários membros de uma rede *Ethereum* que recompensa outros membros da rede com uma determinada quantia em troca da realização de uma tarefa, como por exemplo o preenchimento de um CAPTCHA. Neste caso em particular, optou-se pelos *faucets* `http://faucet.rinkeby.io` para a rede *Rinkeby* e pelo `http://faucet.ropsten.be` para a rede *Ropsten*.

```
{
    "address":"02ef63504f08d5b3118793170985d8af4e472727",
    "crypto":{
        "cipher":"aes-128-ctr",
        "ciphertext":"e8de8c29934bd572244079b0a72a3e3d8c1ebd099d7614c2ef1dc762a917999f",
        "cipherparams":{
            "iv":"2ed61913052d58ebbbd9dca7c997fde0"
        },
        "kdf":"scrypt",
        "kdfparams":{
            "dklen":32,
            "n":262144,
            "p":1,
            "r":8,
            "salt":"8f22b5805b8f5d7cfdfe4d7e15d9fb79bf1bd6d3076b0a532619c5b9fe5499e8"
        },
        "mac":"5223c3b9b1b1950dc33cf709c7ab4b8ffdbec29ed8a0530cb9cbb5850351afb3"
    },
    "id":"c7aad451-d30b-4d62-97f1-d189bb4d682b",
    "version":3
}
```

Listagem 5.5: Conteúdo do ficheiro que persiste a carteira criada.

Estavam então reunidas as condições para criar as *contract accounts* através do *deploy* do *smart contract* especificado no Ponto 4.4. Mas antes disso foi necessário compilar o código fonte do *smart contract* definido em linguagem *Solidity* (disponibilizado no Apêndice 8.1) para os suportes *Application Binary Interface* (ABI) e Binário (BIN), tendo-se utilizado para o efeito o compilador "solc" através da execução do comando apresentado na Listagem 5.6. O suporte ABI é constituído por um conjunto de interfaces que indicarão à EVM dos membros da rede quais as funcionalidades do *smart contract* implementado no suporte BIN.

```
$ solc --optimize -o . --bin --abi Connsent.sol
```

Listagem 5.6: Comando executado para compilar o *smart contract* definido em linguagem *Solidity*.

Como se pretendia recorrer posteriormente a uma aplicação implementada com a biblioteca *web3j* para executar o *deployment* do *smart contract* nas redes *Rinkeby* e *Ropsten*,

foi também necessário invocar a aplicação *web3j* para gerar os *wrappers* do *smart contract* para a biblioteca Java *web3j*. Deste modo passaria a ser possível manipular programaticamente este *smart contract* em aplicações Java com recurso à biblioteca *web3j*. Para esse efeito executou-se o comando apresentado na Listagem 5.7, que recebe como *inputs* o *smart contract* compilado nos suportes ABI e BIN, o caminho do projeto Java no sistema de ficheiros e o *package* do projeto onde os *wrappers* deverão residir.

```
$ web3j solidity generate --javaTypes Connsent.bin Connsent.abi -o ~/
    ↪ ConnsentDeploy/src/main/java -p pt.ipp.estg.mei.contracts
```

Listagem 5.7: Comando executado para gerar os *wrappers* do *smart contract Connsent* para a biblioteca Java *web3j*.

Gerados os *wrappers* ficaram reunidas todas as condições para criar programaticamente as *contract accounts* nas redes *Rinkeby* e *Ropsten*. O *deployment* de ambos os *smart contracts* foi feito com recurso a uma aplicação Java implementada para o efeito (código fonte disponível no Apêndice 8.2) que se liga, dependendo da rede de destino, a um dos nós referidos no Ponto 5.1. À *contract account* na rede *Rinkeby* foi atribuído o endereço "0x0Fc33976d3D9b5E97A9d52C52A798Ce8148aA9e2" que pode ser consultado através da hiperligação `https://rinkeby.etherscan.io/address/0x0fc33976d3d9b5e97a9d52c52a798ce8148aa9e2`, enquanto a *contract account* na rede *Ropsten* ficou associado o endereço "0x0Fc33976d3D9b5E97A9d52C52A798Ce8148aA9e2" podendo este ser consultado através da hiperligação `https://ropsten.etherscan.io/address/0x0fc33976d3d9b5e97a9d52c52a798ce8148aa9e2`. O custo da criação da *contract account* foi o mesmo na rede *Rinkeby* e na rede *Ropsten*, sendo que este comportamento já era esperado tendo em conta que ambas as *contract accounts* foram criadas a partir do mesmo *smart contract*. A operação de criação despendeu 6.160.733 unidades de *gas*, o que, a um custo unitário de 22 *Gwei*, perfez um custo total de 0,135536126 *ether*.

5.3 Comparativo Mapeamento *Versus* Vetor

No Capítulo 4 foi referido que a opção de utilizar mapeamentos para armazenar as estruturas de dados dados complexas que representam as entidades do sistema iria permitir operações de leitura e criação mais eficientes e, por conseguinte, mais baratas. Antes de se optar pela utilização de mapeamentos foi feito um comparativo com o objetivo de aferir a diferença de custo[2] entre armazenar as estruturas de dados complexas em mapeamentos ou em vetores. Para o efeito implementaram-se dois *smart contracts*, um para cada abordagem, que têm como finalidade armazenar dados relativos a sujeitos dos dados numa estrutura de dados idêntica à especificada no *smart contract* do sistema para o mesmo efeito. O código destes *smart contracts*, que é apresentado no Apêndice 8.8, assegura o armazenamento de um sujeito dos dados na *contract account* e garante que não existe mais que um sujeito dos dados com o mesmo identificador. Foi então feito o *deploy* destes *smart contracts* na rede de teste *Rinkeby* seguindo a uma abordagem idêntica à descrita no Ponto 5.2. De seguida é apresentado o endereço das *contract accounts* criadas, das transações de criação, a quantidade de *gas* que foi necessária, e os eventuais custos da

[2]Os valores apresentados neste ponto respeitam à cotação do dia 6 de Novembro de 2018, data em que foram efetuadas as transações que serviram como base ao comparativo, quando a 1 ETH correspondia 183,27 EURO.

transação caso esta tivesse sido efetuada na rede principal (*Mainnet*) e não na rede de teste *Rinkeby*. Como referido no Ponto 2.3, o comportamento da rede principal é imprevisível e não existem dados que permitam calcular o valor médio a pagar por uma unidade de *gas* quando se pretende que uma transação seja confirmada na rede num determinado intervalo de tempo. Posto isto, assumir-se-á como cenário de referência que a um custo de 2 *Gwei* por unidade de *gas* a transação será confirmada em aproximadamente 5 minutos, a um custo de 5 *Gwei* a transação será confirmada em aproximadamente 2 minutos, e que a um custo de 10 *Gwei* a transação será confirmada em menos de 30 segundos. Como se pode comprovar pelos valores apresentados de seguida, a transação de criação da *contract account* que inclui o mapeamento foi 11,6% menos dispendiosa que a criação da *contract account* que recorre ao vetor.

Deploy Smart Contract com Mapeamento

***Contract Account*:** `https://rinkeby.etherscan.io/address/0xc4cecc78fb4fd9b139cdb93d0de1328e5fa5b928`

Transação de Criação: `https://rinkeby.etherscan.io/tx/0x335c6b3d147a99cd327ba07c9f160c1b5d6256cc12436c1cf4523f0e104a1a56`

***Gas* Necessário:** 800.513

Custo (*Gas* a 2 *Gwei*): 0,001601026 ETH / 0,29342003502 EURO

Custo (*Gas* a 5 *Gwei*): 0,004002565 ETH / 0,73355008755 EURO

Custo (*Gas* a 20 *Gwei*): 0,016010260 ETH / 2,9342003502 EURO

Deploy Smart Contract com Vetor

***Contract Account*:** `https://rinkeby.etherscan.io/address/0x6fa08c7d5cb63c347fe2a070a403a7153d32cc15`

Transação de Criação: `https://rinkeby.etherscan.io/tx/0x319ed96036b19dbaf59b5f5d6596110b7f3a469746c730d2fb397c5621663412`

***Gas* Necessário:** 903.945

Custo (*Gas* a 2 *Gwei*): 0,001807890 ETH / 0,3313320003 EURO

Custo (*Gas* a 5 *Gwei*): 0,003615780 ETH / 0,6626640006 EURO

Custo (*Gas* a 20 *Gwei*): 0,018078900 ETH / 3,313320003 EURO

Após a criação de ambas as *contracts accounts* foram executadas 100 transações de armazenamento de um sujeito dos dados em ambas as redes. Foi implementado um *script* que, ao longo de 100 iterações, gerava dados válidos para um novo sujeito dos dados armazenando-os, de seguida, na *contract account* implementada com um mapeamento e posteriormente na *contract account* implementada com um vetor. Esta abordagem permitiu recolher os dados que são listados nos Apêndices 8.9 e 8.10, comparados no gráfico da Figura 5.1[3].

[3] As entradas #0 de ambas as listagens correspondem à criação da *contract account* e foram excluídas do conjunto de valores que deu origem no gráfico apresentado na Figura 5.1 e à Tabela 5.1.

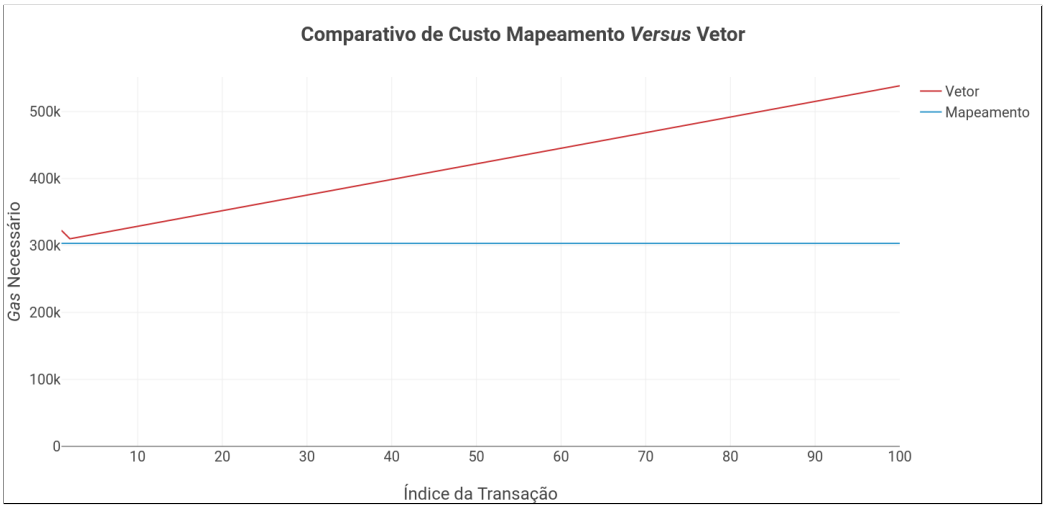

Figura 5.1: Gráfico comparativo entre a evolução do custo das transações na *contract account* implementada com recurso a um mapeamento e a *contract account* implementada com recurso a um vetor.

Através da análise do gráfico da Figura 5.1 e dos dados apresentados nas tabelas apresentadas nos Apêndices 8.9 e 8.10, pode-se retirar que o custo das transações na *contract account* com mapeamento se mantêm praticamente constantes (variando apenas ligeiramente de acordo com a dimensão da informação persistida), enquanto que na *contract account* com vetor o custo tem uma evolução linear crescente. Considerando a lógica do *smart contract* que deu origem à *contract account* com vetor, fica claro que a validação que assegura que o identificador do novo sujeito dos dados ainda não está em uso é responsável por esta evolução linear do custo das transações de criação, tendo em conta que o comprimento do vetor aumenta com cada inserção e que esta validação necessita de realizar uma travessia completa do mesmo. No *smart contract* que implementa o armazenamento com recurso a um mapeamento esta evolução linear não se verifica, uma vez que a disponibilidade do identificador pode ser validada sem recurso a uma travessia completa dos elementos previamente existentes. A Tabela 5.1 apresenta o custo médio, mínimo e máximo das cem transações de inserção nas *contract accounts* em diferentes contextos.

Tendo em conta que a partir da segunda inserção na *contract account* que utiliza o *vetor* aumenta em média 2.333 unidades de *gas*, é possível calcular que a milésima inserção requererá 2.638.342 unidades de gás e que a décima milésima (#10.000) requererá 23.635.342 unidades de gás e que centésima milésima requererá 233.605.342. Atendendo ao facto que o limite de *gas* numa operação na rede principal tende a rondar os 8.000.000 percebe-se que, em teoria, só seria possível armazenar 3.298 sujeitos dos dados. Deste modo, a transação de inserção do sujeito dos dados na posição 3.198 requereria 7.999.576 unidades de *gas*, 0,015999152 *ether* ou 2,932164587 Euro com a unidade de *gas* a 2 *Gwei*, 0,03999788 *ether* ou 7,330411468 Euro com a unidade de *gas* a 5 *Gwei*, ou 0,15999152 *ether* ou 29,32164587 com a unidade de *gas* Euro a 20 *Gwei*. Em comparação, a inserção com o índice #3298 da *contract account* que recorre ao mapeamento não desvia significativamente das 303.266,2 unidades de *gas* referidas na Tabela 5.1 e dos seus respetivos custos. Como a chave de um mapeamento é um resumo criptográfico SHA3-256 que tem um comprimento de 256 *bits*, em teoria um mapeamento suporta 2^{256} entradas, garantido que não ocorre nenhuma colisão entre o resumo criptográficos dos identificadores dos sujeitos dos dados.

Tipo de Custo Armazenamento	**Mapeamento**	**Vetor**
Custo Médio (*Gas*)	303.266,2	423.329,7
Custo Mínimo (*Gas*)	303.215	310.003
Custo Máximo (*Gas*)	303.471	538.642
Custo Médio (*Gas* a 2 *Gwei*)	0,0006065324 ETH 0,111159192948 EURO	0,0008466594 ETH 0,155167268238 EURO
Custo Mínimo (*Gas* a 2 *Gwei*)	0,00060643 ETH 0,1111404261 EURO	0,000620006 ETH 0,11362849962 EURO
Custo Máximo (*Gas* a 2 *Gwei*)	0,000606942 ETH 0,11123426034 EURO	0,19743383868 ETH 0,19743383868 EURO
Custo Médio (*Gas* a 5 *Gwei*)	0,001516331 ETH 0,27789798237 EURO	0,0021166485 ETH 0,387918170595 EURO
Custo Mínimo (*Gas* a 5 *Gwei*)	0,001516075 ETH 0,27785106525 EURO	0,001550015 ETH 0,28407124905 EURO
Custo Máximo (*Gas* a 5 *Gwei*)	0,001517355 ETH 0,27808565085 EURO	0,002693210 ETH 0,4935845967 EURO
Custo Médio (*Gas* a 20 *Gwei*)	0,006065324 ETH 1,11159192948 EURO	0,008466594 ETH 1,55167268238 EURO
Custo Mínimo (*Gas* a 20 *Gwei*)	0,0060643 ETH 1,111404261 EURO	0,00620006 ETH 1,1362849962 EURO
Custo Máximo (*Gas* a 20 *Gwei*)	0,00606942 ETH 1,1123426034 EURO	0,01077284 ETH 1,9743383868 EURO

Tabela 5.1: Valores do custo das 100 transações de inserção nas *contract accounts* do comparativo.

Como o cenário do comparativo é bastante representativo dos cenários de inserção no *smart contract* que suporta o sistema (uma conclusão trivial após analisar o código fonte de ambos), e como ficou provado que o recurso a vetores não é escalável nem é vantajoso em cenário algum, optou-se pela utilização de mapeamentos para armazenar todo o tipo de estruturas de dados complexas necessárias no seio do *smart contract* que suporta o sistema *Connsent*.

5.4 REST API

A REST API, especificada no Ponto 4.5, foi implementada com recurso à linguagem Java tirando partido da *framework* Jersey, uma implementação da especificação *Java API for RESTful Web Services* 2.0. Toda a interação com a rede *Ethereum* é conseguida graças à biblioteca *Web3j* que disponibiliza a lógica necessária à manipulação das *contract accounts* criadas no Ponto 5.2 através dos *wrappers* gerados no mesmo ponto. Foi criada, especificamente para o efeito, a classe "ConnsentSingleton.java" que implementa o padrão de desenvolvimento *Singleton* com o intuito objetivo gerir a instância do serviço *Web3* (recorrendo ao conceito de *lazy loading*) que é consumido pela REST API. O código fonte desta classe está disponível no Apêndice 8.4. A lógica de negócio do componente foi implementada com recurso aos mecanismos nativos disponibilizados pela versão 8 do *Java Development Kit* (JDK), tendo-se utilizado a biblioteca *Bouncy Castle* para todas as operações criptográficas e de codificação de dados. Exemplo disso é a Listagem 8.3 que apresenta o código fonte da rota "/consent" (POST) que valida as assinaturas digitais de pedido de consentimento e para persistir um consentimento dado por um sujeito dos dados a um controlador de dados. A rota "/consent" (POST) seria invocada, por exemplo, pelo comando do cliente HTTP cURL apresentado na Listagem 5.8. Esta Listagem retrata a invocação da rota "/consent" da REST API com o objetivo de persistir o consentimento "211413fe-0d4c-4f78-a8ea-3cbfc5a90ed2" dado pelo sujeito dos dados "08fb82a7-ef19-41fa-bd0a-07d7656aedef" ao controlador de dados "Electro-Portugal".

```
$ curl --request POST
       --url 'https://CONNSENT_REST_API_URL/consent'
       --header 'content-type: application/json'
       --data '{
                 "regulatorName":"CPD",
                 "uid":"211413fe-0d4c-4f78-a8ea-3cbfc5a90ed2",
                 "termsUid":"9dbaa92e-f53f-4d25-8d67-2d3efee4683b",
                 "requestSig":"MD0CHQCAC1HRa4VlyBl8iOa0kdIeTmpV3usfQ3zOkzVqAhwfonZGHyH7
                               ewzY9/K2Fwm3TEtfU0743S7VyWo",
                 "controllerName":"Electro-Portugal",
                 "consentSig":"MD0CHAsIMCzETfrKnnSwuzrAFJUcY/0K6OjhXzOdGw4CHQDTMlfXKrKM
                               qcoXWu1zZvNqX/jpTnBLvq1UJSWk",
                 "subjectUid":"08fb82a7-ef19-41fa-bd0a-07d7656aedef",
                 "dataDigest":"98d3f33cb16d434d876a3799acaad6be05082cff056892a9b0d5061b
                               52de8f0760ad4366c191bef8a37087618e53985f2a6b57584e93545e
                               0669e4eff27713b4"
               }'
```

Listagem 5.8: Invocação da rota "/consent" da REST API

O código fonte da implementação foi transformado em *Java Virtual Machine* (JVM) *bytecode* e empacotado no formato *Web Application Resource* (WAR) pela ferramenta de compilação *Maven*. De maneira a que este componente pudesse ser facilmente executado e distribuído fez-se uso da plataforma *Docker*, que permite embeber o ficheiro WAR

na imagem oficial "tomcat:8-alpine" que já inclui toda a *stack* de *software* necessária à execução da REST API. O conteúdo do ficheiro *Dockerfile* utilizado para gerar a imagem dos *containers* da aplicação *web* é apresentado na Listagem 5.9.

```
FROM tomcat:8-alpine

RUN ln -s /usr/local/tomcat /tomcat
RUN rm -rf /tomcat/webapps/*
COPY /path/to/maven/output /tomcat/webapps
RUN mkdir -p /connsent/data

CMD ["catalina.sh","run"]
```

Listagem 5.9: Conteúdo do ficheiro *Dockerfile* responsável por gerar a imagem utilizada para lançar *containers* da REST API.

Para se gerar a imagem (com recurso ao ficheiro *Dockerfile* apresentado na Listagem 5.9) e lançar o *container* da REST API implementada executaram-se os comandos apresentados na Listagem 5.10.

```
$ docker build -t connsent-api /caminho/para/projeto
$ docker run -d
               -p 80:8080
               -v /caminho/para/dados/container:/connsent
               --env-file /caminho/ficheiro_variaveis_de_ambiente
               --name connsent-api
               connsent-api:latest
```

Listagem 5.10: Comandos executados para gerar imagem *Docker* e lançar o *container* da REST API.

O primeiro comando indica à aplicação *Docker* que deve criar uma imagem com a designação "connsent-api" através do ficheiro *Dockerfile* disponível no diretório "/caminho/para/projeto". A sua execução resultará na criação da imagem "connsent-api:latest", uma vez que a aplicação *Docker* adiciona, por pré-definição, a *tag* ":latest" sempre que não é especificada nenhuma outra *tag* no comando. A execução do segundo comando resultará no lançamento de um *container* em modo *detached* com a designação "connsent-api" que utiliza como base a imagem "connsent-api:latest" (criada com o comando anterior), no mapeamento da porta 8080 do *container* para a porta 80 do sistema anfitrião, no mapeamento do diretório do anfitrião "/caminho/para/dados/container" para o diretório "/connsent" do *container*, e na definição das variáveis especificadas no ficheiro "/caminho/ficheiro_variaveis_de_ambiente" como variáveis de ambiente do sistema do *container*.

As variáveis ambiente que a aplicação *web* da REST API espera que estejam definidas no sistema do *container* são enunciadas na Tabela 5.2.

Estas configurações indicam à aplicação *web* da REST API, entre outros pormenores de menor relevância, qual o nó da rede Ethereum ao qual se deverá conectar, qual é o endereço da *contract account* que deverá utilizar, e o endereço da carteira a utilizar no sistema de ficheiros (e a respetiva *password*).

Designação	Descrição	Valor de Exemplo
PROJECT_NAME	Designação da aplicação *web*	Connsent API
ETHEREUM_NODE	Endereço do nó da rede *Ethereum* a utilizar.	http://127.0.0.1 :8545
ETHEREUM_ TRANSACTION_LINK	Endereço do explorador de blocos associado ao nó de rede a utilizar.	https://rinkeby. etherscan.io/tx/
CONTRACT_ ACCOUNT_ADDRESS	Endereço da *contract account* a utilizar	0x0Fc33976d3D9b 5E97A9d52C52A798 Ce8148aA9e2
ETHEREUM_ WALLET_PATH	Caminho completo para o ficheiro da carteira a utilizar no sistema de ficheiros do *container*.	/connsent/wallet.json
ETHEREUM_ WALLET_SECRET	*Password* para decifra da chave privada da carteira a utilizar.	secure-password

Tabela 5.2: Variáveis de ambiente esperadas pela aplicação *web* da REST API.

5.5 Transações de Teste

Com o intuito de analisar o comportamento e de aferir os custos das transações definidas no *smart contract* (especificado no Ponto 4.4), o sistema foi inicializado com dados representativos de um cenário de utilização real. Para esse efeito, foram realizadas ligações HTTPS a duas instâncias da REST API (especificada no Ponto 4.5), uma configurada para a rede *Rinkeby* e outra para a rede *Ropsten*, que resultaram nas transações de teste detalhadas no Apêndice 8.5. Levadas a cabo estas transações, foi possível analisar o custo de cada tipo de transação. O gráfico apresentado na Figura 5.2 apresenta o custo de cada uma transações de teste efetuadas, destacando-as por tipo (cada cor representa um tipo diferente).

Como seria de esperar após a análise efetuada ao comparativo do Ponto 5.3, as transações de cada um dos tipos têm um custo praticamente constante. Eventuais variações ligeiras ficam-se a dever à dimensão dos dados persistidos; exceto no caso particular das avaliações, onde a inserção da primeira entrada relativa a um consentimento (Índices 1 e 3) terá sempre um custo superior às entradas seguintes relativas ao mesmo consentimento (Índices 2, 4 e 5). Este comportamento é resultado da alocação de um vetor, aquando da primeira entrada, para armazenar todas as entradas de avaliações relativas a esse consentimento. A Tabela 5.3 apresenta o custo médio de cada tipo de transação em unidades de *gas*, *ether* e Euro[4].

[4]Os valores apresentados neste ponto respeitam a cotação do dia 6 de Novembro de 2018 quando a 1 ETH correspondia 183,27 EURO.

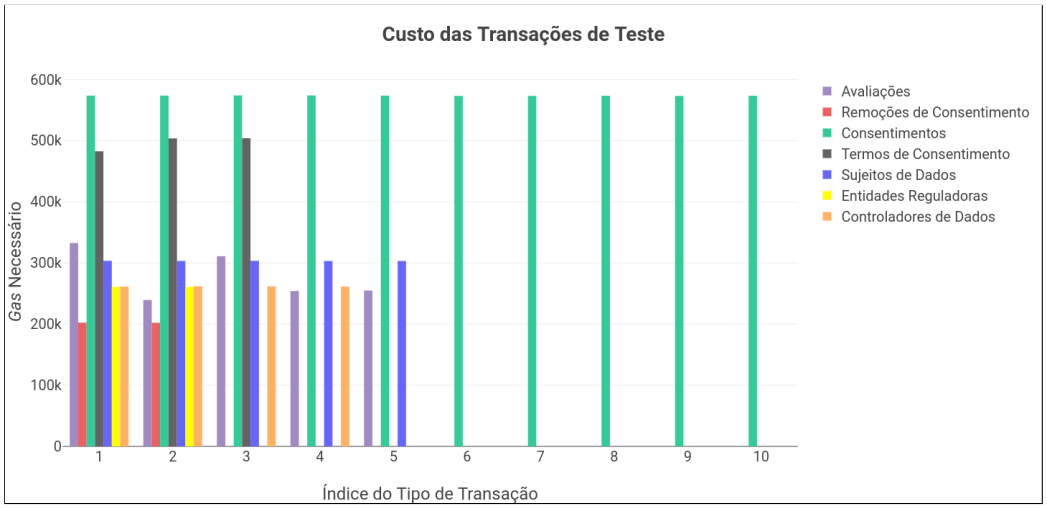

Figura 5.2: Evolução do custo dos diferentes tipos das transações de teste realizadas.

