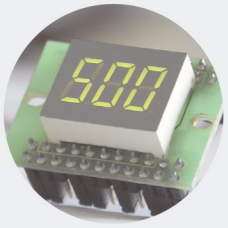
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






Block The Vote - Aplicação descentralizada para a criação de um sistema de votação online

ANA RITA DA CUNHA BARBOSA
Julho de 2022

POLITÉCNICO
DO PORTO

Politécnico do Porto
Instituto Superior de Engenharia do Porto

Block The Vote - Aplicação descentralizada para a criação de um sistema de votação *online*

Ana Rita da Cunha Barbosa

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização em Sistemas e Planeamento Industrial

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto

Julho, 2022

Esta dissertação satisfaz, parcialmente, os requisitos que constam da Ficha de Unidade Curricular de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores, Área de Especialização em Sistemas e Planeamento Industrial.

Candidato: Ana Rita da Cunha Barbosa, Nº 1170647, 1170647@isep.ipp.pt

Orientação Científica: Susana Nicola, sca@isep.ipp.pt

Coorientação Científica: Nuno Bettencourt, nmb@isep.ipp.pt

Empresa: Do iT Lean, Inc.

Orientador: Nuno Dâmaso, nuno.damaso@doitlean.com

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
Rua Dr. António Bernardino de Almeida, 431, 4200–072 Porto

Julho, 2022

Agradecimentos

Esta dissertação representa a conclusão de um percurso repleto de esforço, trabalho e dedicação e o alcançar de uma conquista importante para mim, pelo que gostaria de expressar a minha gratidão pelo apoio recebido durante toda este jornada.

À professora Susana Nicola e Nuno Bettencourt, por me terem acompanhado ao longo desta dissertação, estando sempre disponíveis para o esclarecimento de todas as minhas dúvidas, e incentivando-me a ir mais além.

À empresa Do it Lean, pela oportunidade que me foi dada e pela forma como me acolheu, proporcionando-me todas as condições necessárias para a concretização deste projeto. Com especial atenção à Engenheira Paula Graça e ao Engenheiro Diogo Pacheco que sempre se mostraram disponíveis para me auxiliar, partilhando sugestões e ideias que complementassem o projeto.

Ao Engenheiro Nuno Dâmaso, que acompanhou todo o projeto de perto, facultando o seu conhecimento e apoio para com o trabalho que foi desenvolvido, permitindo-me adquirir novos conhecimentos no âmbito desta dissertação.

Aos meus colegas de curso que fizeram este percurso comigo, por todos os bons momentos passados e de mútuo apoio e pelas boas amizades que tive a oportunidade de fazer, bem como a todos os amigos fora do ambiente universitário que me apoiaram durante este percurso.

Por último, quero agradecer à minha família que me possibilitou este percurso escolar, e pelo apoio e colaboração durante esta etapa que agora termino.

Resumo

Face à evolução da Internet ao longo dos últimos anos, têm surgido cada vez mais tecnologias inovadoras. Um exemplo disso é a tecnologia *blockchain* que tenta combater as fragilidades da área em que poderá ser aplicada e auxiliar o seu funcionamento e eficiência através de uma rede distribuída, descentralizada, imutável e transparente. Uma possível área de aplicabilidade, é nos sistemas de *e-voting* como urnas para armazenar os dados de uma votação.

A presente dissertação teve como principal objetivo o desenvolvimento de uma aplicação capaz de acompanhar os processos inerentes a um sistema de *e-voting*, nomeadamente todos os procedimentos logísticos associadas a uma votação e os seus participantes, bem como todas as ações pertinentes, recorrendo a uma *blockchain* para armazenar, manter a integridade e auxiliar os processos previamente identificados. A aplicação é composta por duas vertentes: uma *reactive web app*, destinada aos gestores de votação e candidatos; e uma *mobile app*, exclusiva aos eleitores.

Desta forma, utilizou-se a plataforma de desenvolvimento *low-code* OutSystems para se construir uma aplicação que compreende as funcionalidades associadas ao processo de votação, ao mesmo tempo que proporciona a interação com uma *blockchain* através de um *smart contract*.

Para analisar a usabilidade da aplicação, desenvolveu-se um questionário composto por cinco questões avaliadas através de uma escala de cinco níveis. O foco dos resultados advindos deste inquérito incidia sobre os dois níveis superiores, pelo que é possível afirmar que a aplicação satisfaz as necessidades dos utilizadores, sendo viável a sua implementação numa situação em concreto.

Palavras-Chave: Blockchain, Ethereum, E-Voting, Smart Contract, Low-code, Outsystems, Desenvolvimento de Software.

Abstract

Given the evolution of the Internet over the past few years, there have been an increasing number of innovative technologies. One example of this is blockchain technology, which tries to fight the weaknesses of the area where it can be applied and help its performance and efficiency through a distributed, decentralized, immutable and transparent network. One possible area of applicability, is in e-voting systems like ballot boxes to store the data of a vote.

This dissertation's main goal was to develop an application capable of following the processes inherent to an e-voting system, namely all the logistical procedures associated with a vote and its participants, as well as all the relevant actions, using a blockchain to store, maintain the integrity and assist the processes previously identified. The application is composed of two parts: a reactive web app, intended for voting managers and candidates; and a mobile app, exclusively for voters.

Therefore, the OutSystems low-code development platform was used to build an application that comprehends the functionalities associated with the voting process, while providing interaction with a blockchain through a smart contract.

To analyze the application's usability, a survey was developed consisting of five questions evaluated through a five-level scale. The focus of the results from this survey was on the two higher levels, so it is possible to state that the application meets the needs of users, being suitable for implementation in a specific situation.

Keywords: Blockchain, Ethereum, E-Voting, Smart Contract, Low-code, Outsystems, Software Development.

Índice

Lista de Figuras

Lista de Tabelas

Listagens de Código

Lista de Acrónimos e Siglas

ABI	*Application Binary Interface*
AP	Arquitetura do Projeto
API	*Application Programming Interface*
CCU	Cenário de Casos de Uso
COMELEC	Comissão Eleitoral das Filipinas
CRUD	*Create, Read, Update and Delete*
DApp	*Decentralized Application*
DE	Diagrama de Estados
DEE	Departamento de Engenharia Electrotécnica
DLT	*Distributed Ledger Technology*
DS	Diagrama de Sequência
DSR	*Design Science Research*
ECDSA	Elliptic Curve Digital Signature Algorithm
ETC	Ethereum Classic
ETH	Ethereum
EVM	*Ethereum Virtual Machine*
IA	Inteligência Artificial
ISEP	Instituto Superior de Engenharia do Porto
MABS	Mobile Application Build Service
MEEC	Mestrado em Engenharia Electrotécnica e Computadores
NONCE	*Number Only Used Once*
P2P	*Peer-to-Peer*

PoS	*Proof of Stake*
PoW	*Proof of Work*
PWA	*Progressive Web App*
SCL	*Smart Contract Language*
SHA	Secure Hash Algorithm
SI	Sistemas de Informação
SOA	Arquiteturas Orientadas a Serviços
TEDI	Tese/Dissertação
UI	*User Interface*
VVPAT	Voter Verifiable Paper Audit Trail
ZKP	*Zero Knowledge Proof*

Capítulo 1

Introdução

O primeiro capítulo desta dissertação é responsável por contextualizar os temas abordados e definir os objetivos propostos, evidenciando, ainda, a empresa onde o projeto foi realizado e as opções metodológicas tomadas. Por último, é apresentado um diagrama de Gantt alusivo às tarefas desenvolvidas, bem como é descrita a estrutura do presente documento.

Este documento descreve o desenvolvimento do projeto no âmbito da unidade currícular de Tese/dissertação (TEDI), integrada no 2ºano do Mestrado de Engenharia Eletrotécnica e de Computadores (MEEC), no Departamento de Engenharia Eletrotécnica (DEE) do Instituto Superior de Engenharia do Porto (ISEP), bem como o estágio realizado na Do iT Lean na execução do mesmo.

1.1 Contextualização

Em 1991, Stuart Haber e W. Scott Stornetta desenvolveram uma rede de blocos protegidos criptograficamente que mais tarde viria a ser chamado de *blockchain* [1]. No ano seguinte, esta rede, que não possibilitava a adulteração do registo de data e hora dos seus documentos pertencentes, passou a integrar Merkle Trees, de modo a permitir que cada bloco pudesse conter mais do que um documento, aumentando assim a sua eficiência [2].

O ano de 2008 foi extremamente impactante para esta tecnologia, uma vez que a mesma foi aplicada na criação da primeira criptomoeda, Bitcoin, pelo pseudónimo Satoshi Nakamoto [3].

Este momento vê-se como o primeiro momento de evolução da tecnologia *blockchain*, denominado de Blockchain 1.0, que se refere à aplicação desta tecnologia para a criação de criptomoedas, sendo a Bitcoin a primeira e a maior até aos dias de hoje [3].

Posteriormente, face às limitações da Bitcoin, surgiram novas plataformas capazes de desenvolver aplicações descentralizadas através da criação de *smart contracts*, isto é, códigos que permitiam a criação de outros tipos de aplicações para além das criptomoedas [3, 4]. Este acontecimento é intitulado de Blockchain 2.0.

O terceiro momento, que é o que se está a viver atualmente, retrata a aplicação da tecnologia *blockchain* nas mais variadas áreas tais como a saúde, ciências, IoT, política, com a presença de decentralized applications (DApps) ou a utilização da *blockchain* como um sistema complementar para o registo de transações [3, 4].

Estes três momentos impulsionaram o crescimento da Web3, isto é, uma nova iteração da World Wide Web baseada na tecnologia *blockchain* que incorpora a ideia da descentralização [5].

Uma das aplicabilidades desta tecnologia são em sistemas de votação, mais concretamente em sistemas de votação eletrónica (*e-voting*), uma vez que a mesma pode trazer mais segurança, transparência, imutabilidade, confiança e mobilidade, permitindo que uma votação seja realizada a partir de qualquer parte do mundo com um dispositivo móvel.

Não obstante, tantos os sistemas de *e-voting* como a tecnologia de *blockchain* levantam algumas preocupações por parte dos seus utilizadores.

Empresa de Acolhimento

A Do it Lean, fundada em 2009, é uma empresa sediada em Leiria que se foca no desenvolvimento de aplicações de *web* e *mobile*, através da utilização da plataforma de *low-code* Outsystems [6, 7]. Esta plataforma proporciona a redução dos custos de desenvolvimentos e o aumento da velocidade de produção [6]. A empresa segue ainda as orientações da metodologia Ágil, mais concretamente o *framework* Scrum, permitindo um contacto constante com o cliente e entregas mais rápidas de produtos funcionais que satisfaçam as necessidades dos clientes [6].

Após a sua criação, em 2010, a empresa consegue os dois primeiros clientes de grande importância, Via Verde e Ministério dos Negócios Estrangeiros, bem como o primeiro cliente nos Estados Unidos [7].

Em 2011, a empresa funde-se com a empresa Simetricircle, também sediada em Leiria, possibilitando a expansão dos seus serviços para outros países como Reino

Unido, Ásia, Espanha e Brasil, tendo sido ainda reconhecida pela Outsystems como *Most Valuable Partner* e *Premier Partner* no ano seguinte [7].

No ano de 2016, para além de adquirir o primeiro cliente remoto nos Estados Unidos da América, conquistou o primeiro prémio *Outsystems Innovation Award* [7].

Em 2017, tornou-se uma empresa verdadeiramente global, obtendo o primeiro cliente na Austrália. Adicionalmente, evidenciando o seu crescimento, a empresa estabeleceu-se na cidade do Porto e América do Norte, e foi reconhecida pela Outsystems como *Elite Partner* assim como obteve o prémio *SME Excellence* [7].

O ano de 2019 marcou a empresa, que abriu novos escritórios nos Açores e alcançou 6 prémios (*OutSystems Partner of the Year Award* e cinco *OutSystems Innovation Awards*) [7].

Apesar da situação pandémica, em 2020 e 2021 expandiu-se para a Índia e Malásia e adquiriu ainda um total de 7 prémios *OutSystems Innovation Awards* [7].

1.2 Questões de investigação e metodologia de investigação

Tendo em consideração a área em que este projeto se encontra, surgem as seguintes questões de investigação:

- Como contribuir para impulsionar a adoção da Web3?
- Que tipo de funcionalidades uma aplicação digital deve conter para que seja possível criar, gerir, participar e analisar uma votação?
- Qual ou quais as ferramentas mais apropriadas para a interação com uma *blockchain*?

No desenvolvimento deste projeto, a metodologia de investigação utilizada foi a *Design Science Research* (DSR), que tem por base o desenvolvimento de artefactos como forma de responder a problemas nunca antes abordados ou, por oposição, melhorar soluções já existentes no mercado [8].

Esta metodologia é bastante usual, principalmente na área dos sistemas de informação e ciências da computação, uma vez que fornece uma abordagem pragmática para a criação de novas tecnologias, focando-se em gerar conhecimento [9].

Para satisfazer os requisitos de uma investigação seguindo a DSR, teve-se em consideração as sete *guidelines* concebidas por Hevner et al. (2004) [8].

1.3 Objetivos

A presente proposta visa, assim, ir ao encontro das questões de investigação levantadas através do desenvolvimento de uma solução digital (aplicação) com os seguintes objetivos:

- Acompanhar os processos inerentes a um sistema de votação;
- Conectar os dispositivos móveis com uma *blockchain*;
- Desenvolver um *smart contract* capaz de atender as necessidades mais comuns/relevantes de uma votação.

1.4 Plano de Trabalho

O diagrama de Gantt, ilustrado na Figura 1.1, resume o plano de trabalho para este projeto.

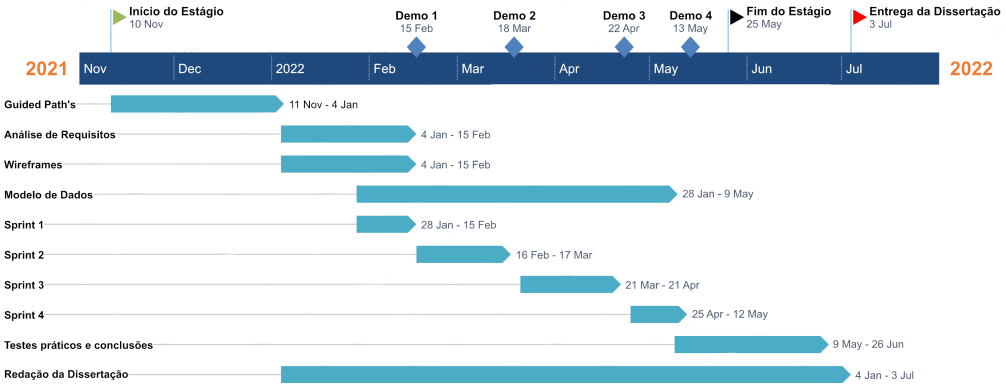


Figura 1.1: Diagrama de Gantt com a calendarização do projeto

1.5 Organização da Dissertação

O presente documento, exibido na Figura 1.2, contém seis secções: Introdução, Revisão Literária, Metodologias, Desenvolvimento de *Software*, Validação da Solução e Conclusões.

O Capítulo 1 serve como capítulo introdutório a todo o projeto, onde é feita contextualização do estágio, são apresentados os objetivos da empresa para o mesmo, bem como a calendarização das diferentes etapas.

No Capítulo 2, é feita a revisão literária que irá auxiliar o desenvolvimento deste projeto. Primeiramente, é exposto o conceito de *E-Voting*, exemplificando alguns casos de sucesso. De seguida, é explorada a noção de *blockchain*, destacando a sua estrutura, principais características e mecanismos de consenso. Por fim, são

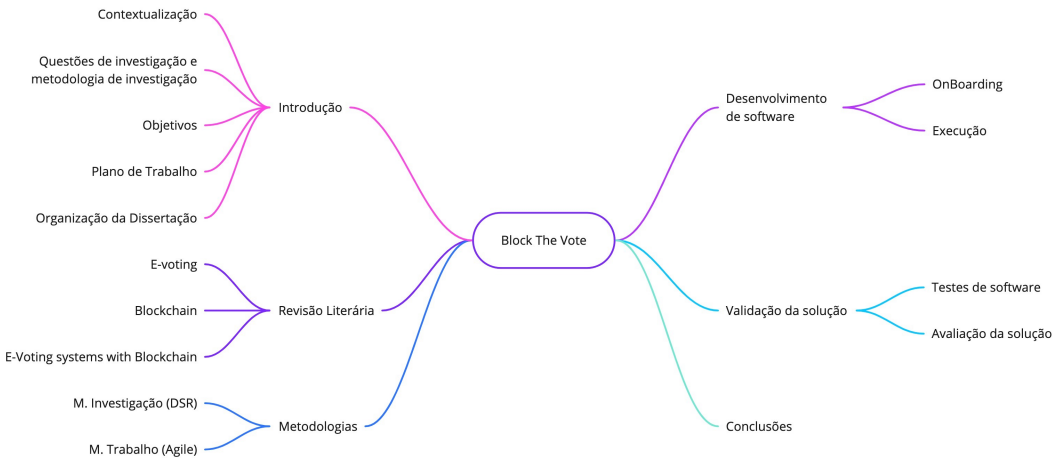


Figura 1.2: Estrutura da dissertação

enunciados alguns exemplos de sistemas de votação *online* que recorrem a uma *blockchain*.

O Capítulo 3 compreende a metodologia de investigação (*Design Science Research*) e a metodologia de desenvolvimento de *software* (Ágil) adotadas durante a execução do projeto, explicitando as mesmas e como estas são implementadas.

O Capítulo 4 engloba a apresentação da plataforma utilizada para o desenvolvimento e a descrição de todos os processos inerentes à construção da aplicação, evidenciando todos os seus sistemas através de tabelas de requisitos, casos de uso, matrizes de rastreabilidade e diagramas de sequência.

No Capítulo 5, são demonstrados os testes de *software* utilizados para a validação da solução implementada, bem como os resultados e discussão dos mesmo.

Por último, no Capítulo 6, são salientadas as conclusões finais do trabalho realizado, passando pelas limitações encontradas ao longo de todo o percurso e realçando algumas sugestões de melhoria.

Capítulo 2

Revisão Literária

O presente capítulo aborda a revisão literária e a fundamentação teórica necessária para o decorrer deste projeto, retratada na Figura 2.1. Inicialmente é explorado o conceito de *E-voting*, evidenciando os intervenientes e requisitos mais importantes e relevantes. São, ainda, referidas as vantagens e desvantagens deste tipo de sistemas, bem como exemplificados alguns sistemas de *e-voting*. De seguida, é exposto o conceito de *Blockchain*, analisando as suas principais características e estrutura. Adicionalmente, são expostas algumas temáticas fundamentais para o bom entendimento de uma *blockchain* e a relevância que a mesma tem para este projeto. Por último, são enunciadas algumas soluções existentes atualmente no mercado face à combinação da temática de *E-voting* e *Blockchain* e as suas especificidades.

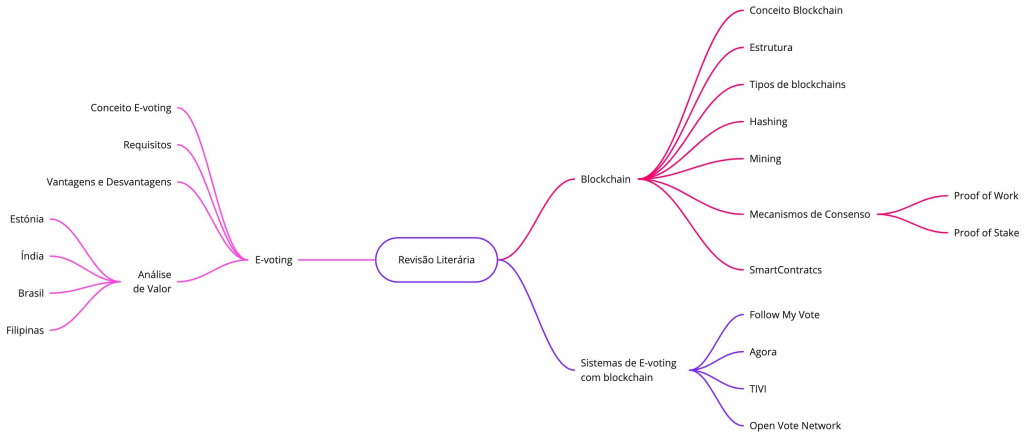


Figura 2.1: Estrutura da Revisão Literária

2.1 *E-Voting*

Esta secção aborda a temática de sistemas de votação eletrónica, evidenciando os seus intervenientes e fases de processamento, bem como os requisitos necessários para um bom funcionamento. Adicionalmente, são explicitadas as suas vantagens e desvantagens, tal como exemplificados quatro casos de sucesso de implementação.

2.1.1 Conceito *E-voting*

O termo *Eletronic Voting*, ou abreviado *E-voting*, é utilizado para descrever um sistema de votação que recorre a equipamentos eletrónicos durante o processo eleitoral, como por exemplo no ato de votar ou contagem dos votos [10, 11].

Esta técnica pode ser aplicada num ambiente controlado, caracterizada por ocorrer num local específico e supervisionado, onde existem máquinas facilmente controláveis pelos operadores e pelos eleitores, ou, por oposição, num ambiente não controlado através de *kiosks* públicos ou da Internet [12].

A Figura 2.2 demonstra em que países o *e-voting* está a ser utilizado e em que modalidade. Dos 178 países em estudo: 25 utilizam *e-voting* num ambiente controlado, entre os quais está o Brasil, Bélgica e Índia; 6 utilizam *e-voting* num ambiente não controlado, entre os quais está a Estónia e Austrália; 3 utilizam *e-voting* num ambiente controlado e não controlado (Canada, México e Mongólia); e os restantes ou não existem dados ou não recorrem ao *e-voting*.

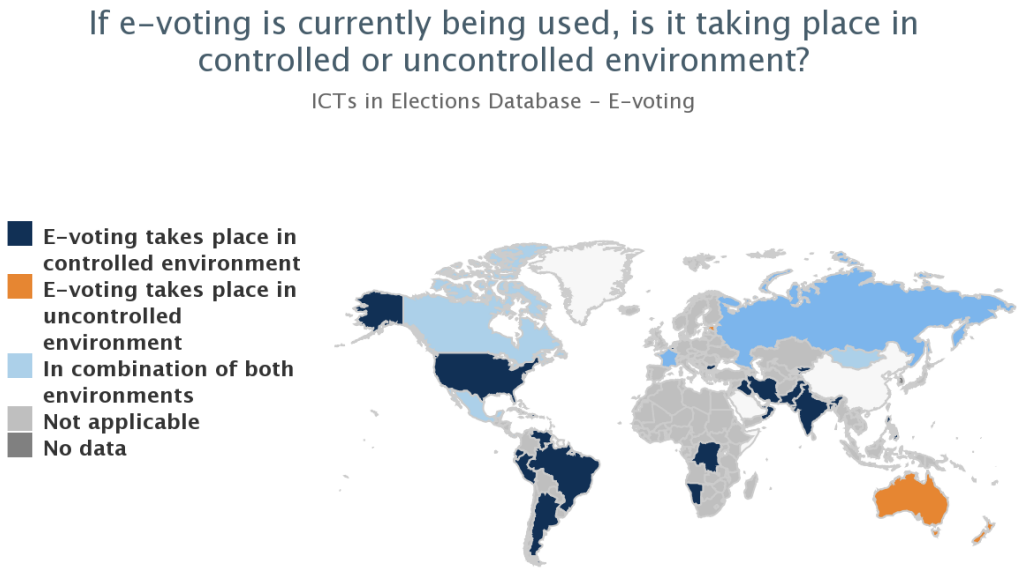


Figura 2.2: Utilização de *E-voting* num ambiente controlado e não controlado [13]

Esta opção tem sido considerada cada vez mais atrativa nas últimas décadas não só pelo impacto que tem no ambiente, mas também na rapidez com que são obtidos os resultados e na redução de eventuais erros [10, 11]. No caso do voto ser realizado através da Internet, o eleitor não é obrigado a deslocar-se, podendo fazê-lo recorrendo apenas a um dispositivo digital dentro do conforto da sua casa.

Essencialmente existem três intervenientes num processo de *e-voting*, independente sistema de *e-voting* e protocolo a implementar: o eleitor, autoridades responsáveis pelo processo de registo dos eleitores, e autoridades responsáveis pela contagem dos votos [14]. O processo de registo de eleitores pode passar por registar, autentificar e autorizar, assegurando que somente os eleitores registados podem votar, e só podem fazê-lo apenas uma vez [14]. Por sua vez, as autoridades responsáveis pela contagem dos votos devem recolher os votos e, posteriormente, proceder à sua contagem [14].