5.6 Subsistema Reputacional

A interação com o subsistema reputacional embutido na solução *Connsent* é possível através de rotas da REST API[5] que permitem deixar *feedback* sobre um controlador de dados (no âmbito de um determinado consentimento) e obter a reputação de um controlador de dados. O código fonte destas rotas está disponível na integra no Apêndice 8.7.

A rota "/controller/feedback" assume particular destaque, uma vez que é nesta que está definida a lógica que permite iterar todos os eventos do tipo "NewFeedbackEntryEvent" com o objetivo de agregar todas as entradas de *feedback* que se qualificam para o cálculo da reputação do controlador de dados. A Listagem 5.11 apresenta o código fonte, que é um excerto do Apêndice 8.7, responsável por iterar todos os eventos do tipo "NewFeedbackEntryEvent". Por qualificar-se entende-se que é a entrada mais recente relativamente a um determinado consentimento, ou seja, para um consentimento como o "211413fe-0d4c-4f78-a8ea-3cbfc5a90ed2" (criado no contexto referido no Ponto 5.5) que tem quatro entradas de *feedback* apenas se qualifica para o cálculo da reputação do controlador de dados a mais recente.

Posto isto, a Listagem 5.12 representa a invocação da rota "/controller/feedback" da REST API para o controlador de dados "Electro-Portugal" e a respetiva resposta.

5.7 Aplicação Móvel

A aplicação móvel *Connsent* foi implementada para o ecossistema *Android*, suportando qualquer dispositivo que tenha instalada pelo menos a versão 5.0 (nome de código *Lollipop*) desse sistema operativo. Para comunicar com a REST API através do protocolo HTTPS recorreu-se à biblioteca *OkHttp*; tendo esta escolha ficado a dever-se à simplicidade, maturidade e desempenho da biblioteca em questão. Com o objetivo de garantir

[5]Para mais detalhes sobre as rotas da REST API consultar a especificação no Ponto 4.5.

```
// Iterating through NewFeedbackEntryEvent events in blockchain
Iterator<Connsent.NewFeedbackEntryEventResponse> it =
        connsent .newFeedbackEntryEventObservable( DefaultBlockParameterName.EARLIEST,
                                                   DefaultBlockParameterName.LATEST)
                .onTerminateDetach()
                .toBlocking()
                .toIterable()
                .iterator();

ArrayList<String> allControllerEntries = new ArrayList<>();
HashSet<String> uniqueConsentEntries = new HashSet<>();

// Iterating through all feedback entries
while(it.hasNext()){

    // Getting next entry's reference
    Connsent.NewFeedbackEntryEventResponse current = it.next();

    // Checking if current entry is about the given controller
    if(current.dataControllerName.equals(controllerName)){
        // Entry is about the given controller, storing consent identifier
        allControllerEntries.add(current.consentUid);
        uniqueConsentEntries.add(current.consentUid);
    }
}
```

Listagem 5.11: Excerto do código fonte da rota "/controller/feedback" da REST API responsável por iterar todos os eventos do tipo "NewFeedbackEntryEvent".

```
curl --request GET
     --url 'https://CONNSENT_API_URL/controller/feedback?name=Electro-Portugal'
{
  "score":1,
  "negative":0,
  "neutralPercentage":"0%",
  "neutral":0,
  "positivePercentage":"100%",
  "totalEntries":1,
  "positive":1,
  "negativePercentage":"0%",
  "entries":[{
              "message":"Estava equivocado relativamente ao comportamento do controlador de
                  ↪ dados.",
              "type":"POSITIVE",
              "timestamp":1542629857
            }]
}
```

Listagem 5.12: Invocação da rota "/controller/feedback" da REST API com o objetivo de obter os dados relativos à reputação do controlador de dados "Electro-Portugal".

que a implementação das operações criptográficas e de codificação de dados realizadas no seio da aplicação móvel se assemelha à da REST API (consultar Ponto 5.4), optou-se pela utilização da biblioteca *Spongy Castle* que não é nada mais nada menos do que uma distribuição da biblioteca *Bouncy Castle* para o ambiente de desenvolvimento *Android*. Quando a aplicação móvel é inicializada pela primeira vez no dispositivo móvel, é gerado um par de chaves ECDSA (que identificará o sujeito dos dados no âmbito do sistema *Connsent*) e é efetuado o registo de um novo sujeito dos dados no sistema *Connsent* através de uma ligação HTTPS à REST API. Findadas estas operações iniciais, dados como o par de chaves gerado e o identificador do sujeito dos dados atribuído na operação de registo são armazenados nas preferências compartilhadas do sistema operativo *Android*. O Apêndice 8.6 apresenta o código fonte das operações de inicialização de um novo sujeito dos dados através da aplicação móvel *Connsent*.

A implementação da interface do utilizador guiou-se pelos *wireframes* e pelos pressupostos definidos no Ponto 4.6. Prova disso é a Figura 5.3 que demonstra a interface do utilizador da aplicação móvel *Connsent* sob a perspetiva do sujeito dos dados "08fb82a7-ef19-41fa-bd0a-07d7656aedef".

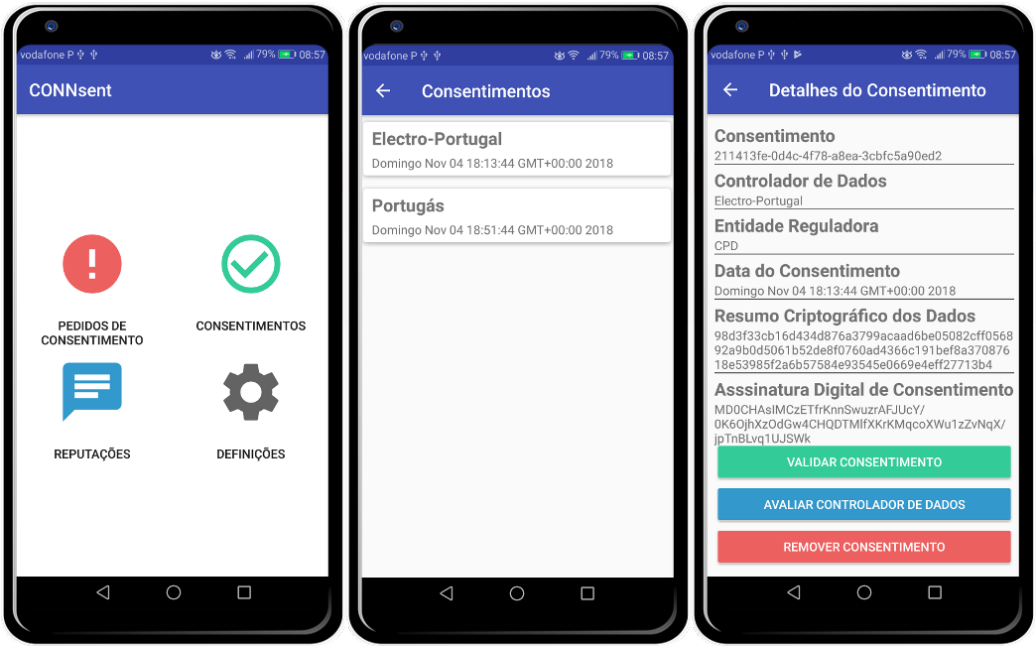

Figura 5.3: Visão geral da interface do utilizador da aplicação móvel *Connsent* sob a perspetiva do sujeito dos dados "08fb82a7-ef19-41fa-bd0a-07d7656aedef".

5.8 Protótipos de Implementações de Referência

No Ponto 4.7 foram especificadas duas eventuais implementações de referência para o sistema *Connsent*; uma plataforma de captura de dados do sujeito dos dados, e uma *dashboard* de monitorização que permitia à entidade reguladora monitorizar os consentimentos que supervisiona. Ora, com o objetivo de demonstrar as potencialidades da solução

Connsent, foram implementados protótipos para as duas implementações de referência mencionadas. Ambas as implementações destinam-se a uma utilização num contexto *web*, tendo sido amplamente utilizada a linguagem de marcação HTML e a linguagem de *scripting Javascript*. A implementação de ambos os protótipos tirou partido da *framework* de *front-end Bootstrap* para agilizar a interface gráfica do protótipo e da biblioteca *jQuery* para o desenvolvimento de funcionalidades em linguagem *Javascript*.

A implementação do protótipo da plataforma de captura de dados teve como principal objetivo recriar um cenário em que os dados do sujeito dos dados são capturados por uma plataforma de inquéritos de satisfação detida por um controlador de dados. O preenchimento do inquérito por parte do sujeito dos dados, a introdução do seu identificador no sistema *Connsent*, e a submissão do inquérito resultaria no envio de uma notificação para o dispositivo móvel associado ao sujeito dos dados. O protótipo da plataforma de inquérito está programado para que, imediatamente após o clique do utilizador no botão de submissão do inquérito, seja calculado o resumo criptográfico dados capturados e computada a assinatura digital de pedido de consentimento com recurso à chave privada do controlador de dados. Terminadas estas operações que ocorrem no perímetro daquilo que seria a infraestrutura tecnológica do sujeito dos dados, serão comunicados à REST API, através da rota "/request", os dados necessários ao pedido de um novo consentimento. O processamento deste pedido por parte da REST API resultaria no envio de uma notificação *push* para o *smartphone* do sujeito dos dados com o objetivo de o alertar para a existência de um novo pedido do seu consentimento. As operações criptográficas realizadas no seio deste protótipo foram implementadas com recurso às biblioteca de *Javascript* "*elliptic*" (`https://github.com/indutny/elliptic`), "js-sha512" (`https://github.com/emn178/js-sha512`), e "hi-base64" (`https://github.com/emn178/hi-base64`). A Figura 5.4 fornece uma visão geral sob o protótipo implementado para a plataforma de captura de dados.

Por sua vez, a implementação do protótipo da *dashboard* da entidade reguladora simula o sistema de monitorização de uma entidade reguladora. Neste cenário, uma entidade reguladora monitoriza em tempo real todos os novos eventos relativos a consentimentos, remoções de consentimento, e avaliações de sujeitos dos dados a controladores de dados. Esta monitorização é possível graças à subscrição de *feeds*, da *contract account* que suporta a solução *Connsent*, que emitem um novo evento sempre que se verifica uma das três situações referidas[6]. No protótipo implementado os novos eventos são agrupados por tipo e ordenados por ordem decrescente de ocorrência (do mais recente para o mais antigo). Está também previsto um gráfico que tem como objetivo fornecer uma representação visual da contagem acumulativa de eventos por tipo. A subscrição de eventos que ocorrem na *blockchain* de uma rede *Ethereum* é conseguida com recurso à biblioteca de *Javascript web3js* (`https://github.com/ethereum/web3.js`). A Figura 5.5 fornece uma visão geral do protótipo implementado para a *dashboard* da entidade reguladora sob a perspetiva da entidade reguladora "CPD".

[6]Os tipos de eventos previstos estão especificados no Ponto 4.4.4.

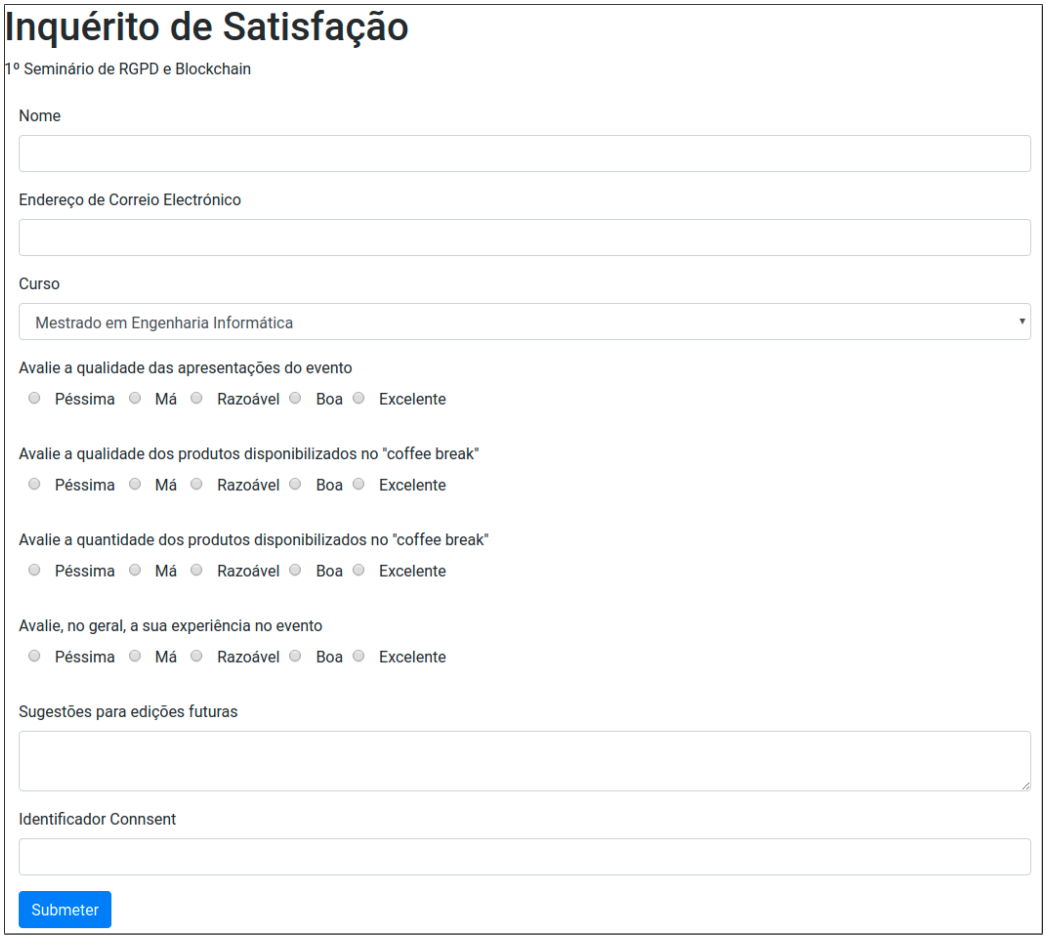

Figura 5.4: Visão geral sob o protótipo da plataforma de captura de dados implementado.

Tipo de Transação	**Custo Médio**
Controlador de Dados	261.675 Unidades de *Gas*
Entidade Reguladora	260.930 Unidades de *Gas*
Sujeito dos Dados	303.408,4 Unidades de *Gas*
Termos de Consentimento	496.859,7 Unidades de *Gas*
Consentimento	573.640,4 Unidades de *Gas*
Remoção de Consentimento	202.347 Unidades de *Gas*
1ª Avaliação	321.848 Unidades de *Gas*
Restantes Avaliações	249.534 Unidades de *Gas*
Controlador de Dados (2 *Gwei*)	0,00052335 ETH / 0,0959143545 EURO
Entidade Reguladora (2 *Gwei*)	0,00052186 ETH / 0,0956412822 EURO
Sujeito dos Dados (2 *Gwei*)	0,0006068168 ETH / 0,111211314936 EURO
Termos de Consentimento (2 *Gwei*)	0,000993719333333 ETH / 0,18211894222 EURO
Consentimento (2 *Gwei*)	0,0011472808 ETH / 0,210262152216 EURO
Remoção de Consentimento (2 *Gwei*)	0,000404694 ETH / 0,07416826938 EURO
1ª Avaliação (2 *Gwei*)	0,000643696 ETH / 0,11797016592 EURO
Restantes Avaliações (2 *Gwei*)	0,000499068 ETH / 0,09146419236 EURO
Controlador de Dados (5 *Gwei*)	0,001308375 ETH / 0,23978588625 EURO
Entidade Reguladora (5 *Gwei*)	0,00130465 ETH / 0,2391032055 EURO
Sujeito dos Dados (5 *Gwei*)	0,001517042 ETH / 0,27802828734 EURO
Termos de Consentimento (5 *Gwei*)	0,002484298333333 ETH / 0,45529735555 EURO
Consentimento (5 *Gwei*)	0,002868202 ETH / 0,52565538054 EURO
Remoção de Consentimento (5 *Gwei*)	0,001011735 ETH / 0,18542067345 EURO
1ª Avaliação (5 *Gwei*)	0,00160924 ETH / 0,2949254148 EURO
Restantes Avaliações (5 *Gwei*)	0,00124767 ETH / 0,2286604809 EURO
Controlador de Dados (20 *Gwei*)	0,0052335 ETH / 0,959143545 EURO
Entidade Reguladora (20 *Gwei*)	0,0052186 ETH / 0,956412822 EURO
Sujeito dos Dados (20 *Gwei*)	0,006068168 ETH / 1,11211314936 EURO
Termos de Consentimento (20 *Gwei*)	0,009937193333333 ETH / 1,8211894222 EURO
Consentimento (20 *Gwei*)	0,011472808 ETH / 2,10262152216 EURO
Remoção de Consentimento (20 *Gwei*)	0,00404694 ETH / 0,7416826938 EURO
1ª Avaliação (20 *Gwei*)	0,00643696 ETH / 1,1797016592 EURO
Restantes Avaliações (20 *Gwei*)	0,00499068 ETH / 0,9146419236 EURO

Tabela 5.3: Custo médio das transações de teste efetuadas.

Figura 5.5: Visão geral do protótipo da plataforma de captura de dados implementado sob a perspetiva da entidade reguladora “CPD”.

Capítulo 6

Conclusões

O Regulamento Geral de Proteção de Dados (RGPD) e a tecnologia *blockchain* romperam com o paradigma estabelecido na regulação para a proteção de dados e nas transações em sistemas descentralizados respetivamente. Ambos os fenómenos vieram dar ao cidadão europeu comum níveis de poder e de proteção que até então seriam impensáveis. Enquanto o RGPD veio proteger universalmente a privacidade dos cidadãos europeus, através da especificação de regras estritas que têm como objetivo a salvaguarda dos seus dados pessoais; a tecnologia *blockchain* está na base da recente vaga de criptomoedas e suporta a primeira geração de sistemas transacionais verdadeiramente descentralizados.

Foi com base nestes dois fenómenos contemporâneos que partiu este trabalho, que tinha como principal objetivo projetar e implementar o protótipo de um sistema que permita obter o consentimento informado de um sujeito dos dados, segundo o RGPD, através do seu *smartphone* e persistir prova do mesmo em tecnologia *blockchain*. Desde o inicio que também ficaram traçados outros objetivo paralelos, como o de englobar as entidades reguladoras no âmbito do sistema e o de incluir no sistema principal um subsistema reputacional que permitisse a um sujeito dos dados avaliar o desempenho de um controlador de dados a quem consentia a utilização dos seus dados pessoais para os fins especificados no termo de consentimento. Ao longo do trabalho foi projetada uma solução que permite a qualquer entidade (controlador de dados) que pretenda obter o consentimento de terceiros (sujeitos dos dados) para processar os seus dados, respeitando o RGPD, o possa fazer de através de um sistema fácil de integrar. Esse mesmo sistema está encarregue de obter o consentimento do sujeito dos dados, de gerar as provas do mesmo, e de as apresentar à entidade reguladora *on demand*. Para persistir todos os dados, o sistema recorre a uma *contract account* na *blockchain* de uma rede *Ethereum*, invocando transações definidas no *smart contract* que permitem persistir novos dados nessa mesma *contract account*. A implementação de transações que garantem um custo constante torna o custo do armazenamento previsível e escalável face à quantidade de dados a armazenar.

Este trabalho resulta num sistema sofisticado que abstrai de todas as partes envolvidas (sujeito dos dados, controlador de dados e entidade reguladora) o fardo tecnológico e legal resultante da necessidade de estar em conformidade com o RGPD e do recurso a uma tecnologia complexa e sofisticada como a *blockchain* enquanto livro de razão, diminuindo assim significativamente a fricção resultante da adaptação às exigências da nova realidade regulatória.

6.1 Resultados Relevantes

A escolha da plataforma *Ethereum* tornou possível estabelecer paralelos exatos com a rede principal com o objetivo de quantificar custos da solução *Connsent* num cenário real. A natureza determinística da *Ethereum Virtual Machine* permitiu testar o sistema em redes de teste com diferentes características e aferir eventuais custos de persistência de dados do sistema na *blockchain* de uma rede *Ethereum* perante diferentes níveis de carga.

A análise detalhada do domínio do problema, da linguagem *Solidity* e do sistema de transações da plataforma *Ethereum* resultou na implementação de um *smart contract* para a solução *Connsent* que garante transações com custos lineares. Ficou deste modo assegurado que a utilização da *blockchain* para persistir dados do sistema[1] consegue ser previsível financeiramente.

O subsistema reputacional incluído na solução incrementa significativamente o poder do sujeito dos dados que, tradicionalmente, é o elo mais fraco da cadeia de processamento de dados. A interação *mobile first* do sujeito dos dados com o sistema através da aplicação móvel, para além de lhe possibilitar dar o seu consentimento de uma forma simples, permite-lhe também remover o seu consentimento de um modo virtualmente instantâneo através do seu *smartphone* e avaliar o desempenho da entidade a quem confiou os seus dados. Esta nova realidade confere-lhe um novo nível de poder que num cenário pré-RGPD seria pouco comum ou até impensável. A arquitetura da solução contempla que o mesmo subsistema reputacional possa ser monitorizado em tempo real pela entidade reguladora de modo automatizado com o objetivo de detetar casos que possam justificar a sua intervenção.

O controlador de dados e a entidade reguladora vêm a sua atividade simplificada e digitalizada através de uma solução que persiste provas criptográficas de consentimento num livro de razão imutável e completamente auditável. O controlador de dados passa a ser capaz de obter o consentimento de um sujeito dos dados, e a respetiva prova, de um modo completamente desmaterializado e em conformidade com o RGPD. A entidade reguladora, por sua vez, passa a ter um sistema que lhe permite consultar provas dos consentimentos que supervisiona *on demand* e sem que seja necessária interação com o controlador de dados para que tal aconteça.

Finalmente, o facto das provas de consentimento estarem persistidas na *blockchain* e serem acessíveis publicamente representa um nível de transparência condizente com os princípios fundamentais do RGPD.

6.2 Trabalho Futuro

Tendo em conta a crescente diversidade da oferta no mercado de soluções *blockchain*, seria pertinente analisar a viabilidade da utilização de outra plataforma, que não a *Ethereum*, para sustentar o sistema *Connsent*.

A introdução de uma funcionalidade que permitisse ao regulador de dados, no âmbito de um consentimento que supervisiona, aceder aos dados recolhidos pelo controlador de dados utilizando como canal seguro de transmissão a solução apresentada. Esta funcionalidade recorreria a um esquema criptográfico híbrido que cifraria os dados no perímetro do sistema controlador de dados recorrendo a um algoritmo de cifra simétrica e a uma

[1] Entre estes dados encontram-se provas de consentimento, de remoção de consentimento e avaliações.

chave de sessão gerada para o efeito. De seguida, essa chave de sessão seria cifrada assimetricamente com a chave pública associada à entidade reguladora no sistema *Connsent*. O último passo antes da transmissão passaria pela assinatura digital da informação a ser transmitida (dados cifrados e chave de sessão cifrada) com chave privada associada à chave pública do controlador de dados no sistema. Esta informação seria então transmitida à entidade reguladora que realizaria a validação da assinatura digital com a chave pública do controlador de dados do sistema (garantindo assim a autenticidade e integridade da mensagem) e decifrada com a sua chave privada.

De modo a garantir que todas as transações de inserção de *feedback* têm um custo linear, como acontece com todos os outros tipos de transações, seria necessário implementar uma estrutura de dados que tivesse como base um mapeamento. Esta estrutura de dados permitiria reconstruir a ordem de inserção original das entradas de *feedback* de um consentimento persistido na *contract account* do sistema. Outra melhoria ao *smart contract* que suporta o sistema passaria pela inclusão da nota e do índice da avaliação nos eventos do tipo "NewFeedbackEntry", permitindo assim a obtenção de todos os dados necessários ao cálculo da reputação de um controlador de dados sem recurso a qualquer outro tipo de consulta aos dados persistidos na *contract account*.

A solução projetada neste trabalho contempla apenas a utilização do algoritmo SHA-512 para cálculo do resumo criptográfico dos dados recolhidos pelo controlador de dados. Uma das evoluções de uma versão futura passaria pelo suporte para múltiplos algoritmos de resumo criptográfico, passando o identificador do algoritmo criptográfico a ser incluído com os restantes dados do consentimento.

Os formatos de assinatura digital beneficiariam com uma análise que identificasse eventuais ataques que possam ser levados a cabo tirando partido da estrutura dos mesmos. Um exemplo que já foi detetado passa pelo reenvio do *payload* de uma avaliação já existente como sendo uma nova avaliação. Esta situação permite que qualquer utilizador da solução possa duplicar qualquer entrada de *feedback* já existente e podia ser facilmente resolvida pela inclusão do índice da avaliação na mensagem assinada pelo sujeito dos dados.

Capítulo 7

Bibliografia

[1] S. Sater, *Blockchain and the European Union's General Data Protection Regulation: A Chance to Harmonize International Data Flows*. SSRN, 2017.