De modo geral, é possível identificar quatro fases no processo de votação, ilustradas na Figura 2.3 [14, 15, 16]:

- Recenseamento: Dá-se a compilação da lista de eleitores aptos a participarem numa dada votação;
- Autenticação/Validação: As credenciais dos eleitores são verificadas, garantindo que apenas os eleitores que estão registados podem votar;
- Votação/Recolha de votos: Os eleitores previamente registados realizam o seu voto e os mesmo são recolhidos;
- Contagem: Procede-se à contagem dos votos e no final os resultados são anunciados.

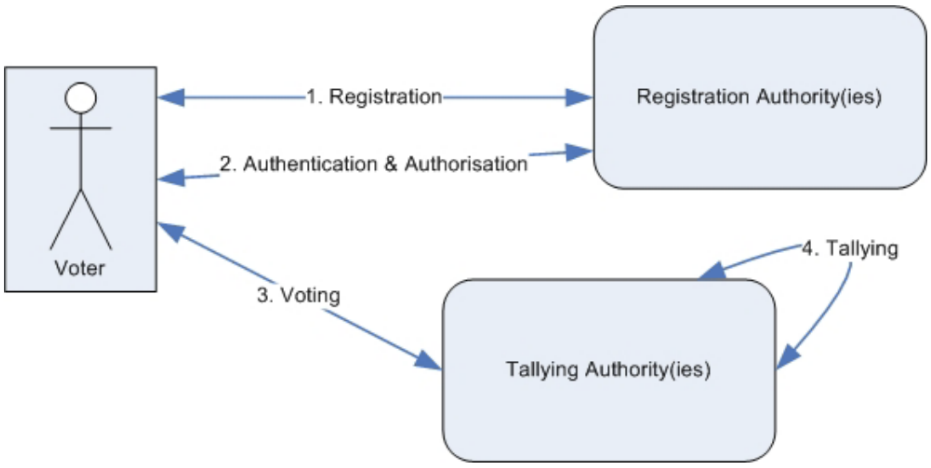


Figura 2.3: Processo típico de *e-voting* [14]

2.1.2 Requisitos

Tal como qualquer outro sistema, um sistema de *e-voting* tem que seguir alguns requisitos para manter um bom funcionamento [16]. Ainda que não haja um completo consenso de quais os princípios a seguir, o seu objetivo é aumentar a eficiência face aos sistemas de votação tradicionais, sem comprometer a segurança, privacidade ou os requisitos legais existentes [16].

Um dos requisitos mais importantes num sistema de *e-voting* é a elegibilidade, isto é, a possibilidade de apenas eleitores elegíveis poderem votar [14, 15, 17]. Para satisfazer esta condição, normalmente são implementados mecanismos de autenticação adequados a cada votação, como por exemplo o uso de impressão digital ou número de cartão de cidadão, algo que seja único a cada eleitor [16].

Durante todo o processo de votação não deve ser possível vincular um eleitor a um voto, sendo que a escolha feita por ele não deverá ser conhecida por outros [14, 15, 16]. Não obstante, o eleitor pode revelar a sua escolha se assim o decidir [16]. Este parâmetro não garante o anonimato, uma vez que na maioria dos casos é necessário autentificar um eleitor e assegurar a unicidade do seu voto [16]. Desta forma, sabe-se que o eleitor foi votar, mas a sua escolha é privada [16].

Relativamente ao voto por si só, este deverá ser único para cada eleitor [15]. Em alguns casos é assumido que o eleitor pode votar várias vezes, sendo que apenas um voto é contado, porém o mais comum é impedir o voto duplo, evitando ataques de clonagem de um voto anteriormente realizado por um eleitor autentificado [14, 16]. Como forma de evitar que um voto seja comprado, os eleitores não devem ser capazes de obter um recibo com o comprovativo do seu voto, e a decisão do eleitor deve recair somente nele mesmo [14, 17]. No entanto, é relevante permitir que o eleitor possa verificar se o seu voto foi contabilizado ou não [16].

Quanto à contagem dos votos, esta deve ser justa e precisa [14, 16]. Por um lado, todos os votos devem ser contados, desprezando os votos realizados por pessoas não autorizadas ou autentificadas, ou seja, apenas os votos válidos devem ser quantificados [14, 16]. Por outro lado, qualquer contagem parcial que seja realizada durante o processo de votação, não deverá ser revelada antes do final do período de votação, de modo a garantir que cada voto é justo e não é influenciado pelos resultados provisórios [14]. Nada impede que em alguns países sejam realizados inquéritos e pesquisas às saídas dos locais de votação [16].

Todo o processo de votação deve ser transparente e comunicado a todas as partes intervenientes e o sistema utilizado deve ser robusto ao ponto de ser capaz de lidar com falhas que possam acontecer, quer sejam falhas técnicas, por exemplo uma máquina de contagem de votos comprometida, quer por ação humana, por exemplo a tentativa de registo de eleitores inelegíveis [14, 16, 17].

2.1.3 Vantagens e desvantagens

Como os sistemas de *e-voting* recorrem a meios eletrónicos para votar e/ou fazer a contagem dos votos, apresentam as seguintes vantagens face aos sistemas de votação tradicionais [10, 15, 18]:

- O processo de contagem de votos é muito mais rápido e preciso;
- Maior comodidade para com os eleitores, pois permite que estes votem a partir de um local conveniente ao invés de uma assembleia de voto específica;
- Maior acessibilidade como por exemplo com a votação pela Internet através da utilização de boletins de voto em áudio para eleitores cegos, bem como para eleitores em casa e eleitores que estão fora do país;
- Possibilidade de utilizar *User Interfaces* com múltiplas linguagens que podem servir a um eleitor multilingue melhor do que as cédulas de papel;
- Potencial aumento da participação e comparecimento, principalmente com o uso de votação pela Internet;
- Eliminação da possibilidade de votos inválidos e duvidosos, visto que os sistemas de votação podem alertar os eleitores sobre um voto inválido;
- Prevenção de fraudes nas assembleias de voto e durante a transmissão e apuramento dos resultados, reduzindo a intervenção humana;
- Redução da quantidade de papel utilizado e consequentemente dos custos de impressão e distribuição dos boletins de voto;
- Maior sintonização com as necessidades de uma sociedade cada vez mais móvel.

Por outro lado, estes sistemas têm algumas desvantagens ou pontos fracos, entre as quais estão [18]:

- Falta de padrões acordados para sistemas de *e-voting* e para a sua certificação;
- Maiores requisitos de segurança para proteger o sistema de votação durante e entre as eleições, inclusive durante o transporte, armazenamento e manutenção;
- Possibilidade de recontagem dos votos limitadas;
- Necessidade de campanhas adicionais de educação do eleitor;
- Possível conflito com a estrutura legal existente;
- Possível falta de confiança do público em eleições baseadas em *e-voting*;

- Aumento dos custos de compra e manutenção de equipamentos dos sistemas de *e-voting*;
- Risco de manipulação ou fraude por pessoas com acesso privilegiado ao sistema;
- Falta de transparência;
- Potencial violação do sigilo do voto, principalmente em sistemas que realizam tanto a autenticação do eleitor quanto a votação.

2.1.4 Análise de valor - Alguns exemplos

Ainda que este conceito tenha surgido há cerca de 30 anos, existem relativamente poucos casos de sucesso que resultaram numa futura implementação [10]. Na presente dissertação, irá ser apresentado o caso da Estónia, Índia, Filipinas e Brasil.

Partindo dos países onde o *e-voting* foi testado, mas não foi implementado (como por exemplo Holanda, Reino Unido, Estado Unidos, Alemanha e Noruega), verifica-se que as razões pelas quais este processo não foi aceite focam-se fundamentalmente na falta de confiança pública e preocupação face à segurança dos sistemas de votação eletrónica [10].

Estónia

O sistema de *e-voting* foi implementado na Estónia pela primeira vez num projeto piloto que começou em 2003 [10]. Em 2007, depois de já ter sido utilizado nas eleições do conselho local (2005), o parlamento estoniano aprovou que o mesmo fosse usado nas eleições parlamentares, passando, assim, a ser o primeiro país a implementar um sistema de *e-voting* recorrendo à Internet [10, 19, 20].

Este sistema tinha como objetivo aumentar a participação nas eleições, tornando o seu processo mais fácil para os eleitores, ainda que fosse um método complementar aos sistemas tradicionais de votação [19]. Na Figura 2.4 é possível observar a evolução da afluência de eleitores de modo geral e através de *e-voting*.

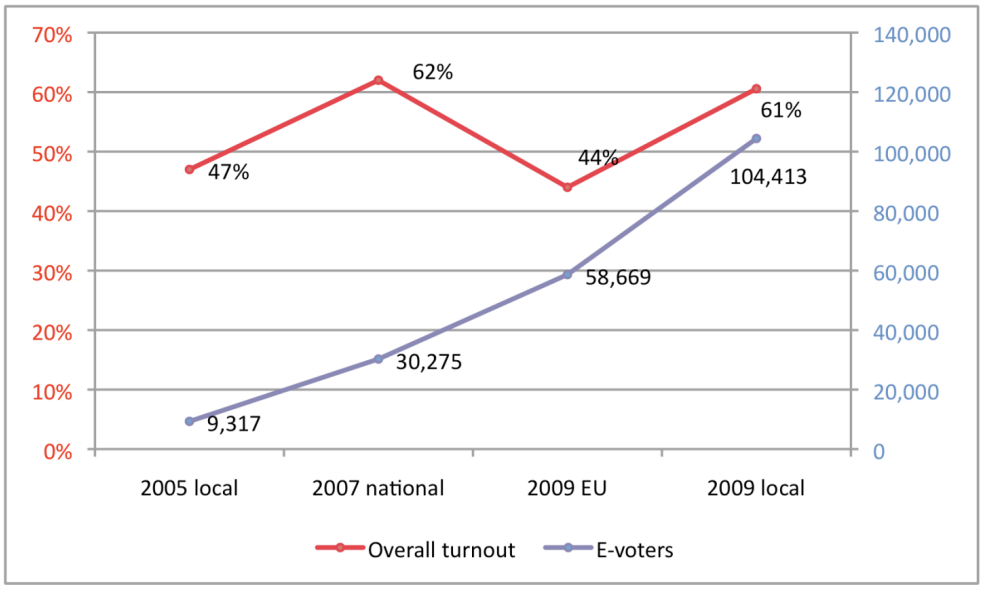


Figura 2.4: Afluência e o número de eleitores através da Internet [20]

O sistema de *e-voting* permitia que os eleitores elegíveis votassem a partir de qualquer lugar no mundo, desde que utilizassem um computador com conexão à Internet [19].

Para votar é necessário que o eleitor faça o *download* de uma aplicação responsável por encriptar o seu voto (E-voter) e um leitor de cartões [19, 20]. O Eleitor insere o seu cartão de identificação no leitor de cartões e acede ao site da Comissão Nacional Eleitoral que lhe permitirá votar [19, 20]. De seguida é verificada a sua identidade, sendo necessário fornecer um código PIN associado ao seu cartão de identificação [19, 20]. Na eventualidade do eleitor ser elegível, ser-lhe-á apresentado a lista de candidatos da respetiva votação e o eleitor faz a sua escolha [19, 20]. Neste momento é necessário confirmar a escolha do eleitor recorrendo a uma assinatura digital utilizando o segundo código PIN do seu cartão de identificação [19, 20]. O voto encriptado funciona como um envelope anónimo dentro de outro envelope que contém os dados pessoais do eleitor [19]. Posto isto, o eleitor recebe uma confirmação com a informação de que o seu voto foi executado com sucesso [19]. Quando se dá o processo de contagem os votos, as assinaturas digitais são removidas e os elementos do Comissão Nacional Eleitoral da Estónia podem abrir os votos anónimos para contagem, como é possível observar na Figura 2.5 [19].

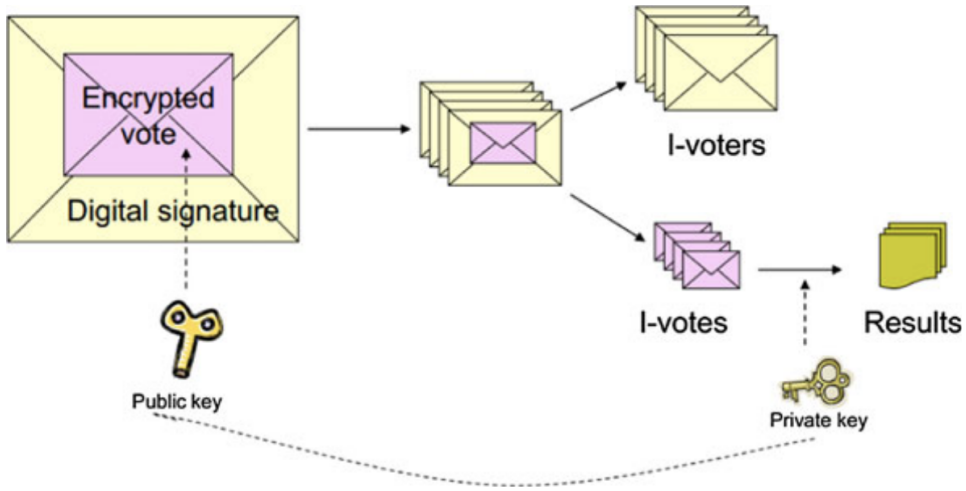


Figura 2.5: O método do envelope [19]

A escolha do eleitor é encriptada com a chave pública do sistema pela aplicação E-voter, os votos são recolhidos e a elegibilidade do eleitor é verificada [19]. No caso de existirem votos duplos ou de eleitores que não lhes é permitido votar, os votos são removidos e não entram no processo de contagem [19]. Posteriormente, os envelopes duplos são separados: os envelopes internos (votos encriptados) são encaminhados ao contador de votos que possui a chave privada do sistema; e os envelopes externos são utilizados para compilar as listas de eleitores [19].

Desta forma, só é permitido realizar o *e-voting* entre o décimo e o quarto dia antes do dia da eleição, uma vez que é necessário enviar as listas de eleitores que recorreram ao *e-voting*, garantindo que os mesmos não votem uma segunda vez [19]. Adicionalmente, o sistema permite ainda que o voto eletrónico de um eleitor seja alterado, quer seja por *e-voting* novamente ou dirigindo-se a uma estação de voto nesse mesmo período de tempo, pelo que o seu voto eletrónico é eliminado [19].

A partir do dia da eleição o voto registado com o processo de *e-voting* passa a definitivo, não podendo ser alterado ou anulado [19]. Após a contagem dos votos terminar, os resultados são anunciados publicamente no site da Comissão Nacional Eleitoral [19].

Índia

Em 1982 na Índia ocorreu um projeto-piloto de votação eletrónica que foi cancelado, pois não estava de acordo com a lei [10]. Após a lei ter sido alterada, a experiência foi retomada em 16 estados em 1989-1990, originando a implementação do *e-voting* em todos os estados da Índia em 2003 [10]. Nas eleições de 2014 verificou-se 814 milhões de eleitores registados e cerca de 930 mil estações de voto com urnas eletrónicas [10].

Inicialmente, o sistema de *e-voting* da Índia era composto por duas unidades: uma máquina de voto, disposta na cabine onde era efetuado o voto; e uma unidade

de controlo que estaria ao encargo do oficial de votação [10]. Porém, a partir de 2014 foi adicionada uma máquina denominada de Voter Verifiable Paper Audit Trail (VVPAT), capaz de imprimir os boletins eleitorais preenchidos pelos eleitores, para serem contados manualmente se assim for desejado [10].

É de notar que os equipamentos utilizados não permitiam o acesso à Internet, nem dependiam da energia elétrica, sendo alimentados por uma bateria [10].

Brasil

Em 2000, após ter sido realizado um estudo de viabilidade, o Supremo Tribunal Eleitoral do Brasil aprovou uma lei que instituía o uso de sistemas de *e-voting*, tendo um exemplo destes sistemas sido implementado parcialmente em 2006 pela comissão eleitoral brasileira na cidade de Santa Catarina [10].

Para este efeito, foram distribuídas cerca de 400 000 kiosks por centros urbanos, bem como urnas eletrónicas integradas com teclados e telas sensíveis ao toque que verificavam a identidade do eleitor e permitiam que ele votasse [10, 21]. Estas máquinas não tinham qualquer ligação à Internet e operavam no dia da votação entre as 8h e as 17h, a partir da qual encerrariam e não eram aceites mais votos [21].

Nesta implementação foram utilizadas máquinas VVPAT para averiguar e detetar possíveis ocorrências durante todo o processo, porém não se verificaram problemas significativos [10].

Filipinas

Em 2010, a Comissão Eleitoral das Filipinas (COMELEC) transmitiu as informações necessárias para que fosse possível implementar uma votação eletrónica nas eleições gerais da Filipinas, através do uso de máquinas chamada PCOS [10].

Estas máquinas eram responsáveis por fornecer aos eleitores uma célula física para estes poderem efetuar o seu voto que, mais tarde, seria inserida numa segunda máquina que lia os boletins de voto [10]. Se por algum motivo a inserção do voto corresse mal, o eleitor teria ainda mais 3 tentativas, antes do seu voto ser rejeitado [22]. No entanto, na maioria dos casos, as células foram aceites na primeira tentativa [22].

Quando as urnas eram encerradas, as máquinas locais transmitiam os respetivos resultados a uma autoridade municipal, que por sua vez, compilava os resultados recebidos e transmitia-os à divisão hierárquica superior até chegar ao servidor central responsável pelos resultados a nível nacional [23].

A Tabela 2.1 resume as tecnologias de *e-voting* utilizadas nestes países.

Tabela 2.1: Comparação das tecnologias de *e-voting* utilizadas na Índia, Brasil, Filipinas e Estónia [10]

Countries → / Technology ↓	**India**	**Brazil**	**Philippines**	**Estonia**
Hardware	EVM (Electronic Voting Machine)	GX-1 Integrated Processor	PCOS	None / Gadget voters /internet voting
Paper Audit Trail	VVPAT Machine	VVPAT Machine	Yes (conventional ballot)	Yes (Digital Receipt)
Internet connection	None	None	Yes (only for counting)	Yes
Wi-Fi / USB	None	None	Yes (only for counting)	Yes
Power	Battery	battery	Battery and electric	Battery and electric
Result	Success, No problem	Success, No problem	Success but many negative comments / claims from the public	Success but many negative comments / claims from the public

2.2 *Blockchain*

> "Blockchain makes it technically possible to ensure the integrity of information and the process of exchanging it." ([24], pág. 3)

O conceito "*blockchain*" surge aplicado pela primeira vez em 2008 com a criação da criptomoeda Bitcoin, uma moeda digital totalmente descentralizada [3], ainda que o conceito remonte a 1991 num estudo levado a cabo por Stuart Haber e W. Scott Stornetta sobre uma cadeia de blocos criptograficamente segura [1]. O *whitepaper* escrito pelo pseudónimo Satoshi Nakamoto (2008) definia uma *blockchain* como um sistema puro peer-to-peer, P2P, de dinheiro eletrónico, em que o mesmo poderia ser enviado da parte A para a parte B sem ter uma instituição financeira a intermediar o processo e assim eliminar gastos desnecessários [25].

2.2.1 Conceito Blockchain

Uma *blockchain* pode ser considerada como um *ledger*, isto é, uma espécie de livro de registos público, onde todas as transações são armazenadas sob a forma de uma lista de blocos ordenadas cronologicamente [26, 27]. Esta cadeia de blocos vai aumentando consoante a necessidade e à medida que são anexadas novas transações, permitindo

que qualquer bloco que é criado se ligue de forma irrefutável ao anterior e ao próximo [26, 28].

Tal como as outras Distributed Ledger Technologies (DLT), uma *blockchain* permite, no fundo, que partes sem qualquer relação de confiança entre si troquem dados digitais dos mais variados tipos (por exemplo: registos médicos, contratos, apólices de seguro, dinheiro, certificados de nascimento ou casamento, bens, serviços, etc.) sem qualquer intermediário ou terceiros [28].

Quando uma transação é solicitada entre duas partes, a mesma é transmitida para uma rede distribuída de nós (*nodes*) para ser validada segundo um mecanismo de consenso, isto é, um conjunto de regras acordadas entre os nós [28]. Posteriormente, a transação, agora aprovada, é agrupada juntamente com outras transações num bloco e adicionada à *blockchain*, garantindo que este bloco está vinculado tanto ao bloco anterior como ao seguinte, formando, assim, uma cadeia de blocos (*Block*-bloco; *Chain*-cadeia). Finalmente, estes registos são compartilhados com todos os nós ou computadores da rede, estando sujeitos a atualizações constantes e sincronizações entre si.

A Figura 2.6 demonstra o processo de uma transação anteriormente enunciado.

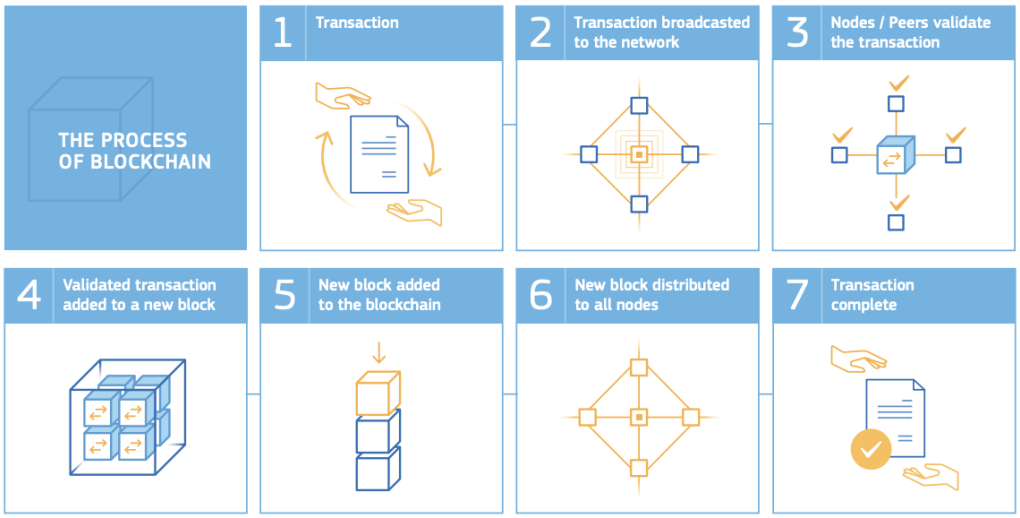


Figura 2.6: Processo de uma transação numa *blockchain* [28]

A tecnologia *blockchain* é também um sistema **descentralizado**, uma vez que os diferentes nós da rede nunca estão todos ligados diretamente entre si, ao contrário do que acontece nos sistemas tradicionais onde, por natureza, existe um nó central [4]. Dessa forma, cada participante da rede tem acesso a toda a base de dados e mantém o seu próprio registo através do processamento de um bloco, e só recorrendo a um mecanismo de consenso é que esse grupo de transações é validado ou não, sendo, consequentemente, atualizado em toda a rede [26, 27].

O facto de não existir uma autoridade central que controla o *ledger*, existindo, portanto, múltiplos nós, permite custos de manutenção menores, processamento mais rápido e maior estabilidade do sistema e dos processos inerentes ao mesmo, pois é mais difícil danificar a maioria dos nós ou até mesmo a rede inteira [4, 28]. A Figura 2.7 transmite a diferença das ligações entre os nós presentes num sistema centralizado por oposição a um sistema descentralizado.

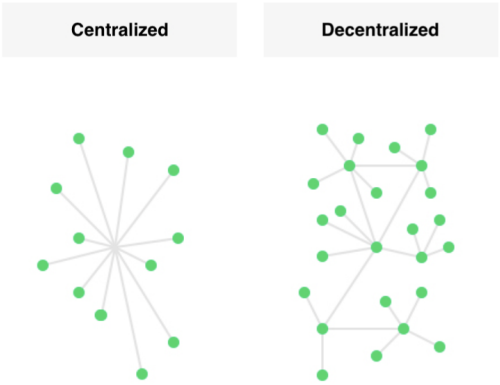


Figura 2.7: Sistemas centralizados vs. Sistemas descentralizados [4]

De um modo geral, a *blockchain* é **imutável**, ou seja, depois de uma transação ser adicionada na mesma, não poderá ser desfeita, o que confere confiabilidade a todos os intervenientes [27]. Esta capacidade deve-se à criptografia utilizada que vincula o bloco a ser adicionado ao bloco anterior, através de um *hash* que contém a informação dos blocos e dos respetivos *hash* anteriores [29]. Na tentativa de adulteração, o *hash* produzido será também alterado, invalidando a rede de informação [27, 29]. Como todos os nós possuem uma cópia da base de dados, nenhuma alteração é passada despercebida [28].

No entanto, embora raro e extremamente difícil, é possível que a *blockchain* seja comprometida se existir um número elevado de nós que se agrupem para tal efeito [27]. Tal acontecimento já se verificou em 2016, quando os fundos (cerca de 70 milhões de dólares) de uma angariação de dinheiro na rede Ethereum foram desviados, devido a uma vulnerabilidade no código base [28, 30]. Posto isto, parte dos participantes da *blockchain* Ethereum acordaram entre si e reenviaram os fundos desviados que, entretanto, foram bloqueados numa conta, aos seus proprietários originais, através de um *hard fork*, alteração radical no protocolo da respetiva *blockchain* [30, 31]. Esta decisão controversa originou a bifurcação em duas redes separadas: uma com o novo protocolo, Ethereum (ETH); e outra contendo o protocolo antigo, Ethereum Classic (ETC) [31] .