[2] New York Times, "Facebook says cambridge analytica may have gained 37m more users' data." https://www.theguardian.com/technology/2018/apr/04/facebook-cambridge-analytica-user-data-latest-more-than-thought, 2018. Online, Accessed: 2018-10-04.

[3] CNN, "Giant Equifax data breach: 143 million people could be affected." http://money.cnn.com/2017/09/07/technology/business/equifax-data-breach/index.html, 2017. Online, Accessed: 2018-01-31.

[4] L.-D. Ibanez, K. O'Hara, and E. Simperl, "On blockchains and the general data protection regulation," project report, July 2018.

[5] European Union, "Regulation (EU) 2016/679 of the European Parliament and of the Council." http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32016R0679, 2016. Online, Accessed: 2018-01-31.

[6] Exame Informática, "Comissão nacional de proteção de dados: ≪haveria toda a vantagem em ter a lei do rgpd já aprovada≫." https://bit.ly/2Nh3T04, 2018. Online, Accessed: 2018-09-22.

[7] I. Bashir, *Mastering Blockchain - Second Edition: Distributed ledger technology, decentralization, and smart contracts explained*. "Packt Publishing", 2018.

[8] J. Mattila *et al.*, "The blockchain phenomenon–the disruptive potential of distributed consensus architectures," tech. rep., The Research Institute of the Finnish Economy, 2016.

[9] T. Aste, P. Tasca, and T. Di Matteo, "Blockchain technologies: The foreseeable impact on society and industry," *computer*, vol. 50, no. 9, pp. 18–28, 2017.

[10] Hyperledger Project, "About - hyperledger." https://www.hyperledger.org/about, 2018. Online, Accessed: 2018-09-25.

[11] Google, "Cryptocurrency - explore - google trends." https://trends.google.com/trends/explore?date=today%205-y&q=%2Fm%2F0vpj4_b, 2018. Online, Accessed: 2018-10-05.

[12] New York Times, "Everyone is getting hilariously rich and you're not." https://www.nytimes.com/2018/01/13/style/bitcoin-millionaires.html, 2018. Online, Accessed: 2018-10-05.

[13] New York Times, "Decoding the enigma of satoshi nakamoto and the birth of bitcoin." https://www.nytimes.com/2015/05/17/business/decoding-the-enigma-of-satoshi-nakamoto-and-the-birth-of-bitcoin.html, 2015. Online, Accessed: 2018-09-18.

[14] A. Phillip, J. Chan, and S. Peiris, "A new look at cryptocurrencies," *Economics Letters*, vol. 163, pp. 6–9, 2018.

[15] S. Raval, *Decentralized Applications: Harnessing Bitcoin's Blockchain Technology*. "O'Reilly Media, Inc.", 2016.

[16] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," http://bitcoin.org/bitcoin.pdf," 2008.

[17] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382–401, 1982.

[18] Google, "Hans peter luhn and the birth of the hashing algorithm." https://spectrum.ieee.org/tech-history/silicon-revolution/hans-peter-luhn-and-the-birth-of-the-hashing-algorithm, 2018. Online, Accessed: 2018-01-30.

[19] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, 2014.

[20] S. D. Galbraith, *Mathematics of public key cryptography*. Cambridge University Press, 2012.

[21] GCHQ, "A note on 'non-secret encryption'." https://www.gchq.gov.uk/note-non-secret-encryption, 2016. Online, Accessed: 2018-10-05.

[22] J.-P. Aumasson, *Serious Cryptography: A Practical Introduction to Modern Encryption*. "No Starch Press", 2017.

[23] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[24] W. F. Ehrsam, C. H. W. Meyer, J. L. Smith, and W. L. Tuchman, "Message verification and transmission error detection by block chaining." https://patents.google.com/patent/US4074066A/en, 1976. Online, Accessed: 2018-09-26.

[25] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press, 2016.

[26] A. Antonopoulos, *Mastering Bitcoin 2nd Edition*. "O'Reilly", 2017.

[27] A. Back *et al.*, "Hashcash-a denial of service counter-measure," 2002.

[28] D. Frisby, *Bitcoin: The Future of Money?* "Unbound", 2014.

[29] I.-C. Lin and T.-C. Liao, "A survey of blockchain security issues and challenges.," *IJ Network Security*, vol. 19, no. 5, pp. 653–659, 2017.

[30] G. Karame, E. Androulaki, and S. Capkun, "Two bitcoins at the price of one? double-spending attacks on fast payments in bitcoin.," *IACR Cryptology ePrint Archive*, vol. 2012, no. 248, 2012.

[31] Coindesk, "Feathercoin hit by massive attack." https://www.coindesk.com/feathercoin-hit-by-massive-attack/, 2013. Online, Accessed: 2018-09-30.

[32] Redman, Jamie, "Small ethereum clones getting attacked by mysterious '51 crew'." https://news.bitcoin.com/ethereum-clones-susceptible-51-attacks/, 2016. Online, Accessed: 2018-09-30.

[33] Gray, Marley, "Introducing project "bletchley- current blockchain ecosystem and evolution." https://github.com/Azure/azure-blockchain-projects/blob/master/bletchley/bletchley-whitepaper.md#current-blockchain-ecosystem-and-evolution, 2017. Online, Accessed: 2018-10-05.

[34] V. Buterin, "What proof of stake is and why it matters," *Bitcoin Magazine, August*, vol. 26, 2013.

[35] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," *self-published paper, August*, vol. 19, 2012.

[36] Ethereum Project, "Bitcoin mining now consumes 1% of the world's electricity, more than the entire state of new york." https://bit.ly/2OuNv0N, 2018. Online, Accessed: 2018-09-30.

[37] Ethereum Project, "Ethereum project - proof of stake faqs." https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQs, 2018. Online, Accessed: 2018-09-30.

[38] N. Szabo, "Smart contracts," *http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html*, 1994.

[39] N. Szabo, "Formalizing and securing relationships on public networks," *First Monday*, vol. 2, no. 9, 1997.

[40] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, 2014.

[41] Wired Magazine, "Ethereum is coding's new wild west." https://www.wired.com/story/ethereum-is-codings-new-wild-west/, 2017. Online, Accessed: 2018-10-05.

[42] Coindesk, "Ethereum launches own 'ether' coin, with millions already sold." https://www.coindesk.com/ethereum-launches-ether-coin-millions-already-sold/, 2014. Online, Accessed: 2018-10-05.

[43] Buterin, Vitalik, "So where did the name ethereum come from?." https://forum.ethereum.org/discussion/comment/3389/#Comment_3389, 2014. Online, Accessed: 2018-10-05.

[44] C. Dannen, *Introducing Ethereum and Solidity*. Springer, 2017.

[45] ThoughtWorks, "Solidity — technology radar — thoughtworks." https://www.thoughtworks.com/radar/languages-and-frameworks/solidity, 2018. Online, Accessed: 2018-10-05.

[46] International Business Times, "Blockchain and big data worth watching in the coming year." https://www.ibtimes.co.uk/blockchain-big-data-worth-watching-2017-1596721, 2016. Online, Accessed: 2018-10-05.

[47] Ethereum Project, "Solidity in depth." https://solidity.readthedocs.io/en/v0.4.21/solidity-in-depth.html, 2018. Online, Accessed: 2018-10-05.

[48] Ethereum Project Team, "Solidity in depth." https://solidity.readthedocs.io/en/v0.4.25/solidity-in-depth.htm, 2018. Online, Accessed: 2018-09-21.

[49] Ethereum Project, "Ethereum clients." http://ethdocs.org/en/latest/ethereum-clients/, 2018. Online, Accessed: 2018-10-05.

[50] D. Crockford, "The application/json media type for javascript object notation (json)." https://www.ietf.org/rfc/rfc4627.txt, 2006.

[51] J.-R. W. Group, "Json-rpc 2.0 specification." https://www.jsonrpc.org/specification, 2010. Online, Accessed: 2018-10-05.

[52] Svensson, Conor, "web3j - lightweight ethereum java and android integration library." https://web3j.io/#features, 2018. Online, Accessed: 2018-10-05.

[53] Ethereum Project, "web3.js - ethereum javascript api." https://web3js.readthedocs.io/en/1.0/, 2018. Online, Accessed: 2018-10-05.

[54] V. Buterin, "Olympic: Frontier pre-release." https://blog.ethereum.org/2015/05/09/olympic-frontier-pre-release/, 2015. Online, Accessed: 2018-10-05.

[55] J. Wilcke, "Homestead release." https://blog.ethereum.org/2016/02/29/homestead-release/, 2016. Online, Accessed: 2018-10-05.

[56] Digix, "Announcing kovan - a stable ethereum public testnet." https://medium.com/@Digix/announcing-kovan-a-stable-ethereum-public-testnet-10ac7cb6c85f, 2017. Online, Accessed: 2018-10-05.

[57] P. Kollock, "The production of trust in online markets," *Advances in group processes*, vol. 16, no. 1, pp. 99–123, 1999.

[58] D. Gambetta *et al.*, "Can we trust trust," *Trust: Making and breaking cooperative relations*, vol. 13, pp. 213–237, 2000.

[59] D. H. McKnight and N. L. Chervany, "The meanings of trust," 1996.

[60] Priberam, "Significado/definição de reputação no dicionário priberam da língua portuguesa." https://dicionario.priberam.org/Reputa%C3%A7%C3%A3o, 2018. Online, Accessed: 2018-10-06.

[61] S. Tadelis, "Firm reputation with hidden information," in *Assets, Beliefs, and Equilibria in Economic Dynamics*, pp. 537–553, Springer, 2004.

[62] B. King-Casas, D. Tomlin, C. Anen, C. F. Camerer, S. R. Quartz, and P. R. Montague, "Getting to know you: reputation and trust in a two-person economic exchange," *Science*, vol. 308, no. 5718, pp. 78–83, 2005.

[63] L. Xiong and L. Liu, "Reputation and trust," *Advances in security and payment methods for mobile commerce*, pp. 19–35, 2005.

[64] J. Palme and M. Berglund, "Anonymity on the internet," *Retrieved August*, vol. 15, p. 2009, 2002.

[65] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman, "Reputation systems," *Commun. ACM*, vol. 43, pp. 45–48, Dec. 2000.

[66] P. Resnick and R. Zeckhauser, "Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system," in *The Economics of the Internet and E-commerce*, pp. 127–157, Emerald Group Publishing Limited, 2002.

[67] statista, "ebay: number of users 2018 — statistic." https://www.statista.com/statistics/242235/number-of-ebays-total-active-users/, 2018. Online, Accessed: 2018-10-15.

[68] P. Resnick, R. Zeckhauser, J. Swanson, and K. Lockwood, "The value of reputation on ebay: A controlled experiment," *Experimental economics*, vol. 9, no. 2, pp. 79–101, 2006.

[69] M. I. Melnik and J. Alm, "Does a seller's ecommerce reputation matter? evidence from ebay auctions," *The journal of industrial economics*, vol. 50, no. 3, pp. 337–349, 2002.

[70] Y. Wang and J. Vassileva, "Trust and reputation model in peer-to-peer networks," in *null*, p. 150, IEEE, 2003.

[71] F. Ali, "Amazon and ebay's us market share nears 50%." https://www.digitalcommerce360.com/2018/06/08/amazon-and-ebays-us-market-share-nears-50/, 2018. Online, Accessed: 2018-10-07.

[72] D. Houser and J. Wooders, "Reputation in auctions: Theory, and evidence from ebay," *Journal of Economics & Management Strategy*, vol. 15, no. 2, pp. 353–369, 2006.

[73] eBay, "All about feedback." https://www.ebay.com/help/buying#feedback, 2018. Online, Accessed: 2018-10-07.

[74] Amazon, "Ordering from a third-party seller." https://www.amazon.com/gp/help/customer/display.html/ref=hp_bc_nav?ie=UTF8&nodeId=201889680, 2018. Online, Accessed: 2018-10-07.

Capítulo 8

Apêndices

8.1 Código Fonte do *Smart Contract* do Sistema

```
1  pragma solidity ^0.4.23;
2
3  contract Connsent {
4
5      // ======================================================
6      // SMART CONTRACT VARIABLES
7      // ======================================================
8      address private owner;
9      mapping (string => DataController) private _dataControllers;
10     mapping (string => DataSubject) private _dataSubjects;
11     mapping (string => Regulator) private _regulators;
12     mapping (string => Consent) private _consents;
13     mapping (string => Terms) private _terms;
14     mapping (string => RemovedConsent) private _removedConsents;
15     mapping (string => Feedback) private _feedbacks;
16
17     // ======================================================
```

```
18      // CONSTRUCTOR
19      // ======================================================
20      constructor () public {
21          owner = msg.sender;
22      }
23
24      // ======================================================
25      // ENUMERATIONS
26      // ======================================================
27      enum FeedbackType {
28          NEGATIVE, NEUTRAL, POSITIVE
29      }
30
31      // ======================================================
32      // SMART-CONTRACT EVENTS
33      // ======================================================
34      event NewConsent(string consentUid, string dataSubjectUid, string dataControllerName, string regulatorName);
35      event NewRemovedConsent(string consentUid, string dataSubjectUid, string dataControllerName, string regulatorName);
36      event NewFeedbackEntry(string consentUid, string dataSubjectUid, string dataControllerName, string regulatorName);
37
38      // ======================================================
39      // STRUCTURES
40      // ======================================================
41      struct DataController {
42          string name;
43          string signature;
44          string publicKey;
45          uint timestamp;
46      }
47
48      struct DataSubject {
49          string uid;
50          string signature;
51          string publicKey;
52          uint timestamp;
53      }
54
55      struct Regulator {
56          string name;
57          string signature;
```

```
58          string publicKey;
59          uint timestamp;
60      }
61
62      struct Terms {
63          string uid;
64          string terms;
65          string dataControllerName;
66          string termsSig;
67          uint timestamp;
68      }
69
70      struct Consent {
71          string uid;
72          string termsUid;
73          string dataSubjectUid;
74          string dataControllerName;
75          string regulatorName;
76          string dataDigest;
77          string requestSig;
78          string consentSig;
79          uint timestamp;
80      }
81
82      struct RemovedConsent {
83          string consentUid;
84          string removalSig;
85          uint timestamp;
86      }
87
88      struct Feedback {
89          string consentUid;
90          FeedbackEntry[] entries;
91      }
92
93      struct FeedbackEntry {
94          FeedbackType feedbackType;
95          string message;
96          string feedbackSig;
97          uint timestamp;
```

```
98  }
99
100 // ============================================================
101 // DATA SUBJECT
102 // ============================================================
103 // ADD DATA SUBJECT
104 // Adds a data subject to the smart contract's storage
105 function addDataSubject(string uid, string publicKey, string signature) public
106     returns (string result){
107
108     // Making sure the function was called by the smart-contract owner
109     require(msg.sender==owner,"Only the smart-contract owner can call this transaction!");
110
111     // Not empty validations
112     require(bytes(uid).length>0,"'uid' must not be empty!");
113     require(bytes(signature).length>0,"'signature' must not be empty!");
114     require(bytes(publicKey).length>0,"'publicKey' must not be empty!");
115
116
117     DataSubject storage dataSubject = _dataSubjects[uid];
118
119     // Making sure the new data subject name is not already in use
120     require(bytes(dataSubject.uid).length==0, "'uid' already in use!");
121
122     _dataSubjects[uid] = DataSubject(uid,signature,publicKey,block.timestamp);
123
124     return "Data subject created!";
125 }
126
127 // RETRIEVE DATA SUBJECT
128 // Retrieves a data subject from the smart contract's storage
129 function getDataSubject(string dataSubjectUid) public view
130     returns (string uid, string publicKey, string signature, uint timestamp) {
131
132     require(bytes(dataSubjectUid).length>0,"'uid' must not be empty!");
133
134     // Getting reference
135     DataSubject storage subject = _dataSubjects[dataSubjectUid];
136
137     // Making sure the new data subject exists
```

```
138         require(bytes(subject.uid).length!=0, "Data subject not found!");
139 
140         return (subject.uid,subject.signature,subject.publicKey,subject.timestamp);
141     }
142 
143 
144     // ============================================================
145     // DATA CONTROLLER
146     // ============================================================
147     // ADD DATA CONTROLLER
148     function addDataController(string name, string publicKey, string signature) public
149         returns (string result) {
150 
151         // Making sure the function was called by the smart-contract owner
152         require(msg.sender==owner, "Only the smart-contract owner can call this transaction!");
153 
154         // Not empty validations
155         require(bytes(name).length>0,"'name' must not be empty!");
156         require(bytes(signature).length>0,"'signature' must not be empty!");
157         require(bytes(publicKey).length>0,"'publicKey' must not be empty!");
158 
159 
160         DataController storage dataController = _dataControllers[name];
161 
162         // Making sure new data controller uid is not already in use
163         require(bytes(dataController.name).length==0, "'name' already in use");
164 
165         _dataControllers[name] = DataController(name,signature,publicKey,block.timestamp);
166 
167         return "Data controller successfully created!";
168     }
169 
170     // RETRIEVE DATA CONTROLLER
171     function getDataController(string dataControllerName) public view
172         returns (string name, string publicKey, string signature, uint timestamp) {
173 
174         require(bytes(dataControllerName).length>0,"'dataControllerName' must not be empty!");
175 
176         // Getting reference
177         DataController storage controller = _dataControllers[dataControllerName];
```

```
178
179         // Making sure the data controller exists
180         require(bytes(controller.name).length!=0, "Data controller not found!");
181
182         return (controller.name,controller.signature,controller.publicKey,controller.timestamp);
183     }
184
185     // ============================================================
186     // REGULATOR
187     // ============================================================
188     // ADD REGULATOR
189     function addRegulator(string name, string publicKey, string signature) public
190                 returns (string result) {
191
192         // Making sure the function was called by the smart-contract owner
193         require(msg.sender==owner,"Only the smart-contract owner can call this transaction!");
194
195         // Not empty validations
196         require(bytes(name).length>0,"'name' must not be empty!");
197         require(bytes(signature).length>0,"'signature' must not be empty!");
198         require(bytes(publicKey).length>0,"'publicKey' must not be empty!");
199
200
201         Regulator storage regulator = _regulators[name];
202
203         // Making sure the new data controller uid is not already in use
204         require(bytes(regulator.name).length==0,"'name' already already in use");
205
206         _regulators[name] = Regulator(name,signature,publicKey,block.timestamp);
207
208         return "Regulator successfully created!";
209     }
210
211     // RETRIEVE REGULATOR
212     function getRegulator(string regulatorName) public view
213                 returns (string name, string publicKey, string signature, uint timestamp) {
214
215         require(bytes(regulatorName).length>0,"'regulatorName' must not be empty!");
216
217         // Getting reference
```

```
218         Regulator storage regulator = _regulators[regulatorName];
219 
220         // Making sure the regulator exists
221         require(bytes(regulator.name).length!=0, "Regulator not found!");
222 
223         return (regulator.name,regulator.signature,regulator.publicKey,regulator.timestamp);
224     }
225 
226     // ============================================================
227     // TERMS
228     // ============================================================
229     // ADD TERMS
230     function addTerms(string uid, string termsText, string dataControllerName, string termsSig) public
231         returns (string result){
232 
233         // Making sure the function was called by the smart-contract owner
234         require(msg.sender==owner,"Only the smart-contract owner can call this transaction!");
235 
236         // Not empty validations
237         require(bytes(uid).length>0,"'uid' must not be empty!");
238         require(bytes(termsText).length>0,"'termsText' must not be empty!");
239         require(bytes(dataControllerName).length>0,"'dataControllerName' not be empty!");
240         require(bytes(termsSig).length>0,"'termsSig' must not be empty!");
241 
242         // Making sure the data controller actually exists
243         DataController storage dataController = _dataControllers[dataControllerName];
244         require(bytes(dataController.name).length!=0, "Data controller does not exist!");
245 
246         // Getting terms reference
247         Terms storage terms = _terms[uid];
248 
249         // Making sure the new terms uid is not already in use
250         require(bytes(terms.uid).length==0, "Terms uid already in use!");
251 
252         _terms[uid] = Terms(uid,termsText,dataControllerName,termsSig,block.timestamp);
253 
254         return "Terms successfully created!";
255 
256     }
257 
```

```
258     // RETRIEVE TERMS
259     function getTerms(string termsUid) public view
260         returns(string uid, string consentTerms, string dataControllerName, string termsSig, uint timestamp){
261
262         require(bytes(termsUid).length>0,"'termsUid' must not be empty!");
263
264         // Getting reference
265         Terms storage terms = _terms[termsUid];
266
267         // Making sure the terms exist
268         require(bytes(terms.uid).length!=0, "Terms not found!");
269
270         return (terms.uid,terms.terms,terms.dataControllerName,terms.termsSig,block.timestamp);
271     }
272
273     // ============================================================
274     // CONSENT
275     // ============================================================
276     // ADD CONSENT
277     function addConsent(string uid, string termsUid, string dataSubjectUid, string dataControllerName, string regulatorName, string dataDigest,
            ↪ string requestSig, string consentSig) public
278         returns (string result) {
279
280         // Making sure the function was called by the smart-contract owner
281         require(msg.sender==owner, "Only the smart-contract owner can call this transaction!");
282
283         // Not empty validations
284         require(bytes(uid).length>0,"'uid' must not be empty!");
285         require(bytes(termsUid).length>0,"'termsUid' must not be empty!");
286         require(bytes(dataSubjectUid).length>0,"'dataSubjectUid' not be empty!");
287         require(bytes(dataControllerName).length>0,"'dataControllerName' must not be empty!");
288         require(bytes(regulatorName).length>0,"'regulatorName' must not be empty!");
289         require(bytes(dataDigest).length>0,"'dataDigest' must not be empty!");
290         require(bytes(consentSig).length>0,"'consentSig' must not be empty!");
291         require(bytes(requestSig).length>0,"'requestSig' must not be empty!");
292
293         // Making sure the terms actually exist
294         Terms storage terms = _terms[termsUid];
295         require(bytes(terms.uid).length!=0, "Terms do not exist!");
296
```

```
297         // Making sure the data subject actually exists
298         DataSubject storage dataSubject = _dataSubjects[dataSubjectUid];
299         require(bytes(dataSubject.uid).length!=0, "Data subject does not exist!");
300 
301         // Making sure the data controller actually exists
302         DataController storage dataController = _dataControllers[dataControllerName];
303         require(bytes(dataController.name).length!=0, "Data controller does not exist!");
304 
305         // Making sure the terms were created by the data controller
306         require(compareStrings(bytes(terms.dataControllerName),bytes(dataControllerName)),"Terms were not created by the data controller!");
307 
308         // Making sure the regulator actually exists
309         Regulator storage regulator = _regulators[regulatorName];
310         require(bytes(regulator.name).length!=0, "Regulator does not exist!");
311 
312         // Getting consent reference
313         Consent storage consent = _consents[uid];
314 
315         // Making sure the new consent uid is not already in use
316         require(bytes(consent.uid).length==0, "Consent uid already in use!");
317 
318         _consents[uid] = Consent(uid,termsUid,dataSubjectUid,dataControllerName,regulatorName,dataDigest,requestSig,consentSig,block.timestamp);
319 
320         emit NewConsent(uid,dataSubjectUid,dataControllerName,regulatorName);
321 
322         return "Consent successfully created!";
323 
324     }
325 
326     // RETRIEVE CONSENT
327     function getConsentParts(string consentUid) public view
328         returns(string dataSubjectUid, string dataControllerName, string termsUid, string regulatorName){
329 
330         require(bytes(consentUid).length>0,"'consentUid' must not be empty!");
331 
332         // Getting reference
333         Consent storage consent = _consents[consentUid];
334 
335         // Making sure the consent exists
336         require(bytes(consent.uid).length!=0, "Consent not found!");
```

```
337
338         return (consent.dataSubjectUid,consent.dataControllerName,consent.termsUid,consent.regulatorName);
339     }
340
341     function getConsentValidationData(string consentUid) public view
342         returns(string dataSubjectUid, string termsUid, string dataDigest, string requestSig, string consentSig, uint timestamp){
343
344         require(bytes(consentUid).length>0,"'consentUid' must not be empty!");
345
346         // Getting reference
347         Consent storage consent = _consents[consentUid];
348
349         // Making sure the consent exists
350         require(bytes(consent.uid).length!=0, "Consent not found!");
351
352         return (consent.dataSubjectUid,consent.termsUid,consent.dataDigest,consent.requestSig,consent.consentSig,consent.timestamp);
353     }
354
355     // ============================================================
356     // REMOVED CONSENT
357     // ============================================================
358     // ADD REMOVED CONSENT
359     function addRemovedConsent(string consentUid, string removalSig) public
360         returns (string result){
361
362         // Making sure the function was called by the smart-contract owner
363         require(msg.sender==owner, "Only the smart-contract owner can call this transaction!");
364
365         // Not empty validations
366         require(bytes(consentUid).length>0,"'consentUid' must not be empty!");
367         require(bytes(removalSig).length>0,"'removalSig' must not be empty!");
368
369         // Making sure the consent actually exists
370         Consent storage consent = _consents[consentUid];
371         require(bytes(consent.uid).length!=0, "Consent does not exist!");
372
373         // Getting removed consent reference
374         RemovedConsent storage removedConsent = _removedConsents[consentUid];
375
376         // Making sure the new terms uid is not already in use
```

```
377         require(bytes(removedConsent.consentUid).length==0, "Consent already removed'!");
378
379         _removedConsents[consentUid] = RemovedConsent(consentUid,removalSig,block.timestamp);
380
381         emit NewRemovedConsent(consentUid,consent.dataSubjectUid,consent.dataControllerName,consent.regulatorName);
382
383         return "Consent Removal successfully created!";
384     }
385
386     // RETRIEVE REMOVED CONSENT
387     function getRemovedConsent(string consentUid) public view
388         returns(string removalSig, uint timestamp){
389
390         require(bytes(consentUid).length>0,"'consentUid' must not be empty!");
391
392         // Getting reference
393         RemovedConsent storage removedConsent = _removedConsents[consentUid];
394
395         // Making sure the removed consent exists
396         require(bytes(removedConsent.consentUid).length!=0, "Removed consent not found!");
397
398         return (removedConsent.removalSig,removedConsent.timestamp);
399     }
400
401
402     // ========================================================
403     // FEEDBACK
404     // ========================================================
405     // ADD FEEDBACK ENTRY
406     function addFeedbackEntry(string consentUid, uint feedbackType, string message, string feedbackSig) public
407         returns (string result, uint entryIndex){
408
409         // Making sure the function was called by the smart-contract owner
410         require(msg.sender==owner, "Only the smart-contract owner can call this transaction!");
411
412         // Not empty validations
413         require(bytes(consentUid).length>0,"'consentUid' must not be empty!");
414         require(bytes(message).length>0,"Message must not be empty!");
415         require(bytes(feedbackSig).length>0,"Feedback Signature must not be empty!");
416
```

```
417         // Feedback type enumeration validation
418         require(feedbackType==0||feedbackType==1||feedbackType==2, "Invalid Feedback type value!");
419 
420         // Making sure the consent actually exists
421         Consent storage consent = _consents[consentUid];
422 
423         require(bytes(consent.uid).length!=0, "Consent does not exist!");
424 
425         // Getting removed consent reference
426         Feedback storage feedback = _feedbacks[consentUid];
427 
428         FeedbackType entryFeedbackType;
429         if(feedbackType == 0){
430             entryFeedbackType = FeedbackType.NEGATIVE;
431         } else if (feedbackType == 1){
432             entryFeedbackType = FeedbackType.NEUTRAL;
433         } else if (feedbackType == 2){
434             entryFeedbackType = FeedbackType.POSITIVE;
435         }
436 
437         if(bytes(feedback.consentUid).length==0){
438             _feedbacks[consentUid].consentUid = consentUid;
439             _feedbacks[consentUid].entries.length++;
440             _feedbacks[consentUid].entries[_feedbacks[consentUid].entries.length-1] = FeedbackEntry(entryFeedbackType,message,feedbackSig,block.
                    ↪ timestamp);
441         } else {
442             _feedbacks[consentUid].entries.push(FeedbackEntry(entryFeedbackType,message,feedbackSig,block.timestamp));
443         }
444 
445         emit NewFeedbackEntry(consentUid,consent.dataSubjectUid,consent.dataControllerName,consent.regulatorName);
446 
447         return ("Feedback entry successfully created!",_feedbacks[consentUid].entries.length-1);
448     }
449 
450     // RETRIEVE FEEDBACK
451     function getFeedbackEntry(string consentUid, uint entryIndex) public view
452         returns(uint feedbackType, string message, string feedbackSig, uint timestamp){
453 
454         require(bytes(consentUid).length>0,"'consentUid' must not be empty!");
455 
```

```
456         // Getting reference
457         Feedback storage feedback = _feedbacks[consentUid];
458 
459         // Making sure the feedback exists
460         require(bytes(feedback.consentUid).length!=0, "No feedback entries for given consent uid!");
461 
462         require(entryIndex>=0&&entryIndex<feedback.entries.length,"Entry index out of range!");
463 
464 
465         return (uint(feedback.entries[entryIndex].feedbackType), feedback.entries[entryIndex].message, feedback.entries[entryIndex].feedbackSig, feedback.
                ↪ entries[entryIndex].timestamp);
466     }
467 
468     // COUNT FEEDBACK ENTRIES
469     function getFeedbackEntryCount(string consentUid) public view
470         returns(uint count){
471 
472         require(bytes(consentUid).length>0,"'consentUid' must not be empty!");
473 
474         // Getting reference
475         Feedback storage feedback = _feedbacks[consentUid];
476 
477         // Making sure the feedback exists
478         require(bytes(feedback.consentUid).length!=0, "No feedback entries for given consent uid!");
479 
480         return feedback.entries.length;
481     }
482 
483     function compareStrings (bytes a, bytes b) private pure returns (bool){
484        return keccak256(a) == keccak256(b);
485     }
486 }
```