Em contrapartida, num *soft fork* dá-se na mesma a alteração do protocolo, porém todos os participantes aceitam as alterações efetuadas e migram para a rede com o

novo protocolo, visto que apenas uma das redes passa a ser válida [32]. A Figura 2.8 demonstra a diferença entre *hard* e *soft fork*.

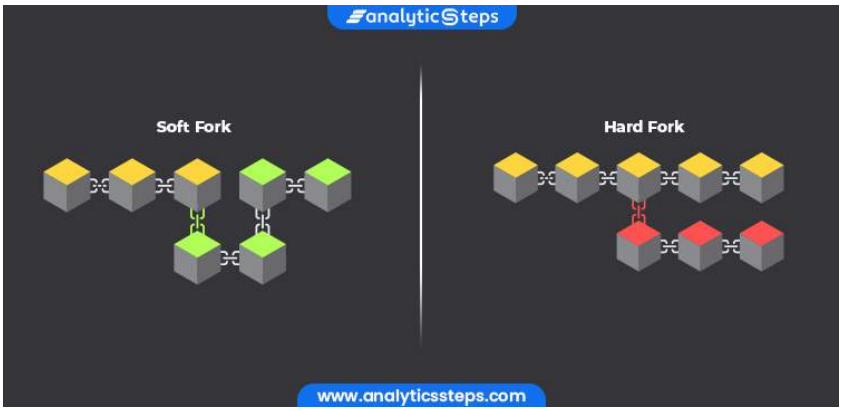


Figura 2.8: Hard Fork vs Soft Fork [33]

Dado que as transações são visíveis a qualquer pessoa com acesso ao sistema, pode afirmar-se que existe **transparência** [29]. No caso de ser uma *blockchain* privada, o acesso está condicionado a certos participantes, porém numa *blockchain* pública, todos os intervenientes têm os mesmos direitos de acesso e/ou atualização do *legder* dependendo do mecanismo de consenso imposto, aumentando, assim, a auditabilidade e a confiança na rede [28]. Na eventualidade de existirem dados que não deveriam estar acessíveis publicamente ou necessitam de serem alterados a posteriori, o facto de existir transparência deixa de ser vantajoso [28].

Face a este problema, uma solução possível seria o armazenamento dos dados confidenciais fora dessa *blockchain*, sendo possível armazenar uma maior quantidade de dados, onde os mesmos estariam vinculados à *blockchain* através de *hashes* restritos apenas a partes autorizadas [28]. Adicionalmente, pode recorrer-se a uma *blockchain* privada ou híbrida, onde é possível configurar diferentes níveis de acesso a informações pessoais com base na diferenciação entre participantes [27, 28].

No que diz respeito ao **anonimato/privacidade**, qualquer utilizador que interaja com a *blockchain*, fá-lo através de um endereço e, dado que as transações são visíveis pela rede, as contas dos utilizadores não são anónimas, ainda que a sua identidade seja [26, 27]. Assim, as *blockchains* públicas tendem a utilizar **pseudónimos**, visto que as contas dos utilizadores não requerem, no momento da sua criação, de identificação ou autorização, sendo meramente identificadas pelos seus endereços [27]. Por outro lado, as *blockchains permissioned* podem exigir que a identidade do utilizador seja verificada aquando do seu acesso e utilização [27]. Tomando como exemplo a Bitcoin, qualquer pessoa pode transferir Bitcoin para outras pessoas, através das chaves públicas e privadas, sem qualquer informação pessoal [28].

Cada utilizador possui um par de chave privada e chave pública, conhecidas como assinaturas digitais, às quais se recorre em dois momentos: na assinatura e

na verificação [26, 29]. A primeira é utilizada para assinar as transações que, mais tarde, serão transmitidas por toda a rede, pelo que deve ser mantida em segredo [26]. A chave pública é usada para verificar a veracidade da transação [29]. Por exemplo, pretende efetuar-se uma transação entre duas pessoas. Para assinar esta transação, dispõe-se de um algoritmo que, através da chave privada do remetente e da transação a efetuar, gera uma assinatura digital [26, 29]. Normalmente, o algoritmo de assinatura digital utilizado numa blockchain é o Elliptic Curve Digital Signature Algorithm (ECDSA) [26, 29]. Após a encriptação, tanto a transação original como a assinatura digital são enviadas para o recetor [26, 29]. Numa fase de verificação, o recetor descodifica a transação recorrendo à chave pública do remetente e compara-a com a transação original, permitindo verificar a veracidade e integridade da mesma, bem como comprovar o autor da transação [4, 26, 29].

2.2.2 Estrutura

Como referido anteriormente, uma *blockchain* é composta por blocos de informação interligados sequencialmente entre si (Figura 2.9), em que cada bloco (Figura 2.10) tem um cabeçalho constituído por [3, 26, 29]:

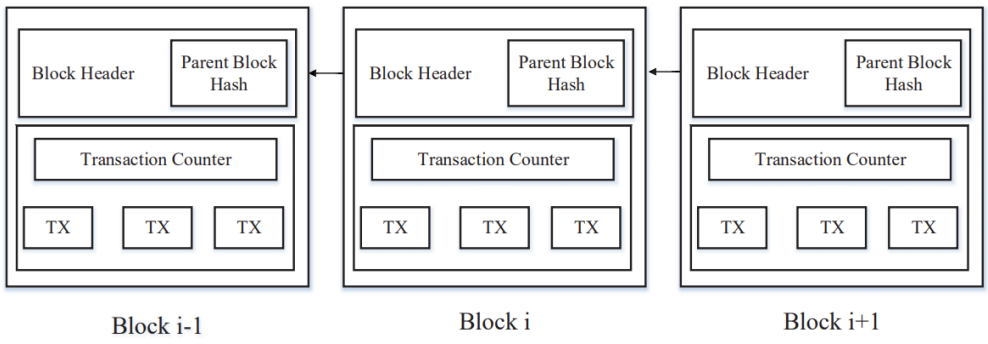


Figura 2.9: Estrutura da cadeia de blocos (*blockchain*) [26]

- *Block Version*: Indica a versão do protocolo utilizada pelo bloco;
- *Merkle Tree Root Hash*: Valor de *hash* de todas as transações validas presentes no bloco;
- *Timestamp*: Data e hora de criação do bloco;
- *NBits*: Corresponde ao nível de dificuldade base do sistema no momento em que o bloco foi proposto;
- *Number Only Used Once* (NONCE): Número único utilizado para a resolução do desafio criptográfico;
- *Parent Block Hash*: *Hash value* do bloco anterior.

O corpo do bloco consiste em diversas transações, número este que depende do tamanho do bloco e do tamanho de cada transação [26].

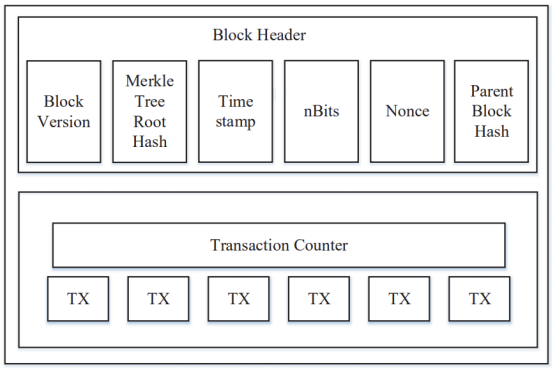


Figura 2.10: Arquitetura interna de um bloco [26]

2.2.3 Tipos de *Blockchains*

Tendo em consideração a infraestrutura onde a cadeia de blocos é implantada e permissão de acesso, podem existir vários tipos de *blockchains*, encontrando-se resumidos na Tabela 2.2.

Por um lado, ao analisar o nível de autorização à rede, pode fazer-se a distinção em dois tipos [27, 29]:

- *Permissioned*: É necessário que uma autoridade central forneça aos utilizadores permissões para ler, guardar ou validar novas transações a serem inseridas na cadeia e, como nem todos os participantes têm permissão para guardar dados, o consenso na rede é obtido mais rapidamente;
- *Permissionless*: Qualquer pessoa pode entrar, ler, guardar e validar transações, pelo que é necessário que o sistema seja descentralizado. Desta forma, é fundamental existir transparência e confiança na rede, uma vez que qualquer informação pode ser validada desde que o algoritmo de consenso o permita.

Por outro lado, se for tido em conta a abertura da plataforma, a rede pode ser classificada em [3, 27, 29]:

- Pública: Rede aberta a todos os participantes, onde não é necessário pedir permissão para participar na mesma. Qualquer pessoa pode fazer *download* do código e correr em qualquer nó público no seu dispositivo pessoal e, por conseguinte, decidir se querem, ou não, adicionar blocos à cadeia. Esta abordagem está presente na maioria dos sistemas de cripto moedas (como por exemplo Bitcoin e Ethereum), visto que existem inúmeros nós e não necessário conhecer a identidade real dos participantes;

- Privada: Rede fechada num ambiente controlado, com um número reduzido de nós, onde é possível identificar os participantes da rede e verifica-se um maior nível de centralização, pelo que quem mandar na *blockchain* pode a qualquer altura fazer alterações;
- Consórcio: É a combinação entre uma rede pública e privada, podendo existir um acesso público ou restrito a certos participantes, no entanto mantém algum nível de centralização.

Tabela 2.2: Tipos de blockchains [27]

			READ	WRITE	COMMIT	EXAMPLE
BLOCKCHAIN TYPES	OPEN	Public permissionless	Open to anyone	Anyone	Anyone	Bitcoin, Ethereum
		Public permissioned	Open to anyone	Authorised participants	All or subset of authorised participants	Supply chain ledger for retail brand viewable by public
	CLOSED	Consortium	Restricted to an authorised set of participants	Authorised participants	All or subset of authorised participants	Multiple banks operating a shared ledger
		Private permissioned "enterprise"	Fully private or restricted to a limited set of authorised nodes	Network operator only	Network operator only	External bank ledger shared between parent company and subsidiaries

2.2.4 *Hashing*

De modo a garantir a integridade dos *ledgers*, isto é, a capacidade de detetar adulterações dos dados, as *blockchains* utilizam técnicas criptográficas, promovendo a confiança nestes sistemas [29]. Aqui surge o conceito de função *hash* ou *hashing*, que consiste numa função matemática capaz de transformar qualquer informação, independentemente do seu tamanho, numa string de tamanho fixo [3, 28]. O output obtido é denominado de *hash value* ou somente *hash*, é, basicamente, uma impressão digital, exclusiva para cada parte de dados da *blockchain* [27].

A função é considerada livre de colisões, ou seja, para duas variáveis diferentes a probabilidade de estas originarem o mesmo *hash* é nula, conferindo veracidade às transações [4]. Se o input da função *hash* for o mesmo, a saída produzida por ela também será a mesma, independentemente do número de vezes que é executada [27, 28]. Em contrapartida, se se alterar a entrada de dados desta função, o *hash* obtido será um conjunto de caracteres completamente diferente [27, 28]. Ainda assim, qualquer *hash* produzido não revela nenhuma informação sobre os dados de entrada que o originaram [28].

Na Tabela 2.3, é possível observar alguns exemplos de *hashes*, recorrendo ao Secure Hash Algorithm (SHA 256) utilizado pela Bitcoin. A pequena diferença entre uma letra minúscula e maiúscula, produz um *hash* completamente diferente.

Tabela 2.3: Exemplos de encriptação (SHA-256)

Blockchain	625DA44E4EAF58D61CF048D168AA6F5E492DEA166D8BB54EC06C30DE07DB57E1
Hello	185F8DB32271FE25F561A6FC938B2E264306EC304EDA518007D1764826381969
hello	2CF24DBA5FB0A30E26E83B2AC5B9E29E1B161E5C1FA7425E73043362938B9824

A função *hash* surge também interligada à chave pública de um utilizador, sendo o *hash* produzido representante de um endereço [3, 28]. Este endereço, semelhante a um número de uma conta bancária, remete para uma carteira digital, através da qual o utilizador poderá interagir com a *blockchain*, que contém a chave privada do utilizador com a qual as transações serão assinadas [3]. Para validar e desencriptar uma transação é utilizada a chave-pública corresponde, garantindo que só as pessoas para quem a transação era destinada, podem ler e processar a informação enviada [3, 4].

Atualmente, a carteira mais popular é o MetaMask, uma carteira digital sob a forma de uma extensão no *browser* que proporciona interfaces de comunicação com aplicações descentralizadas (DApps) na rede Ethereum, permitindo que o utilizador as execute sem ter a necessidade de possuir um *software* especializado para esse efeito [3].

Paralelamente ao conceito de *hashing*, surge o conceito de *Merkle Tree* fundamental para manter a integridade do sistema, cujo resultado da mesma (*Merkle Root Hash*, previamente mencionado) será posteriormente inserido em cada bloco de uma *blockchain* [4]. Representa o *top hash*, correspondente ao *hash* da arvóre inteira, advindo da concatenação das suas ramificações, ou seja, uma *Merkle Tree* é representada por um único *hash*, apesar de possuírem várias ramificações/*hashes* associados [29]. Estas ramificações retratam as transações presentes num bloco e são agrupadas aos pares [4]. Ora, na eventualidade de existir um número ímpar de entradas de dados, a última entrada é copiada e associada com ela mesma [4].

Se um bloco for composto por 106 transações, as mesmas serão agrupadas em 54 pares (a última transação é agrupada com a sua cópia) e obtido os 54 *hashes* correspondentes. Por sua vez, estes são agrupados em 28 pares e são novamente produzidos os *hashes* correspondes. Este processo é repetido ($28 > 14 > 7$ (8) $> 4 > 2 > 1$) até se alcançar um único *hash*, *Merkle Tree Root Hash* [4].

Na seguinte figura (Figura 2.11) está exemplificada uma *Merkle Tree*.

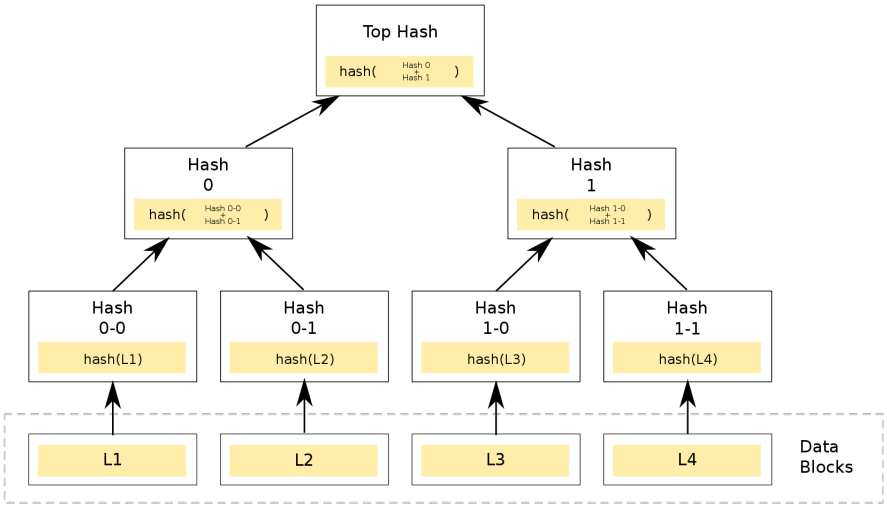


Figura 2.11: Exemplo de uma Merkle Tree [29]

2.2.5 *Mining*

Mining é um processo através do qual conseguimos adicionar novos registos de transações ao *ledger* [27]. Por sua vez, os *miners* são nós na rede que garantem a validação das transações e que seguem o mecanismo de consenso escolhido pela plataforma para adicionar um novo bloco ao *ledger* [27].

Quando uma transação é efetuada, esta é enviada para toda a rede para ser verificada, juntamente com outras transações em todo o mundo [28]. Os *miners* agregam as transações pendentes num bloco com outras informações (data, tamanho do bloco e alguns dados aleatórios inseridos pelo *miner*) e recorre à função *hash* para determinar o *hash* desse bloco [28]. O propósito de um *miner* é obter um *hash* tendo em conta um número pré-definido de zeros, símbolo da dificuldade, através de um processo tentativa erro, onde o *miner* vai alterando alguns parâmetros dentro do bloco, nomeadamente o *Nonce* [28]. Quando o *hash* desejado é encontrado, a solução é transmitida para toda a rede que verifica se esta está correta e, após o bloco ser considerado válido, é adicionado no final da cadeia [28].

Como recompensa por encontrar o bloco correto, o *miner* recebe moedas novas que entraram apenas agora em circulação (neste momento um *miner* recebe 6,25 Bitcoins para minerar um novo bloco na rede Bitcoin) e todas as taxas de transação incluídas nas transações desse bloco [28, 34].

2.2.6 Mecanismos de Consenso

Nenhum bloco pode ser adicionado ao *ledger* ser a aprovação de determinados *nodes* da rede. Os mecanismos pelo qual a *blockchain* atinge o consenso na validação e

verificação de transações, denominados de mecanismos de consenso, são essenciais para garantir a validação de cada bloco e ao mesmo tempo assegurar que todos os participantes estejam de acordo [4]. Trata-se do conjunto de regras a seguir para que haja um acordo entre os diferentes nós do *ledger* para a criação de blocos de transações na *blockchain* [4]. Existem diversos mecanismos de consenso, pelo que a escolha do mesmo passa pelo tipo de rede e dados pretendidos [4].

***Proof of Work* (PoW)**

O *Proof of Work* é o mecanismo de consenso onde os *miners* competem entre si para criarem novos blocos [3]. Este mecanismo, inicialmente proposto em 2002, foi adaptado em 2008 com a criação da Bitcoin, sendo, portanto, o algoritmo mais antigo [4].

Os *miners* tentam resolver um problema matemático, de modo a verificar a legitimidade das transações inseridas nos blocos [4, 35]. Quando é encontrada a solução do problema, o nó que o resolveu primeiro recebe uma recompensa e fica responsável de transmitir a toda a rede a solução, e inserir o novo bloco [4, 29].

Porém, é um mecanismo que necessita de um grande poder computacional e energético, reduzindo o incentivo de participação numa *blockchain* com este mecanismo [4, 35].

***Proof of Stake* (PoS)**

Contrariamente ao PoW, este mecanismo não depende do poder computacional nem da realização de cálculos matemáticos por parte dos *miners* [35]. Funciona como um sorteio, no qual o sistema escolhe qual o nó que criará um novo bloco com base numa probabilidade que reflete a quantidade de moedas que esse mesmo nó possui. Por exemplo, se um nó contém 10% de todas as moedas em circulação, ele tem também 10% de probabilidade de minerar o próximo bloco [4, 28].

No entanto, este mecanismo leva a uma concentração de riqueza, uma vez que quem possui mais moedas tem também uma maior probabilidade de criar novos blocos e, consequentemente, adquirir mais riqueza [35].

A *blockchain* Ethereum, que inicialmente adotou o PoW está neste momento a fazer a transição para o PoS, visto que é mais seguro, consume menos energia e está mais apto para implementar novas soluções de escalabilidade [26, 36].

2.2.7 *Smart Contract*

A Ethereum é uma plataforma *blockchain* que, para além de possuir como moeda principal o Ether, permite a criação de *smart contracts* para serem executados numa Ethereum Virtual Machine (EVM) como uma aplicação descentralizada (DApp) [3].

Os *smart contracts* são programas informáticos capazes de cumprir os termos de um contrato estabelecido entre diferentes partes [4, 29]. Quando as regras por

ele descritas e acordadas pelas partes que com ele irão interagir são cumpridas, o contrato será automaticamente aplicado [29]. Podem ser assim considerados auto verificáveis, auto executáveis e resistentes à adulteração, uma vez que depois de implementado não poderá ser alterado [29].

Para a criação de um *smart contract*, é necessária recorrer a uma linguagem de alto-nível orientada a contratos, Smart Contract Language (SCL), como por exemplo *Solidity*, e de acesso a uma plataforma capaz de compilar e efetuar o *deploy* do contrato, isto é a implantação do mesmo [3, 29].

Para além de permitir que várias partes realizem uma transação confiável sem a necessidade de intermediários, um *smart contract* oferece atualizações em tempo real [28, 29].

Tomando com exemplo uma máquina de venda automática, cujas regras de compra e venda de produtos são previamente programadas e verificadas ao longo de todo o processo de compra, a máquina irá executar um contracto programado a priori quando for selecionado um item e inseridas moedas [29]. Este contrato confirma se o dinheiro colocado é suficiente para o produto em questão e, só nesse caso libertaria o item escolhido, bem como verificava qual seria a quantia a devolver [29]. Este exemplo em específico demonstra a oferta de um serviço de disponibilidade absoluta [29].

2.3 Sistemas de *E-Voting* com *Blockchain*

A tecnologia *blockchain* pode ser aplicada nas mais variadas áreas, entre as quais no setor eleitoral, mais concretamente num sistema de *e-voting*. Esta aplicabilidade surge como tentativa de combater as fragilidades ou reforçar as caraterísticas dos sistemas atuais através de uma rede distribuída, descentralizada, imutável e transparente.

Existem várias propostas, algumas delas já consolidadas, para a utilização da *blockchain* num sistema de *e-voting* como urnas para armazenar os dados de uma votação.

As mesmas etapas de um sistema *e-voting*, isto é, recenseamento, autenticação, votação e contagem, aplicam-se quando o sistema utiliza tecnologia *blockchain* [37].

De modo geral, a arquitetura de um sistema de *e-voting* que utiliza tecnologia *blockchain* encontra-se retratada na Figura 2.12.

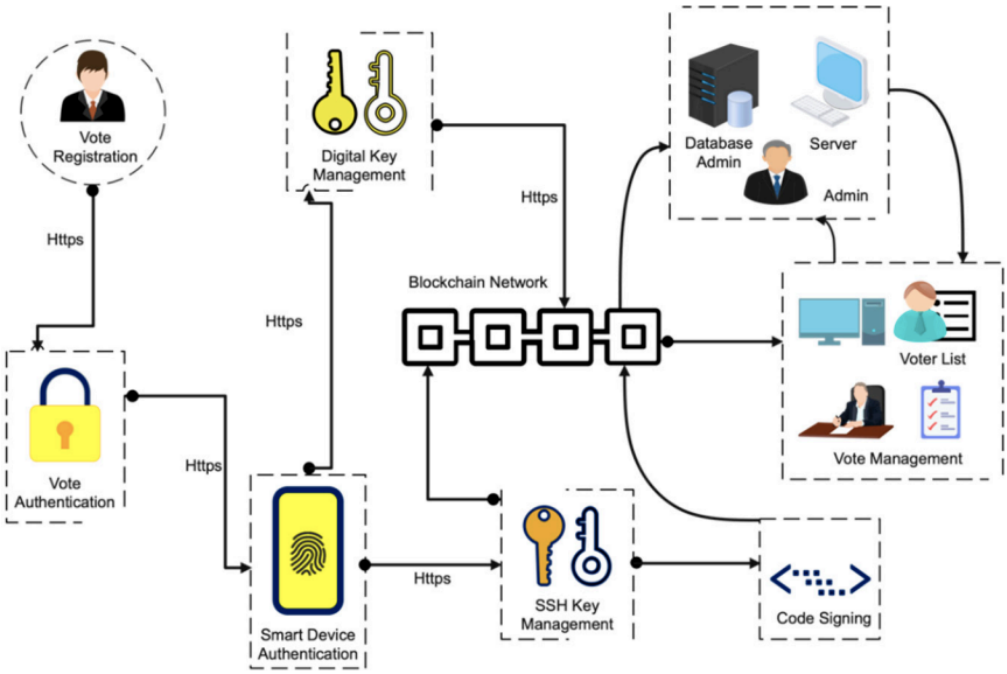


Figura 2.12: Arquitetura dos sistemas de votação com *blockchain* [37]

Qualquer voto registado numa *blockchain* será tratado como um dado encriptado que será armazenado publicamente nesta rede que, por si só é distribuída e composta por vários nós, ao invés de ficar apenas armazenado num único servidor [37].

Os votos encriptados são validados e adicionados ao *legder* através de um mecanismo de consenso, podendo ser verificados por qualquer elemento e não exclusivamente aos elementos da organização da votação [37].

Como a *blockchain* é um sistema descentralizado e público, assegurando na mesma que os eleitores estão protegidos, qualquer pessoa pode efetuar a contagem dos votos de uma respetiva votação, todavia, não consegue interligar um voto a um eleitor, isto é, não consegue determinar quem é que votou em quem [37].

De seguida, serão discutidas algumas implementações de sistemas de *e-voting* com recurso a uma *blockchain*.

2.3.1 Follow My Vote

Follow My Vote [38], é um sistema de e-voting com blockchain capaz de analisar o desenvolvimento de uma votação a tempo real [37, 39, 40].