Listagem 8.1: Código fonte do *smart contract* que suporta o sistema *Connsent*

8.2 Código Fonte Java para *Deploy* do *Smart Contract*

```
1 package pt.ipp.estg.mei;
2 
```

```
3  import org.web3j.crypto.Credentials;
4  import org.web3j.crypto.WalletUtils;
5  import org.web3j.protocol.Web3j;
6  import org.web3j.protocol.http.HttpService;
7  import org.web3j.tx.Contract;
8  import pt.ipp.estg.mei.contracts.Connsent;
9  import java.math.BigInteger;
10
11 public class ConnsentDeploy {
12     public static void main(String[] args) throws Exception {
13         new ConnsentDeploy().run();
14     }
15
16     void run() throws Exception {
17         // Connecting to Ethereum node
18         Web3j web3j = Web3j.build(new HttpService("http://ETHEREUM-NODE-WITH-RPC-SERVER-ENABLED:PORT"));
19         System.out.println("Connected to Ethereum client version: " + web3j.web3ClientVersion().send().getWeb3ClientVersion());
20
21         // Loading wallet
22         Credentials credentials = WalletUtils.loadCredentials("WALLET-PASSWORD ", "/full/path/to/wallet.json");
23         System.out.println("Loaded wallet " + credentials.getAddress() + "!");
24
25         // Smart contract deployment
26         BigInteger GAS_LIMIT = BigInteger.valueOf(7000000);
27         Connsent contract = Connsent.deploy(web3j, credentials, Contract.GAS_PRICE, GAS_LIMIT).send();
28
29         // Contract account info
30         System.out.println("Smart contract deployed to address " + contract.getContractAddress());
31         System.out.println("View contract account at https://ETHEREUM-NETWORK.etherscan.io/address/" + contract.getContractAddress());
32     }
33 }
```

Listagem 8.2: Código fonte Java para o *deploy* do *smart contract* em redes *Ethereum*.

8.3 Código Fonte da Rota “/consent” (@POST)

```
1 @POST
2 @Path("consent")
3 @Consumes(MediaType.APPLICATION_JSON)
```

```
4  @Produces(MediaType.APPLICATION_JSON)
5  public Response addConsent(JSONObject reqJSON) throws Exception {
6
7      // Checking required fields
8      if( !reqJSON.has("uid") || !reqJSON.has("subjectUid") || !reqJSON.has("termsUid") || !reqJSON.has("controllerName") ||
9          !reqJSON.has("regulatorName") || !reqJSON.has("dataDigest") || !reqJSON.has("consentSig")|| !reqJSON.has("requestSig")){
10         return Response.status(400).entity(new JSONObject().put("message","Missing mandatory attributes!")).build();
11     }
12
13     String consentUid = reqJSON.getString("uid");
14
15     String subjectUid = reqJSON.getString("subjectUid");
16     Tuple4<String,String,String, BigInteger> subjectBC;
17     try {
18         // Getting data subject from blockchain
19         subjectBC = ConsentSingleton.getInstance().getDataSubject(subjectUid).send();
20     } catch (ArrayIndexOutOfBoundsException | StringIndexOutOfBoundsException e){
21         return Response.status(404).entity(new JSONObject().put("message","Subject '"+subjectUid+"' not found in smart-contract's storage!")).
               ↪ build();
22     }
23
24     String termsUid = reqJSON.getString("termsUid");
25     try {
26         // Getting consent terms from blockchain
27         ConsentSingleton.getInstance().getTerms(termsUid).send();
28     } catch (ArrayIndexOutOfBoundsException | StringIndexOutOfBoundsException e){
29         return Response.status(404).entity(new JSONObject().put("message","Terms '"+termsUid+"' not found in smart-contract's storage!")).build
               ↪ ();
30     }
31
32     String controllerName = reqJSON.getString("controllerName");
33     Tuple4<String,String,String,BigInteger> controllerBC;
34     try {
35         // Getting controller data from blockchain
36         controllerBC = ConsentSingleton.getInstance().getDataController(controllerName).send();
37     } catch (ArrayIndexOutOfBoundsException | StringIndexOutOfBoundsException e){
38         return Response.status(404).entity(new JSONObject().put("message","Controller '"+controllerName+"' not found in smart-contract's
               ↪ storage!")).build();
39     }
40
```

```
41      String regulatorName = reqJSON.getString("regulatorName");
42      try {
43          // Getting regulator data from blockchain
44          ConnsentSingleton.getInstance().getRegulator(regulatorName).send();
45      } catch (ArrayIndexOutOfBoundsException | StringIndexOutOfBoundsException e) {
46          return Response.status(404).entity(new JSONObject().put("message","Regulator '"+regulatorName+"' not found in smart-contract's
                ↪ storage!")).build();
47      }
48
49      String dataDigest = reqJSON.getString("dataDigest");
50
51      String requestSig = reqJSON.getString("requestSig");
52
53      // Validating consent request signature
54      String requestSignatureMessage = consentUid+"/"+termsUid+"/"+subjectUid+"/"+controllerName+"/"+regulatorName+"/"+dataDigest;
55
56      String controllerPublicKey = controllerBC.getValue3()
57      if(!CryptoHelper.verifySignature(controllerPublicKey,requestSignatureMessage.getBytes("UTF-8"),requestSig)){
58          // Invalid request signature
59          return Response.status(400).entity(new JSONObject().put("message","Request signature verification failed!")).build();
60      }
61
62      // Validating consent signature
63      String consentSig = reqJSON.getString("consentSig");
64      String consentMessageSignature =requestSignatureMessage+"/"+requestSig;
65
66      String subjectPublicKey = subjectBC.getValue3();
67      if(!CryptoHelper.verifySignature(subjectPublicKey,consentMessageSignature.getBytes("UTF-8"),consentSig)){
68          // Invalid consent signature
69          return Response.status(400).entity(new JSONObject().put("message","Consent signature verification failed!")).build();
70      }
71
72      TransactionReceipt receipt;
73      try{
74          // Persisting consent in blockchain
75          receipt = ConnsentSingleton.getInstance().addConsent(consentUid,termsUid,subjectUid,controllerName,regulatorName,dataDigest,requestSig,
                ↪ consentSig).send();
76      } catch (Exception e){
77          // Internal server error
78          return Response.status(500).entity(new JSONObject().put("message","Error while persisting consent in blockchain! Try again later...")).
```

```
            ↪ build();
79      }
80
81      return Response.ok(new JSONObject() .put("message","Consent successfully persisted!")
82                                          .put("transactionHash",receipt.getTransactionHash())
83                                          .put("link",Env.ETHEREUM_TRANSACTION_LINK.getValue()+receipt.getTransactionHash())
84                                          .put("contractAddress",receipt.getContractAddress())).build();
85  }
86
87  static final public boolean verifySignature(String publicKeyBase64, byte[] data, String signatureBase64) throws InvalidKeyException,
        ↪ NoSuchProviderException, NoSuchAlgorithmException, SignatureException, InvalidKeySpecException {
88      Signature verifySig = Signature.getInstance("SHA512withECDSA","BC");
89      PublicKey pubKey = KeyFactory.getInstance("ECDSA","BC").generatePublic(new X509EncodedKeySpec(Base64.decode(publicKeyBase64)));
90      verifySig.initVerify(pubKey);
91      verifySig.update(data);
92      return verifySig.verify(Base64.decode(signatureBase64));
93  }
```

Listagem 8.3: Código fonte da implementação da rota "/consent" (@POST) da REST API.

8.4 Código Fonte da Classe ConnsentSingleton.java

```
1   package pt.ipp.estg.mei.connsent.smartcontracts;
2
3   import okhttp3.OkHttpClient;
4   import org.web3j.crypto.CipherException;
5   import org.web3j.crypto.Credentials;
6   import org.web3j.crypto.WalletUtils;
7   import org.web3j.protocol.Web3j;
8   import org.web3j.protocol.core.DefaultBlockParameter;
9   import org.web3j.protocol.http.HttpService;
10  import org.web3j.tx.Contract;
11  import org.web3j.tx.ManagedTransaction;
12  import pt.ipp.estg.mei.connsent.utils.Env;
13  import java.io.IOException;
14  import java.util.concurrent.TimeUnit;
15
16  public final class ConnsentSingleton {
17
```

```
18      static private Connsent connsent = null;
19      static private Web3j web3j = null;
20      static private DefaultBlockParameter initialBlock = null;
21
22      public static final Connsent getInstance() throws IOException, CipherException {
23
24          if(connsent == null){
25
26              // Getting references for environment variables
27              String nodeAddr=Env.ETHEREUM_NODE.getValue();
28              String walPath=Env.ETHEREUM_WALLET_PATH.getValue();
29              String walSec=Env.ETHEREUM_WALLET_SECRET.getValue();
30              String contAddr=Env.BLOCKCHAIN_SMART_CONTRACT_ADDRESS
31                              .getValue();
32
33              // Building HTTP client
34              OkHttpClient.Builder builder =
35                      new OkHttpClient.Builder();
36              Long tos = 300L;
37              builder.connectTimeout(tos, TimeUnit.SECONDS);
38              builder.readTimeout(tos, TimeUnit.SECONDS);
39              builder.writeTimeout(tos, TimeUnit.SECONDS);
40              OkHttpClient client = builder.build();
41              // Building web3j service
42              web3j = Web3j.build(new HttpService(nodeAddr,
43                                                  client,
44                                                  false));
45
46
47
48
49              // Loading credentials
50              Credentials creds = WalletUtils.loadCredentials(walSec,
51                                                              walPath);
52
53              // Loading contract from blockchain
54              connsent = Connsent.load(contAddr,
55                                       web3j,
56                                       creds,
57                                       ManagedTransaction.GAS_PRICE,
```

```
58                                          Contract.GAS_LIMIT);
59          }
60          return connsent;
61      }
62
63      public static final DefaultBlockParameter getSmartContractInitialBlock(){
64
65          if(initialBlock==null){
66              if(Env.BLOCKCHAIN_SMART_CONTRACT_INITIAL_BLOCK.isSet()){
67                  initialBlock = new DefaultBlockParameterNumber(BigInteger.valueOf(Long.valueOf(Env.
                        ↪ BLOCKCHAIN_SMART_CONTRACT_INITIAL_BLOCK.getValue())));
68              } else {
69                  initialBlock = DefaultBlockParameterName.EARLIEST;
70              }
71          }
72
73          return initialBlock;
74      }
75  }
```

Listagem 8.4: Código Fonte da Classe ConnsentSingleton.java.

8.5 Transações de Teste

Entidades Reguladoras

CPD

Chave Pública: ME4wEAYHKoZIzj0CAQYFK4EEACEDOgAEuXDKqndcZcnr52GFK02DynvTSCDlAqNChiP+X2iU7w3Bm+fb7yGLiW4VHCTXpatcBhuXinHZ4gY=
Valor Assinado: CPD/ME4wEAYHKoZIzj0CAQYFK4EEACEDOgAEuXDKqndcZcnr52GFK02DynvTSCDlAqNChiP+X2iU7w3Bm+fb7yGLiW4VHCTXpatcBhuXinHZ4gY=
Assinatura Digital de Registo: MDwCHGyw2xXquyscY2nduVm/iOzsBBOLvizwNUUpzTkCHHJcp9Sic+rOdjFhZeO8TpIP5jthS9xnhzfBhWA=
Transação Rinkeby: `https://rinkeby.etherscan.io/tx/0x33d8b321bbc10e7f4646214b1ecf0f3b43db9da5c8445586d96eef5487dd338f`
Transação Ropsten: `https://ropsten.etherscan.io/tx/0x33d8b321bbc10e7f4646214b1ecf0f3b43db9da5c8445586d96eef5487dd338f`
Bloco Rinkeby: 3281125
Bloco Ropsten: 4364682
***Gas* Necessário:** 260930

Controladores de Dados

Lusocom

Chave Pública: ME4wEAYHKoZIzj0CAQYFK4EEACEDOgAEvVHTboHWzqNjRjBDssfRLBK27+CzbF6AQcrfYBi+I7812NIZHXzH4oqIcEEuVPCuZemuQowvPS0=
Valor Assinado: Lusocom/ME4wEAYHKoZIzj0CAQYFK4EEACEDOgAEvVHTboHWzqNjRjBDssfRLBK27+CzbF6AQcrfYBi+I7812NIZHXzH4oqIcEEuVPCuZemuQowvPS0=
Assinatura Digital de Registo: MDwCHBJzEafrz1zR3fwiGsBqbqGsNC3Pxxd3berJKwUCHDEKtppkCCyKzOSO45hLk16VTuzcg7v+iMqkTxw=
Transação Rinkeby: `https://rinkeby.etherscan.io/tx/0xccdd83b508947563c229d00359984767110618846d6af6f4927b7711aaa68d22`
Transação Ropsten: `https://ropsten.etherscan.io/tx/0xccdd83b508947563c229d00359984767110618846d6af6f4927b7711aaa68d22`
Bloco Rinkeby: 3281147
Bloco Ropsten: 4364714
***Gas* Necessário:** 261403

Electro-Portugal

Chave Pública: ME4wEAYHKoZIzj0CAQYFK4EEACEDOgAEZdV/dtEdnlZnd5nv4tToyIunno/ZIFZxTayX9Givwu+w8WCOsaqTrW4qgxXuFW2WUn2CHafQVeU=
Valor Assinado: Electro-Portugal/ME4wEAYHKoZIzj0CAQYFK4EEACEDOgAEZdV/dtEdnlZnd5nv4tToyIunno/ZIFZxTayX9Givwu+w8WCOsaqTrW4qgxXuFW2WUn2CHafQVeU=
Assinatura Digital de Registo: MD0CHE2ABMj3BGPKZYMm/nJ6iBReRuFSKdpw3ARFN+ECHQCPrfaCVmc2JBr8mNRE7JYU7EWBVufxg2gvEBVS

Transação Rinkeby: https://rinkeby.etherscan.io/tx/0x5bcae4f6fd707a67e5d928ee36fb8facf574d5968c0750af834b6f525ee798b0
Transação Ropsten: https://ropsten.etherscan.io/tx/0x2162ea80c5ac6b9d4fe2153fb87427c8c35bf285f248aaab02d8d59370198eb6
Bloco Rinkeby: 3281200
Bloco Ropsten: 4364764
***Gas* Necessário:** 261979

Portugás

Chave Pública: ME4wEAYHKoZIzj0CAQYFK4EEACEDOgAEGBWAq1jktBW/qKdVSydg+VIwpa4CJrqcbGw+kKvxekiadDgD0zApRJVbTJfE6vz0e+zvKjQ94Zc=
Valor Assinado: Portugás/ME4wEAYHKoZIzj0CAQYFK4EEACEDOgAEGBWAq1jktBW/qKdVSydg+VIwpa4CJrqcbGw+kKvxekiadDgD0zApRJVbTJfE6vz0e+zvKjQ94Zc=
Assinatura Digital de Registo: MDwCHDifkUWSBm2CdxM2PpGKdVTTHqVJM2W+RuN/CgYCHA0ggfSGR+4zjZuKdSZ3qog8PRND6VoBnC7rmoo=
Transação Rinkeby: https://rinkeby.etherscan.io/tx/0x03946840b3fdbd5fb93d6c2bb634117c3a6255b468c9389d2cee52cc82b56c18
Transação Ropsten: https://ropsten.etherscan.io/tx/0xc17c776ebb67f578b1d776418c5bf10fd2c9a86d1a6d2bd6f326247af4610ad9
Bloco Rinkeby: 3281598
Bloco Ropsten: 4365226
***Gas* Necessário:** 261531

PPorto - ESTG

Chave Pública: ME4wEAYHKoZIzj0CAQYFK4EEACEDOgAEcm2xa0LVJ6hseryYHeT2EN5g8lHJ03WJ7vtIs8iGhtY3i38lDvcG1dJliZR7eScfVOV1brN+EwM=
Valor Assinado: PPorto-ESTG/ME4wEAYHKoZIzj0CAQYFK4EEACEDOgAEcm2xa0LVJ6hseryYHeT2EN5g8lHJ03WJ7vtIs8iGhtY3i38lDvcG1dJliZR7eScfVOV1brN+EwM=
Assinatura Digital de Registo: MD0CHCWiiuqPZBZFI4Szm+eKiWIoAaUB+27iSi6huMQCHQCW4s8vMwPcIQ6tAa3f6WEr8hDZZOebJvTd1ohO
Transação Rinkeby: https://rinkeby.etherscan.io/tx/0x106031a4fd6cb98ad4a0f3ffc7d79c32d34c43357ccdf72ac83b25c71a11c27d
Transação Ropsten: https://ropsten.etherscan.io/tx/0x68d513f7b1243a2d688da639463c23ca8df6e99205936a1d171d6e089608cc0a
Bloco Rinkeby: 3340558
Bloco Ropsten: 4430684
***Gas* Necessário:** 261787

Sujeitos dos Dados

08fb82a7-ef19-41fa-bd0a-07d7656aedef

Chave Pública: ME4wEAYHKoZIzj0CAQYFK4EEACEDOgAEr+BsWY9vZ4gBM6oz7fQl66DUVw9GiTM7rb2YypgrAp3SQx8tcKO1pKhoMuzuLizHC5V+Z1M1NrY=

Valor Assinado: 08fb82a7-ef19-41fa-bd0a-07d7656aedef/ME4wEAYHKoZIzj0CAQYFK4EEACEDOgAEr+BsWY9vZ4gBM6oz7fQl66DUVw9GiTM7rb2YypgrAp3SQx8tcKO1pKhoMuzuLizHC5V+Z1M1NrY=

Assinatura Digital de Registo: MD4CHQDN8mZgRANXnRpXLXnpDW+xNv45Ck0jLC3CCN23Ah0AqBt4sCQ9y1i7zwPVv4Kr+89KBx4zaRC9ay17VQ==

Transação Rinkeby: https://rinkeby.etherscan.io/tx/0x382899095f4500eda85de1d6039cf92f1d67648fb58ea0690c2d63f44417f53e

Transação Ropsten: https://ropsten.etherscan.io/tx/0x044c6073ed1e4c961c861f264929f7ba4d91071020d1b606244f8b9d7a9265fb

Bloco Rinkeby: 3281260

Bloco Ropsten: 4364844

***Gas* Necessário:** 303562

d8a216e0-b694-4e93-b678-ee8eec7a4d0d

Chave Pública: ME4wEAYHKoZIzj0CAQYFK4EEACEDOgAEcDGrF93xz+rrFMX1wSemVOfrxlPZAQq8Ga0GOedUGeDy3P5vUUDx9HdXDVv3WbZt5ApCEKIEpkg=

Valor Assinado: d8a216e0-b694-4e93-b678-ee8eec7a4d0d/ME4wEAYHKoZIzj0CAQYFK4EEACEDOgAEcDGrF93xz+rrFMX1wSemVOfrxlPZAQq8Ga0GOedUGeDy3P5vUUDx9HdXDVv3WbZt5ApCEKIEpkg=

Assinatura Digital de Registo: MD0CHQCG2G8Jr3Lq1YSAGj/LobdXoQe0VdMm7fGpBv4fAhw4ovNhSspsUrY69gLU2o1Rbni5sqwQmHJkdrN+

Transação Rinkeby: https://rinkeby.etherscan.io/tx/0x98fd74f112cf11c6e196fe6d6a205c71f9d46b61c9f919e9a759f0ee79010746

Transação Ropsten: https://ropsten.etherscan.io/tx/0x648217a75a7d35a7d618a246e4b7e257c931b50780b118e997f93db29b5cae0d

Bloco Rinkeby: 3281282

Bloco Ropsten: 4364864

***Gas* Necessário:** 303306

ab3f5c76-c00a-4171-b81b-a4ca3879479a

Chave Pública: ME4wEAYHKoZIzj0CAQYFK4EEACEDOgAEcDGrF93xz+rrFMX1wSemVOfrxlPZAQq8Ga0GOedUGeDy3P5vUUDx9HdX

Valor Assinado: ab3f5c76-c00a-4171-b81b-a4ca3879479a/ME4wEAYHKoZIzj0CAQYFK4EEACEDOgAEE0AwLqGM5UN52nXk2MfyFgj/DYLRfrio0VOPFRw13OApd8y7FNBbHqTpsUbCs3GSXwnJDXiWGqY=

Assinatura Digital de Registo: MD4CHQDNcovtP+rT6I/DmUHLgoijBiRlo1ndoC052ayVAh0AnJDnuHJztNjPUxV1y6rabJUOaWUB6KlQBXdzWw==

Transação Rinkeby: https://rinkeby.etherscan.io/tx/0x2857c90554697c5be52e55b71fed090e434a2e2a881e4bb21f4d7275734945ce

Transação Ropsten: https://ropsten.etherscan.io/tx/0x1183cf69a22578760b08a498f9c95c0ed3afb1111960be585519cb3d6492a59e