Inicialmente, existe uma fase de recenseamento, na qual o eleitor tem que fornecer uma foto e o seu cartão de cidadão para serem verificados por uma autoridade confiável que determina se o mesmo é elegível ou não [39, 41, 42]. Posto isto, o eleitor pode votar remotamente.

Para manter o anonimato e autenticidade dos eleitores, o sistema utiliza criptografia através de *blind signature* e conjuntos de chaves públicas e privadas, em

que uma dela está ligada à identidade do eleitor [41, 43, 44]. Por sua vez, o eleitor pode utilizar a sua identificação para localizar o seu voto e verificar se o mesmo está correto [37, 39, 40, 44].

Este sistema possibilita ainda ao eleitor votar antecipadamente ou de efetuar múltiplos votos, desde que seja até à data limite da votação [42]. Em contrapartida, como os votos são realizados sem serem encriptados, podem existir vendas de votos ou coerção [39, 43, 44].

Para além disso, o cálculo dos resultados parciais e a possibilidade de alteração dos votos, advinda do facto da autoridade possuir todas as senhas dos eleitores, comprometendo a integridade dos votos, demonstram-se ser desvantagens [39, 43, 44].

Em 2016, como tentativa de demonstrar a eficácia do sistema, tentou realizar-se uma votação paralela a uma eleição presidencial, no entanto a equipa não conseguiu terminar o desenvolvimento do sistema a tempo do dia da eleição, falhando, assim, na demonstração [42].

2.3.2 Agora Voting

Agora [45] é uma plataforma de votação digital com *blockchain* criada em 2015, tendo sido implementada parcialmente em 2018 durante a eleição presidencial de Serra Leoa [37, 40].

Este processo dá-se em 6 etapas, ilustradas na Figura 2.13, que garantem segurança, transparência e que os resultados não sejam adulterados, ao mesmo tempo que permite que qualquer elemento verifique os dados da votação, mantendo a privacidade do eleitor [39, 40].

Primeiramente, o administrador da votação cria um ficheiro responsável pela configuração da mesma, onde contém algumas informações como os eleitores elegíveis, as datas de começo e término, a lista de candidatos, etc. [39, 45]. De seguida, os eleitores podem votar a partir do seu dispositivo pessoal ou em locais específicos, onde o voto é posteriormente encriptado [39, 45].

O anonimato dos eleitores garante-se na fase seguinte, na qual os votos são baralhados numa rede com esse propósito, evitando a possibilidade de coerção e compra de votos [39, 44].

Posto isto, os votos anónimos são desencriptados pelas autoridades competentes, gerando *Zero Knowledge Proofs* (ZKPs) para atestar se a desencriptação está correta [39, 45].

Os votos previamente desencriptados e agora expostos em texto simples são contados manualmente por uma entidade competente, ou caso seja permitido, o Agora faz a contagem dos votos automaticamente [39, 45]. Como os votos são públicos, qualquer pessoa pode realizar a contagem dos votos [45].

Por último, as pessoas designadas para tal verificam os resultados da votação em todas as suas fases e é então assinada uma declaração com a chave privada dos auditores, demonstrando que a votação ocorreu sem incidentes [39, 45].

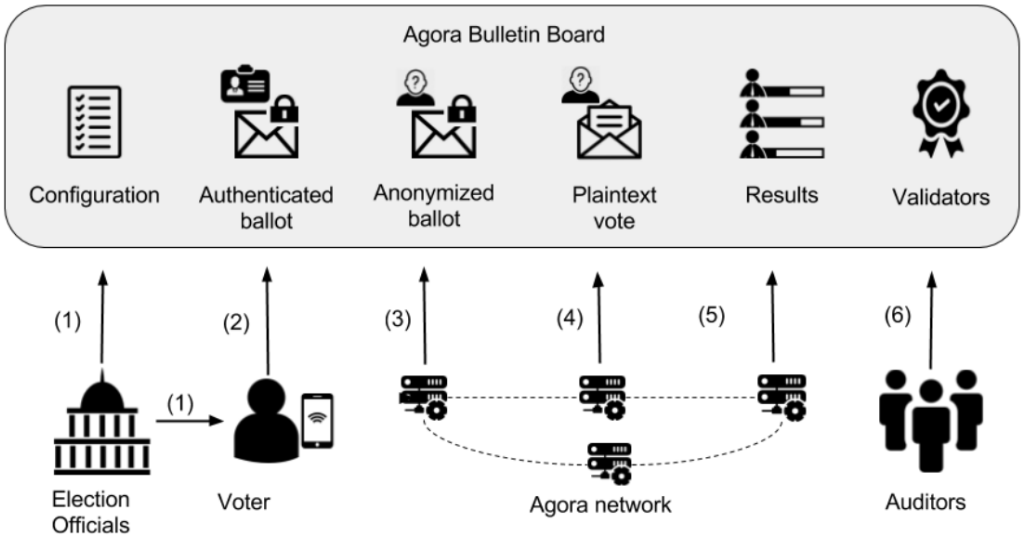


Figura 2.13: Processo de votação com o Agora [45]

2.3.3 TIVI

Desenvolvido pela empresa Smartmatic, TIVI [46, 47] é um sistema de *e-voting* com tecnologia *blockchain* baseado na autentificação biométrica. Oferece a flexibilidade e a conveniência de um voto ser efetuado através de um dispositivo digital, a partir de qualquer lugar no mundo [46]. De modo a verificar a identidade do eleitor, antes de votar, um eleitor necessita de submeter uma foto sua, que consequentemente será comparada com as informações obtidas na fase de registo dos eleitores por uma autoridade externa [39, 41].

A *blockchain* age como uma urna que armazena os dados relativamente ao processo de votação, mantendo a integridade dos votos [39, 41]. Contém, ainda, um sistema criptográfico capaz de desvincular as informações de identificação de um eleitor do seu voto encriptado utilizando um sistema de duplo envelope semelhante ao sistema de *e-voting* Estoniano [39, 41, 47].

Adicionalmente, permite que o eleitor confirme que o seu voto foi aceite e está de acordo com o que era pretendido através de um *QR code* único, fornecido no momento do voto, que é verificado por uma aplicação de verificação localizada no *smartphone* do eleitor [39, 47].

Por outro lado, este sistema inibe a possibilidade de múltiplos votos ou qualquer mecanismo com a finalidade de proteger os eleitores de coerção [39].

O TIVI foi utilizado com sucesso no Utah, Noruega e Chile, obtendo resultados positivos e uma grande adesão por parte dos eleitores [46].

2.3.4 Open Vote Network

O Open Vote Network, é um protocolo inicialmente apresentado por Patrick McCorry et. al. [48], destinado a um sistema de votação online descentralizado de auto contagem escrito como um *smart contract* na rede Ethereum [39, 41].

Este contrato está entregue a um administrador encarregue de configurar a eleição e autentificar os eleitores, garantindo que apenas os eleitores elegíveis podem votar [39].

Não necessita de uma autoridade responsável pelo processo de contagem dos votos e proteção do anonimato de um eleitor, uma vez que o próprio faz a contagem dos votos e a privacidade de um voto é controlada pelo seu eleitor correspondente e os mesmos são encriptados [39, 41]. Por sua vez, caso um eleitor queira verificar o seu voto, pode fazê-lo inspecionando a *blockchain* [39].

Em contrapartida, este sistema suporta apenas votações com duas opções, sim ou não, não é resistente à coerção, e está limitada a 50 eleitores, devido às ferramentas matemáticas que utiliza [39]. Tendo em conta a forma como é construído, impossibilita o cálculo de resultados parciais, portanto o resultado calculado corresponde ao resultado final calculado somente depois de todos os eleitores registados terem votado [39, 47]. A Figura 2.14 representa as etapas de uma votação implementadas neste protocolo.

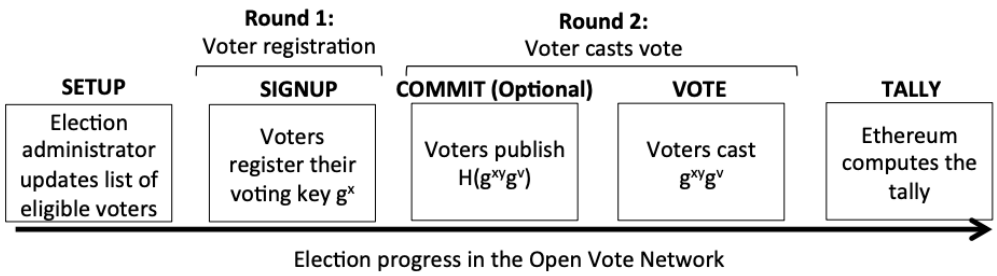


Figura 2.14: Processo de votação com o Open Vote Network [48]

A Tabela 2.4 retrata o culminar das características dos sistemas previamente abordados. Qualquer um destes sistemas está apenas apto para ser utilizado em pequena escala [37].

Tabela 2.4: Comparação entre os sistemas de *e-voting* baseados em *blockchain* analisados, adaptado de [39]

Propriedades	Follow My Vote	Agora	TIVI	Open Vote Network
Elegibilidade	✓	✓	✓	✓
Integridade	X	✓	✓	✓
Justiça	X	✓	✓	✓
Verificabilidade	✓	✓	✓	✓
Privacidade do voto	X	✓	✓	✓
Isenção de recibo	X	X	X	X
Resistência à coerção	X	X	X	X
Política de voto	Voto múltiplo	–	Voto único	Voto único

Em suma, um sistema de votação eletrónica (*e-voting*) recorre a equipamentos eletrónicos durante o processo eleitoral, nomeadamente durante as etapas de recenseamento, autenticação, votação e contagem. Este tipo de sistemas possuem poucos casos de sucesso que resultaram numa futura implementação, demonstrando a falta de confiança pública e preocupação face à segurança dos sistemas de votação eletrónica.

Uma *blockchain* trata-se de uma espécie de livro de registos público, onde todas as transações são armazenadas sob a forma de uma lista de blocos ordenadas cronologicamente. Os blocos são validados por determinados *nodes* da rede seguindo um mecanismo de consenso escolhido pela plataforma para adicionar um novo bloco ao *ledger* e assegurar que todos os participantes estejam de acordo.

Esta tecnologia pode ser aplicada à temática anterior como tentativa de combater as fragilidades ou reforçar as caraterísticas dos sistemas atuais através de uma rede distribuída, descentralizada, imutável e transparente.

Capítulo 3

Metodologias

Neste capítulo é elucidada a metodologia de investigação utilizada, bem como a metodologia de desenvolvimento de *software* adotada, evidenciando os seus conceitos principais e como as mesmas são implementadas. Perante o presente projeto, adotou-se, como metodologia de investigação, a abordagem *Design Science Research* (DSR), tendo em conta as *guidelines* de Hevner e, para o desenvolvimento de *software*, foi considerado a metodologia Ágil, nomeadamente o *framework* Scrum.

3.1 Metodologia de Investigação

O campo dos sistemas de informação (SI) suscitou interesse nos métodos e paradigmas de pesquisa, levando a crer que a pesquisa interpretativa tem um lugar significativo na pesquisa de (SI) [49]. Não obstante, a pesquisa crítica também é altamente relevante para os sistemas de informação.

O conceito *design science* surge pela primeira vez em 1965 por Richard Buckminster Fuller, onde foi definida como uma forma sistemática de projetar ou conceber coisas [9]. Mais tarde, Gregory estabelece a distinção entre a conceção pelo método científico (*design science research*) e o método de projeto ou conceção em si (*design*), alegando que a última não era uma ciência [9].

Na década de 90, Simon (1996) define *design science* como um paradigma de resolução de problemas através do desenvolvimento de artefactos [9]. Segundo (Denning 1997; Tsichritzis 1998), esta metodologia permite a criação de inovações que definem as ideias, práticas, capacidades técnicas e produtos através dos quais a análise, o

design, a implementação, a gestão e o uso de sistemas de informação podem ser realizados de forma eficaz e eficiente [8]. conjunto de ideias, procedimentos Posteriormente, Hevner et al. (2004) exemplifica o conjunto de ideias e procedimentos a seguir para a DSR e, numa abordagem pragmática, afirma que a DSR não anseia alcançar grandes teorias, mas almeja identificar e compreender os problemas do mundo real e propor soluções apropriadas, fazendo avançar o conhecimento teórico da área [9].

Ora, a DSR é exatamente um tipo de metateoria que tem dois objetivos essenciais: (1) desenvolver um artefacto para resolver um problema prático num contexto específico e (2) gerar novos conhecimentos técnicos e científicos [50].

3.1.1 As 7 *guidelines* da metodologia *Design Science Research*

De acordo com Hevner et al. (2004) foram criadas 7 diretrizes com o intuito de auxiliar os investigadores a conhecer e a compreender um problema de *design* e sua solução, através da construção e aplicação de um artefacto [8]. Os autores consideram que cada uma dessas diretrizes deve ser abordada de alguma maneira para que a pesquisa em *Design Science* seja completa, no entanto a maneira em como são aplicadas fica ao encargo dos pesquisadores/editores [51].

Neste artigo, o autor evidencia ainda a aplicação destas diretrizes em três estudos diferentes relativamente a pesquisas com sistemas de informações.

Na seguinte tabela, Tabela 3.1, estão expostas as 7 *guidelines* da DSR definidas por Hevner et al (2004).

Tabela 3.1: As 7 *Guidelines* da *Design Science Research* (traduzido de [8])

Guidelines	Descrição
1 - Design como um artefacto	A DSR deve produzir um artefacto viável na forma de um *construct*, modelo, método ou instanciação.
2 - Relevância do problema	O objetivo da DSR é desenvolver soluções baseadas em tecnologia para problemas importantes e relevantes.
3 - Avaliação do Design	A utilidade, qualidade e eficácia do artefacto devem ser rigorosamente demonstradas através de métodos de avaliação bem executados.
4 - Contribuições da pesquisa	A DSR deve promover contribuições claras e verificáveis nas áreas especificas dos artefactos desenvolvidos, nas fundamentações de design e/ou nas metodologias de design.
5 - Rigor na pesquisa	A DSR centra-se na aplicação de métodos rigorosos na construção e na avaliação do design do artefacto.
6 - Design como um processo de pesquisa	O desenvolvimento de um artefacto eficaz implica a utilização adequada dos meios disponíveis para alcançar os fins desejados, desde que satisfaçam as leis no ambiente do problema.
7 - Comunicação da pesquisa	A DSR deve ser apresentada tanto ao publico orientado para a tecnologia como para o publico orientado para a gestão.

Por um lado, um artefacto pode assumir diferentes formas, dentro das quais [9, 52]:

- ***Construct*** (vocabulários e símbolos): Fornecem uma linguagem na qual os problemas e soluções são definidos e comunicados dentro de um domínio;
- **Modelo** (abstrações e representações): É uma abstração intencional de entidades do mundo real que reduzem a complexidade ao incluir declarações e proposições sobre o problema e o espaço de solução;
- **Método** (algoritmos e práticas): Contém os passos a serem seguidos dentro do espaço de solução de um modelo para a obtenção de resultados tangíveis;
- **Instanciação** (sistemas implementados e protótipos): Implementação dos objetos anteriores utilizada para demonstrar viabilidade, possibilitando a avaliação concreta da adequação de um artefacto ao seu propósito pretendido.

Por outro lado, a avaliação de um artefacto pode ser realizada da seguinte maneira: Observação, Análise, Experimentação, Testagem e/ou Descrição [52].

3.1.2 Modelo de processamento da *Design Science Research*

Aliado às *guidelines* mencionada no ponto acima, recorreu-se à utilização do *framework* também proposto por Hevner et al. Este modelo de processamento consiste em três ciclos: Ciclo de Relevância, Ciclo de *Design* e o Ciclo de Rigor [51, 52, 53].

O ciclo de relevância permite a ligação das atividades de *design* e dos seus correspondentes ambientes de aplicação, fornecendo os requisitos para a pesquisa e os critérios de aceitação para a avaliação dos resultados da investigação.

O ciclo do rigor conecta as atividades de *design* com pesquisas e bases de conhecimento relevantes para o contexto do objeto de pesquisa. Deste modo, são transmitidos os conhecimentos científicos necessários para assegurar o desenvolvimento de uma inovação.

O ciclo do *design* itera a construção de um artefacto a ser desenvolvido com base na sua avaliação e *feedback*, aprimorando o *design* do mesmo.

No presente trabalho, adotou-se a metodologia DSR mencionada anteriormente, pelo que resultou na seguinte metodologia de conceção e construção de produto, Figura 3.1.

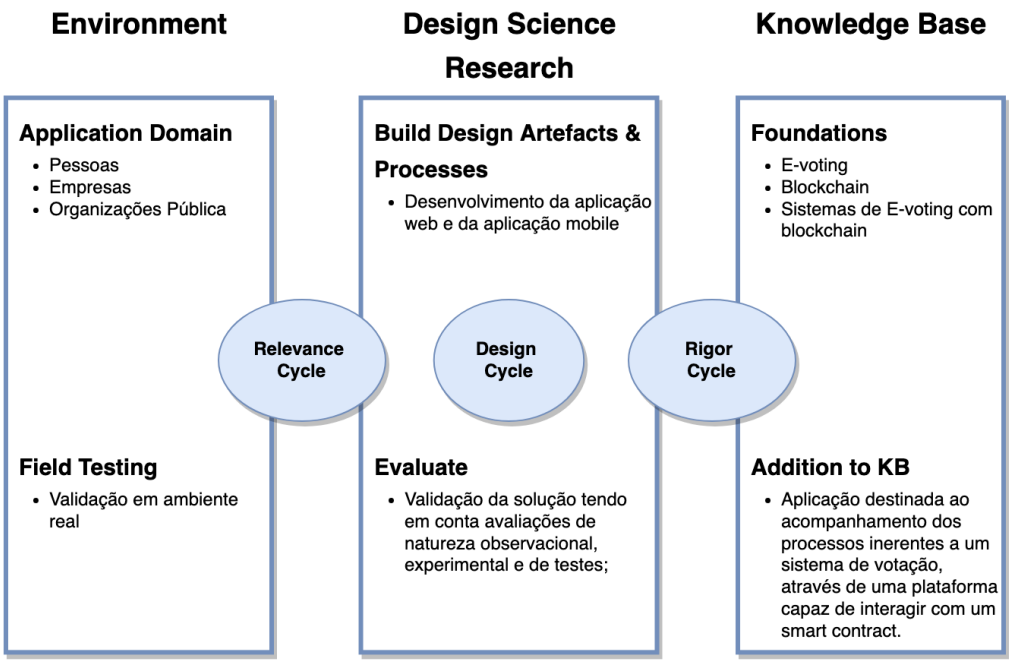


Figura 3.1: Metodologia DSR figura adaptada do artigo [8]

3.2 Metodologia de Trabalho

Em 2001, um grupo de 17 trabalhadores da indústria de *software* juntou-se com o objetivo de elaborar uma nova metodologia de desenvolvimento de *software* [54, 55]. Daqui, surgiu um documento (Manifesto para o Desenvolvimento Ágil de Software [56]) contendo todos os princípios para a obtenção de melhores resultados face aos métodos tradicionais. Estes princípios assentam sobre os seguintes valores [56]:

- Indivíduos e iterações são mais importantes do que processos e ferramentas;
- *Software* funcional é mais importante do que documentação completa;
- Colaboração com o cliente é mais importante do que negociação de contratos;
- Adaptação a mudanças é mais importante do que seguir o plano inicial.

Esta abordagem é iterativa e baseada na equipa, tentando, por um lado, promover a comunicação e colaboração entre todos os participantes e, por outro, reduzir o desperdício de recursos, tempo de desenvolvimento e esforço [54]. Contrariamente aos métodos tradicionais, onde é efetuado um planeamento aprofundado no início de um projeto, os métodos ágeis focam-se em entregar incrementos de produtos que agreguem valor ao cliente, pelo que tanto os requisitos como a solução final evoluem continuamente [54].

3.2.1 Ciclo de Vida Agile

As metodologias ágeis regem-se por um ciclo de vida considerado flexível, onde, em oposição ao modelo tradicional Waterfall, os eventos dão-se de forma progressiva, sem uma sucessão estrita, podendo ainda ocorrer paralelamente [54]. Este processo (Figura 3.2) divide-se em 6 fases [54]:

1. Análise de requisitos: Todos os participantes do projeto reúnem-se a fim de identificar os requisitos do mesmo, como este será utilizado e por quem;
2. Planeamento: Após existir uma ideia concreta, o projeto é divido em tarefas mais pequenas, priorizando e realocando-as em diferentes secções de execução;
3. Design: A equipa define a resolução que vá ao encontro dos requisitos pretendidos, bem como a estratégia de testagem;
4. Desenvolvimento: Decorre o processo de desenvolvimento propriamente dito, onde as funcionalidades são implementadas;
5. Testagem: São efetuados testes de modo a assegurar que o produto segue as indicações dos requisitos e atende às necessidades dos clientes;
6. Implantação: Dá-se a entrega do produto aos clientes, seja este uma entrega parcial ou o projeto finalizado.

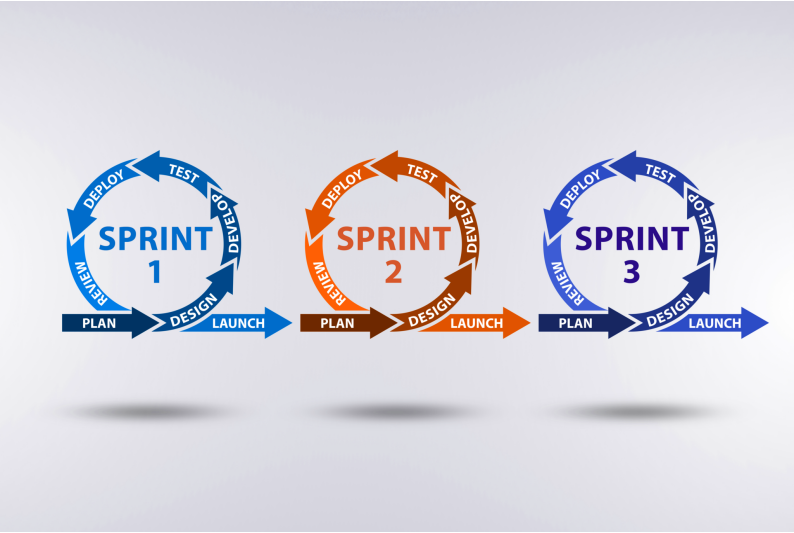


Figura 3.2: Ciclo de vida da metodologia Ágil [57]

3.2.2 Agile vs. Tradicional

A seguinte tabela (Tabela 3.2) contém uma comparação entre estas duas metodologias.

Tabela 3.2: Metodologia Ágil vs Metodologia Tradicional, traduzido de [58]

Parâmetro	Metodologia Tradicional	Metodologia Ágil
Facilidade de modificação	Difícil	Fácil
Abordagem de desenvolvimento	Preditiva	Adaptativa
Orientação do desenvolvimento	Orientado para o processo	Orientado para o cliente
Dimensão do projeto	Grande	Pequena ou Média
Escala de planeamento	Longo prazo	Curto prazo
Estilo de gestão	Comando e controlo	Liderança e colaboração
Processo de aprendizagem	Contínua	Secundária
Documentação	Elevada	Baixa
Tipo de organização	Difícil	Fácil
Tamanho da organização	Grande	Pequeno
Orçamento	Alto	Baixo
Número de equipas	Múltiplo	Um
Tamanho da equipa	Médio	Pequeno

As metodologias tradicionais de desenvolvimento de software seguem uma abordagem orientada ao processo, tendo princípios bastante rígidos e pouco propícios à mudança, aumentando o custo de possíveis alterações ao longo do projeto [58]. Estas metodologias são aplicadas a projetos de elevada dimensão, uma vez que o planeamento é também realizado a longo prazo, e, consequentemente, a documentação do mesmo necessita de ser maior e mais detalhada [58].

Por outro lado, as metodologias ágeis, consideradas leves, são altamente adaptáveis, sendo crucial a colaboração entre os clientes e a equipa que irá implementar o projeto [58]. A dimensão do projeto é menor, pelo que o seu planeamento é elaborado a curto prazo, e ainda a documentação obtida também é mais reduzida [58].

3.2.3 *Framework* Scrum

Scrum é um *framework* criado por Ken Schwaber e Jeff Sutherland, com o intuito de fornecer as etapas necessárias para controlar e gerir o processo de desenvolvimento de um *software*, tentando oferecer o maior valor no menor tempo possível [54]. Caracteriza-se por intervalos de tempo iterativos e incrementais, denominados de *sprints*, nos quais um determinado conjunto de tarefas deve ser realizado, originando entregas contínuas e evitando que os problemas que possam surgir sejam apenas encontrados no final do processo [54].

A sua atratividade deve-se à simplicidade e ao facto de o mesmo se focar na gestão do projeto ao invés de práticas técnicas de desenvolvimento de *software*, tornando-se uma das metodologias ágil mais utilizadas, demonstrando a sua vasta aplicabilidade [58]. Este *framework* possibilita auxiliar projetos com diferentes níveis de complexidade, mantendo-se flexível quanto ao procedimento a ser seguido.

Scrum *Roles*

Neste *framework*, existem três *personas* de modo a garantir o seu devido funcionamento [54, 55, 58, 59, 60]:

- *Scrum Master*: Tem o papel de moderador entre todos os membros da equipa, sendo responsável por transmitir e garantir que os princípios e as práticas desta abordagem sejam respeitados, e que o projeto avance, eliminando qualquer impedimento que surja;
- *Product Owner*: Serve como intermediário entre a equipa e os seus clientes, estando encarregue de definir os objetivos do projeto, incorporar os requisitos pretendidos no *Product Backlog*, priorizar o que será incluído no próximo *sprint* e rever o sistema no final de cada *sprint*;
- *Scrum Team*: É uma equipa multifuncional, que colabora entre si com o intuito de atingir as metas do *sprint*, criando um produto incremental no final de cada *sprint*. Está incumbida de analisar os requisitos identificados e planear, desenvolver, testar e validar o *software* elaborado.

Ciclo de Vida Scrum

O Ciclo de Vida Scrum, ilustrado na Figura 3.3, inicia-se com o levantamento dos objetivos pretendidos para o sistema a ser projetado, bem como a identificação de todos os elementos da equipa que serão incorporados no projeto e ferramentas necessárias para a realização do mesmo [58]. Adicionalmente, é delineado o *Product Backlog*, correspondente a uma lista que contém os requisitos e as funcionalidades a implementar na solução, estando sujeito a atualizações durante todo o processo de desenvolvimento [55].

A fase de desenvolvimento propriamente dita consiste numa sucessão de *sprints* estendidos entre 2 e 4 semanas, cujo resultado é um incremento do sistema [58]. O planeamento do *sprint* dá-se numa reunião no início de um *sprint* (*Sprint Planning Meeting*) com a finalidade de definir o objetivo desse mesmo *sprint*, selecionar itens do *Product Backlog* para serem incluídos no mesmo e planear como é que estas tarefas serão realizadas [55, 58].

Como forma da equipa se sincronizar em relação ao andamento do projeto, existem reuniões diárias breves - *Daily Scrum* -, visando verificar o que foi efetuado no dia anterior, identificar as tarefas a desenvolver nesse dia e resolver quaisquer problemas que a equipa possa enfrentar [54, 55, 58].

Por fim, na conclusão de cada *sprint* são realizados dois eventos: *Sprint Review Meeting*, onde a equipa mostra o que foi produzido durante o *sprint* e são determinadas futuras adaptações face ao produto pretendido; e a *Sprint Retrospective*,

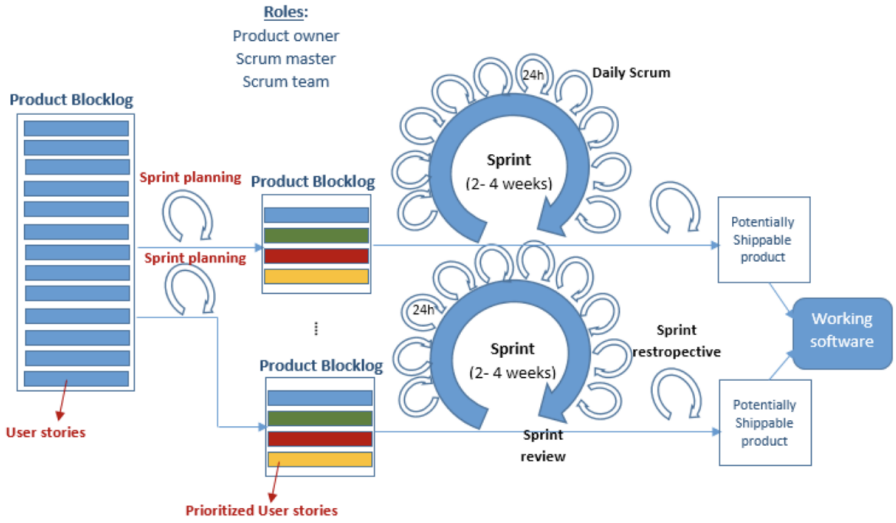


Figura 3.3: Ciclo de vida do Scrum [58]

cujo objetivo é identificar formas de aumentar a qualidade e a eficácia de trabalho, através da revisão do que foi feito e não feito, sendo definidas as ações a melhorar no próximo *sprint* [54, 55, 58].

São executados vários *sprints* até o produto obtido corresponder à visão pensada e estarem satisfeitos todos os requisitos, dando o projeto por concluído.

Em suma, a *Design Science Research* almeja desenvolver um artefacto para resolver um problema prático num contexto específico e gerar novos conhecimentos técnicos e científicos, sendo composta por 7 *guidelines* com o intuito de auxiliar os investigadores a conhecer e a compreender um problema de *design* e sua solução, através da construção e aplicação de um artefacto, seguindo um modelo de processamento constituído por 3 ciclos.

A metodologia de trabalho utilizada tenta, por um lado, promover a comunicação e colaboração entre todos os participantes e, por outro, reduzir o desperdício de recursos, tempo de desenvolvimento e esforço. Caracteriza-se pela aplicabilidade de 6 etapas impostas por 3 *personas* e dispostas em intervalos de tempo iterativos e incrementais (*sprints*), originando entregas contínuas e evitando que os problemas que possam surgir sejam apenas encontrados no final do processo.

Capítulo 4

Desenvolvimento

No presente capítulo será exposta a plataforma utilizada para o desenvolvimento deste projeto. Adicionalmente, é apresentada a modelação do problema, que engloba todos os intervenientes do sistema e cenários de utilização, e ainda, as *wireframes* resultantes destas interações. Os diagramas e as *wireframes* foram desenvolvidos utilizando respetivamente a aplicação draw.io e a ferramenta *web* de edição gráfica Balsamiq. Para o desenvolvimento do *smart contract* recorreu-se à aplicação *web* Remix IDE.

4.1 *OnBoarding*

Para o decorrer deste projeto, recorreu-se à plataforma Outsystems. É uma plataforma de desenvolvimento de aplicações para diversos dispositivos projetada para acelerar drasticamente o desenvolvimento das mesmas, oferecendo, ainda, todo o acompanhamento durante o seu ciclo de vida.

Possui um ambiente visual de desenvolvimento *full-stack* (Figura 4.1) integrado com uma abordagem de *low-code*, capaz de facilitar todo o processo de desenvolvimento e, consequentemente, diminuindo o seu tempo [61].

Esta plataforma dispõe de *templates* e padrões personalizáveis para a construção das interfaces com as quais os utilizadores irão interagir, bem como permite que a realização de APIs, serviços *web*, fluxos de trabalho, regras de negócios e modelação do banco de dados sejam efetuados de forma visual através de um sistema *drag-and-drop* [61].

Adicionalmente, é possível usufruir de conectores de código ou um assistente de integração para se conectar a pacotes de *software*, outras aplicações ou bancos de dados existentes e, ainda, consumir ou expor visualmente serviços *web* [61].

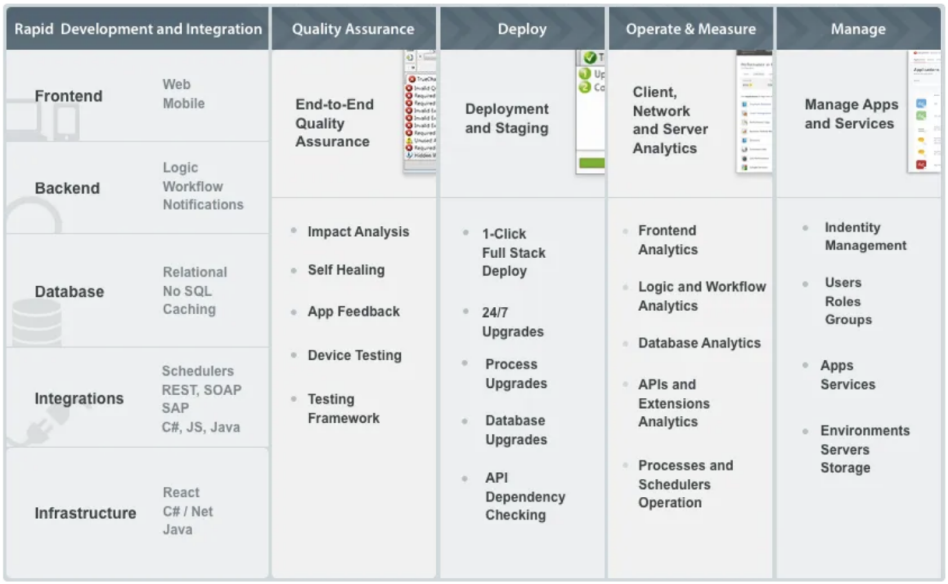


Figura 4.1: *Full-stack* da Outsystems [61]

4.1.1 Componentes da plataforma Outsystems

A plataforma Outsystems oferece diversas ferramentas e serviços, presentes na Figura 4.2, para desenvolver e monitorizar toda a evolução das aplicações, através de um ambiente visual infundido com inteligência artificial (IA), dentro das quais [62, 63]:

- **ServiceStudio**: É o ambiente de desenvolvimento visual que permite criar todos os componentes para o desenvolvimento de um projeto, desde o modelo de dados, a *User Interfaces* (UI), integrações com outros sistemas, à lógica de toda a aplicação ou até mesmo políticas de segurança. Este processo é assistido por inteligência artificial, onde é sugerido as próximas etapas a realizar e é preenchido automaticamente certos campos;
- **Integration Studio**: É um ambiente capacitado a criar componentes capazes de integrarem outros sistemas e micro serviços, podendo estes serem reutilizáveis por todas as aplicações da plataforma Outsystems;
- **Forge**: Contém um repositório, acessível a todos os utilizadores, de componente de UI, bibliotecas e conectores construídos e partilhados pelos membros da comunidade OutSystems, devidamente auditados;

- **Lifetime**: É um ambiente de gestão centralizada de todos os ambientes de desenvolvimento, controle de qualidade e produção. Aqui, é monitorizado o desempenho das aplicações, para, caso seja necessário, serem feitas as devidas alterações a fim de promover o bom funcionamento das aplicações, bem como se dá a gestão das permissões da equipa de desenvolvimento;
- **Service Center**: Ferramenta capaz de gerir os aspectos operacionais de um ambiente, possibilitando a reversão para uma versão anterior da aplicação, configuração de propriedades da aplicação, análise dos seus utilizadores ou possíveis erros que surjam;
- **Plataform Server**: É o núcleo da plataforma Outsystems responsável por gerar, compilar e implementar as aplicações desenvolvidas num servidor de aplicações da *web* padrão ou empacotar numa aplicação móvel nativa, aliando-se a serviços especializados para fazer a compilação e a difusão automatizada do código para vários servidores, o agendamento de tarefas e monitorização de erros e questões de performance;
- **Mobile Application Build Service (MABS)**: É um serviço responsável por colocar uma aplicação móvel num pacote nativo pronto para ser instalado nos dispositivos móveis, à distância de um clique.

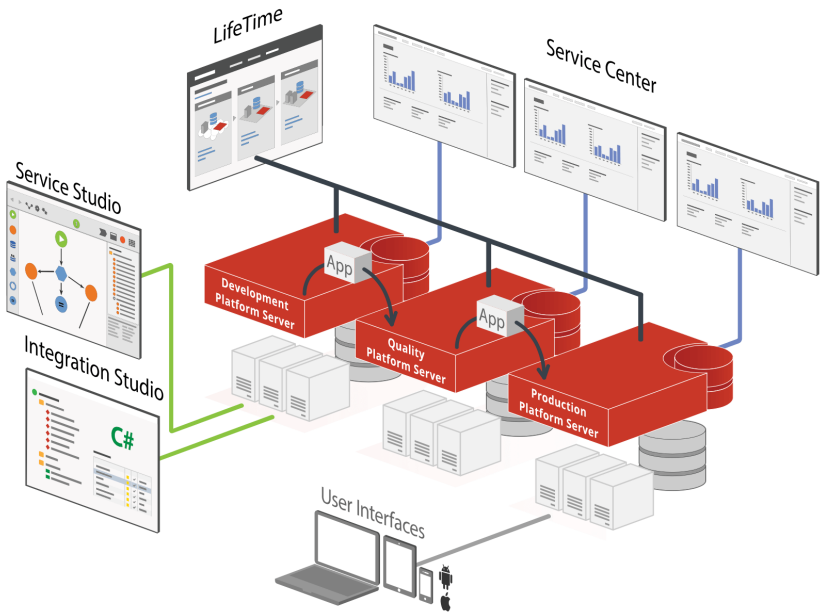


Figura 4.2: Componentes da plataforma Outsystems [64]

4.1.2 *Architecture Canvas*

Considerando que, em Outsystems, um projeto é construído baseando-se numa arquitetura com várias camadas de abstração, surge a ferramenta *Architecture Canvas*, como forma de simplificar o design de Arquiteturas Orientadas a Serviços (SOA) [65]. Na eventualidade de serem desenvolvidos vários módulos, esta garante a correta abstração de (micro) serviços reutilizáveis, tal como o correto isolamento de módulos funcionais distintos, promovendo uma arquitetura mais económica, fácil de manter e, consequentemente, evoluir [65].

Por um lado, recorre-se à *Architecture Canvas* para identificar as necessidades funcionais, não funcionais e de integração, fazendo o levantamento dos requisitos de forma estruturada e sistemática; e, por outro, este ajuda a definir os módulos que irão implementar os conceitos identificados [65]. Ainda assim, este processo é contínuo ao longo de todo o desenvolvimento de uma solução, possibilitando o aparecimento de novas necessidades e conceitos [65].

A *Architecture Canvas* é composto por 3 *layers* (Figura 4.3) representativas do tipo de funcionalidade encontradas nos seus módulos correspondentes [66, 67]:

- *End-User*: Engloba as *user interfaces* proporcionando a interação do utilizador com a aplicação, estando implementadas as *user stories* do sistema;
- *Core*: Abrange o modelo de dados e todas as regras de negócio que irão ser usadas na camada *End User*, fornecendo os serviços necessários para o bom funcionamento da mesma;
- *Library* ou *Foundation*: Contém todos os serviços necessários para uma possível conexão a sistemas externos, bem como recursos reutilizáveis, padrões de UI e temas.

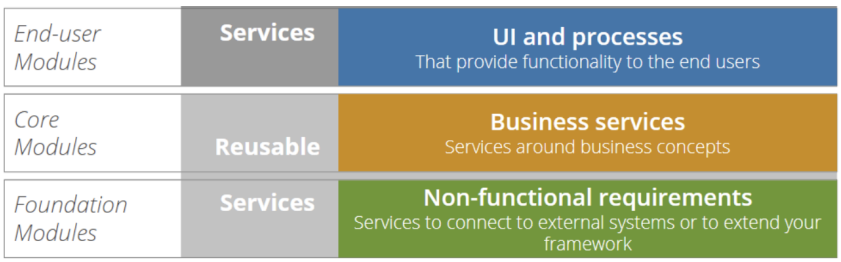


Figura 4.3: *Layers* da *Architecture Canvas*, adaptado de [65]

É de notar que para uma arquitetura bem projetada é necessário respeitar 3 regras, ilustradas na Figura 4.4. Primeiramente, não devem existir referências ascendentes, pois estas tendem a criar uma dependência circular, demonstrando que os serviços não estão devidamente isolados [68]. Posteriormente, a camada dos módulos *End-User* não deve ter referências laterais, isto é, a outros módulos nesta mesma

camada, uma vez que os mesmos devem ter diferentes ciclos de vida e não devem fornecer serviços reutilizáveis [68]. Por último, não devem encontrar-se ciclos nas camadas inferiores (*Core* e *Foundation*), como forma de evitar impactos inesperados e códigos difíceis de gerir, onde os conceitos não estão corretamente abstraídos [68].

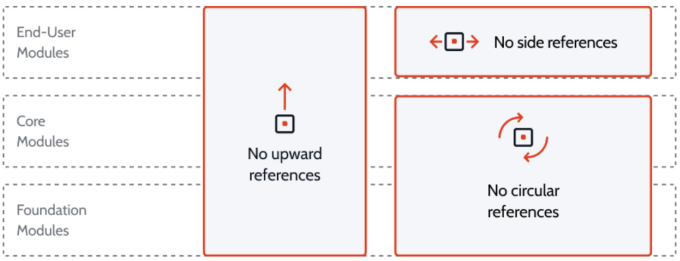


Figura 4.4: Regras para a construção da *Architecture Canvas* [68]

Para chegar-se à *Architecture Canvas* deste projecto, inicialmente foram identificados todos os conceitos e requisitos funcionais e não funcionais inerentes a cada camada, tendo em consideração todas as possíveis *user stories*, *personas* e *roles*, tecnologia a ser integrada para um bom funcionamento e promover a experiência do utilizador [69, 70], e organizados nas respetivas camadas. Posto isto, chegou-se à representação ilustrativa das etapas *Disclose* e *Organize*, exposta na Figura 4.5.

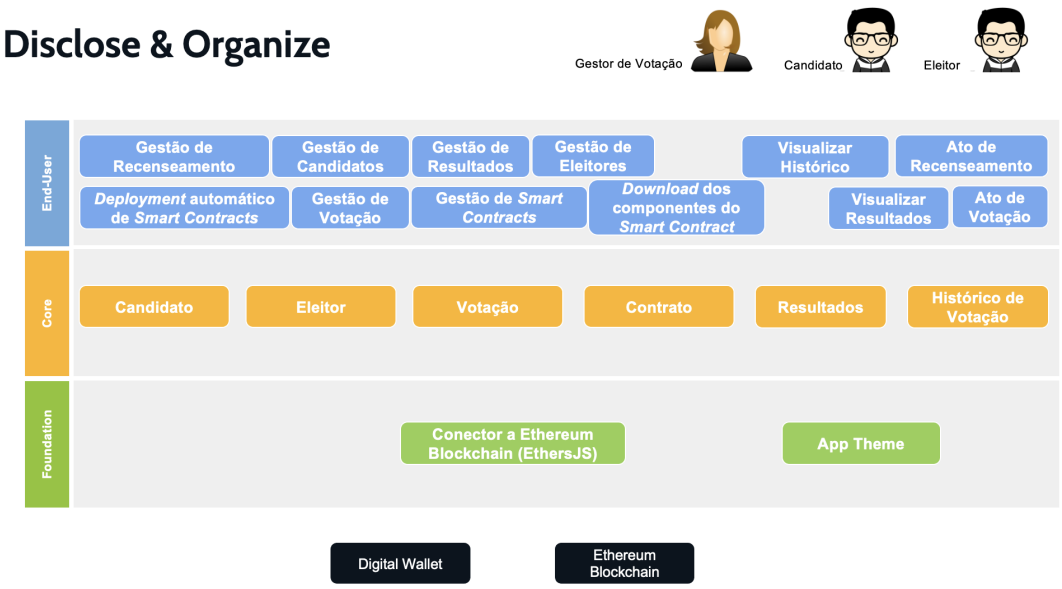


Figura 4.5: Diagrama das etapas *Disclose* e *Organize*

Por último, todos os conceitos foram agrupados em sistemas (Figura 4.6) a serem desenvolvidos e retratadas as ligações de comunicação entre os mesmos [69]. Desta forma, obteve-se duas vertentes aplicacionais (*web* e *mobile*) na camada *End-User*, de modo a satisfazer as necessidades de cada utilizador. Na camada *Core* situa-se um módulo apenas (Voting_CS), representativo de todas as tabelas de base de dados, as

suas relações, e todas as *server actions* necessárias. A última camada, *Foundation*, contém as ferramentas necessárias para permitir a comunicação com a *blockchain* e o tema dos módulos aplicacionais. A documentação das APIs do sistema *Backend* em Outsystems pertencentes à camada *Core* está exposta no Anexo A.

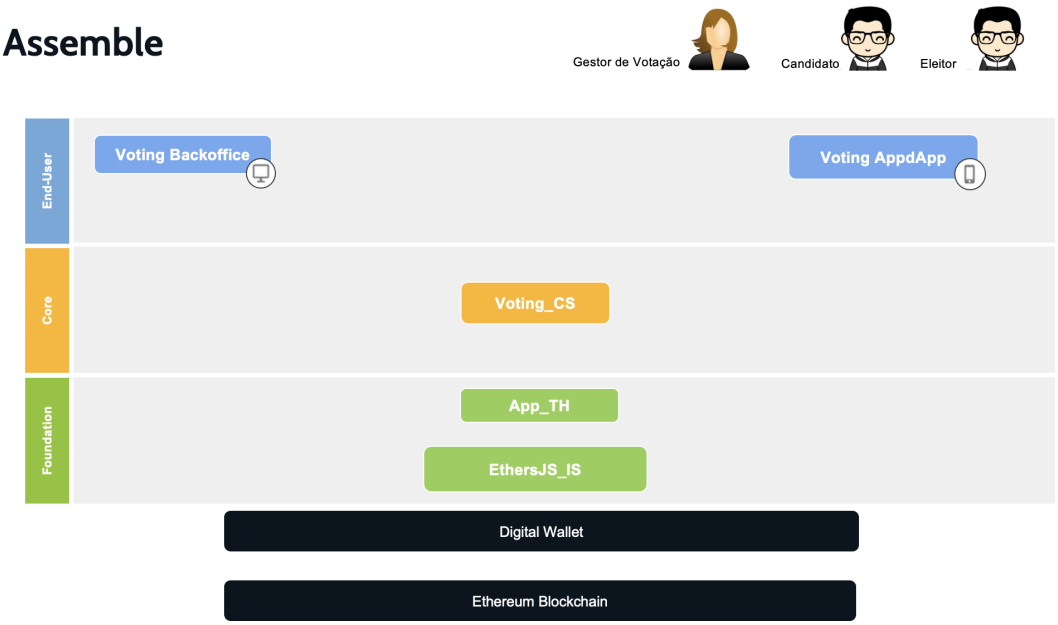


Figura 4.6: Diagrama da etapa *Assemble*

Posto isto, foi possível elaborar diagrama de arquitetura final do projeto, contendo as dependências entre cada sistema, exibido na Figura 4.7.

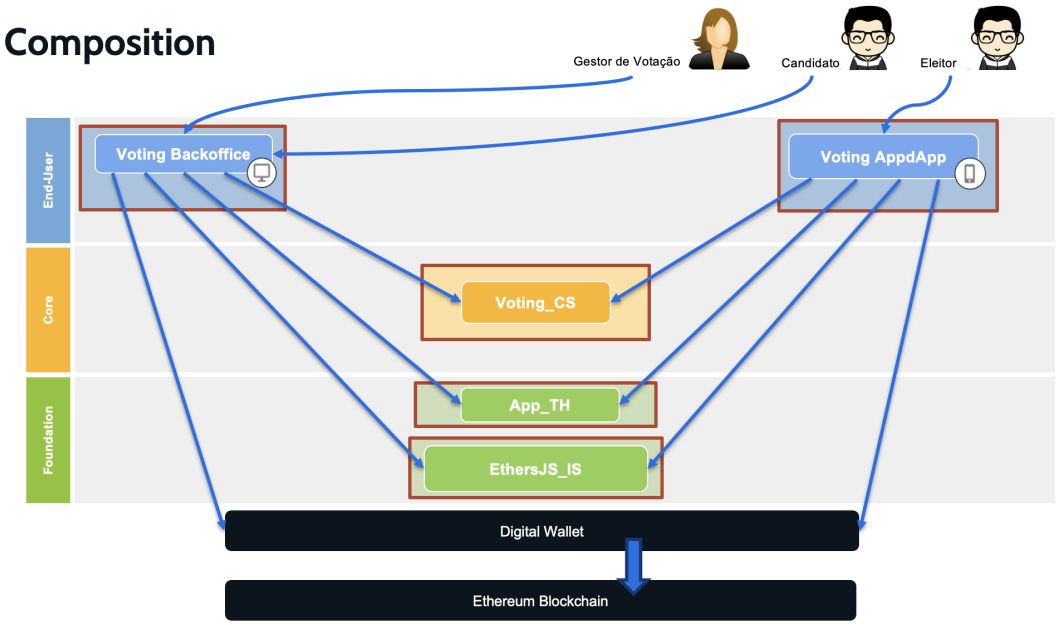


Figura 4.7: *Architecture Canvas*

4.2 Execução

No caso específico desta dissertação, o projeto é constituído por 4 componentes: a componente *web* (*BackOffice*), a componente *mobile*, a plataforma OutSystems e o *smart contract* (Figura 4.8), sendo necessários que ambas as componentes *mobile* e *web* tenham conexão à *internet*, uma vez que estas são suportadas pelos dados presentes no servidor OutSystems. Por outro lado, para permitir aos utilizadores alterarem os dados existentes nas bases de dados, recorre-se a *server actions*, demonstrando-se na relação bidirecional entre o servidor e os dispositivos *web* e *mobile*. Por sua vez, a comunicação com o *smart contract* é feita através da inserção em todos os módulos aplicacionais de um ficheiro *javascript* denominado de *EthersJS* [71], sendo este uma biblioteca *open source* e compacta que permite interagir com a *blockchain* Ethereum e todo o seu ecosistema, com o auxílio da extensão do *browser* MetaMask.