Bloco Rinkeby: 3281291

Bloco Ropsten: 4364872

***Gas* Necessário:** 303562

b493a9fe-32fb-4e29-910e-2254dd44b474

Chave Pública: ME4wEAYHKoZIzj0CAQYFK4EEACEDOgAEmaBiDpuXrMCVmuESzG9zrkYC/pPH0JG8QpLg+XEtwN1FpaWq0owE708VXAEMcnznNNVBrSpQL4E=
Valor Assinado: b493a9fe-32fb-4e29-910e-2254dd44b474/ME4wEAYHKoZIzj0CAQYFK4EEACEDOgAEmaBiDpuXrMCVmuESzG9zrkYC/pPH0JG8QpLg+XEtwN1FpaWq0owE708VXAEMcnznNNVBrSpQL4E=
Assinatura Digital de Registo: MDwCHFoDYd4p6k/onnZnhyWcj49a9nLP+V1CgvOE8WwCHAXBmjbvsOLBWKPuO/iA9IRFdkHpfYEwdKokfqU=
Transação Rinkeby: https://rinkeby.etherscan.io/tx/0xbed758c0dd00ca9c356c0f6a7369ce1fca037c6b0bbf372bb885da39cc85d874
Transação Ropsten: https://ropsten.etherscan.io/tx/0x4432315412c290757f6d1b4f2503e6dcc2a11c4bf4555c406c493e35b23b0b05
Bloco Rinkeby: 3281301
Bloco Ropsten: 4364885
***Gas* Necessário:** 303306

5478b805-0fad-495c-8295-d67873e31ba0

Chave Pública: ME4wEAYHKoZIzj0CAQYFK4EEACEDOgAE6UNNFd43NFY2OF+xAbUGzRWoE+uCQ4Y1smqx+wH59yP9TzEHJarve7/GeL/WAi8dVhK6EVpcg6k=
Valor Assinado: 5478b805-0fad-495c-8295-d67873e31ba0/ME4wEAYHKoZIzj0CAQYFK4EEACEDOgAE6UNNFd43NFY2OF+xAbUGzRWoE+uCQ4Y1smqx+wH59yP9TzEHJarve7/GeL/WAi8dVhK6EVpcg6k=
Assinatura Digital de Registo: MD0CHH7UYivXi50tsaiWgLhvSqAfHRTOM3zEGR3FNe0CHQDgoLOdLBZ/qcQrMUh6L0Ohr779xZBnL7IZZsBZ
Transação Rinkeby: https://rinkeby.etherscan.io/tx/0x09b2dcc07b4ff4c547ea61286207da8303abacafb5c38c006dc3e62dc9e047a1
Transação Ropsten: https://ropsten.etherscan.io/tx/0xd3f83bd13b5680e9787daa1c020e0dacd2200f94a2050af1c5a1c116281a3de3
Bloco Rinkeby: 3281318
Bloco Ropsten: 4364904
***Gas* Necessário:** 303306

Termos de Consentimento

9dbaa92e-f53f-4d25-8d67-2d3efee4683b

Controlador de Dados: Electro-Portugal
Texto do Termo: Lorem ipsum dolor sit amet, dicam tractatos mei ne, per dicit nonumy menandri an, his at consul legere phaedrum. Ad eos ullum noster docendi, qui at odio graeco honestatis, ius eu putent facete. Minim inimicus dissentias ei sea. Laudem honestatis est ex, erroribus assueverit ne sea. Sed at euismod tibique, vis at erant praesent similique.
Valor Assinado: 9dbaa92e-f53f-4d25-8d67-2d3efee4683b/Lorem ipsum dolor sit amet, dicam tractatos mei ne, per dicit nonumy menandri an, his at consul legere phaedrum. Ad eos ullum noster docendi, qui at odio graeco honestatis, ius eu putent facete. Minim inimicus dissentias ei sea. Laudem honestatis est ex, erroribus assueverit ne sea. Sed at euismod tibique, vis at erant praesent similique./Electro-Portugal

Assinatura Digital de Registo de Termo de Consentimento: MD0CHQD3Kpwk HWiiDAT/8FpO8wFrBg5U33wKG8LxqRsCAhx21yNwFu4Sc4KhUprp7WVZw2qjmTXl 0HHHI15P
Transação Rinkeby: https://rinkeby.etherscan.io/tx/0xcc4c73644d7ec6013f85 6a1506c3bd92800a9e5df34be4f8b078b6019f5e49f3
Transação Ropsten: https://ropsten.etherscan.io/tx/0xfcd5af631e66e28bd8bb 92133714cec4be9f122411c8176527469bc0166d945e
Bloco Rinkeby: 3281244
Bloco Ropsten: 4364827
***Gas* Necessário:** 482665

36dbaae2-930d-41ec-a4e6-d232b286137a

Controlador de Dados: Portugás
Texto do Termo: Pro ad prima mediocrem. Mea no causae ornatus, id mel utroque interesset, pri invidunt tractatos cu. Brute consequat democritum an sed. Dolore electram urbanitas qui et, cu quo sumo meliore reprimique, an mel elit consul nostro. Ad illum graeci aliquip mei, pri oporteat consequat cu, at ius velit nostro laboramus. Sea tota conceptam ad, ne pro primis sensibus.
Valor Assinado: 36dbaae2-930d-41ec-a4e6-d232b286137a/Pro ad prima mediocrem. Mea no causae ornatus, id mel utroque interesset, pri invidunt tractatos cu. Brute consequat democritum an sed. Dolore electram urbanitas qui et, cu quo sumo meliore reprimique, an mel elit consul nostro. Ad illum graeci aliquip mei, pri oporteat consequat cu, at ius velit nostro laboramus. Sea tota conceptam ad, ne pro primis sensibus./Portugás
Assinatura Digital de Registo de Termo de Consentimento: MD0CHQD8VGFH biYPrNYVwoTQugs8Arzjj9FN+ONGrdGyAhxZKGUTsESqv/s4aNIrs+dlfHhrQlWkVq WIW+Ac
Transação Rinkeby: https://rinkeby.etherscan.io/tx/0x2ac1970660f9968b9284 dc64c6f9065d9be812e23550ec6f0dbd9f5151b48267
Transação Ropsten: https://ropsten.etherscan.io/tx/0xbf281724a357c8ec9150 f47fcddfed5b8221ecbbe39a9ec3ad3d9cbf797644b2
Bloco Rinkeby: 3281619
Bloco Ropsten: 4365243
***Gas* Necessário:** 503829

94d53e49-5193-416b-ad0d-bbee44ac47c2

Controlador de Dados: PPorto - ESTG
Texto do Termo: Pro ad prima mediocrem. Mea no causae ornatus, id mel utroque interesset, pri invidunt tractatos cu. Brute consequat democritum an sed. Dolore electram urbanitas qui et, cu quo sumo meliore reprimique, an mel elit consul nostro. Ad illum graeci aliquip mei, pri oporteat consequat cu, at ius velit nostro laboramus. Sea tota conceptam ad, ne pro primis sensibus.
Valor Assinado: 94d53e49-5193-416b-ad0d-bbee44ac47c2/Pro ad prima mediocrem. Mea no causae ornatus, id mel utroque interesset, pri invidunt tractatos cu. Brute consequat democritum an sed. Dolore electram urbanitas qui et, cu quo sumo meliore reprimique, an mel elit consul nostro. Ad illum graeci aliquip mei, pri oporteat consequat cu, at ius velit nostro laboramus. Sea tota conceptam ad, ne pro primis sensibus./PPorto - ESTG

Assinatura Digital de Registo de Termo de Consentimento: MD0CHQCiYszIw0BlzWIWgug32LZq6VT/kxZKbeYSF2LIAhwXou5cK2Gl+dM5mvQq/cli0zmiahCnEgNvKFte
Transação Rinkeby: https://rinkeby.etherscan.io/tx/0x06b410496643e1d53ebe8acbeac38b1f2e7082af6649b707ec9124b7ed3d7d93
Transação Ropsten: https://ropsten.etherscan.io/tx/0x6086b3cf749cbc049d379fa8b2b8c16888506a71554dce284a5ee93068d94e6c
Bloco Rinkeby: 3358087
Bloco Ropsten: 4455300
***Gas* Necessário:** 504085

Consentimentos

211413fe-0d4c-4f78-a8ea-3cbfc5a90ed2

Controlador de Dados: Electro-Portugal
Termo de Consentimento: 9dbaa92e-f53f-4d25-8d67-2d3efee4683b
Entidade Reguladora: CPD
Sujeito dos Dados: 08fb82a7-ef19-41fa-bd0a-07d7656aedef
Resumo Criptográfico dos Dados: 98d3f33cb16d434d876a3799acaad6be05082cff056892a9b0d5061b52de8f0760ad4366c191bef8a37087618e53985f2a6b57584e93545e0669e4eff27713b4
Valor Assinado Para Pedido de Consentimento: 211413fe-0d4c-4f78-a8ea-3cbfc5a90ed2/9dbaa92e-f53f-4d25-8d67-2d3efee4683b/08fb82a7-ef19-41fa-bd0a-07d7656aedef/Electro-Portugal/CPD/98d3f33cb16d434d876a3799acaad6be05082cff056892a9b0d5061b52de8f0760ad4366c191bef8a37087618e53985f2a6b57584e93545e0669e4eff27713b4
Assinatura Digital de Pedido de Consentimento: MD0CHQCAC1HRa4VlyBl8iOa0kdIeTmpV3usfQ3zOkzVqAhwfonZGHyH7ewzY9/K22Fwm3TEtfU0743S7VyWo
Valor Assinado Para Consentimento: 211413fe-0d4c-4f78-a8ea-3cbfc5a90ed2/9dbaa92e-f53f-4d25-8d67-2d3efee4683b/08fb82a7-ef19-41fa-bd0a-07d7656aedef/Electro-Portugal/CPD/98d3f33cb16d434d876a3799acaad6be05082cff056892a9b0d5061b52de8f0760ad4366c191bef8a37087618e53985f2a6b57584e93545e0669e4eff27713b4/MD0CHQCAC1HRa4VlyBl8iOa0kdIeTmpV3usfQ3zOkzVqAhwfonZGHyH7ewzY9/K22Fwm3TEtfU0743S7VyWo
Assinatura Digital de Consentimento: MD0CHAsIMCzETfrKnnSwuzrAFJUcY/0K6OjhXzOdGw4CHQDTMlfXKrKMqcoXWu1zZvNqX/jpTnBLvq1UJSWk
Transação Rinkeby: https://rinkeby.etherscan.io/tx/0xf88d01f1728724a39d8d4b119345f8b9c9f417b13b7bd1c0fe94fb6e888d59d5
Transação Ropsten: https://ropsten.etherscan.io/tx/0x2e8d5c60c35d5fd33e73cad0bb97ed80e98f4fd568404e2865e2b1602a63f62b
Bloco Rinkeby: 3281781
Bloco Ropsten: 4365459
***Gas* Necessário:** 573762

a4bdd3f6-225f-4264-9dbd-538e0d04b450

Controlador de Dados: Electro-Portugal
Termo de Consentimento: 9dbaa92e-f53f-4d25-8d67-2d3efee4683b
Entidade Reguladora: CPD
Sujeito dos Dados: d8a216e0-b694-4e93-b678-ee8eec7a4d0d

Resumo Criptográfico dos Dados: d73f6b38da92a4a5ca16d5fd5f6e96459f91af6b276f2d499e82d969675ce5b95b01584ca4bece501bc1f98f3cacb54e1e3cdc7c6bd137aef9ea569f0da1f480
Valor Assinado Para Pedido de Consentimento: a4bdd3f6-225f-4264-9dbd-538e0d04b450/9dbaa92e-f53f-4d25-8d67-2d3efee4683b/d8a216e0-b694-4e93-b678-ee8eec7a4d0d/Electro-Portugal/CPD/d73f6b38da92a4a5ca16d5fd5f6e96459f91af6b276f2d499e82d969675ce5b95b01584ca4bece501bc1f98f3cacb54e1e3cdc7c6bd137aef9ea569f0da1f480
Assinatura Digital de Pedido de Consentimento: MDwCHHi3qgdI6qCQa9WMBI2eZTDs6YBp6igL/1XlSwYCHHuLm6ZpFIr1ch7MhqgFWCTkd65HHiHudLKhbSI=
Valor Assinado Para Consentimento: a4bdd3f6-225f-4264-9dbd-538e0d04b450/9dbaa92e-f53f-4d25-8d67-2d3efee4683b/d8a216e0-b694-4e93-b678-ee8eec7a4d0d/Electro-Portugal/CPD/d73f6b38da92a4a5ca16d5fd5f6e96459f91af6b276f2d499e82d969675ce5b95b01584ca4bece501bc1f98f3cacb54e1e3cdc7c6bd137aef9ea569f0da1f480/MDwCHHi3qgdI6qCQa9WMBI2eZTDs6YBp6igL/1XlSwYCHHuLm6ZpFIr1ch7MhqgFWCTkd65HHiHudLKhbSI=
Assinatura Digital de Consentimento: MD0CHQCPny9IFrDu/XM3utgI0R5sjtwyQsutFicjDAmcAhwXAazlt/pBrecGVRVWHfo/bOG+d+lzhh/tKEXJ
Transação Rinkeby: `https://rinkeby.etherscan.io/tx/0xf88d01f1728724a39d8d4b119345f8b9c9f417b13b7bd1c0fe94fb6e888d59d5`
Transação Ropsten: `https://ropsten.etherscan.io/tx/0x2e8d5c60c35d5fd33e73cad0bb97ed80e98f4fd568404e2865e2b1602a63f62b`
Bloco Rinkeby: 3281843
Bloco Ropsten: 4365520
***Gas* Necessário:** 573762

baf6d32c-06cd-4b17-b5d5-e6f6549d38c2

Controlador de Dados: Electro-Portugal
Termo de Consentimento: 9dbaa92e-f53f-4d25-8d67-2d3efee4683b
Entidade Reguladora: CPD
Sujeito dos Dados: ab3f5c76-c00a-4171-b81b-a4ca3879479a
Resumo Criptográfico dos Dados: d73f6b38da92a4a5ca16d5fd5f6e96459f91af6b276f2d499e82d969675ce5b95b01584ca4bece501bc1f98f3cacb54e1e3cdc7c6bd137aef9ea569f0da1f480
Valor Assinado Para Pedido de Consentimento: baf6d32c-06cd-4b17-b5d5-e6f6549d38c2/9dbaa92e-f53f-4d25-8d67-2d3efee4683b/ab3f5c76-c00a-4171-b81b-a4ca3879479a/Electro-Portugal/CPD/d73f6b38da92a4a5ca16d5fd5f6e96459f91af6b276f2d499e82d969675ce5b95b01584ca4bece501bc1f98f3cacb54e1e3cdc7c6bd137aef9ea569f0da1f480
Assinatura Digital de Pedido de Consentimento: MD4CHQCR/OX2KfT07CuAIF1m/CJ1qCX1tXiPYaDWCnzhAh0ApXq4wxBF8xEqM69D6ilICK9QT1ViYw+dQfE7iA==
Valor Assinado Para Consentimento: baf6d32c-06cd-4b17-b5d5-e6f6549d38c2/9dbaa92e-f53f-4d25-8d67-2d3efee4683b/ab3f5c76-c00a-4171-b81b-a4ca3879479a/Electro-Portugal/CPD/d73f6b38da92a4a5ca16d5fd5f6e96459f91af6b276f2d499e82d969675ce5b95b01584ca4bece501bc1f98f3cacb54e1e3cdc7c6bd137aef9ea569f0da1f480/MD4CHQCR/OX2KfT07CuAIF1m/CJ1qCX1tXiPYaDWCnzhAh0ApXq4wxBF8xEqM69D6ilICK9QT1ViYw+dQfE7iA==
Assinatura Digital de Consentimento: MD0CHQCEMsVi3jioJ1JLmzlvnXw25tocHKejEbZE4xnxAhwU69o+opQudGhz/2a6TrgXLJiY1/ql2LpgN96w

Transação Rinkeby: https://rinkeby.etherscan.io/tx/0xd31c920b0adc3fa1bf57fd4e0c1c58889f90e833b537c8aa1023055722e8f4aa
Transação Ropsten: https://ropsten.etherscan.io/tx/0xf2ee76b371825f6c23a9b14d23c5c7d4f766d9b35412cb7534b795c6cc32b372
Bloco Rinkeby: 3281858
Bloco Ropsten: 4365541
***Gas* Necessário:** 574018

7f249424-c21c-4522-816e-74ceaa3a65ed

Controlador de Dados: Electro-Portugal
Termo de Consentimento: 9dbaa92e-f53f-4d25-8d67-2d3efee4683b
Entidade Reguladora: CPD
Sujeito dos Dados: b493a9fe-32fb-4e29-910e-2254dd44b474
Resumo Criptográfico dos Dados: d73f6b38da92a4a5ca16d5fd5f6e96459f91af6b276f2d499e82d969675ce5b95b01584ca4bece501bc1f98f3cacb54e1e3cdc7c6bd137aef9ea569f0da1f480
Valor Assinado Para Pedido de Consentimento: 7f249424-c21c-4522-816e-74ceaa3a65ed/9dbaa92e-f53f-4d25-8d67-2d3efee4683b/b493a9fe-32fb-4e29-910e-2254dd44b474/Electro-Portugal/CPD/d73f6b38da92a4a5ca16d5fd5f6e96459f91af6b276f2d499e82d969675ce5b95b01584ca4bece501bc1f98f3cacb54e1e3cdc7c6bd137aef9ea569f0da1f480
Assinatura Digital de Pedido de Consentimento: MD0CHGlYmp7jkMlXB5/mx8BJflOv0BBHkA4SbbrKAx0CHQCLFaTS5NvR0/slo1LhESTKoDx6svyBPuSc9CDA
Valor Assinado Para Consentimento: 7f249424-c21c-4522-816e-74ceaa3a65ed/9dbaa92e-f53f-4d25-8d67-2d3efee4683b/b493a9fe-32fb-4e29-910e-2254dd44b474/Electro-Portugal/CPD/d73f6b38da92a4a5ca16d5fd5f6e96459f91af6b276f2d499e82d969675ce5b95b01584ca4bece501bc1f98f3cacb54e1e3cdc7c6bd137aef9ea569f0da1f480/MD0CHGlYmp7jkMlXB5/mx8BJflOv0BBHkA4SbbrKAx0CHQCLFaTS5NvR0/slo1LhESTKoDx6svyBPuSc9CDA
Assinatura Digital de Consentimento: MD4CHQCBJAKG8piz5B1tLRdKRbBxbHkFf1VnJ3QEOqPuAh0AtIG12YWHROfkdITGDt+X3ZENacVyaS5f12l/dw==
Transação Rinkeby: https://rinkeby.etherscan.io/tx/0x7f262962b4a2711d2d6f9794043e2cc734fb91a549793743519cfaf394a76346
Transação Ropsten: https://ropsten.etherscan.io/tx/0x57f2aa45e1232d67dbe9064c61005a99f08fde1a001f9acfdf326165fad406f2
Bloco Rinkeby: 3281875
Bloco Ropsten: 4365556
***Gas* Necessário:** 574018

66f050cc-5555-49b0-b916-dba5f1b32b5e

Controlador de Dados: Electro-Portugal
Termo de Consentimento: 9dbaa92e-f53f-4d25-8d67-2d3efee4683b
Entidade Reguladora: CPD
Sujeito dos Dados: 5478b805-0fad-495c-8295-d67873e31ba0
Resumo Criptográfico dos Dados: d73f6b38da92a4a5ca16d5fd5f6e96459f91af6b276f2d499e82d969675ce5b95b01584ca4bece501bc1f98f3cacb54e1e3cdc7c6bd137aef9ea569f0da1f480

Valor Assinado Para Pedido de Consentimento: 66f050cc-5555-49b0-b916-dba5f1b32b5e/9dbaa92e-f53f-4d25-8d67-2d3efee4683b/5478b805-0fad-495c-8295-d67873e31ba0/Electro-Portugal/CPD/d73f6b38da92a4a5ca16d5fd5f6e96459f91af6b276f2d499e82d969675ce5b95b01584ca4bece501bc1f98f3cacb54e1e3cdc7c6bd137aef9ea569f0da1f480
Assinatura Digital de Pedido de Consentimento: MD0CHB3nR7AJqxbUG5l2uKH3gIPc1iPFfohO0wL147wCHQCswnFn69l6BEKKlfe2L5FjLcXLGey4tvNcoY0n
Valor Assinado Para Consentimento: 66f050cc-5555-49b0-b916-dba5f1b32b5e/9dbaa92e-f53f-4d25-8d67-2d3efee4683b/5478b805-0fad-495c-8295-d67873e31ba0/Electro-Portugal/CPD/d73f6b38da92a4a5ca16d5fd5f6e96459f91af6b276f2d499e82d969675ce5b95b01584ca4bece501bc1f98f3cacb54e1e3cdc7c6bd137aef9ea569f0da1f480/MD0CHB3nR7AJqxbUG5l2uKH3gIPc1iPFfohO0wL147wCHQCswnFn69l6BEKKlfe2L5FjLcXLGey4tvNcoY0n
Assinatura Digital de Consentimento: MDwCHBXzn9+t14LRXri5SdtQ2DOHk9mldIabv9eLOC0CHBnEdp3zWnb1okKIJz2iit/ufSnZrKbJcHkoTLI=
Transação Rinkeby: `https://rinkeby.etherscan.io/tx/0xc81eb225f574ae7974615e3dfff4770c0335a6c6cecc29c61b6afd51ca902407`
Transação Ropsten: `https://ropsten.etherscan.io/tx/0xb084b28b036c5d98797826a8ffde156cbbbde3939b045683137c56b5a5d24017`
Bloco Rinkeby: 3281887
Bloco Ropsten: 4365569
***Gas* Necessário:** 573762

072dc286-c959-4220-8d48-438ab1d2e3b7

Controlador de Dados: Portugás
Termo de Consentimento: 36dbaae2-930d-41ec-a4e6-d232b286137a
Entidade Reguladora: CPD
Sujeito dos Dados: 08fb82a7-ef19-41fa-bd0a-07d7656aedef
Resumo Criptográfico dos Dados: d73f6b38da92a4a5ca16d5fd5f6e96459f91af6b276f2d499e82d969675ce5b95b01584ca4bece501bc1f98f3cacb54e1e3cdc7c6bd137aef9ea569f0da1f480
Valor Assinado Para Pedido de Consentimento: 072dc286-c959-4220-8d48-438ab1d2e3b7/36dbaae2-930d-41ec-a4e6-d232b286137a/08fb82a7-ef19-41fa-bd0a-07d7656aedef/Portugás/CPD/df733df02a3d936ed144f37640fa15b213ce1d8db9865ad1add7be05b35962292238fc387820aca38f7cad4824468a34257d76e
Assinatura Digital de Pedido de Consentimento: MD0CHQCvQYiDyocp/t95DJQt8iDfrAFph85rvhvBbGR9Ahwt5HZmStdJDogWC+qszpaMpTt0HJYhwMejJhsU
Valor Assinado Para Consentimento: 072dc286-c959-4220-8d48-438ab1d2e3b7/36dbaae2-930d-41ec-a4e6-d232b286137a/08fb82a7-ef19-41fa-bd0a-07d7656aedef/Portugá/CPD/df733df02a3d936ed144f37640fa15b213ce1d8db9865ad1add7be05b35962292238fc387820aca38f7cad4824468a34257d76e0d0b611cdb1d2d7316f5177d8/MDwCHA5YUjgYVFYX5napMLrJ3nc/psM1nT4z+eE7w58CHHQS/uQ9NoywVCkQbsM4/3p/ICA8qIiFEf39ohE=
Assinatura Digital de Consentimento: MDwCHBXzn9+t14LRXri5SdtQ2DOHk9mldIabv9eLOC0CHBnEdp3zWnb1okKIJz2iit/ufSnZrKbJcHkoTLI=
Transação Rinkeby: `https://rinkeby.etherscan.io/tx/0x0895bc51dabca8f12f63e8afab04590411f70a3ccc0f479495fdd099b0a696bc`
Transação Ropsten: `https://ropsten.etherscan.io/tx/0xaa2af772a7ce2e47d54b711aadadbb2454c8442d29436b239ea3e16bc5e33042`
Bloco Rinkeby: 3281933
Bloco Ropsten: 4365621