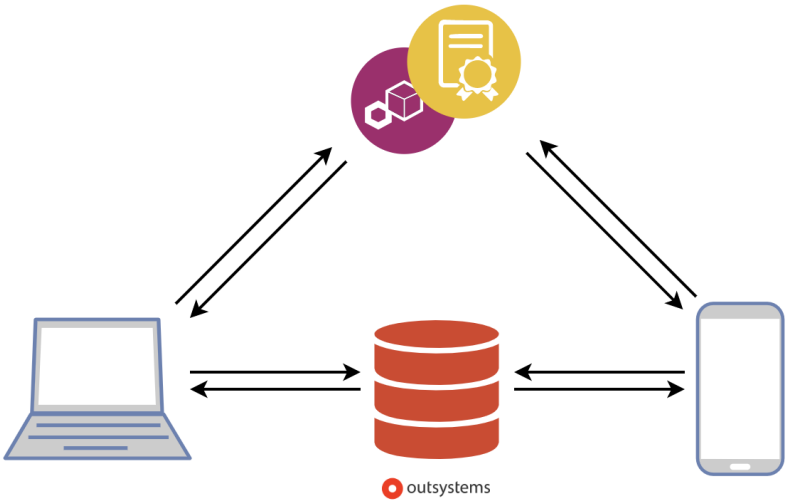


Figura 4.8: Arquitetura do projeto

O *EthersJS* surge como alternativa ao *Web3Js*, a modo de oferecer outra opção de utilização aos *developers* desta área, sendo mais pequena, possui uma documentação extensa, que é atualizada regularmente, capaz de orientar tanto a sua utilização como para o uso do vocabulário associado a uma *blockchain*, com inúmeros casos de testes, e ainda uma licença do tipo MIT, proporcionando a sua utilização sem qualquer restrição [72, 73].

Como outra possibilidade, existem ainda algumas plataformas de desenvolvimento de *blockchain* como o Infura ou o Alchemy, que fazem uma conexão rápida e confiável com a *blockchain*, através de *nodes* e das suas APIs para armazenar e acessar arquivos na *blockchain*, e possuiem diversas ferramentas para testar e monitorizar com segurança as aplicações descentralizadas criadas [74, 75].

Ainda assim, a escolha do *EthersJs* deu-se pela facilidade de utilização e implementação através dos blocos *javascript* presentes no ambiente de desenvolvimento utilizado.

Por último, é possível evidenciar que a *blockchain* utilizada foi a Ropsten, uma rede de testes associada ao ecossistema Ethereum, que permite que os projetos sejam testados num ambiente semelhante à rede principal, neste caso Ethereum, antes de serem implementadas na mesma, neste caso Ethereum.

Após a *Architecture Canvas* ter sido desenvolvida, foi possível a construção do modelo de dados (Figura 4.9), estando este inserido na camada *Core* e englobando todos os conceitos importantes do negócio e funcionalidades indispensáveis para a satisfação dos casos de uso definidos.

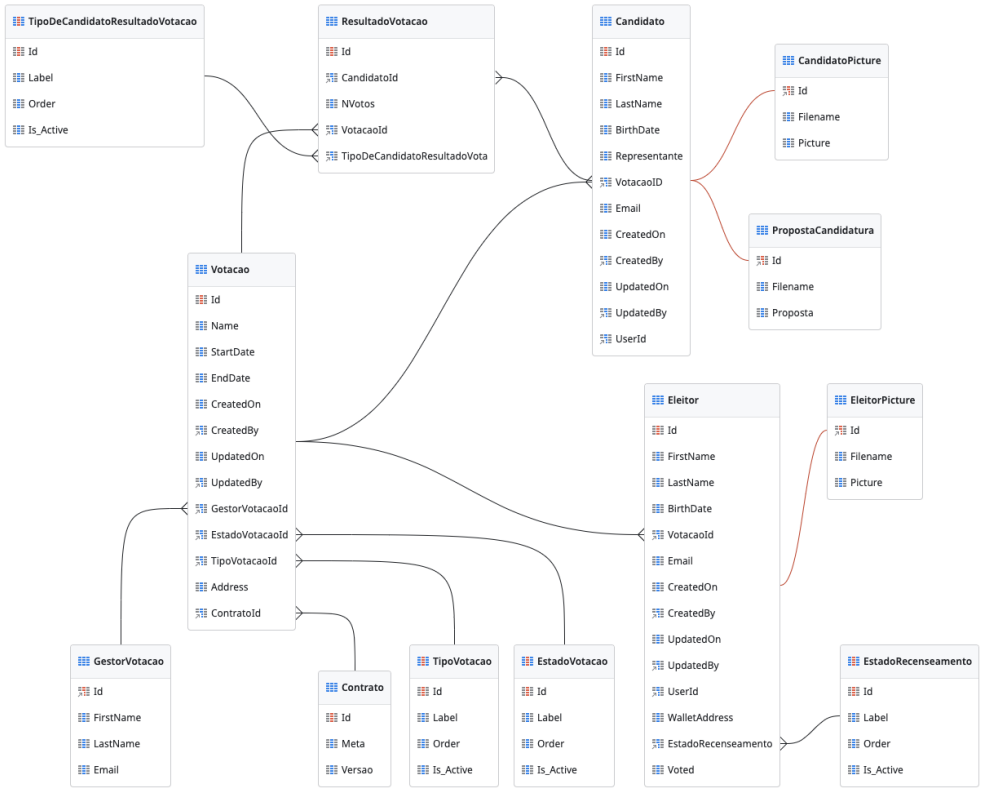


Figura 4.9: Modelo de dados do projeto

4.2.1 Atores

Previamente à implementação deste projeto, foi necessário proceder-se à identificação dos atores que irão interagir com todo o sistema. Deste modo, foram considerados os seguintes intervenientes:

- **Eleitor**: O eleitor pode votar numa votação e, posteriormente, confirmar o seu voto, verificar as informação de uma votação, incluindo os seus candidatos e a respetiva proposta, e analisar os devidos resultados de uma votação;
- **Candidato**: O candidato está encarregue de editar o seu perfil para uma determinada votação, podendo adicionar uma proposta de candidatura;
- **Gestor de Votação**: Este ator tem a responsabilidade de inserir uma votação no sistema, bem como todas as informações inerentes a esta, adicionar eleitores e candidatos a uma votação, encerrar ou, até mesmo, eliminar uma votação.

4.2.2 Block The Vote - Mobile

A aplicação *mobile* é composta por um único módulo que proporciona que o ator interaja com o sistema. No caso da aplicação *mobile*, o ator em questão é unicamente o eleitor.

Requisitos

Na seguinte tabela, Tabela 4.1, estão expostos os requisitos aos quais a aplicação móvel deve atender, em que F e NF representam um requisito do tipo Funcional e Não Funcional, respetivamente.

Tabela 4.1: Requisitos da aplicação móvel Block The Vote

Id	Tipo	Descrição
1.F.M	F	Deverá permitir que os eleitores façam login no sistema através dos seus dados exclusivos, fornecidos pelo Admin.
2.F.M	F	Deverá permitir que os eleitores se recenseiam numa votação.
3.F.M	F	Deverá permitir que os eleitores conectem a sua carteira digital.
4.F.M	F	Deverá permitir que o eleitor visualize informações de uma votação, tal como as suas datas de início e término e os respetivos candidatos.
5.F.M	F	Deverá permitir que o eleitor visualize informações os candidatos de uma determinada votação.
6.F.M	F	Deverá permitir que os eleitores votem somente num candidato.
7.F.M	F	Deverá permitir que os eleitores votem somente uma vez.
8.F.M	F	Deverá permitir que o eleitor visualize os resultados das votações em que se recenseou anteriormente.
9.F.M	F	Deverá permitir que o eleitor visualize em detalhe o seu histórico de votações.

Continua na próxima página

Tabela 4.1: Requisitos da aplicação móvel Block The Vote (continuação)

Id	Tipo	Descrição
10.NF.M	NF	A aplicação deverá ser capaz de interagir com um *smart contract*.
11.NF.M	NF	A aplicação deverá ser capaz de interagir com uma carteira digital.
12.NF.M	NF	A aplicação deverá suportar login próprio para a app.
13.NF.M	NF	A aplicação deverá ter um tema de *User Interface*.
14.NF.M	NF	A aplicação deverá conhecer o modelo de dados.

Cenários

Posteriormente, foram levantados os cenários, expostos nas seguintes tabelas, que retratam as interações do eleitor com o sistema, estando em concordância com o diagrama de casos de uso ilustrado na Figura 4.10.

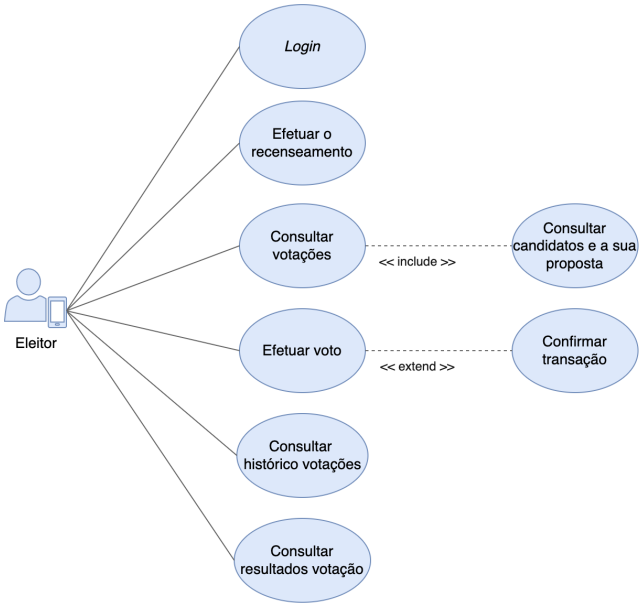


Figura 4.10: Diagrama de casos de uso da aplicação móvel

A Tabela 4.2 retrata o Cenário de Casos de Uso 1 da aplicação *Mobile* correspondente a efetuar o *login* na aplicação.

Tabela 4.2: Cenário 1: efetuar *login* na aplicação

Id	Cenário de Casos de Uso 1 (CCU1): *Login* na aplicação
Descrição	O eleitor pretende efetuar *login* na aplicação
Ator	Eleitor
Cenário	1. O eleitor abre a *app*; 2. Preenche os campos assinalados com o seu *username* e *password*; 3. Carrega no botão "*Login*".

A Tabela 4.3 retrata o Cenário de Casos de Uso 2 da aplicação *Mobile* correspondente a efetuar o recenseamento numa votação.

Tabela 4.3: Cenário 2: efetuar o recenseamento numa votação

Id	CCU2: Recenseamento numa votação
Descrição	O eleitor pretende efetuar o recenseamento numa votação
Ator	Eleitor
Cenário	1. O eleitor abre a *app*; 2. Seleciona "*Registration*" no menu de navegação; 3. Carrega sobre a votação pretendida; 4. Preenche os campos assinalados com os seus dados; 5. Carrega no botão "*Save*"; 6. Carrega no botão "*Connect Wallet*"; 7. Realiza a confirmação no pop-up exibido.

A Tabela 4.4 retrata o Cenário de Casos de Uso 3 da aplicação *Mobile* correspondente a analisar uma dada votação.

Tabela 4.4: Cenário 3: analisar uma dada votação

Id	CCU3: Analisar uma dada votação
Descrição	O eleitor pretende analisar os detalhes de uma votação
Ator	Eleitor
Cenário	1. O eleitor abre a *app*; 2. Seleciona "*Votings*" no menu de navegação; 3. Carrega sobre a votação pretendida; 4. Apresentação da informação de uma dada votação.

A Tabela 4.5 retrata o Cenário de Casos de Uso 4 da aplicação *Mobile* correspondente a analisar os detalhes dos candidatos de uma dada votação.

Tabela 4.5: Cenário 4: analisar os detalhes dos candidatos

Id	CCU4: Analisar os detalhes dos candidatos
Descrição	O eleitor pretende analisar as informações dos candidatos de uma votação, bem como a sua proposta de candidatura.
Ator	Eleitor
Cenário	1. O eleitor abre a *app*; 2. Seleciona "*Votings*" no menu de navegação; 3. Carrega sobre a votação pretendida; 4. Carrega sobre o candidato pretendido; 5. Apresentação da informação do candidato.

A Tabela 4.6 retrata o Cenário de Casos de Uso 5 da aplicação *Mobile* correspondente a votar numa determinada votação.

Tabela 4.6: Cenário 5: votar

Id	CCU5: Efetuar voto
Descrição	O eleitor pretende votar numa determinada votação
Ator	Eleitor
Cenário	1. O eleitor abre a *app*; 2. Seleciona "*Votings*" no menu de navegação; 3. Carrega sobre a votação pretendida; 4. Carrega no botão "*Vote*"; 5. Realiza a confirmação no pop-up exibido; 6. Seleciona o candidato em quem pretende votar; 7. Realiza a confirmação no pop-up exibido; 8. Efetua a confirmação da transação.

A Tabela 4.7 retrata o Cenário de Casos de Uso 6 da aplicação *Mobile* correspondente a consultar o histórico de votações.

Tabela 4.7: Cenário 6: consultar o histórico de votações

Id	CCU6: Consultar o histórico de votações
Descrição	O eleitor pretende consultar o histórico de votações
Ator	Eleitor
Cenário	1. O eleitor abre a *app*; 2. Seleciona "*History*" no menu de navegação; 3. Carrega sobre a votação pretendida; 4. Apresentação da informação de uma dada votação.

A Tabela 4.8 retrata o Cenário de Casos de Uso 7 da aplicação *Mobile* correspondente a consultar os resultados de uma votação.

Tabela 4.8: Cenário 7: consultar resultados votações

Id	CCU7: Consultar resultados votações
Descrição	O eleitor pretende consultar os resultados de uma determinada votação
Ator	Eleitor
Cenário	1. O eleitor abre a *app*; 2. Seleciona "*Results*" no menu de navegação; 3. Carrega sobre a votação pretendida; 4. Apresentação da informação relativamente aos resultados da votação.

Wireframes

Tendo por base os requisitos previamente recolhidos e os cenários existentes, procedeu-se ao desenvolvimento das respetivas *Wireframes*, recorrendo à ferramenta Balsamiq.

CCU1. Como Eleitor, quero fazer *login* na aplicação.

Detalhes: O eleitor pode efetuar o *login* na aplicação, ilustrado na Figura 4.11, utilizando o seu *username* e *password*.

Figura 4.11: *Wireframe* de *login*

CCU2. Como Eleitor, quero recensear-me numa votação.

Detalhes: Para se recensear numa votação (Figura 4.12), o eleitor tem que navegar até à página de "*Registration*" e depois selecionar a votação pretendida. Após confirmar os seus dados e guardá-los, o eleitor deve conectar a sua carteira digital com a aplicação.

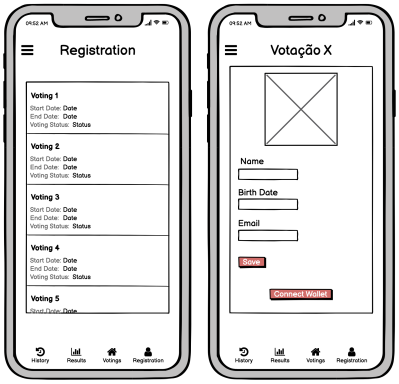


Figura 4.12: *Wireframe* do Recenseamento numa votação

CCU3. Como Eleitor, pretendo analisar os detalhes de uma votação.

Detalhes: Para visualizar os detalhes de uma votação, ilustrado na Figura 4.13, basta aceder à votação pretendida, sendo possível observar a sua data de início e término, o tipo de votação, o seu estado e os candidatos associados à mesma.

Figura 4.13: *Wireframe* dos detalhes de uma votação

CCU4. Como Eleitor, pretendo analisar os detalhes dos candidatos.

Detalhes: Para visualizar o perfil de um dado candidato associado a uma votação, ilustrado na Figura 4.14, basta aceder à votação pretendida e selecionar o candidato sobre o qual deseja obter informações.

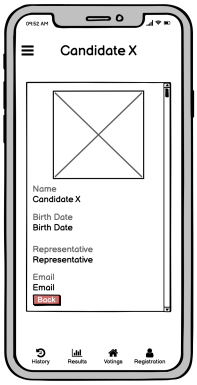


Figura 4.14: *Wireframe* de um candidato

CCU5. Como Eleitor, pretendo votar.

Detalhes: A fim de votar (Figura 4.15), é necessário entrar previamente no ecrã de detalhes de uma data votação e, de seguida, proceder à escolha do candidato em quem pretende votar. Após a confirmação da transação, este recebe uma notificação de que o seu voto foi aceite.

Figura 4.15: *Wireframe* de votar

CCU6. Como Eleitor, pretendo consultar o histórico de votações.

Detalhes: Para verificar o seu voto, o eleitor tem que aceder às votações terminadas que se encontram na página "*History*" e depois selecionar a votação pretendida, como é possível observar na Figura 4.16.

Figura 4.16: *Wireframe* do histórico de uma votação

CCU7. Como Eleitor, pretendo analisar os resultados de uma votação.

Detalhes: De modo a consultar os resultados de uma votação (Figura 4.17), o eleitor deve navegar até à página "*Results*" e, posteriormente, escolher uma votação. Neste ecrã é possível averiguar a contagem dos votos para cada candidato.

Figura 4.17: *Wireframe* dos resultados de uma votação

Matriz de rastreabilidade entre requisitos e os casos de uso

A seguinte tabela, Tabela 4.9, ilustra a matriz de rastreabilidade entre os requisitos da aplicação *mobile* com os casos de uso respetíveis. A Arquitetura do Projeto (AP) responsável por abater os Requisitos 13 e 14, pode ser representada através das Figuras 4.7 e 4.8, previamente expostas.

Tabela 4.9: Matriz de rastreabilidade entre requisitos e casos de uso

Req. Id	CCU1	CCU2	CCU3	CCU4	CCU5	CCU6	CCU7	AP
1.F.M	X							
2.F.M		X						

Continua na próxima página

Tabela 4.9: Matriz de rastreabilidade entre requisitos e casos de uso (continuação)

Req. Id	CCU1	CCU2	CCU3	CCU4	CCU5	CCU6	CCU7	AP
3.F.M		X						
4.F.M			X					
5.F.M				X				
6.F.M					X			
7.F.M					X			
8.F.M							X	
9.F.M						X		
10.NF.M					X			
11.NF.M		X			X			
12.NF.M	X							
13.NF.M								X
14.NF.M								X

Diagramas de Sequência

Os diagramas de sequência (DS) representativos dos casos de uso do sistema *Mobile* estão evidenciados nas seguintes figuras.

A Figura 4.18 ilustra o diagrama de sequência relativo ao Cenário de Casos de Uso 1 da aplicação *Mobile* correspondente a efetuar o *login* na aplicação.

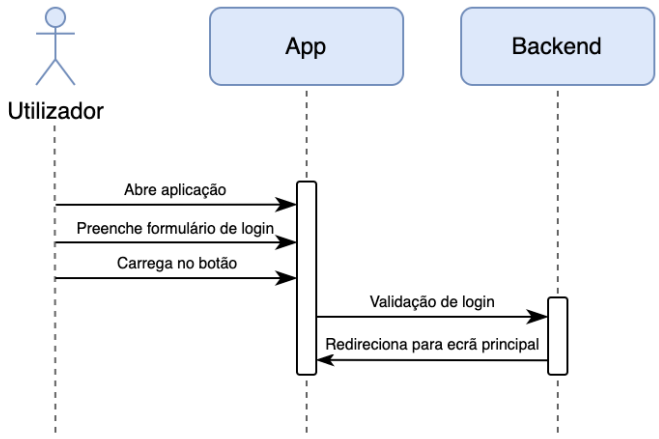


Figura 4.18: DS1 - Diagrama de sequência dos casos de uso 1

A Figura 4.19 ilustra o diagrama de sequência relativo ao Cenário de Casos de Uso 2 da aplicação *Mobile* correspondente a efetuar o recenseamento numa votação.

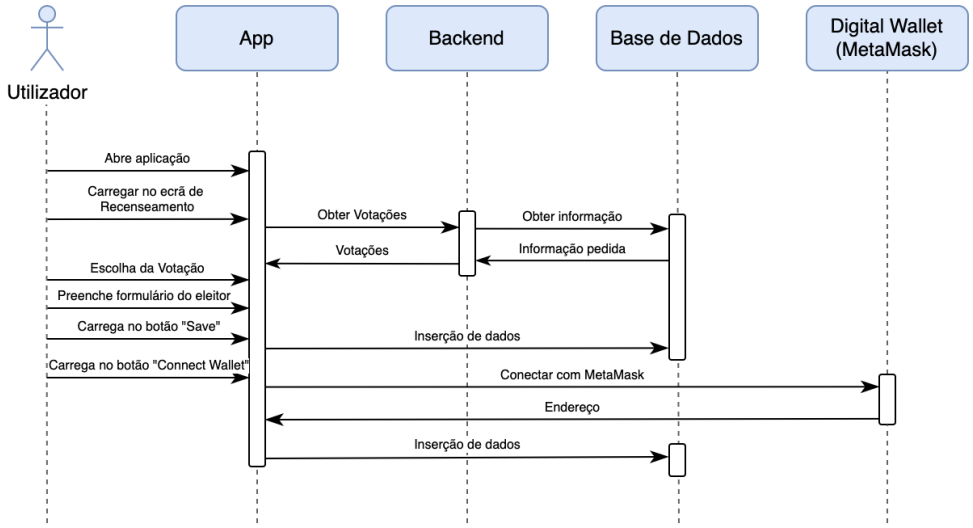


Figura 4.19: DS2 - Diagrama de sequência dos casos de uso 2

A Figura 4.20 ilustra o diagrama de sequência relativo ao Cenário de Casos de Uso 3 da aplicação *Mobile* correspondente a analisar uma dada votação.

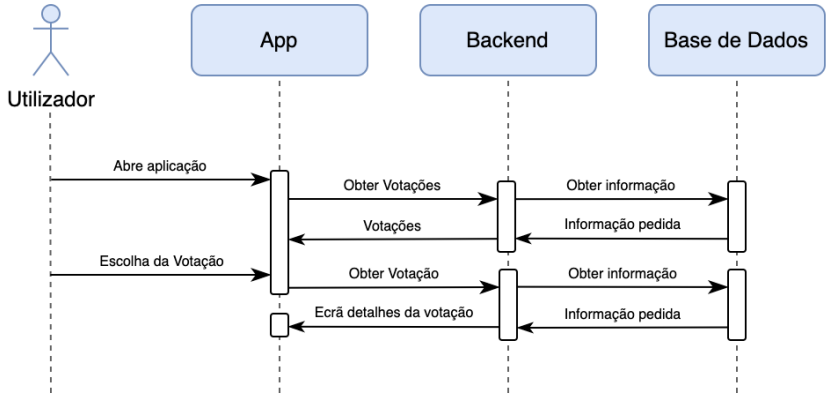


Figura 4.20: DS3 - Diagrama de sequência do casos de uso 3

A Figura 4.21 ilustra o diagrama de sequência relativo ao Cenário de Casos de Uso 4 da aplicação *Mobile* correspondente a analisar os detalhes dos candidatos de uma dada votação.

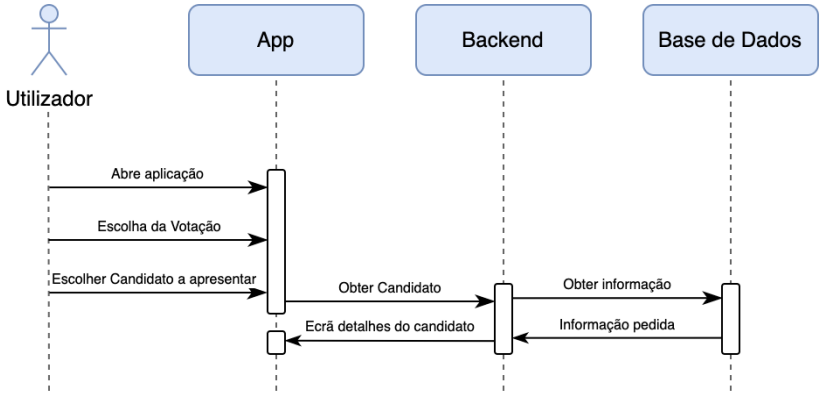


Figura 4.21: DS4 - Diagrama de sequência do casos de uso 4

A Figura 4.22 ilustra o diagrama de sequência relativo ao Cenário de Casos de Uso 5 da aplicação *Mobile* correspondente a votar numa determinada votação.