***Gas* Necessário:** 573314

52b14c89-cee6-445c-9351-3adc676a27b9

Controlador de Dados: Portugás
Termo de Consentimento: 36dbaae2-930d-41ec-a4e6-d232b286137a
Entidade Reguladora: CPD
Sujeito dos Dados: d8a216e0-b694-4e93-b678-ee8eec7a4d0d
Resumo Criptográfico dos Dados: 06b897db59b4410330e85cd036d438969f28b7f0ae5f4bfc2974b0ad2e52bcf42b916ea11f8e1e82f329d501a546a70395007c338b44faf4a30e9672e0d6b980
Valor Assinado Para Pedido de Consentimento: 52b14c89-cee6-445c-9351-3adc676a27b9/36dbaae2-930d-41ec-a4e6-d232b286137a/d8a216e0-b694-4e93-b678-ee8eec7a4d0d/Portugás/CPD/06b897db59b4410330e85cd036d438969f28b7f0ae5f4bfc2974b0ad2e52bcf42b916ea11f8e1e82f329d501a546a70395007c338b44faf4a30e9672e0d6b980
Assinatura Digital de Pedido de Consentimento: MD0CHQCSPvra04+AGV+mHpmHjOrBEwMMxifersCVlUe2Ahwm9Yvic0HBCbELebXneMmXUhdUMmq807IV1pNK
Valor Assinado Para Consentimento: 52b14c89-cee6-445c-9351-3adc676a27b9/36dbaae2-930d-41ec-a4e6-d232b286137a/d8a216e0-b694-4e93-b678-ee8eec7a4d0d/Portugás/CPD/06b897db59b4410330e85cd036d438969f28b7f0ae5f4bfc2974b0ad2e52bcf42b916ea11f8e1e82f329d501a546a70395007c338b44faf4a30e9672e0d6b980/MD0CHQCSPvra04+AGV+mHpmHjOrBEwMMxifersCVlUe2Ahwm9Yvic0HBCbELebXneMmXUhdUMmq807IV1pNK
Assinatura Digital de Consentimento: MD0CHAfzQvLi5KBNHj7nPz1tmLXaWNyn0/51uf/OUYACHQC6qpeCQLf/PD1gPPZG2gUjZc7aTn1/Teq/DkiR
Transação Rinkeby: `https://rinkeby.etherscan.io/tx/0x286f1c388e9cb68f7c77bb6ed5639494cb2d7113bdabca4584db3d0ae4e3e4a0`
Transação Ropsten: `https://ropsten.etherscan.io/tx/0xaa2af772a7ce2e47d54b711aadadbb2454c8442d29436b239ea3e16bc5e33042`
Bloco Rinkeby: 3281942
Bloco Ropsten: 4365633
***Gas* Necessário:** 573314

4ce66245-d171-4fc9-87dc-cf5b87c670ba

Controlador de Dados: Portugás
Termo de Consentimento: 36dbaae2-930d-41ec-a4e6-d232b286137a
Entidade Reguladora: CPD
Sujeito dos Dados: ab3f5c76-c00a-4171-b81b-a4ca3879479a
Resumo Criptográfico dos Dados: bbd7d0e894ab9e855b82bd28e8bd4bd92c35b5124ef8913cd4946745ccf4135d2e9cb83a1525896d5426ce684363fa7fd1d8d29a210a3aaae2833336c28a72bf
Valor Assinado Para Pedido de Consentimento: 4ce66245-d171-4fc9-87dc-cf5b87c670ba/36dbaae2-930d-41ec-a4e6-d232b286137a/ab3f5c76-c00a-4171-b81b-a4ca3879479a/Portugás/CPD/bbd7d0e894ab9e855b82bd28e8bd4bd92c35b5124ef8913cd4946745ccf4135d2e9cb83a1525896d5426ce684363fa7fd1d8d29a210a3aaae2833336c28a72bf
Assinatura Digital de Pedido de Consentimento: MD4CHQDmSQvorYlgi7WU4Ls5mwsLhf39aHXk3PD5e2pUAh0A6wogqG1exKh/NWB57hVh6XLKltZoRQmsU9d69A==

Valor Assinado Para Consentimento: 4ce66245-d171-4fc9-87dc-cf5b87c670ba/36dbaae2-930d-41ec-a4e6-d232b286137a/ab3f5c76-c00a-4171-b81b-a4ca3879479a/Portugás/CPD/bbd7d0e894ab9e855b82bd28e8bd4bd92c35b5124ef8913cd4946745ccf4135d2e9cb83a1525896d5426ce684363fa7fd1d8d29a210a3aaae2833336c28a72bf/MD4CHQDmSQvorYlgi7WU4Ls5mwsLhf39aHXk3PD5e2pUAh0A6wogqG1exKh/NWB57hVh6XLKltZoRQmsU9d69A==
Assinatura Digital de Consentimento: MD0CHAHtzssYpAkLfbLVTiqfX0lODlQCvDn3tihSYzcCHQDLE0kaJ0iZ8lDYn58OSV//cGts895fhYUm8T8y
Transação Rinkeby: https://rinkeby.etherscan.io/tx/0x2b902d7c3b424fdadd4f3b4842a1bd950182fe7dde288551cec872fafea6cd28
Transação Ropsten: https://ropsten.etherscan.io/tx/0xe2b317ef2595a92974e15d3796ef353ebf8f902c39ba082776cda4a9846c9f6d
Bloco Rinkeby: 3281960
Bloco Ropsten: 4365657
***Gas* Necessário:** 573570

9f662f9b-6fc9-46b3-86f1-e3e048a48534

Controlador de Dados: Portugás
Termo de Consentimento: 36dbaae2-930d-41ec-a4e6-d232b286137a
Entidade Reguladora: CPD
Sujeito dos Dados: b493a9fe-32fb-4e29-910e-2254dd44b474
Resumo Criptográfico dos Dados: 82f39d41a351b144fe6e37f674b6edd31553402f082b00752b26b590ed4438deb5c1cd2142d7bae955cda02dbece1106a8608dfe9186b906f0e8127b3ccd6ebd
Valor Assinado Para Pedido de Consentimento: 9f662f9b-6fc9-46b3-86f1-e3e048a48534/36dbaae2-930d-41ec-a4e6-d232b286137a/b493a9fe-32fb-4e29-910e-2254dd44b474/Portugás/CPD/82f39d41a351b144fe6e37f674b6edd31553402f082b00752b26b590ed4438deb5c1cd2142d7bae955cda02dbece1106a8608dfe9186b906f0e8127b3ccd6ebd
Assinatura Digital de Pedido de Consentimento: MD0CHQDmtvUahb9ckDMSJTgi62WQgLuLExMWI6YlaaKKAhxlabPNEd0q/fTBnA+na4YE0bp2doR4BKbDvKRA
Valor Assinado Para Consentimento: 9f662f9b-6fc9-46b3-86f1-e3e048a48534/36dbaae2-930d-41ec-a4e6-d232b286137a/b493a9fe-32fb-4e29-910e-2254dd44b474/Portugás/CPD/82f39d41a351b144fe6e37f674b6edd31553402f082b00752b26b590ed4438deb5c1cd2142d7bae955cda02dbece1106a8608dfe9186b906f0e8127b3ccd6ebd/MD0CHQDmtvUahb9ckDMSJTgi62WQgLuLExMWI6YlaaKKAhxlabPNEd0q/fTBnA+na4YE0bp2doR4BKbDvKRA
Assinatura Digital de Consentimento: MDwCHFIlq3s1oi79u4IuFkBgj2GXBo2eIU63vIBZUA8CHB7TIA1aWTCTRJUapaXvC7/tQiPwWJAO2X3Bfus=
Transação Rinkeby: https://rinkeby.etherscan.io/tx/0xbbb5aa84300469d96ac2b37e8d624fe1a3a17f597f518abe2f1d80b433064801
Transação Ropsten: https://ropsten.etherscan.io/tx/0x13d870a5bd1ad550a8be3546ef45d34ba107719d316914d631789c432a75874c
Bloco Rinkeby: 3281972
Bloco Ropsten: 4365671
***Gas* Necessário:** 573314

a960bb5e-91d6-4c43-811d-1733c039f160

Controlador de Dados: Portugás

Termo de Consentimento: 36dbaae2-930d-41ec-a4e6-d232b286137a
Entidade Reguladora: CPD
Sujeito dos Dados: 5478b805-0fad-495c-8295-d67873e31ba0
Resumo Criptográfico dos Dados: 76f7e959db32af0a48f3ba97b2b1b870f0b7af6c62d88d8ef26c1a0355ada0dd6f70919ced3c6bbe0f123ac46e6e144697ddaed094a2e8a733922d1a1742bb28
Valor Assinado Para Pedido de Consentimento: a960bb5e-91d6-4c43-811d-1733c039f160/36dbaae2-930d-41ec-a4e6-d232b286137a/5478b805-0fad-495c-8295-d67873e31ba0/Portugás/CPD/76f7e959db32af0a48f3ba97b2b1b870f0b7af6c62d88d8ef26c1a0355ada0dd6f70919ced3c6bbe0f123ac46e6e144697ddaed094a2e8a733922d1a1742bb28
Assinatura Digital de Pedido de Consentimento: MD4CHQD/s3o5HZVMNqW3KBwkteF9fgT0a/CN3igwpaGkAh0Am3bBRLFZ7YDmnCsPIr2VJnNgBfVX7sLrTN2g7w==
Valor Assinado Para Consentimento: a960bb5e-91d6-4c43-811d-1733c039f160/36dbaae2-930d-41ec-a4e6-d232b286137a/5478b805-0fad-495c-8295-d67873e31ba0/Portugás/CPD/76f7e959db32af0a48f3ba97b2b1b870f0b7af6c62d88d8ef26c1a0355ada0dd6f70919ced3c6bbe0f123ac46e6e144697ddaed094a2e8a733922d1a1742bb28/MD4CHQD/s3o5HZVMNqW3KBwkteF9fgT0a/CN3igwpaGkAh0Am3bBRLFZ7YDmnCsPIr2VJnNgBfVX7sLrTN2g7w==
Assinatura Digital de Consentimento: MDwCHFt4cdXEzB3BF0fVFkiWfnhICiBRg92JPXiA3KgCHDxKaiBgzdS0Sq+wa/8S7kZh8d0Jh1D1wly06go=
Transação Rinkeby: https://rinkeby.etherscan.io/tx/0xaab1accf217b828244cba80bc1e67b3fab6c2a8c57f1a36a07f3e162f88358b3
Transação Ropsten: https://ropsten.etherscan.io/tx/0x3923d875ad3dd56d3d5be5bfa5ffdf4da269744207f552cc3bc04b75c806faca
Bloco Rinkeby: 3281988
Bloco Ropsten: 4365692
***Gas* Necessário:** 573570

Remoções de Consentimento

211413fe-0d4c-4f78-a8ea-3cbfc5a90ed2

***Timestamp* do Bloco do Consentimento Rinkeby:** 1541355224
***Timestamp* do Bloco do Consentimento Ropsten:** 1541355239
Valor Assinado Para Remoção de Consentimento Rinkeby: 211413fe-0d4c-4f78-a8ea-3cbfc5a90ed2/1541355224
Valor Assinado Para Remoção de Consentimento Ropsten: 211413fe-0d4c-4f78-a8ea-3cbfc5a90ed2/1541359897
Assinatura Digital de Remoção de Consentimento Rinkeby: MD4CHQCSy1jH8WeomudNOktvK05iwUdpZnbBSkSh6UBqAh0A97Y+3rLr8Tgh49zMXMWfKdXZULgzcn7A7VT3zA==
Assinatura Digital de Remoção de Consentimento Ropsten: MDwCHFDvykaPEjHkN2FFtHZLWlCzcgfyI1+K+B9dGQUCHAI3k5qRe9AYbvIeVJU/LrNJAJ7IyAubwZ5Kx+c=
Transação Rinkeby: https://rinkeby.etherscan.io/tx/0x4147a965947f917dffd559ea41d47726b5d344ce2d466e6932fbec2041a2378c
Transação Ropsten: https://ropsten.etherscan.io/tx/0xfbdb50e6c067b4b8b16921eb0216cf437573b89e0b3ff3acc68760cec978f77f

Bloco Rinkeby: 3282084
Bloco Ropsten: 4365818
***Gas* Necessário:** 202475

a960bb5e-91d6-4c43-811d-1733c039f160

***Timestamp* do Bloco do Consentimento Rinkeby:** 1541358329
***Timestamp* do Bloco do Consentimento Ropsten:** 1541358344
Valor Assinado Para Remoção de Consentimento Rinkeby: a960bb5e-91d6-4c43-811d-1733c039f160/1541358329
Valor Assinado Para Remoção de Consentimento Ropsten: a960bb5e-91d6-4c43-811d-1733c039f160/1541358344
Assinatura Digital de Remoção de Consentimento Rinkeby: MD0CHQCT7RHbe6AAJKiuh1fsGjJ9j7UTqPScSZaaOG5KAhwI9CQghYFZv+Vq8Mfvce44O7gNygg1cpzOVAl9
Assinatura Digital de Remoção de Consentimento Ropsten: MD0CHQCH2Rf/hfaU+xKoSWEWcGKNuYSA+bZBGv/sSOYzAhxVBcQjEzWe1R0HgDEEZ5/9LX0OA3U1uF2CgxlL
Transação Rinkeby: `https://rinkeby.etherscan.io/tx/0x8ae2d89296d3a64e0ab54666c5543b2c68550a007089b9d32d47623c1c862dc3`
Transação Ropsten: `https://ropsten.etherscan.io/tx/0xeaa8075c61f6c29323162c026350b6a3d98efb00d015e47b28e029711068e14e`
Bloco Rinkeby: 3366845
Bloco Ropsten: 4461164
***Gas* Necessário:** 202219

Entradas de *Feedback*

211413fe-0d4c-4f78-a8ea-3cbfc5a90ed2

Índice da Entrada: 1
Nota da Avaliação: Positiva (2)
Mensagem da Avaliação: O controlador de dados tem respeitado o termo de consentimento escrupulosamente.
Valor Assinado: 211413fe-0d4c-4f78-a8ea-3cbfc5a90ed2/2/Ocontroladordedadostemrespeitadootermodeconsentimentoescrupulosamente.
Assinatura Digital de Entrada de *Feedback*: MDwCHDnl14VBT+Mi+Jcy7qJpX16S2B7VKgnXzA0iXEsCHHm9ewe2iSKofUxA4InxrfgFm59twEuHbITgDLM=
Transação Rinkeby: `https://rinkeby.etherscan.io/tx/0x32f8402977ea26984c51e9859ca823e293bd777338c1af0baf9358c1ce3c89b0`
Transação Ropsten: `https://ropsten.etherscan.io/tx/0xf8c4bf202e5de3d931dc2b99070d417b82f03dd113a1d31be0695a4f2a43685e`
Bloco Rinkeby: 3282062
Bloco Ropsten: 4365786
***Gas* Necessário:** 332635

211413fe-0d4c-4f78-a8ea-3cbfc5a90ed2

Índice da Entrada: 2
Nota da Avaliação: Negativa (1)
Mensagem da Avaliação: O controlador de dados deixou de respeitar o termo de consentimento.
Valor Assinado: 211413fe-0d4c-4f78-a8ea-3cbfc5a90ed2/0/Ocontroladordedadosdeixou derespeitarotermodeconsentimento.
Assinatura Digital de Entrada de *Feedback*: MD4CHQCGbdvAvgjC+BIec97NwmrUAQTsy1EIkEZ8uVH5Ah0AhQ/JRxFy/sVn54fUNPTxuLrCCBjVgnUx7lqPzA==
Transação Rinkeby: https://rinkeby.etherscan.io/tx/0xe615eb14c467d59709b480935f88e8ff7e3b4eb9173e8012839a872da32ed74f
Transação Ropsten: https://ropsten.etherscan.io/tx/0x5a8170fc889124699b6d5c7359b7f574a32cbd338f3f8c2fe4f54ebae740d557
Bloco Rinkeby: 3282111
Bloco Ropsten: 4365840
***Gas* Necessário:** 239458

072dc286-c959-4220-8d48-438ab1d2e3b7

Índice da Entrada: 1
Nota da Avaliação: Neutra (1)
Mensagem da Avaliação: O controlador tem vindo a cumprir o termo de consentimento.
Valor Assinado: 072dc286-c959-4220-8d48-438ab1d2e3b7/1/Ocontroladortemvindoacumprirotermodeconsentimento.
Assinatura Digital de Entrada de *Feedback*: MD0CHGz6hEeSUkPDM7V9umj9uQaD7yEZdZ2cBGuq+uwCHQCMli9tJ08kw84pYsnHkHfQkN0Cnt7olkVTwPBz
Transação Rinkeby: https://rinkeby.etherscan.io/tx/0x6987276cd738c51fdfdf01a09818938f1679bf3757fbc818031c9bc4327de94e
Transação Ropsten: https://ropsten.etherscan.io/tx/0xc336669b6f8fc5c249a8a6903d281490e23c48a6b1b0a1a0af29405f90c9b42b
Bloco Rinkeby: 3282137
Bloco Ropsten: 4365867
***Gas* Necessário:** 311061

211413fe-0d4c-4f78-a8ea-3cbfc5a90ed2

Índice da Entrada: 3
Nota da Avaliação: Neutra (1)
Mensagem da Avaliação: O controlador provou que estava a cumprir o termo de consentimento.
Valor Assinado: 211413fe-0d4c-4f78-a8ea-3cbfc5a90ed2/1/Ocontroladorprovouqueestavaacumprirotermodeconsentimento.
Assinatura Digital de Entrada de *Feedback*: MDwCHEOa6vypa+cUtv+a+jWcPCzOiN6XZYyrxNWqWWkCHA58jJHE3zyZI0EzdIovr+S30MDYErWv6tnQ28w=
Transação Rinkeby: https://rinkeby.etherscan.io/tx/0xb06df66b3f46697e1903fae61b5aa9f78ac5558093d11c579110fc1d6b6111cb
Transação Ropsten: https://ropsten.etherscan.io/tx/0x627ef12ad6688528a1a2922dcc0ec2eeb89cae97c708325391661f5527cc8980
Bloco Rinkeby: 3366736

Bloco Ropsten: 4461038
***Gas* Necessário:** 254239

211413fe-0d4c-4f78-a8ea-3cbfc5a90ed2

Índice da Entrada: 4
Nota da Avaliação: Positiva (2)
Mensagem da Avaliação: Estava equivocado relativamente ao comportamento do controlador de dados.
Valor Assinado: 211413fe-0d4c-4f78-a8ea-3cbfc5a90ed2/2/Estavaequivocadorelativamenteaocomportamentodocontroladordedados.
Assinatura Digital de Entrada de *Feedback*: MD4CHQCcmvmh472FdOzQ7skkhUiw65fIe9+hfGHaq3YUAh0AoSIsIxAAWhLcV2nZBw1c1UuV85wtbsdWpzzmzA==
Transação Rinkeby: `https://rinkeby.etherscan.io/tx/0xd8359fce4a1b974f9bf2b5c9112d166f144ddc75b2a0bf110613bd00de667079`
Transação Ropsten: `https://ropsten.etherscan.io/tx/0x28cdc577bc61d4b94818c036d7bfbf40bec2d67d86297c80d465d3005eff51e1`
Bloco Rinkeby: 3366748
Bloco Ropsten: 4461048
***Gas* Necessário:** 254905

8.6 Código Fonte da Inicialização da Aplicação Móvel

```
1  import android.content.Context;
2  import android.content.SharedPreferences;
3  import android.support.v7.app.AppCompatActivity;
4  import android.os.Bundle;
5  import android.util.Base64;
6  import android.widget.Toast;
7  import org.json.JSONException;
8  import org.json.JSONObject;
9  import java.io.IOException;
10 import java.io.UnsupportedEncodingException;
11 import java.security.InvalidKeyException;
12 import java.security.KeyFactory;
13 import java.security.KeyPair;
14 import java.security.KeyPairGenerator;
15 import java.security.NoSuchAlgorithmException;
16 import java.security.Provider;
17 import java.security.Signature;
18 import java.security.SignatureException;
19 import java.security.spec.InvalidKeySpecException;
20 import java.security.spec.PKCS8EncodedKeySpec;
21 import java.util.UUID;
22 import okhttp3.MediaType;
23 import okhttp3.OkHttpClient;
24 import okhttp3.Request;
25 import okhttp3.RequestBody;
26 import okhttp3.Response;
27 import static org.spongycastle.util.encoders.Base64.*;
28
29 public class MainActivity extends AppCompatActivity {
30
31     static final protected String CONNSENT_SHARED_PREFS = "connsent";
32     static final protected String CONNSENT_PRIV_PREF_KEY = "privKey";
33     static final protected String CONNSENT_PUB_PREF_KEY = "pubKey";
34     static final protected String CONNSENT_SUBJECT_PREF_KEY = "subjectUid";
35
36     @Override
37     protected void onCreate(Bundle savedInstanceState) {
38         super.onCreate(savedInstanceState);
```

```
39          setContentView(R.layout.activity_main);
40
41          SharedPreferences sp = this.getApplicationContext().getSharedPreferences(CONNSENT_SHARED_PREFS, Context.MODE_PRIVATE);
42
43          // Checking if application is initialized
44          if (!sp.contains(CONNSENT_PRIV_PREF_KEY)) {
45              // Application is not initialized
46              try {
47                  // Generating key pair
48                  KeyPair kp = CryptoHelper.generatePrivateKey();
49                  String privKeyBase64 = Base64.encodeToString(kp.getPrivate().getEncoded(), Base64.DEFAULT);
50                  String pubKeyBase64 = Base64.encodeToString(kp.getPublic().getEncoded(), Base64.DEFAULT);
51
52                  // Generating data subject identifier
53                  String subjectUid = UUID.randomUUID().toString();
54
55                  // Signing registration data
56                  String signatureMessage = subjectUid + "/" + privKeyBase64;
57                  String signature = CryptoHelper.sign(privKeyBase64, signatureMessage.getBytes("UTF-8"));
58
59
60                  // Data subject registration through REST API
61                  // Building request object
62                  JSONObject registrationData = new JSONObject() .put("uid", subjectUid)
63                                                                 .put("signature", signature)
64                                                                 .put("publicKey", pubKeyBase64);
65
66                  OkHttpClient client = new OkHttpClient();
67                  // Building request
68                  MediaType reqMediaType = MediaType.get("application/json; charset=utf-8");
69                  RequestBody body = RequestBody.create(reqMediaType,registrationData.toString());
70                  Request request = new Request.Builder().url("https://<CONNSENT_API_URL>/subject").post(body).build();
71
72                  // Sending request
73                  Response response = client.newCall(request).execute();
74
75                  // Validating if request was successful
76                  if (response.isSuccessful()) {
77
78                      // Request was successful
```

```
79                  SharedPreferences.Editor e = sp.edit();
80                  e.putString(CONNSENT_PRIV_PREF_KEY, privKeyBase64);
81                  e.putString(CONNSENT_PUB_PREF_KEY, pubKeyBase64);
82                  e.putString(CONNSENT_SUBJECT_PREF_KEY, subjectUid);
83
84
85                  // Trying to store data in shared preferences
86                  if (e.commit()) {
87                      // Initialization successful
88                      Toast.makeText(this, "Successfully initialized as subject '" + subjectUid + "'!", Toast.LENGTH_LONG)
89                              .show();
90                      return;
91                  } else {
92                      // Failed to store in shared preferences
93                      Toast.makeText(this, "Initialization failed! Couldn't store data in shared preferences!", Toast.LENGTH_LONG)
94                              .show();
95                  }
96              } else {
97                  // Request was not successful
98                  Toast.makeText(this, "Initialization failed! Unexpected response from REST API!", Toast.LENGTH_LONG)
99                          .show();
100             }
101
102
103         } catch (NoSuchAlgorithmException | InvalidKeyException | SignatureException | UnsupportedEncodingException | InvalidKeySpecException e)
                ↪ {
104             // Exception while performing crypto related operation
105             Toast.makeText(this, "Initialization failed! Crypto exception '" + e.getMessage() + "'!" + e.getMessage(), Toast.LENGTH_LONG)
106                     .show();
107         } catch (JSONException e) {
108             // Exception while building HTTP request object
109             Toast.makeText(this, "Initialization failed! Failed to build request object!" + e.getMessage(), Toast.LENGTH_LONG)
110                     .show();
111         } catch (IOException e) {
112             // Exception while communication with REST API
113             Toast.makeText(this, "Initialization failed! Communication with REST API failed!" + e.getMessage(), Toast.LENGTH_LONG)
114                     .show();
115         }
116
117         // Closes application in case of failure
```

```
118             finishAndRemoveTask();
119         } else {
120             // Application already initialized
121             Toast.makeText(this, "Welcome data subject '" + sp.getString(CONNSENT_SUBJECT_PREF_KEY, "") + "'!", Toast.LENGTH_LONG)
122                     .show();
123         }
124     }
125 }
126
127 final class CryptoHelper {
128
129     private CryptoHelper() {}
130
131     private static final Provider provider = new org.spongycastle.jce.provider.BouncyCastleProvider();
132
133     static final protected KeyPair generatePrivateKey() throws NoSuchAlgorithmException {
134         // Getting key generator instance
135         KeyPairGenerator keyGen = KeyPairGenerator.getInstance("ECDSA",provider);
136         // Initializing key generator
137         keyGen.initialize(224);
138         // Generating key pair
139         return keyGen.generateKeyPair();
140     }
141
142     static final public String sign(String privateKeyBase64, byte[] data)
143             throws NoSuchAlgorithmException, SignatureException, InvalidKeySpecException, InvalidKeyException {
144         // Getting signature instance
145         Signature removalSig = Signature.getInstance("SHA512withECDSA",provider);
146         // Rebuilding Private Key
147         PKCS8EncodedKeySpec privKey = new PKCS8EncodedKeySpec(decode(privateKeyBase64));
148         // Initializing instance with private key
149         removalSig.initSign(KeyFactory.getInstance("ECDSA",provider).generatePrivate(privKey));
150         // Feeding data to be signed
151         removalSig.update(data);
152         // Performing signature
153         return toBase64String(removalSig.sign());
154     }
155
156
```

```
157 }
```

Listagem 8.5: Código fonte da inicialização da aplicação móvel *Connsent*.

8.7 Rotas da REST API para o Subsistema Reputacional

```
1  import org.bouncycastle.util.encoders.Base64;
2  import java.security.*;
3  import java.security.spec.InvalidKeySpecException;
4  import java.security.spec.X509EncodedKeySpec;
5  import org.bouncycastle.util.encoders.Base64;
6  import org.bouncycastle.util.encoders.DecoderException;
7  import org.json.JSONArray;
8  import org.json.JSONException;
9  import org.json.JSONObject;
10 import org.web3j.crypto.CipherException;
11 import org.web3j.protocol.core.DefaultBlockParameterName;
12 import org.web3j.protocol.core.methods.request.EthFilter;
13 import org.web3j.protocol.core.methods.response.TransactionReceipt;
14 import org.web3j.tuples.generated.Tuple2;
15 import org.web3j.tuples.generated.Tuple4;
16 import org.web3j.tuples.generated.Tuple5;
17 import org.web3j.tuples.generated.Tuple6;
18 import pt.ipp.estg.mei.connsent.enums.FeedbackType;
19 import pt.ipp.estg.mei.connsent.exceptions.RecordNotFoundException;
20 import pt.ipp.estg.mei.connsent.smartcontracts.Connsent;
21 import pt.ipp.estg.mei.connsent.smartcontracts.ConnsentSingleton;
22 import pt.ipp.estg.mei.connsent.utils.CryptoHelper;
23 import pt.ipp.estg.mei.connsent.utils.DataHelper;
24 import pt.ipp.estg.mei.connsent.utils.Env;
25 import javax.ws.rs.*;
26 import javax.ws.rs.core.MediaType;
27 import javax.ws.rs.core.Response;
28 import java.math.BigInteger;
29 import java.util.ArrayList;
30 import java.util.HashSet;
31 import java.util.Iterator;
32
33 @Path("/")
```

```
34  public class RouteRoot {
35
36      @GET
37      @Path("controller/feedback")
38      @Produces(MediaType.APPLICATION_JSON)
39      public Response getControllerFeedback(@QueryParam("name")String controllerName) throws Exception {
40
41          // Making sure controller name is not null
42          if(controllerName==null){
43              return Response.status(400).entity((new JSONObject().put("message","No 'name' sent!"))).build();
44          }
45
46          // Getting contract account wrapper instance
47          Connsent connsent = ConnsentSingleton.getInstance();
48
49          // Making sure controller exists
50          Tuple4<String,String,String,BigInteger> controllerBC;
51          try {
52              controllerBC = connsent.getDataController(controllerName).send();
53          } catch (ArrayIndexOutOfBoundsException | StringIndexOutOfBoundsException e){
54              return Response.status(404).entity(new JSONObject().put("message","Controller '"+controllerName+"' not found!")).build();
55          }
56
57          // Iterating through NewFeedbackEntryEvent events in blockchain
58          Iterator<Connsent.NewFeedbackEntryEventResponse> it = connsent .newFeedbackEntryEventObservable( DefaultBlockParameterName.EARLIEST,
59                                                                                                            DefaultBlockParameterName.LATEST)
60                                                                  .onTerminateDetach()
61                                                                  .toBlocking()
62                                                                  .toIterable()
63                                                                  .iterator();
64
65          ArrayList<String> allControllerEntries = new ArrayList<>();
66          HashSet<String> uniqueConsentEntries = new HashSet<>();
67
68          // Iterating through all feedback entries
69          while(it.hasNext()){
70
71              // Getting next entry's reference
72              Connsent.NewFeedbackEntryEventResponse current = it.next();
73
```

```
74              // Checking if current entry is about the given controller
75              if(current.dataControllerName.equals(controllerName)){
76                  // Entry is about the given controller, storing consent identifier
77                  allControllerEntries.add(current.consentUid);
78                  uniqueConsentEntries.add(current.consentUid);
79              }
80          }
81
82
83          int[] latestEntryIndex = new int[uniqueConsentEntries.size()];
84          int uniqueConsentEntriesIndex = 0;
85
86          // Iterating through all unique consent entries
87          for (String i:uniqueConsentEntries) {
88
89              // Since the current consent wasn't matched yet, its index is -1
90              int currentConsentEntryIndex = -1;
91
92              // Iterating all non unique entries
93              for(String j:allControllerEntries){
94                  // Checking if current entry matches unique entry
95                  if(j.equals(i)){
96                      // It matches, incrementing index
97                      currentConsentEntryIndex++;
98                  }
99              }
100
101             // Saving latest consent entry index
102             latestEntryIndex[uniqueConsentEntriesIndex]=currentConsentEntryIndex;
103         }
104
105
106
107
108         // Getting latest consent entry index for all unique entries
109         int negative = 0, neutral = 0, positive = 0;
110         uniqueConsentEntriesIndex = 0;
111         JSONArray entries = new JSONArray();
112         for(String i : uniqueConsentEntries){
113
```

```
114             // Getting current feedback entry
115             BigInteger currentConsentIndex = BigInteger.valueOf(latestEntryIndex[uniqueConsentEntriesIndex])
116             Tuple4<BigInteger,String,String,BigInteger> current = connsent.getFeedbackEntry(i,currentConsentIndex).send();
117
118             JSONObject currentEntry = new JSONObject();
119
120             // Adding feedback type
121             BigInteger entryType = current.getValue1();
122             if(entryType.equals(BigInteger.ZERO)){
123                 negative++;
124                 currentEntry.put("type","NEGATIVE");
125             } else if (entryType.equals(BigInteger.ONE)){
126                 neutral++;
127                 currentEntry.put("type","NEUTRAL");
128             } else {
129                 positive++;
130                 currentEntry.put("type","POSITIVE");
131             }
132
133             // Adding message
134             currentEntry.put("message",current.getValue2());
135             // Adding timestamp
136             currentEntry.put("timestamp",current.getValue4());
137
138             // Adding current entry to array
139             entries.put(currentEntry);
140
141             uniqueConsentEntriesIndex++;
142         }
143
144         // Calculating percentages
145         int totalEntries = negative+neutral+positive;
146         String positivePercentage = Float.toString((100/totalEntries)*positive)+"%";
147         String neutralPercentage = Float.toString((100/totalEntries)*neutral)+"%";
148         String negativePercentage = Float.toString((100/totalEntries)*negative)+"%";
149
150         // Creating response object
151         JSONObject response = new JSONObject() .put("totalEntries",totalEntries)
152                                                .put("positivePercentage",positivePercentage)
153                                                .put("neutralPercentage",neutralPercentage)
```

```
154                                                 .put("negativePercentage",negativePercentage)
155                                                 .put("score",positive−negative)
156                                                 .put("negative",negative)
157                                                 .put("neutral",neutral)
158                                                 .put("positive",positive)
159                                                 .put("entries",entries);
160
161         return Response.ok(response).build();
162     }
163
164     @POST
165     @Path("feedback")
166     @Consumes(MediaType.APPLICATION_JSON)
167     @Produces(MediaType.APPLICATION_JSON)
168     public Response addFeedback(JSONObject request) throws Exception {
169
170
171         // Making sure request has expected fields
172         if( !request.has("consentUid") || !request.has("feedbackType") || !request.has("message") || !request.has("feedbackSig")){
173             return Response.status(400).entity(new JSONObject().put("message","Missing mandatory attributes!")).build();
174         }
175
176         // Reading expected fields
177         String consentUid = request.getString("consentUid");
178         BigInteger feedbackType = BigInteger.valueOf(request.getLong("feedbackType"));
179         String message = request.getString("message");
180         String feedbackSig = request.getString("feedbackSig");
181
182         // Validating feedback type value
183         try {
184             FeedbackType.fromValue(feedbackType);
185         } catch (NotFoundException e){
186             return Response.status(400).entity(new JSONObject().put("message","Invalid feedback type!")).build();
187         }
188
189         // Making sure message length doesn't exceed the specified limit
190         if(message.length()>500){
191             return Response.status(400).entity(new JSONObject().put("message","Message exceeds 500 characters!")).build();
192         }
193
```

```
194         // Making sure the consent exists
195         Tuple6<String,String,String,String,String,BigInteger> consentBC;
196         try {
197             consentBC = ConnsentSingleton.getInstance().getConsentValidationData(consentUid).send();
198         } catch (ArrayIndexOutOfBoundsException | StringIndexOutOfBoundsException e){
199             return Response.status(404).entity(new JSONObject().put("message","Consent '"+consentUid+"' not found in smart-contract's storage!
                ↪ ")).build();
200         }
201
202         // Getting consent's subject identifier
203         Tuple4<String,String,String,BigInteger> subjectBC = ConnsentSingleton.getInstance().getDataSubject(consentBC.getValue1()).send();
204
205         // Building signature message
206         String signatureMessage = consentUid+"/"+feedbackType.toString()+"/"+message;
207
208         // Verifying feedback signature
209         if(!CryptoHelper.verifySignature(subjectBC.getValue3(),signatureMessage.getBytes("UTF-8"),feedbackSig)){
210             return Response.status(400).entity(new JSONObject().put("message","Feedback signature verification failed!")).build();
211         }
212
213         // Performing transaction
214         TransactionReceipt receipt = ConnsentSingleton.getInstance().addFeedbackEntry(consentUid,feedbackType,message,feedbackSig).send();
215
216         // Sending response data
217         return Response.ok(new JSONObject() .put("message","Feedback successfully persisted!")
218                                             .put("transactionHash",receipt.getTransactionHash())
219                                             .put("link",Env.ETHEREUM_TRANSACTION_LINK.getValue()+receipt.getTransactionHash())
220                                             .put("contractAddress",receipt.getContractAddress())).build();
221     }
222
223 }
224
225 public class CryptoHelper {
226
227     private CryptoHelper() {
228     }
229
230     static final public boolean verifySignature(String publicKeyBase64, byte[] data, String signatureBase64)
231         throws InvalidKeyException, NoSuchProviderException, NoSuchAlgorithmException, SignatureException, InvalidKeySpecException {
232
```

```
233         // Getting signature instance
234         Signature verifySig = Signature.getInstance("SHA512withECDSA","BC");
235         // Rebuilding public key
236         PublicKey pubKey = KeyFactory.getInstance("ECDSA","BC").generatePublic(new X509EncodedKeySpec(Base64.decode(publicKeyBase64)));
237         // Initializing signature instance with public key
238         verifySig.initVerify(pubKey);
239         // Feeding data to be verified
240         verifySig.update(data);
241         // Performing verification
242         return verifySig.verify(Base64.decode(signatureBase64));
243     }
244 }
```

Listagem 8.6: Código fonte da rota da REST API responsável por computar a reputação de um controlador de dados.

8.8 *Smart Contracts* do Comparativo Mapeamentos *Versus* Vetores

```
1  pragma solidity ^0.4.23;
2  contract ConnsentArray {
3
4      // ============================================================
5      // CONTRACT VARIABLES
6      // ============================================================
7      address private owner;
8      DataSubject[] private _subjects;
9
10     // ============================================================
11     // CONSTRUCTOR
12     // ============================================================
13     constructor () public {
14         owner = msg.sender;
15     }
16
17     // ============================================================
18     // DATA STRUCTURES
19     // ============================================================
20     struct DataSubject {
21         string uid;
22         string signature;
```

```
23          string publicKey;
24          uint timestamp;
25      }
26
27
28      // ADD DATA SUBJECT
29      // Adds a data subject to the smart contract's storage
30      function addDataSubject(string uid, string publicKey, string signature) public
31          returns (string result){
32
33          // Making sure the function was called by the smart-contract owner
34          require(msg.sender==owner,"Only the smart-contract owner can call this transaction!");
35
36          // Not empty validations
37          require(bytes(uid).length>0,"'uid' must not be empty!");
38          require(bytes(signature).length>0,"'signature' must not be empty!");
39          require(bytes(publicKey).length>0,"'publicKey' must not be empty!");
40
41          for(uint i=0;i<_subjects.length;i++){
42              // Making sure the new subject uid is not already in use
43              require(!compareStrings(bytes(_subjects[i].uid),bytes(uid)),"'uid' is already in use!");
44          }
45
46          // Adding subject to array
47          _subjects.push(DataSubject(uid,publicKey,signature,block.timestamp));
48
49          return "Data subject created!";
50      }
51
52      // RETRIEVE DATA SUBJECT
53      // Retrieves a data subject from the smart contract's storage
54      function getDataSubject(string dataSubjectUid) public view
55          returns (string uid, string publicKey, string signature, uint timestamp) {
56
57          require(bytes(dataSubjectUid).length>0,"'uid' must not be empty!");
58
59          // Looking for subject
60          for(uint i=0;i<_subjects.length;i++){
61              if(compareStrings(bytes(_subjects[i].uid),bytes(dataSubjectUid))){
62                  // Found subject
```

```
63                  return (_subjects[i].uid,_subjects[i].publicKey,_subjects[i].signature,_subjects[i].timestamp);
64              }
65          }
66
67          revert("Data subject not found!");
68      }
69
70      function compareStrings (bytes a, bytes b) private pure returns (bool){
71          return keccak256(a) == keccak256(b);
72      }
73  }
```

Listagem 8.7: Código fonte do *smart contract* do comparativo que persiste os sujeitos dos dados em vetores.

```
1   pragma solidity ^0.4.23;
2
3   contract ConnsentMapping {
4
5       // ============================================================
6       // CONTRACT VARIABLES
7       // ============================================================
8       address private owner;
9       mapping (string => DataSubject) private _dataSubjects;
10
11      // ============================================================
12      // CONSTRUCTOR
13      // ============================================================
14      constructor () public {
15          owner = msg.sender;
16      }
17
18
19      // ============================================================
20      // DATA STRUCTURES
21      // ============================================================
22      struct DataSubject {
23          string uid;
24          string signature;
25          string publicKey;
26          uint timestamp;
```

```
27      }
28
29
30      // ADD DATA SUBJECT
31      // Adds a data subject to the smart contract's storage
32      function addDataSubject(string uid, string publicKey, string signature) public
33          returns (string result){
34
35          // Making sure the function was called by the smart-contract owner
36          require(msg.sender==owner,"Only the smart-contract owner can call this transaction!");
37
38          // Not empty validations
39          require(bytes(uid).length>0,"'uid' must not be empty!");
40          require(bytes(signature).length>0,"'signature' must not be empty!");
41          require(bytes(publicKey).length>0,"'publicKey' must not be empty!");
42
43
44          DataSubject storage dataSubject = _dataSubjects[uid];
45
46          // Making sure the new data subject name is not already in use
47          require(bytes(dataSubject.uid).length==0, "'uid' already in use!");
48
49          _dataSubjects[uid] = DataSubject(uid,signature,publicKey,block.timestamp);
50
51          return "Data subject created!";
52      }
53
54      // RETRIEVE DATA SUBJECT
55      // Retrieves a data subject from the smart contract's storage
56      function getDataSubject(string dataSubjectUid) public view
57          returns (string uid, string publicKey, string signature, uint timestamp) {
58
59          require(bytes(dataSubjectUid).length>0,"'uid' must not be empty!");
60
61          // Getting reference
62          DataSubject storage subject = _dataSubjects[dataSubjectUid];
63
64          // Making sure the new data subject exists
65          require(bytes(subject.uid).length!=0, "Data subject not found!");
66
```

```
67          return (subject.uid,subject.signature,subject.publicKey,subject.timestamp);
68      }
69  }
```

Listagem 8.8: Código fonte do *smart contract* do comparativo que persiste os sujeitos dos dados em mapeamentos.

8.9 Resultados do Comparativo (Mapeamento)

#	Endereço da Transação (Rede *Rinkeby*)	Bloco	Hora da Transação	Gas
0	0x335c6b3d147a99cd327ba07c9f160c1b5d6256cc12436c1cf4523f0e104a1a56	3293556	11/6/2018 7:17:55 PM	800513
1	0xfacc0490393ae4c36e1f84d8a469083af9db132155033982ac1cd277916e972d	3293560	11/6/2018 7:18:55 PM	303215
2	0x883385c0448d8039e40000b937a9d0df6f5b8bfcf856c5ca5d1bf8a60f745e56	3293563	11/6/2018 7:19:40 PM	303215
3	0x97805cd03fb004391e74c7c1d117707544093490ed5042d9eb732d58a3b354fa	3293566	11/6/2018 7:20:25 PM	303215
4	0x2d5d7ebb3d3a2e20032e79fdf629bfa69b8a508d4c8c51d8cf1cd26fa37cdb52	3293568	11/6/2018 7:20:55 PM	303215
5	0x04b11733cd75501eae7116cb97bf2a950a338a20aeed37a481e3c5eb117fa9a1	3293572	11/6/2018 7:21:55 PM	303215
6	0x18b2f559b9bd59263468dca0eeaa5a3c420ad909a08e8fd27898105efa6c7953	3293575	11/6/2018 7:22:40 PM	303215
7	0x4e7eba0a168a81fa86b0ac8a6746b4f173133442086c85a1acf1a2c640bd6111	3293578	11/6/2018 7:23:25 PM	303215
8	0x8163f158a44224812e94bc8184c0088edf15368a8a48e040ac80d17afa9edbd1	3293582	11/6/2018 7:24:25 PM	303215
9	0xada448752688151d287d7728a54b49932b33802e2caf30e87837f2cd7588c638	3293585	11/6/2018 7:25:10 PM	303471
10	0x7ce0180f9faf0ac27cc9e9718c75a63c18a254158f0e0f987e33a31d85e63081	3293588	11/6/2018 7:25:55 PM	303215
11	0x3fda91065b0a4558a9f316380484ea05cc2f86c502f83735a01fa1c801f4cca0	3293590	11/6/2018 7:26:25 PM	303215
12	0xb55b467397610ac840e2d93c6b23c6f1d7d368c4c31b418065bc0c651c36eaf8	3293593	11/6/2018 7:27:10 PM	303215
13	0xa386cc8058812a6353bf2632fc9a6e739f86cce82e7ed6709090b6330aa70c0c	3293596	11/6/2018 7:27:55 PM	303215
14	0x031aea9cfbd1e5553d1c841eb08c17805c0b7a1162c73fa1ef0a93f8a2282a30	3293600	11/6/2018 7:28:55 PM	303471

#	Endereço da Transação (Rede *Rinkeby)*	Bloco	Hora da Transação	Gas
15	0x5b140c1bcdec4aad7efda970058d589f53164b8eb628a9d698364eeaf54bde6d	3293603	11/6/2018 7:29:40 PM	303215
16	0x90745b3671574b1709a2f605e6c9a2edf1eae862b404798ca35339361c829e39	3293606	11/6/2018 7:30:25 PM	303215
17	0xfc15c0156b8c9c8c07e9cda412f71eb52d01fab553a8ce083eb367dfe8f603cc	3293609	11/6/2018 7:31:10 PM	303215
18	0xa1da86390195d9109187192dc4f4da2090d74eed8959140cb261352f6c334bff	3293611	11/6/2018 7:31:40 PM	303215
19	0x8eb0b793f20d9b9d4116a3a58be60aa83f2b2e8517a684d46c1227539ea2a054	3293614	11/6/2018 7:32:25 PM	303471
20	0xf97a3a9ef9cc95517ae13149a33122139077e10ee8851f60652bb8e672577344	3293617	11/6/2018 7:33:10 PM	303215
21	0xf267248161156d7709c2ec73f7a55ede84b2333fbd7f7b69248ba8c3b77713184	3293619	11/6/2018 7:33:40 PM	303215
22	0xcfc4ee75c9c3e012ab65bbf48c91cecc312348f0e4cebf9c2855f984f17aaac0	3293622	11/6/2018 7:34:25 PM	303215
23	0x17d1f1d24a68211ccbaca5ee12e08fff17550435ca9148481682 2e5c985d6ecd	3293625	11/6/2018 7:35:10 PM	303215
24	0xd2fbd0b7d1f930c2ace1394e8ba9166704310 90ebb8ca8a6443e8e6ef240b8a6	3293628	11/6/2018 7:35:55 PM	303215
25	0xaa56ffe1f104da88d23491e0144a5b1ccde8e2f1ff1954c7843710192f335a50	3293632	11/6/2018 7:36:55 PM	303215
26	0x8d7d4399d040adb3ca4982dc2cfa7d6726c174a45cffbf29e9c917d4393e8ace	3293635	11/6/2018 7:37:40 PM	303471
27	0x0c7b2aad33c5bc9e3af2c2521a300c49caa8990d4a9fee91406a2d35bae4c6c7	3293638	11/6/2018 7:38:25 PM	303215
28	0xe0e4cd79b636ddea3d43cd722d6a6e969a568daeaddd93359813193dbb146977	3293641	11/6/2018 7:39:10 PM	303215
29	0x40a1da7ef3a55de719d97c6d38734ed95a48a2a60051c128367cca58de48bea9	3293644	11/6/2018 7:39:55 PM	303215
30	0x155e32833f49f1d2cc27c9e66b69482 9874ccd6b9ae994b22597d9f354457bb	3293647	11/6/2018 7:40:40 PM	303471
31	0x5d6a8ccf213ea006c34091a59124e8c4d8ec1379693f5665cc2cb198203619d0	3293650	11/6/2018 7:41:25 PM	303471
32	0x414f5accb10f6f4e4e6455786b8aa4e98e2a0e60934832c2806bfd3d6bf7ee77	3293654	11/6/2018 7:42:25 PM	303215
33	0x00ddd6615716ee6c52960f0ef0c14f8b102c2bbfdbda46761fb074148b6e0759	3293657	11/6/2018 7:43:10 PM	303471
34	0x8e56cf8b2f0197b711e9b45d8d1bff9c077e07f6935b96872bcd2582a3f956e4	3293660	11/6/2018 7:43:55 PM	303215

#	Endereço da Transação (Rede *Rinkeby*)	Bloco	Hora da Transação	Gas
35	0x933a9fe378ff915fc36e47232d6aa268ec401f8d950c64da436eb0036d48368b	3293664	11/6/2018 7:44:55 PM	303215
36	0xdc8ae291dc87d76c10cdc937b81d6fe0fd413e1650ffe992fe438ab127d2492e	3293668	11/6/2018 7:45:55 PM	303215
37	0x61683da7dc0e9cbb5c5d6549d160738fd0dd13a64d3742f4dc5ac1e9883c3a8e	3293671	11/6/2018 7:46:40 PM	303215
38	0x5721522a877d219cce1eb79d067a83fcea15d94bbf7a9af7a7496e58478b8dc7	3293673	11/6/2018 7:47:10 PM	303471
39	0x95fb02ad65356a82cd9060efafca20afbc9fbdcccd8ebc92c75ca90858d208a1	3293676	11/6/2018 7:47:55 PM	303215
40	0xa99173e35829dab2b3780aa05f0a435ac5bd169887c5116a18678ac60f045660	3293679	11/6/2018 7:48:40 PM	303215
41	0xc53f7bfde4498494e25c7c2a5e3c3654ba437cd5d1368bd61945041a0a2af0a3	3293681	11/6/2018 7:49:10 PM	303215
42	0xb604c7736c78c776f844a33978a4c1cf15632bbf3a7722e7311429c4f87b2bd5	3293685	11/6/2018 7:50:10 PM	303215
43	0x01e4d31d96a0a3836b553d5237f4f8b664364838acd2d3ecb85932acc7e743cf	3293688	11/6/2018 7:50:55 PM	303471
44	0x7adbe3eb48b8b96212292917a153f2eb68ca6ad153f84f648291948de026f759	3293691	11/6/2018 7:51:40 PM	303215
45	0x7a3a084abd20167339941041b96f68aa0586291 2b7b1972e67cac0648aca5c0ac	3293694	11/6/2018 7:52:25 PM	303215
46	0x20e0580faef446d7206067db5020205bbf88830cadfd4cbe10998bb36b968f46	3293697	11/6/2018 7:53:10 PM	303215
47	0x700b222c7bbe391c0634ed522c50e311ea0a6d92833146489012 8b4f27a9edd7	3293700	11/6/2018 7:53:55 PM	303215
48	0x0c800b41dab0a5f7e224a7bf7cdc8aef399fa150e3e829dda3eba41dbf27a273	3293702	11/6/2018 7:54:25 PM	303471
49	0x05cfe33feea12648e918ebdbc6b943bd8cf6b34d724dbd51a15312ff1fe9f85c	3293705	11/6/2018 7:55:10 PM	303215
50	0x71e91553ba365dc78c2de3474af1c6a22eb86ab4f170287 66a4d3f6830627126	3293708	11/6/2018 7:55:55 PM	303215
51	0xdc6438853410259 1b5a192fd50671dc07d9378b928f0ff9dea01843c6a06e93c	3293711	11/6/2018 7:56:40 PM	303215
52	0xfd8a5a06e0fd1620f0a90e6bbc31a991e2a7120a400bd92d4d3f103fc5d2bf3e	3293714	11/6/2018 7:57:25 PM	303215
53	0x28a50cc80f4fe122d536f34237ddc473a41497bd5b511c3d67ba31994c195edb	3293717	11/6/2018 7:58:10 PM	303215
54	0xe3e57dc4865b4342acf163e489246c1cd46d21c7ee766718a149ecefd34ac032	3293720	11/6/2018 7:58:55 PM	303215