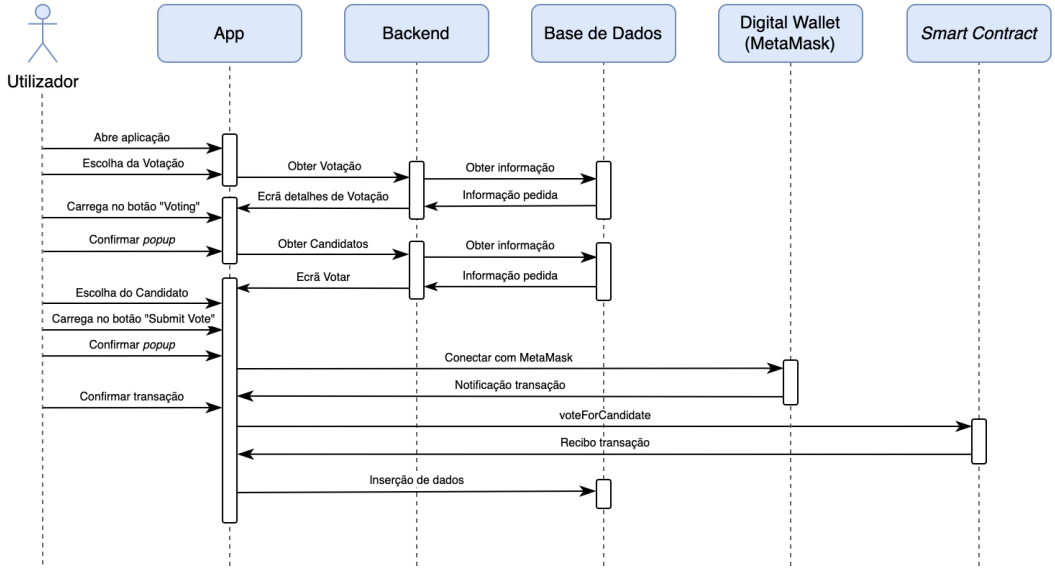


Figura 4.22: DS5 - Diagrama de sequência dos casos de uso 5

A Figura 4.23 ilustra o diagrama de sequência relativo ao Cenário de Casos de Uso 6 da aplicação *Mobile* correspondente a consultar o histórico de votações.

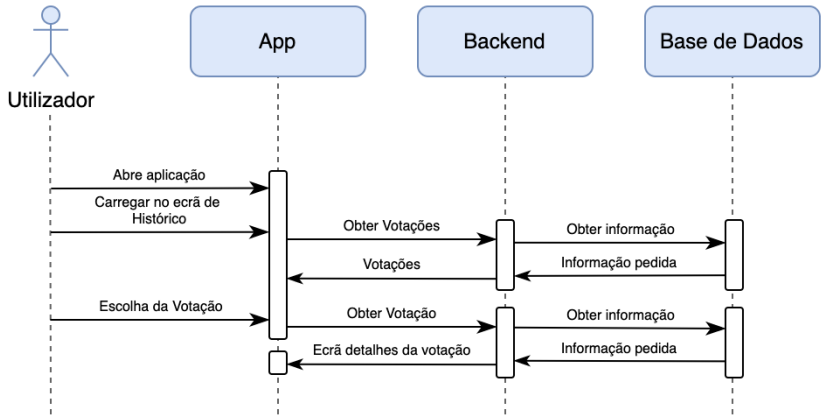


Figura 4.23: DS6 - Diagrama de sequência dos casos de uso 6

A Figura 4.24 ilustra o diagrama de sequência relativo ao Cenário de Casos de Uso 7 da aplicação *Mobile* correspondente a consultar os resultados de uma votação.

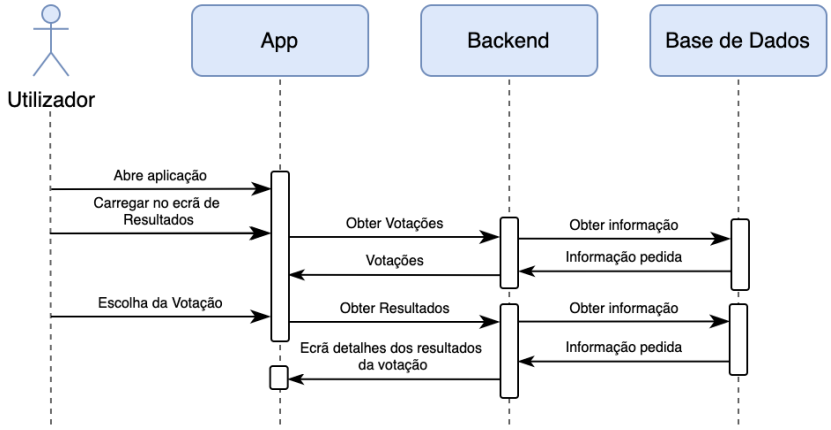


Figura 4.24: DS7 - Diagrama de sequência dos casos de uso 7

Matriz de rastreabilidade entre os casos de uso e diagramas de sequência
A matriz de rastreabilidade interligando os casos de uso da aplicação *mobile* com os diagramas de sequência, está exposta na Tabela 4.10.

Tabela 4.10: Matriz de rastreabilidade entre casos de uso e diagramas de sequência

CU. Id	DS1	DS2	DS3	DS4
CCU1	X			
CCU2	X			
CCU3		X		
CCU4		X		

Continua na próxima página

Tabela 4.10: Matriz de rastreabilidade entre casos de uso e diagramas de sequência (continuação)

CU. Id	DS1	DS2	DS3	DS4
CCU5			X	
CCU6				X
CCU7				X

A modo de exemplo, a Figura 4.25 representa a função de votar presente na aplicação móvel. Primeiramente, caso o utilizador não tiver escolhido nenhum candidato, é atribuído a opção de "Branco" como alegoria a um voto branco. De seguida, é verificado se o dispositivo contém a extensão do MetaMask, sendo que se o mesmo não tiver, este recebe uma notificação para proceder à sua instalação. Após traduzir o ficheiro meta do *smart contract*, que se encontra em dados binários, para texto, procede-se interação com o *smart contract* propriamente dita.

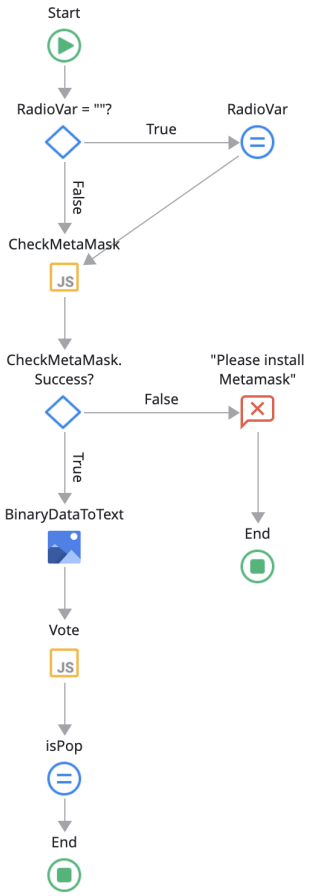


Figura 4.25: Função de votar

Dentro do bloco *javascript* com o nome "Vote" está o código presente no Anexo B.1, onde é iniciado de antemão o *provider* a ser utilizado, neste caso a rede de

testes Ropsten, bem como as variáveis necessárias para a interação com o *smart contract* advindas dos *inputs* deste bloco, sendo elas o endereço do *smart contract* e o *Application Binary Interface* (ABI).

Posteriormente, é solicitado o acesso à carteira digital do utilizador e conectado o *signer* à sua conta do MetaMask, e definido o objeto do *smart contract* usando o seu endereço, ABI e *signer*.

Agora já é possível interagir com as funções dentro do contrato, nomeadamente *voteForCandidate*, onde é passada a versão binária do nome do candidato escolhido. Depois da transação ser executada, o utilizador recebe uma notificação do seu voto e é guardada na base de dados que este eleitor já votou.

4.2.3 Block The Vote - BackOffice

Este sistema foi concebido com a finalidade de permitir ao gestor de votação efetuar as operações básicas Create-Read-Update-Delete (CRUD) na perspetiva de uma votação ou lista de eleitores e candidatos de cada votação, e ainda algumas operações relacionadas com o *smart contract* tais como o *deploy* do contrato e o *download* dos seus componentes. Já o candidato, é capaz de adicionar e editar o seu perfil para cada votação, podendo inserir informações vitais à sua candidatura.

Requisitos

Na seguinte tabela, Tabela 4.11, estão expostos os requisitos aos quais a aplicação *web* deve atender, em que F e NF representam um requisito do tipo Funcional e Não Funcional, respetivamente.

Tabela 4.11: Requisitos da aplicação *web* Block The Vote

Id	Tipo	Descrição
1.F.B	F	Deverá permitir ao gestor de votação criar uma votação.
2.F.B	F	Deverá permitir ao gestor de votação listar uma votação.
3.F.B	F	Deverá permitir ao gestor de votação editar uma votação.
4.F.B	F	Deverá permitir ao gestor de votação eliminar uma votação.
5.F.B	F	Deverá permitir ao candidato inserir dados sobre a sua respetiva candidatura.
6.F.B	F	Deverá permitir ao gestor de votação iniciar uma votação.
7.F.B	F	Após o início de uma votação, o sistema não deverá permitir qualquer alteração sobre a respetiva votação.
8.F.B	F	Deverá permitir ao gestor de votação terminar uma votação.
9.F.B	F	Deverá permitir ao gestor de votação solicitar os resultados de uma votação.
10.F.B	F	Deverá permitir ao gestor de votação efetuar o *deploy* do contrato associado a uma votação.

Continua na próxima página

Tabela 4.11: Requisitos da aplicação *web* Block The Vote (continuação)

Id	Tipo	Descrição
11.F.B	F	Deverá permitir ao gestor de votação aprovar o recenseamento de um eleitor.
12.F.B	F	Deverá permitir ao gestor de votação gerir a lista de eleitores (criar, ler, editar e eliminar).
13.F.B	F	Deverá permitir ao gestor de votação gerir a lista de candidatos (criar, ler, editar e eliminar).
14.F.B	F	Deverá permitir ao gestor de votação inserir um *smart contract*.
15.F.B	F	Deverá permitir ao gestor de votação fazer o *download* do *meta* de um *smart contract*.
16.F.B	F	Deverá permitir ao gestor de votação fazer o *download* do ABI de um *smart contract*.
17.F.B	F	Deverá permitir ao gestor de votação fazer o *download* do código de um *smart contract*.
18.F.B	F	Deverá permitir ao gestor de votação fazer o *download* do *bytecode* de um *smart contract*.
19.NF.B	NF	A aplicação deverá ser capaz de interagir com um *smart contract*.
20.NF.B	NF	A aplicação deverá ser capaz de interagir com uma carteira digital.
21.NF.B	NF	A aplicação deverá suportar *login* próprio para a app.
22.NF.B	NF	A aplicação deverá ter um tema de *User Interface*.
23.NF.B	NF	Deverá implementar um modelo de dados de suporte ao sistema.
24.NF.B	NF	Deverá fornecer APIs de comunicação com o modelo de dados.

Cenários

De seguida, serão expostos diversos cenários tal como o respetivo diagrama de casos de uso (Figura 4.26), resultantes das interações do gestor de votação e candidato com este sistema.

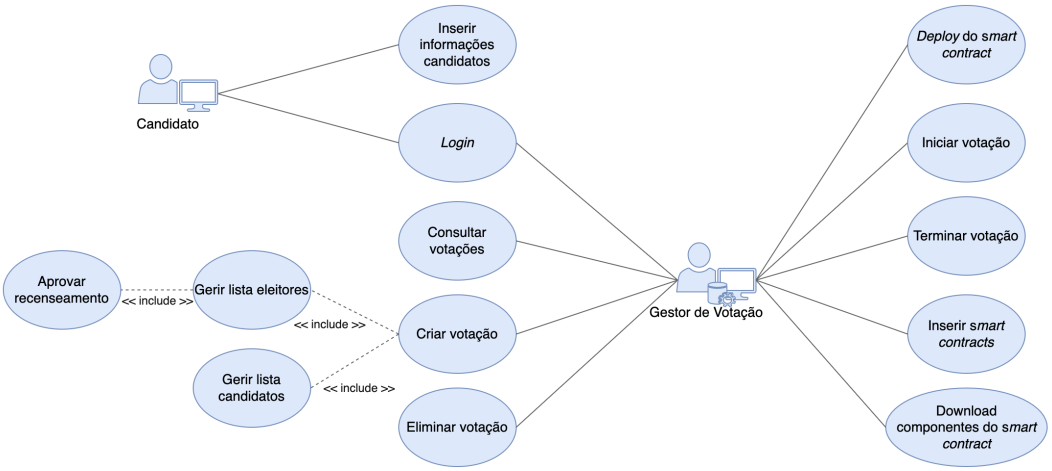


Figura 4.26: Diagrama de casos de uso da aplicação *web*

A Tabela 4.12 retrata o Cenário de Casos de Uso 1 da aplicação *BackOffice* correspondente a efetuar *login* na página *web* associada ao *Backoffice*.

Tabela 4.12: Cenário 1: efetuar *login* na página *web* correspondente ao *Backoffice*

Id	CCU1: *Login* na aplicação
Descrição	O Utilizador pretende efetuar *login* na página *web* correspondente ao *Backoffice*
Ator	Gestor de Votação e Candidato
Cenário	1. O utilizador acede à página *web*; 2. Preenche os campos assinalados com o seu *username* e *password*; 3. Carrega no botão "*Login*".

A Tabela 4.13 retrata o Cenário de Casos de Uso 2 da aplicação *BackOffice* correspondente a consultar as votações.

Tabela 4.13: Cenário 2: consultar as votações

Id	CCU2: Consultar as votações
Descrição	O utilizador pretende consultar as votações com que poderá interagir
Ator	Gestor de Votação e Candidato
Cenário	1. O utilizador acede à página *web*; 2. Seleciona "*Active Votings*" no menu de navegação; 3. Visualiza a lista de votação que lhe é apresentada.

A Tabela 4.14 retrata o Cenário de Casos de Uso 3 da aplicação *BackOffice* correspondente a criar uma votação.

Tabela 4.14: Cenário 3: criar uma votação

Id	CCU3: Criar uma votação
Descrição	O utilizador pretende criar uma votação
Ator	Gestor de Votação
Cenário	1. O utilizador acede à página *web*; 2. Seleciona "*Active Votings*" no menu de navegação; 3. Carrega sobre o botão "*Add Voting*"; 4. Preenche os campos assinalados com os detalhes da votação; 5. Carrega sobre o botão "*Save*".

A Tabela 4.15 retrata o Cenário de Casos de Uso 4 da aplicação *BackOffice* correspondente a analisar uma dada votação.

Tabela 4.15: Cenário 4: analisar uma dada votação

Id	CCU4: Analisar uma dada votação
Descrição	O utilizador pretende analisar os detalhes de uma votação
Ator	Gestor de Votação e Candidato
Cenário	1. O utilizador acede à página *web*; 2. Seleciona "*Active Votings*" no menu de navegação; 3. Carrega sobre a votação pretendida; 4. Apresentação da informação de uma dada votação.

A Tabela 4.16 retrata o Cenário de Casos de Uso 5 da aplicação *BackOffice* correspondente a eliminar uma dada votação.

Tabela 4.16: Cenário 5: eliminar uma dada votação

Id	CCU5: Eliminar uma dada votação
Descrição	O utilizador pretende eliminar uma votação
Ator	Gestor de Votação
Cenário	1. O utilizador acede à página *web*; 2. Seleciona "*Active Votings*" no menu de navegação; 3. Carrega sobre a votação pretendida; 4. Apresentação da informação de uma dada votação; 5. Carrega sobre o botão "*Delete Voting*"; 6. Realiza a confirmação no pop-up exibido.

A Tabela 4.17 retrata o Cenário de Casos de Uso 6 da aplicação *BackOffice* correspondente a um candidato a uma votação.

Tabela 4.17: Cenário 6: adicionar um candidato a uma votação

Id	CCU6: Adicionar um candidato a uma votação
Descrição	O utilizador pretende adicionar um candidato à lista de candidatos de uma votação
Ator	Gestor de Votação
Cenário	1. O utilizador acede à página *web*; 2. Seleciona "*Active Votings*" no menu de navegação; 3. Carrega sobre a votação pretendida; 4. Carrega sobre o botão "*Manage Candidates List*"; 5. Carrega sobre o botão "*Add Candidate*"; 6. Preenche os campos assinalados com os detalhes do candidato; 5. Carrega sobre o botão "*Save*".

A Tabela 4.18 retrata o Cenário de Casos de Uso 7 da aplicação *BackOffice* correspondente a editar o perfil de um candidato numa votação.

Tabela 4.18: Cenário 7: editar o perfil de um candidato numa votação

Id	CCU6: Editar o perfil de um candidato numa votação
Descrição	O utilizador pretende editar a sua informação respetiva a uma votação
Ator	Candidato
Cenário	1. O utilizador acede à página *web*; 2. Seleciona "*Active Votings*" no menu de navegação; 3. Carrega sobre a votação pretendida; 4. Apresentação da informação do candidato; 6. Preenche os campos assinalados com os detalhes do candidato; 5. Carrega sobre o botão "*Save*".

A Tabela 4.19 retrata o Cenário de Casos de Uso 8 da aplicação *BackOffice* correspondente a efetuar o *deploy* do contrato.

Tabela 4.19: Cenário 8: *deploy* do contrato

Id	CCU8: *Deploy* do contrato
Descrição	O utilizador pretende fazer o *deploy* do contrato
Ator	Gestor de Votação
Cenário	1. O utilizador acede à página *web*; 2. Seleciona "*Active Votings*" no menu de navegação; 3. Carrega sobre a votação pretendida; 4. Apresentação da informação de uma dada votação; 3. Carrega sobre o botão "*Deploy*"; 4. Realiza a confirmação da transação no pop-up exibido.

A Tabela 4.20 retrata o Cenário de Casos de Uso 9 da aplicação *BackOffice* correspondente a adicionar um eleitor a uma votação.

Tabela 4.20: Cenário 9: adicionar um eleitor a uma votação

Id	CCU9: Adicionar um eleitor a uma votação
Descrição	O utilizador pretende adicionar um eleitor à lista de eleitores de uma votação
Ator	Gestor de Votação
Cenário	1. O utilizador acede à página *web*; 2. Seleciona "*Active Votings*" no menu de navegação; 3. Carrega sobre a votação pretendida; 4. Carrega sobre o botão "*Manage Voters List*"; 5. Carrega sobre o botão "*Add Voter*"; 6. Preenche os campos assinalados com os detalhes do eleitor; 5. Carrega sobre o botão "*Save*".

A Tabela 4.21 retrata o Cenário de Casos de Uso 10 da aplicação *BackOffice* correspondente a aprovar o recenseamento de um eleitor.

Tabela 4.21: Cenário 10: aprovar o recenseamento de um eleitor

Id	CCU10: Aprovar o recenseamento de um eleitor
Descrição	O utilizador pretende aprovar o recenseamento de um eleitor, adicionando-o na lista de eleitores elegíveis
Ator	Gestor de Votação
Cenário	1. O utilizador acede à página *web*; 2. Seleciona "*Active Votings*" no menu de navegação; 3. Carrega sobre a votação pretendida; 4. Carrega sobre o botão "*Manage Voters List*"; 5. Carrega sobre o botão "*Aprove Registration*" do eleitor correspondente; 6. Realiza a confirmação da transação no pop-up exibido.

A Tabela 4.22 retrata o Cenário de Casos de Uso 11 da aplicação *BackOffice* correspondente a iniciar uma votação.

Tabela 4.22: Cenário 11: iniciar uma votação

Id	CCU11: Iniciar uma votação
Descrição	O utilizador pretende iniciar uma votação
Ator	Gestor de Votação
Cenário	1. O utilizador acede à página *web*; 2. Seleciona "*Active Votings*" no menu de navegação; 3. Carrega sobre a votação pretendida; 4. Carrega sobre o botão "*Start Voting*"; 5. Realiza a confirmação no pop-up exibido.

A Tabela 4.23 retrata o Cenário de Casos de Uso 12 da aplicação *BackOffice* correspondente a terminar uma votação.

Tabela 4.23: Cenário 12: terminar uma votação

Id	CCU12: Terminar uma votação
Descrição	O utilizador pretende terminar uma votação
Ator	Gestor de Votação
Cenário	1. O utilizador acede à página *web*; 2. Seleciona "*Active Votings*" no menu de navegação; 3. Carrega sobre a votação pretendida; 4. Carrega sobre o botão "*End Voting*"; 5. Realiza a confirmação no pop-up exibido.

A Tabela 4.24 retrata o Cenário de Casos de Uso 13 da aplicação *BackOffice* correspondente a consultar o histórico de votações.

Tabela 4.24: Cenário 13: consultar o histórico de votações

Id	CCU13: Consultar o histórico de votações
Descrição	O utilizador pretende verificar as votações terminadas
Ator	Gestor de Votação e Candidato
Cenário	1. O utilizador acede à página *web*; 2. Seleciona "*History*" no menu de navegação; 3. Carrega sobre a votação pretendida; 4. Apresentação da informação de uma dada votação.

A Tabela 4.25 retrata o Cenário de Casos de Uso 14 da aplicação *BackOffice* correspondente a consultar os resultados de uma determinada votação.

Tabela 4.25: Cenário 14: consultar resultados votações

Id	CCU14: Consultar resultados votações
Descrição	O utilizador pretende consultar os resultados de uma determinada votação
Ator	Gestor de Votação e Candidato
Cenário	1. O utilizador acede à página *web*; 2. Seleciona "*Results*" no menu de navegação; 3. Carrega sobre a votação pretendida; 4. Apresentação da informação relativamente aos resultados da votação.

A Tabela 4.26 retrata o Cenário de Casos de Uso 15 da aplicação *BackOffice* correspondente a inserir um contrato na aplicação *Backoffice*.

Tabela 4.26: Cenário 15: inserir um contrato

Id	CCU15: Inserir um contrato
Descrição	O utilizador pretende inserir um contrato
Ator	Gestor de Votação
Cenário	1. O utilizador acede à página *web*; 2. Seleciona "*Contracts*" no menu de navegação; 3. Carrega sobre o botão "*Add Contract*"; 4. Preenche os campos assinalados com os detalhes do contrato; 5. Carrega sobre o botão "*Save*".

A Tabela 4.27 retrata o Cenário de Casos de Uso 16 da aplicação *BackOffice* correspondente a efetuar o *download* dos componentes de um contrato.

Tabela 4.27: Cenário 16: *download* dos componentes de um contrato

Id	CCU16: *Download* dos componentes de um contrato
Descrição	O utilizador pretende fazer o *download* dos componentes de um contrato
Ator	Gestor de Votação
Cenário	1. O utilizador acede à página *web*; 2. Seleciona "*Contracts*" no menu de navegação; 3. Carrega sobre o contrato pretendido; 4. Carrega sobre o componente pretendido.

Wireframes

Tendo por base os requisitos previamente recolhidos e os cenários existentes, procedeu-se ao desenvolvimento das respetivas *Wireframes*, recorrendo à ferramenta Balsamiq.

CCU1. Como Utilizador, pretendo efetuar *login* na página *web* correspondente ao *Backoffice*.

Detalhes: O utilizador pode efetuar o *login* na aplicação, ilustrado na Figura 4.27, utilizando o seu *username* e *password*.

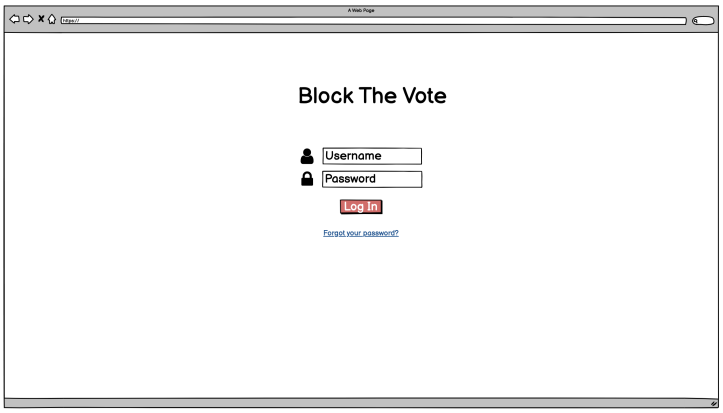


Figura 4.27: *Wireframe* de *login* da aplicação *Web*

CCU2. Como Utilizador, pretendo consultar as votações com que poderei interagir.

CCU3. Como Utilizador, pretendo criar uma votação.

Detalhes: Após o *login* na aplicação, serão apresentadas as votações com as quais o utilizador conectado poderá interagir (Figura 4.28). Na eventualidade de o utilizador ser um *Gestor de Votação*, este poderá criar uma votação.