#	Endereço da Transação (Rede *Rinkeby*)	Bloco	Hora da Transação	Gas
55	0x5880b82c59bbeb79baadafd632cb5770930e3218d0c57d3510ad7901db1f31c1	3293724	11/6/2018 7:59:55 PM	303215
56	0x03d16bf7648e31c2963ac13428c24a99c8e10a3db0f12205faaf851e557ae3a1	3293727	11/6/2018 8:00:40 PM	303215
57	0xa47b9f96a5593156a2b0abcee995f335658dacc4400f29910f1f7cf4e895516e	3293730	11/6/2018 8:01:25 PM	303471
58	0xbaafee3fff041a5e58c1671bfc04217d12b23bb61b6f1c63aa3abf7f36a57e53	3293733	11/6/2018 8:02:10 PM	303215
59	0xab6827dfcbda58e0c0bc31ddd3117d86da3ed2a283beeb2d8df89b24194a2bc5	3293736	11/6/2018 8:02:55 PM	303215
60	0x1fca68408e1f7ce34d09321cc041b2387314229c0e5e8c3f990d81d608dc1a8	3293739	11/6/2018 8:03:40 PM	303215
61	0x2c51cabc68ba0b656635d61c2a0b9549aceefcf6c1801031e51e8bab77590d81	3293741	11/6/2018 8:04:10 PM	303215
62	0x0001ac6a8f0d3f148f57decfddf9d0ad765c3d6dbd42407e3d52ff01ae07fe95	3293745	11/6/2018 8:05:10 PM	303471
63	0x92463a929e9e6a6a457b0286f9e42d19aa4a44df7e309735fd88101bc7991daf	3293748	11/6/2018 8:05:55 PM	303215
64	0x29b80fdd8d4a2e5de235dca1243e18f186ab9e4e0bfec7cdf8936d8d6c89d8a1	3293751	11/6/2018 8:06:40 PM	303215
65	0xec9291af84a7cbd5bf2061cae3fad450327d30eee09807dc39280dc28f29ea01	3293755	11/6/2018 8:07:40 PM	303215
66	0x37287a3c4c658b3b291570d630a5518caa13c728133a68cb76fd68202ab8c65c	3293758	11/6/2018 8:08:25 PM	303215
67	0x20fe89d80b5246f24ba1d16599409a06774f6a65a9b3109b00f3b809b4683184	3293761	11/6/2018 8:09:10 PM	303471
68	0xcc7e2dd1ffb3eb6569379a9484e5c2485df7a4334f52f8a41ffaf2eedf7cd441	3293763	11/6/2018 8:09:40 PM	303215
69	0x18ad32e5d04db6321401008001e6b1c8ee14c06c2a6da8aabf1f302d2b60a676	3293766	11/6/2018 8:10:25 PM	303215
70	0xc3e31b5d8abb18fec4a64e052a899ccc4d3baabcab5cf34d2c037905dca5ce97	3293769	11/6/2018 8:11:10 PM	303215
71	0x55d421ed7cac75f6f615c2ce85e7e2ec401acca9f63b5c4fdfad5d4a5ec03e3d	3293772	11/6/2018 8:11:55 PM	303471
72	0xf42d1ff002cdf56673e839c7d6f18c775b19e8ed434bdc3ec2e9f8baf4a0f3ce	3293775	11/6/2018 8:12:40 PM	303215
73	0x750021d58b49e8228a662ed9177e9c1beb8d83c52d68119709c74df58962592f	3293778	11/6/2018 8:13:25 PM	303215
74	0x89d93c4d06dacead6e6f3b2633c8edb79dff4a3824b9739408032a0f8f3e8e9d	3293781	11/6/2018 8:14:10 PM	303215

#	Endereço da Transação (Rede *Rinkeby*)	Bloco	Hora da Transação	Gas
75	0x6b7e838a32f4bc793163fe1d077be3e54240a3f253d73ecff93dc4f69816bf24	3293785	11/6/2018 8:15:10 PM	303471
76	0x1f6df592ceab1ab5ec02e70365434435b45175622a985dc9ecfa439986969f95	3293788	11/6/2018 8:15:55 PM	303215
77	0xe1a7b15e6c275d9a38660d270f4e3799ac9f57922a83d84dea73ee3a63b067741	3293791	11/6/2018 8:16:40 PM	303471
78	0x1d7032bf6b70d9156955e4f7eaf50be56ff8c9c1335c691576907f23b754efb9	3293793	11/6/2018 8:17:10 PM	303215
79	0x4102d9e505450bf971173ac04be12803bab9530eb4c88d9707cad414446ff2cea	3293796	11/6/2018 8:17:55 PM	303215
80	0xa8815f8563bd881066e5d806b873e69651892c88e6fb907f6b2673349c8c0f20	3293799	11/6/2018 8:18:40 PM	303215
81	0x54a4ef37c2f1171647l1a2b4533a4fc14cd4c53002288f080be66169d356249b2	3293802	11/6/2018 8:19:25 PM	303215
82	0x9f2b833f78bee5e3a5c8af86e1e8d272ec1345b0b0de6ec3bdfb5f00a771ba3e	3293805	11/6/2018 8:20:10 PM	303471
83	0xd7f8256316601be062d1b3ba0143ee73ae26c73e4531919164ece0a27c9cbf47b0	3293808	11/6/2018 8:20:55 PM	303471
84	0xa769780e31ffa0b1dfba0345801621230adeedd0e0db67c41895ea5e26e9cd11	3293811	11/6/2018 8:21:40 PM	303471
85	0xc9e8312831ca47a5714249f5149598e498d8037998670f233c101f3543988b80	3293815	11/6/2018 8:22:40 PM	303215
86	0xd6bad203b0ada9953eb756ee0c8a0391e853c4ebfd9b59834098da56912a6ef8	3293818	11/6/2018 8:23:25 PM	303471
87	0x30bbf5f9b800bc8c2e7b0f3658d709aef6fcbc997ce1205c30756b5d9b4fd4a3	3293821	11/6/2018 8:24:10 PM	303215
88	0x3aee8634641311bbab81a8552ec676c5358b060e691b3facf7890df5561802979	3293823	11/6/2018 8:24:40 PM	303215
89	0x70999f98c059bf129869ed13b8c9893f1e1efe7915986e96796ab480d00df72f	3293826	11/6/2018 8:25:25 PM	303215
90	0xbe1e1b40a4ffcef971a1ecc4387073ce6f22d0fa855c2d7fb8ab123cc480903b	3293829	11/6/2018 8:26:10 PM	303215
91	0xfe1504d28e65178932c9cc16952efabb92c4531b46a73ad70861f14769c54109e	3293832	11/6/2018 8:26:55 PM	303215
92	0x4ae41a5306724d4d95c86ccc02377acf1a69991052d8c253b0ba10cd865d9ea1	3293836	11/6/2018 8:27:55 PM	303215
93	0xca90dc6d40979b42cd47c381dd511af0d7f0177fb66232 6130ba468a8fd6c764	3293839	11/6/2018 8:28:40 PM	303215
94	0x0728dff191cf5fff63180bd95a753d33b19b807fc3eed2ea81539dbcd2c863a8	3293842	11/6/2018 8:29:25 PM	303215

#	Endereço da Transação (Rede *Rinkeby*)	Bloco	Hora da Transação	Gas
95	0x1d74511994f423defd7dedcfa2205671142ffaefb0feaeb1a09f7a54ace4475991	3293846	11/6/2018 8:30:25 PM	303215
96	0x19426677a46fc4d2706bdc92c52579c7f9daa027cf99c9692fb2e9fafa8754164	3293849	11/6/2018 8:31:10 PM	303215
97	0x9f063c1306f8a5142a6b1994658da56427e5db1b99e09cb3184a127722798b2e	3293852	11/6/2018 8:31:55 PM	303215
98	0xf07ac5c9d6cd64f4636c1348bd22c3c1d5505fa157672173961c2e489783907e	3293854	11/6/2018 8:32:25 PM	303215
99	0x16d7b804b6019a0a9fde9b90dc7ea59877fb66d6dd7183b731bf683e7bd8dde8	3293857	11/6/2018 8:33:10 PM	303215
100	0x22bb1a5b25518bf3e14e640d0937105a21dfe4552dd01c3b7744aca985563cad	3293860	11/6/2018 8:33:55 PM	303215

8.10 Resultados do Comparativo (Vetor)

#	Endereço da Transação (Rede *Rinkeby*)	Bloco	Hora da Transação	Gas
0	0x319ed96036b19dbaf59b5f5d6596110b7f3a469746c730d2fb397c5621663412	3293554	11/6/2018 7:17:25 PM	903945
1	0xc07b3444d1f72b6595ac20747ff901085ca946fa13f18737891c78daf1d7135f	3293561	11/6/2018 7:19:10 PM	322672
2	0x915bd295ff90dc7efeddec96455b97b4ffbacb1ccb17258d34f6d5f87150dbe1	3293564	11/6/2018 7:19:55 PM	310003
3	0x5eb13487f67f19d8a5f3905194afd6a9e527a27c37c3e78707a3eb4b133c3671	3293567	11/6/2018 7:20:40 PM	312334
4	0x79970bfc4e6b7df0c496d62a85b4b1f7e5c22790258f6e54886841d6f668469b	3293570	11/6/2018 7:21:25 PM	314666
5	0xe8021061fedb8fc487cc70b5e245904ac2624c6a2876953abd3f29aed296a47e	3293573	11/6/2018 7:22:10 PM	316997
6	0x3c17d87b3c62f9a6eb3e7b50da1a2d45414895422238252cbd366f477e0d420c	3293576	11/6/2018 7:22:55 PM	319329
7	0x7526715a9acb0ceba26c71d10ea981ce197e7c3270ff2dc1d7c26d95389f7e9d	3293580	11/6/2018 7:23:55 PM	321660
8	0x6b9ebc1d432276095ff796514517a2913f5d39e748158e7b52159f0376da4fc9	3293583	11/6/2018 7:24:40 PM	323992
9	0x1ee6366c262126cb1e5ea8c26b81fb5cd0910347cc7547a9607349fd8afc1db6	3293586	11/6/2018 7:25:25 PM	326579
10	0xf22365b4e259cc3a1b3b427891401a30e6e37ea13bd1b555b7a6355223260e90	3293589	11/6/2018 7:26:10 PM	328655
11	0x261129203e43c3754f31b286bdd5e557b56a0009e4abd4fe4b83bf8ac5e378ef	3293591	11/6/2018 7:26:40 PM	330986
12	0xffb4354eabb38f0f5779eeed7e9edfa82abad21fc02ed27cfe398f7aa7d61d51	3293594	11/6/2018 7:27:25 PM	333318
13	0xbd505c105df2aacecffa00c82ec0745179c8269c60994ce83bbffd910fbfec1c	3293597	11/6/2018 7:28:10 PM	335650
14	0xef0499b7c43abcb38ad65df505ef0ecbbdfef79286ed5d0ee81ed467a8284bab	3293601	11/6/2018 7:29:10 PM	338238
15	0x19d809c4cd0c2e5b2e00bff4cc0a91059954b3883e11656c528850adf67aa684	3293604	11/6/2018 7:29:55 PM	340313
16	0x4e4835538bea57a227394295febc7e06be22e2437ae7624c058fed9dbdc1ae85	3293607	11/6/2018 7:30:40 PM	342645
17	0xe6a11953d755264fdabcb94cf1f7c73e81153d50f15bd97d32bb27d6d4f200ff	3293610	11/6/2018 7:31:25 PM	344977

#	Endereço da Transação (Rede *Rinkeby*)	Bloco	Hora da Transação	Gas
18	0xa3d4a625b5ac08bbc5ffde529c5e2aee41e29fa88f799499bbfd05f39ab9afd7	3293612	11/6/2018 7:31:55 PM	347309
19	0x73e3e9a1d9e653e8318b9bc840da4866ec6061f920ae13b5ca0edb79d83695e1	3293615	11/6/2018 7:32:40 PM	349897
20	0x72250528078c8646dfc2e582fcfe622937c1084f8fb7c9a1ea541789b7b15aa7	3293618	11/6/2018 7:33:25 PM	351973
21	0x096c41d2faef54d92484a2248eb1837dc1b02c4f0654f9682cbe753b413c1fb1	3293620	11/6/2018 7:33:55 PM	354305
22	0x2bfb35d46dd5743fe13bd3ce5d6163c11ad128072ae1b01eeced43ecf4644296	3293623	11/6/2018 7:34:40 PM	356637
23	0x183f9a8fca2da824316dc22af0035f82e597fcde91f9d1a188570384a4405b71	3293626	11/6/2018 7:35:25 PM	358969
24	0x311ee7150c1d98a264aa397c1fa276c21454f352af1b332fcd88d8d1b7ae7d36	3293630	11/6/2018 7:36:25 PM	361301
25	0x6625b74b77a84513870cfabd57cd94cf06543013fd1e3956f978af59dcc3dd44	3293633	11/6/2018 7:37:10 PM	363633
26	0x78b2f1d58d8e3dca149406e22222acc753c7bb20dbece698924edcde6b086c4d	3293637	11/6/2018 7:38:10 PM	366221
27	0xf7ea62adf720d5f1de2310637e373c97f9de930f2a44a7b34c74cd3037cfe7e9	3293640	11/6/2018 7:38:55 PM	368297
28	0x9a577377d9a2be91ebb9187308a13c5b10ca08a6140b19d6998bd4b6e4dd3e4c	3293642	11/6/2018 7:39:25 PM	370629
29	0xa057f0d494ca425f8d20fdb559f030b46745c22df3ad4ebe8220fe421c4131ad	3293645	11/6/2018 7:40:10 PM	372962
30	0x32c0eb82578a7f9c6e383be48adc399287c1788fc7a67c4340757c32e129dbbb	3293648	11/6/2018 7:40:55 PM	375550
31	0xc00ccf0770c441adfdfa457a147e8a4d68f84729c2f65691ade493db87625fd1	3293652	11/6/2018 7:41:55 PM	377882
32	0xaf06df3853d74036d40d3fef83e097ba9399260630007f8319a737cd9f9e6c07d	3293655	11/6/2018 7:42:40 PM	379959
33	0x687ff5a5aaf1fdf05332a66c6f71a9fe9ba93d6bf4c6e42ab3ef68f5ad34c89d	3293658	11/6/2018 7:43:25 PM	382547
34	0x220d7f62cef198b2826e149ed3f5a1c5faa914898881beed784b6626b527fdd2	3293662	11/6/2018 7:44:25 PM	384624
35	0x09a7d88333eca7611c308737e05f12066c6c2d190d5de521e575a5f823e97cdc	3293666	11/6/2018 7:45:25 PM	386956
36	0x1af1cc27359af7ea5784437b31553070b6bcad9f6b3e510df22b881476af51f0	3293669	11/6/2018 7:46:10 PM	389289
37	0xd76e7e45ef7ad7962e91e392f2576204c58331e85f506b47d226d3765f5a2905	3293672	11/6/2018 7:46:55 PM	391621

#	Endereço da Transação (Rede *Rinkeby*)	Bloco	Hora da Transação	Gas
38	0xf44aa45ab1d49d9c42aae6df561176443f289bc5680c2efedbb09af5fcd6340f	3293674	11/6/2018 7:47:25 PM	394210
39	0x07205acd2e2f236dfc93647ff63216f6496a45020b4018090ddb174fe33a7860	3293677	11/6/2018 7:48:10 PM	396286
40	0x33c3100d00af37eabeee8b802ec9d105ee7e9f13fbeb71b940a528457a93d3ec	3293680	11/6/2018 7:48:55 PM	398619
41	0x3b526e043ba1dc37a168ef84c4dc4f02508ff9c09b163bcdc91bce253b7d30ab	3293683	11/6/2018 7:49:40 PM	400952
42	0x1b92ed98a11499e3a9561cc6925b37825a0225c250e21ffd254e8ea25383a020	3293686	11/6/2018 7:50:25 PM	403284
43	0x6925c6adddd4759d8cde36762f045a90ab14460c664f4011d493de46ae812c22f	3293689	11/6/2018 7:51:10 PM	405873
44	0x235498d8e58eb6cccac7d5b1a2fb2e314aca342c744a696fcd50748e0d38e7ac	3293692	11/6/2018 7:51:55 PM	407950
45	0x6f4d2735adf6fd5284d94485bdc89a5253dc448d7e1889b27cb25eca0aca079d	3293695	11/6/2018 7:52:40 PM	410283
46	0xba5a46976b90df8cf731cf795f4f7555e40efd403ae661963e2871f1610ee46e	3293698	11/6/2018 7:53:25 PM	412616
47	0x75df501a0b535d82277f4c8dd63a54978c20184423405laa37cbd831e62f32ba	3293701	11/6/2018 7:54:10 PM	414948
48	0x469e4f9904684dfdee88eddb9fe2efd60f6d75e423d21b45b3bed147eaafeee9	3293703	11/6/2018 7:54:40 PM	417537
49	0xe421a5873459f6cfb45f0871e656ce240889d8b72048a7a04436d8a3f884d04f	3293706	11/6/2018 7:55:25 PM	419614
50	0x89d4fac3f159a39ca9a8ca1bf86d267cf1d52960be4065b8e13a6d8f0b34364d	3293709	11/6/2018 7:56:10 PM	421947
51	0x9fa587f9af5037298ea18d9fc32e3f4ce449c396cc140b244dc01e3df6dd2769	3293712	11/6/2018 7:56:55 PM	424280
52	0xbae9011b3cc95d62b9bfa5c19b2d14db608e1a56f90077e0cd223f1f26d05765	3293715	11/6/2018 7:57:40 PM	426613
53	0x4f68693c3981e2b3a9ddcdd1ba0d2fc67b7b679a7ea344fec6a22537b7e106ea	3293718	11/6/2018 7:58:25 PM	428946
54	0x4bfd2390ee0a7d214c663fafe4d160cb7d0813c9369183986a69b2ee6d10c78a	3293722	11/6/2018 7:59:25 PM	431280
55	0xc82848eb3b1332807e8b73970f6d3166edff392e3440ef7628c6834a8df9509e	3293725	11/6/2018 8:00:10 PM	433613
56	0xab0ddde5a72992ddc850691d9538221cbc33ef6119721a168bf415a0e255538e1	3293728	11/6/2018 8:00:55 PM	435946
57	0x6abcaed18c2a153d099129718f828326a4e2f7d10611977dfe3015ef4a0d7b9d	3293732	11/6/2018 8:01:55 PM	438535

#	Endereço da Transação (Rede *Rinkeby*)	Bloco	Hora da Transação	Gas
58	0x89139b0d666a569d509218f393f2bca6b56c29762d745d73377ccc85eb941a1b	3293734	11/6/2018 8:02:25 PM	440613
59	0x446209efbc8bf4e6efd9adc6168f34c6be429466271fc84b7de2e3880f674d44	3293737	11/6/2018 8:03:10 PM	442946
60	0xdc6c4fa6b007825ddcb29cc64de5dd8b774dd7924be20c2ceb54a1c1d72c16af	3293740	11/6/2018 8:03:55 PM	445279
61	0x762a0fc30ef58a2228cd8cf296fb107d6942e8d7f1e8a49dcb4f10380026d463	3293743	11/6/2018 8:04:40 PM	447613
62	0x81c3115c54d0a95c67f10d5cbcb0226be88ed989cc9c46fc65a22424b8b1dab1	3293746	11/6/2018 8:05:25 PM	450202
63	0x6e454c20f456356e275d44914208f7b3914fd7e28fa2dd7111956ee33a9fb3a48	3293749	11/6/2018 8:06:10 PM	452279
64	0xa38f22a1ded083c261845db377d8dc1ff5bf2be91d89d0147813b57f5d084ed1	3293753	11/6/2018 8:07:10 PM	454613
65	0x1377972557da6562618d6d4039f3dc1851e889cf545a51decd6ddb4fb23b0b36	3293756	11/6/2018 8:07:55 PM	456946
66	0x3a3e74bac842fcb0e1a4a2544f7b5e32ec7dc041162c57c24d656c633d22ebee	3293759	11/6/2018 8:08:40 PM	459280
67	0x6d08a03638bbb2671ece65b2d3cd8f1e02ab02eee70a305cda6e8645a439234e	3293762	11/6/2018 8:09:25 PM	461870
68	0x3298dcf685875d394a9adf77c2a8358a151b50dccf55c5e38bac073eff122356	3293764	11/6/2018 8:09:55 PM	463947
69	0x3518d6d9c0ad7184b432f4adbff6ec8eaf01e55c99212cd832708cb2e06ada97	3293767	11/6/2018 8:10:40 PM	466281
70	0x196a7afa7210bcd3bde7d99bc43e321f9e194a2a2a9c3b9d92d6f7eee9300664	3293770	11/6/2018 8:11:25 PM	468615
71	0x0b96397e99a8bb9017238bc27a338a157b72505b97387c383175f18ebb8a9a05	3293773	11/6/2018 8:12:10 PM	471204
72	0xe3201806584b8dfcf305ccbdb1bdf90322469805eff564199e155c161a8b3a38	3293776	11/6/2018 8:12:55 PM	473282
73	0x1235ddbf594a2794b43733677a9461a89ba70e7b8e2852bd1b8ec0b9e64f8e6d	3293779	11/6/2018 8:13:40 PM	475616
74	0x2ab8dcc3398ea15e29ff515bfd564d2877097aeb7bedd7b39bc916b88f1b04cd	3293783	11/6/2018 8:14:40 PM	477950
75	0x03153b67ca77dd412f7b1c799a28bb89b1202c7ace6e8c4754d0c39df5fe5269	3293786	11/6/2018 8:15:25 PM	480540
76	0x11265b5ba8b22137bb79bddc7bf7818aa8df7ab0fe8d4ca4ed95c608eb43361	3293789	11/6/2018 8:16:10 PM	482618
77	0x3e52425d42fea4228e899163d37003c5dfe7637e38b6bca4c84ffb6f7272ae9f	3293792	11/6/2018 8:16:55 PM	485208

#	Endereço da Transação (Rede *Rinkeby*)	Bloco	Hora da Transação	Gas
78	0xe37329e67cdd0cf6158279e9421de47f37e7940c9d51da9484a8486694415 26e	3293794	11/6/2018 8:17:25 PM	487286
79	0x6e734bdf08396b821ff542f2e8905054a7c840b2a5fbaece86b2463da47a0228	3293797	11/6/2018 8:18:10 PM	489620
80	0xd4b2a61e1ea47ad1e905c14f5045fb3b660fb65cc6fe1e61c2aeb22548a4edf1	3293800	11/6/2018 8:18:55 PM	491954
81	0x6907eadd2a074fe2a3f20b42332be3e906af5ec4887e2a58f3bb47c2a66604fb	3293803	11/6/2018 8:19:40 PM	494288
82	0x387846b222d8be9532926dd3d2b1ec48da613f1ac80354a570e46ca6a92361d7	3293806	11/6/2018 8:20:25 PM	496878
83	0x81b064daf75129886e570c3999dad565582174 99e4e8f9679a3f1e6f12cee0a5	3293809	11/6/2018 8:21:10 PM	499212
84	0x5fa4559d8c2378c993bec213b5a47f85bd7bfbfb53e778fd125711c1ce720e38	3293813	11/6/2018 8:22:10 PM	501546
85	0x7b7f9cf1b0b5a991b021cd35852c0535d09c8a59f04c497831f93eb1c443caaa	3293816	11/6/2018 8:22:55 PM	503624
86	0x2b4c3d1fae3f1dba7926a6543dd8d4cc8fb106b52bf89684ecd91a706cdd5225	3293819	11/6/2018 8:23:40 PM	506215
87	0x06d1131dbd00372dd34ff83695ed11938ca2fa15ac3eb59a6b47edbd06c68cf8	3293822	11/6/2018 8:24:25 PM	508293
88	0x40d84eb39e76cc79e0e7e0ded1b7487bad1eeb8b69e4a5215b6c21d89c260b04	3293824	11/6/2018 8:24:55 PM	510627
89	0x62de593a97f23b6114905643 8b46a7b719e5fd00e814c49220aa3fae64d95899	3293827	11/6/2018 8:25:40 PM	512962
90	0xf63ac3a95969a4292826b245c629636bd9bc6c9d586c56ce9f98d28e79e2c768	3293830	11/6/2018 8:26:25 PM	515296
91	0xd5b2cd0e2989b60e1313f11511a47fc8a33bbdb535d028c79e53cf4f16b9ef48	3293834	11/6/2018 8:27:25 PM	517630
92	0xa6bc63a8a54944eea52e0db60f223f88693e2dbd9f5295ebbb743aec58bef71a	3293837	11/6/2018 8:28:10 PM	519965
93	0xd74a6c83efeba6e9f7b40b981d6416a44c515fbd4d99f5a714deb0bb23bb4f86	3293840	11/6/2018 8:28:55 PM	522299
94	0xceb40fe59b1047850f8d831fe8e1833cb171d768f4fb50e0f8d0233850b85d97	3293844	11/6/2018 8:29:55 PM	524634
95	0xbce0c57fe0a8b8d9d298bfba190cdad28e9ab9842d1be079d33572ba5da44dd1	3293847	11/6/2018 8:30:40 PM	526969
96	0x261716e5ae958631f04106f94c5f7585bfc147e0c06d5e4570b9b1abb878dc52	3293850	11/6/2018 8:31:25 PM	529303
97	0xda0d6cd933ca9e65a68ea425b73dbe86b4114705c2dabb99404279daad7936fe	3293853	11/6/2018 8:32:10 PM	531638

#	Endereço da Transação (Rede *Rinkeby*)	Bloco	Hora da Transação	Gas
98	0xb696ee395e4ef6530ab798c7048d9984c321c069eef196db90f49a11f232c6a2	3293855	11/6/2018 8:32:40 PM	533973
99	0x96dd6ff2c47b4878652587094c4e9f8482e7faba76175395c2975a25df747f4	3293858	11/6/2018 8:33:25 PM	536307
100	0x21f1f0b9783e7df6d29a05a0cd31cb3dfd409c5eaa52e70617ea937f48c0e5ab	3293861	11/6/2018 8:34:10 PM	538642