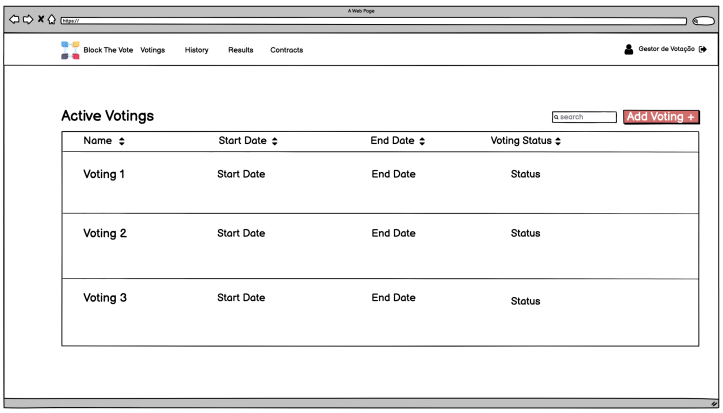


Figura 4.28: *Wireframe* da página do Gestor de Votação

CCU4. Como Utilizador, pretendo analisar os detalhes de uma votação.
CCU5. Como Utilizador, pretendo eliminar uma votação.
CCU8. Como Utilizador, pretendo fazer o *deploy* de um contrato.
CCU11. Como Utilizador, pretendo iniciar uma votação.
CCU12. Como Utilizador, pretendo terminar uma votação.

Detalhes: Para consultar uma votação, o utilizador tem de aceder à página de detalhes da mesma, exposta na Figura 4.29. Caso o ator seja do tipo *Candidato*, só lhe é permitido consultar os detalhes de uma votação. Porém, se for um *Gestor de Votação*, este poderá, adicionalmente, editar e eliminar uma votação, ou ainda interagir com um *smart contract*, realizando o seu *deploy*, a partir do qual é permitido iniciar e terminar uma votação.

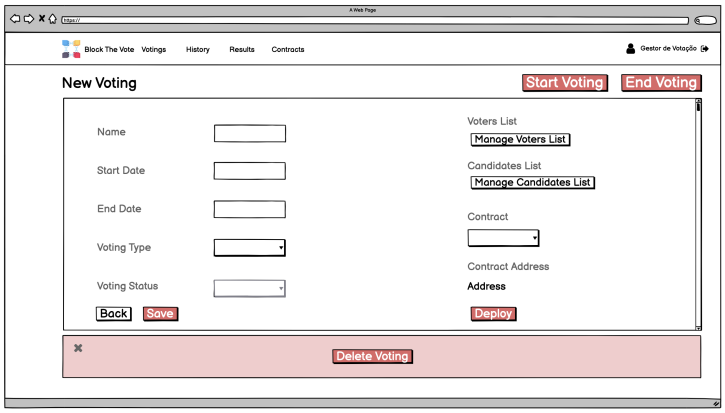
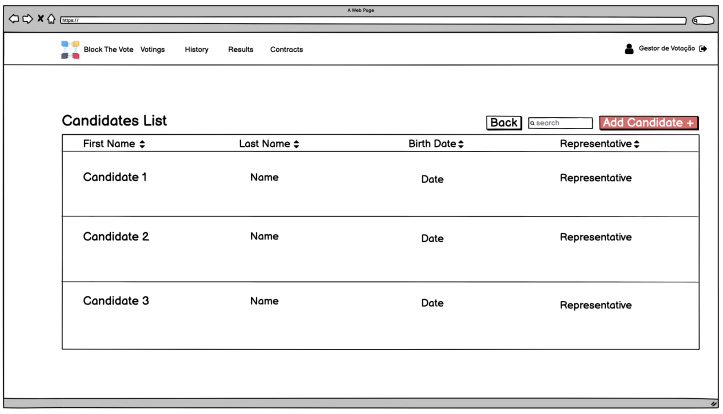


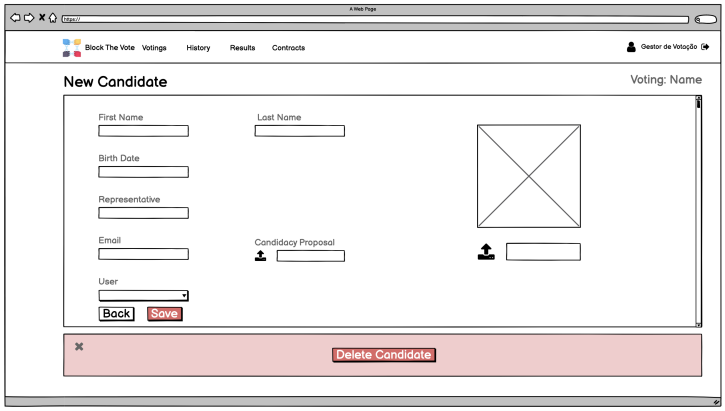
Figura 4.29: *Wireframe* da página de detalhes de uma votação

CCU6. Como Utilizador, pretendo adicionar um candidato à lista de candidatos de uma votação.
CCU7. Como Candidato, pretendo editar a minha informação respetiva a uma votação.

Detalhes: Para adicionar um candidato, o utilizador tem de aceder à página da Lista dos Candidatos da respetiva votação, exposta na Figura 4.30a. Aqui, o *Gestor de Votação* poderá criar, consultar, editar ou eliminar um candidato (Figura 4.30b). No caso do ator ser do tipo *Candidato*, este poderá apenas editar as suas informações.



(a) Ecrã da Lista de Candidatos



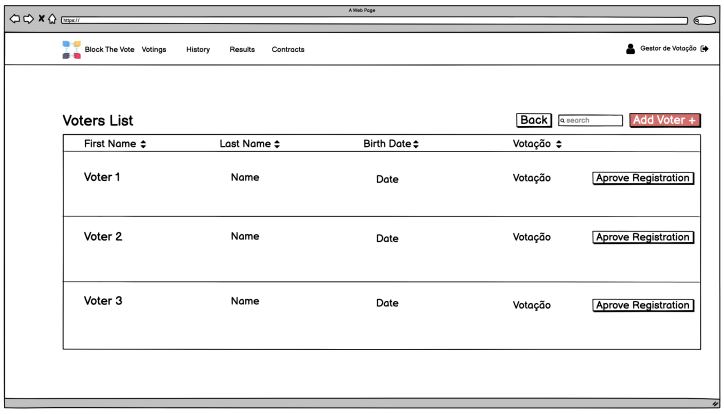
(b) Ecrã dos detalhes de um Candidato

Figura 4.30: *Wireframe* da página da Lista de Candidatos (a) e respetivos detalhes (b)

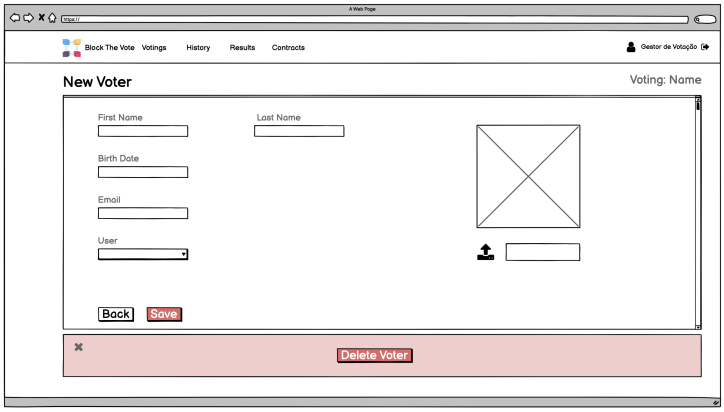
CCU9. Como Utilizador, pretendo adicionar um eleitor à lista de eleitores de uma votação.

CCU10. Como Utilizador, pretendo aprovar o recenseamento de um eleitor.

Detalhes: Para adicionar um eleitor, o utilizador tem de aceder à página da Lista dos Eleitores da respetiva votação, exposta na Figura 4.31a. Aqui, o *Gestor de Votação* poderá criar, consultar, editar, eliminar ou aprovar o recenseamento de um eleitor com o auxílio da sua página de detalhes, Figura 4.31b.



(a) Ecrã da Lista de Eleitores



(b) Ecrã dos detalhes de um Eleitor

Figura 4.31: *Wireframe* da página da Lista de Eleitores (a) e respetivos detalhes (b)

CCU13. Como Utilizador, pretendo verificar as votações terminadas.

Detalhes: Para verificar as votações terminadas, basta o utilizador navegar até à página "*History*", como é possível observar na Figura 4.32, e, se pretender, selecionar a votação pretendida para obter mais detalhes.

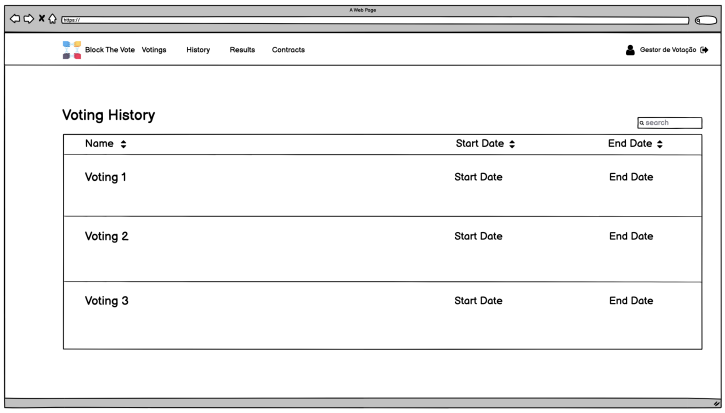
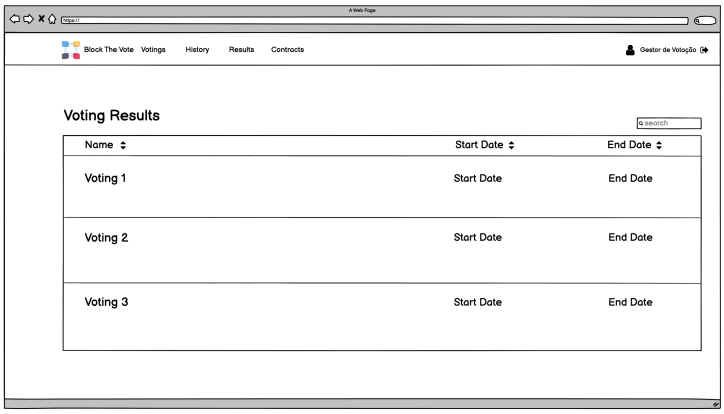


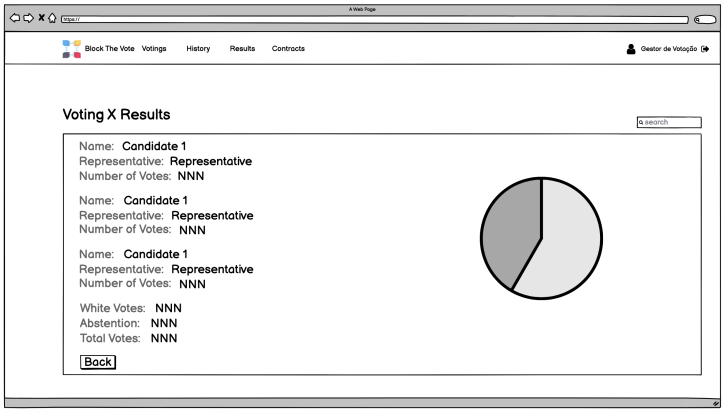
Figura 4.32: *Wireframe* da página de Histórico referente às votações terminadas

CCU14. Como Utilizador, pretendo consultar os resultados de uma determinada votação.

Detalhes: De modo a consultar os resultados de uma votação, o Utilizador deve navegar até à página "*Results*" (Figura 4.33a) e, posteriormente, escolher uma votação. Neste ecrã é possível averiguar a contagem dos votos para cada candidato (Figura 4.33b).



(a) Ecrã de Resultados das Votações



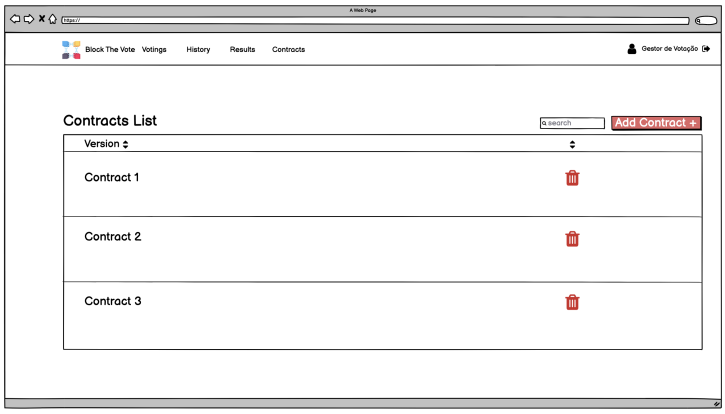
(b) Ecrã de detalhes de Resultados das Votações

Figura 4.33: *Wireframe* da página de Resultados referente às votações terminadas (a) e respetivos detalhes (b)

CCU15. Como Utilizador, pretendo inserir um contrato.

CCU16. Como Utilizador, pretendo fazer o *download* de componentes de um contrato.

Detalhes: Para inserir um contrato, basta o utilizador navegar até à página "*Contracts*", Figura 4.34a, onde este poderá eliminar ou inserir contratos, através da sua página de detalhes (Figura 4.34b). Aqui, também poderá efetuar o *download* de componentes de um contrato, tal como o seu código correspondente, o ABI, o ficheiro *meta* e o *bytecode*.



(a) Ecrã de Contratos

(b) Ecrã de detalhes de Contratos

Figura 4.34: *Wireframe* da página de Contratos (a) e respetivos detalhes (b)

Matriz de rastreabilidade entre requisitos e os casos de uso

A seguinte tabela, Tabela 4.28, ilustra a matriz de rastreabilidade entre os requisitos da aplicação *web* com os casos de uso respetíveis, em que CX representa o cenário do caso de uso X e AP representa a arquitetura do projeto exposta nas Figuras 4.7 e 4.8.

Tabela 4.28: Matriz de rastreabilidade entre requisitos e casos de uso

Req. Id	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	AP
1.F.B			X														
2.F.B		X															
3.F.B				X													
4.F.B					X												
5.F.B							X										
6.F.B											X						
7.F.B											X						
8.F.B												X					
9.F.B														X			
10.F.B								X									
11.F.B										X							
12.F.B									X								
13.F.B						X											
14.F.B															X		

Continua na próxima página

Tabela 4.28: Matriz de rastreabilidade entre requisitos e casos de uso (continuação)

Req. Id	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	AP
15.F.B																X	
16.F.B																X	
17.F.B																X	
18.F.B																X	
19.NF.B								X		X	X	X				X	
20.NF.B								X		X	X	X				X	
21.NF.B	X																
22.NF.B																	X
23.NF.B																	X
24.NF.B																	X

Diagramas de Sequência

Os diagramas de sequência representativos dos casos de uso do sistema *BackOffice* estão evidenciados nas seguintes figuras.

A Figura 4.35 ilustra o diagrama de sequência relativo ao Cenário de Casos de Uso 1 da aplicação *BackOffice* correspondente a efetuar *login* na página *web* associada ao *Backoffice*.

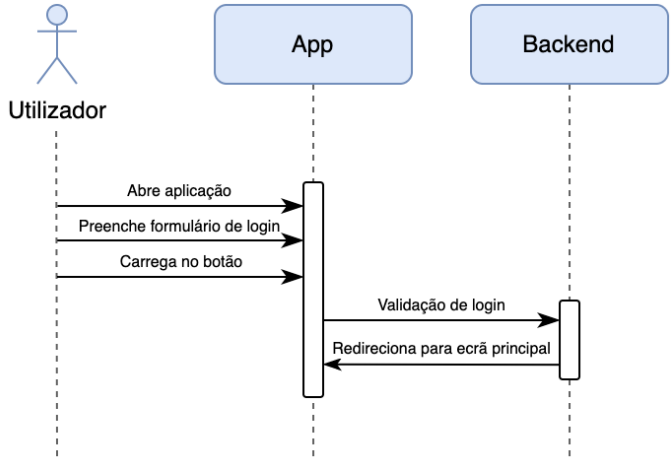


Figura 4.35: DS1 - Diagrama de sequência dos casos de uso 1

A Figura 4.36 ilustra o diagrama de sequência relativo ao Cenário de Casos de Uso 2 da aplicação *BackOffice* correspondente a consultar as votações.

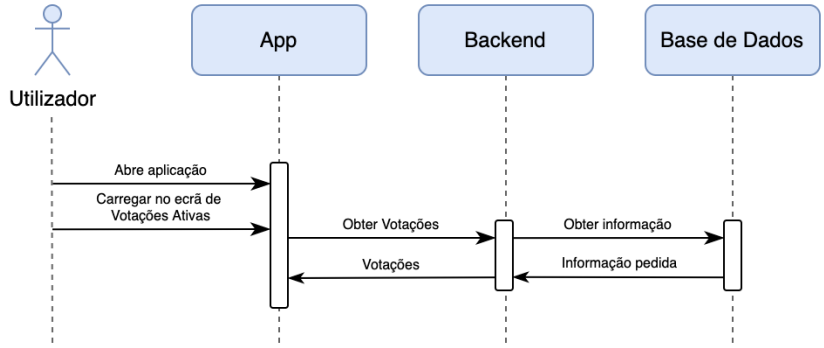


Figura 4.36: DS2 - Diagrama de sequência dos casos de uso 2

A Figura 4.37 ilustra o diagrama de sequência relativo ao Cenário de Casos de Uso 3 da aplicação *BackOffice* correspondente a criar uma votação.

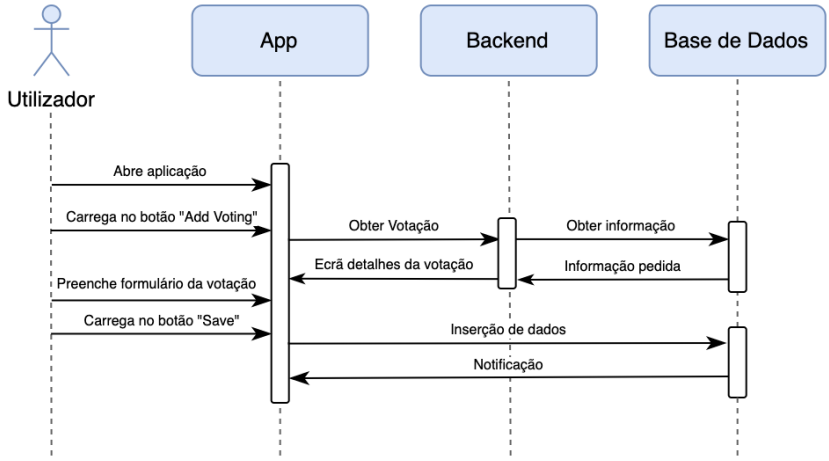


Figura 4.37: DS3 - Diagrama de sequência do casos de uso 3

A Figura 4.38 ilustra o diagrama de sequência relativo ao Cenário de Casos de Uso 4 da aplicação *BackOffice* correspondente a analisar uma dada votação.

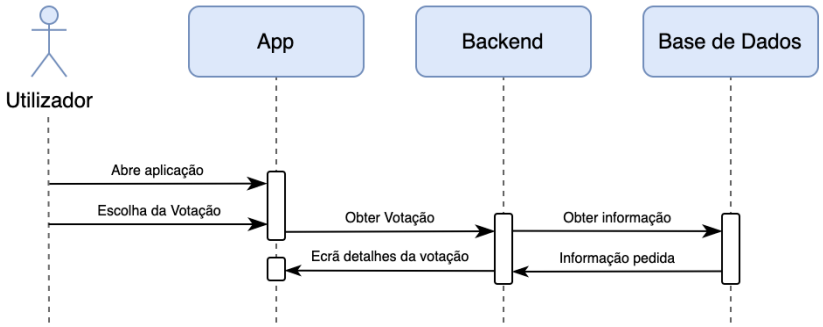


Figura 4.38: DS4 - Diagrama de sequência dos casos de uso 4

A Figura 4.39 ilustra o diagrama de sequência relativo ao Cenário de Casos de Uso 5 da aplicação *BackOffice* correspondente a eliminar uma dada votação.

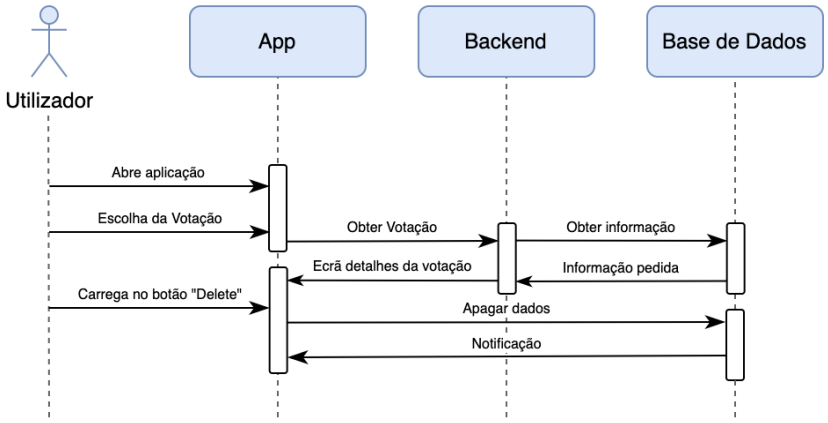


Figura 4.39: DS5 - Diagrama de sequência do casos de uso 5

A Figura 4.40 ilustra o diagrama de sequência relativo ao Cenário de Casos de Uso 6 da aplicação *BackOffice* correspondente a um candidato a uma votação.

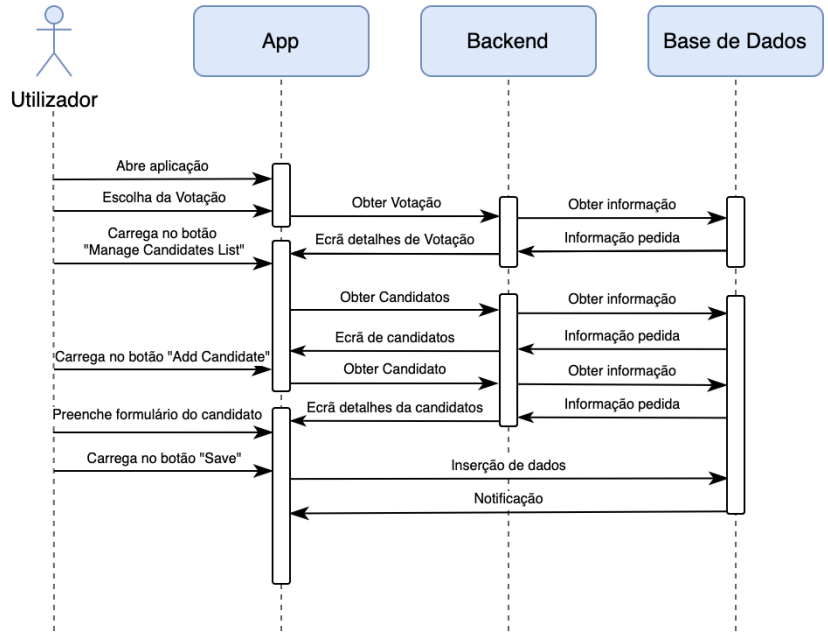


Figura 4.40: DS6 - Diagrama de sequência dos casos de uso 6

A Figura 4.41 ilustra o diagrama de sequência relativo ao Cenário de Casos de Uso 7 da aplicação *BackOffice* correspondente a editar o perfil de um candidato numa votação.

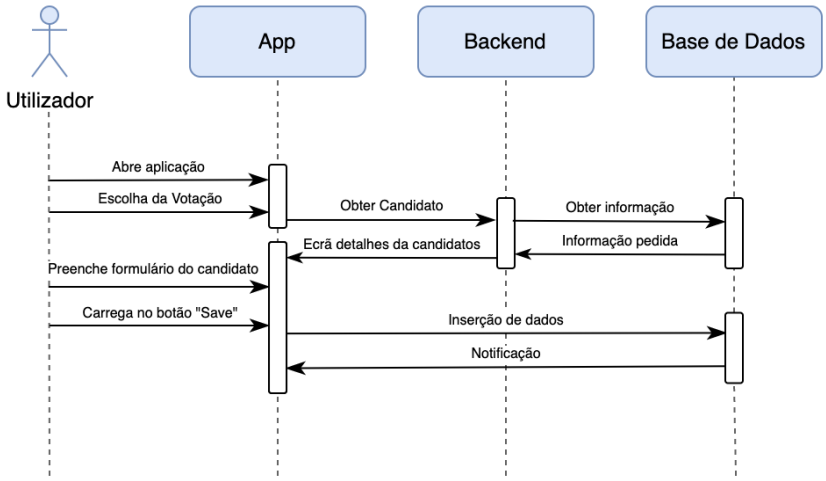


Figura 4.41: DS7 - Diagrama de sequência dos casos de uso 7

A Figura 4.42 ilustra o diagrama de sequência relativo ao Cenário de Casos de Uso 8 da aplicação *BackOffice* correspondente a efetuar o *deploy* do contrato.

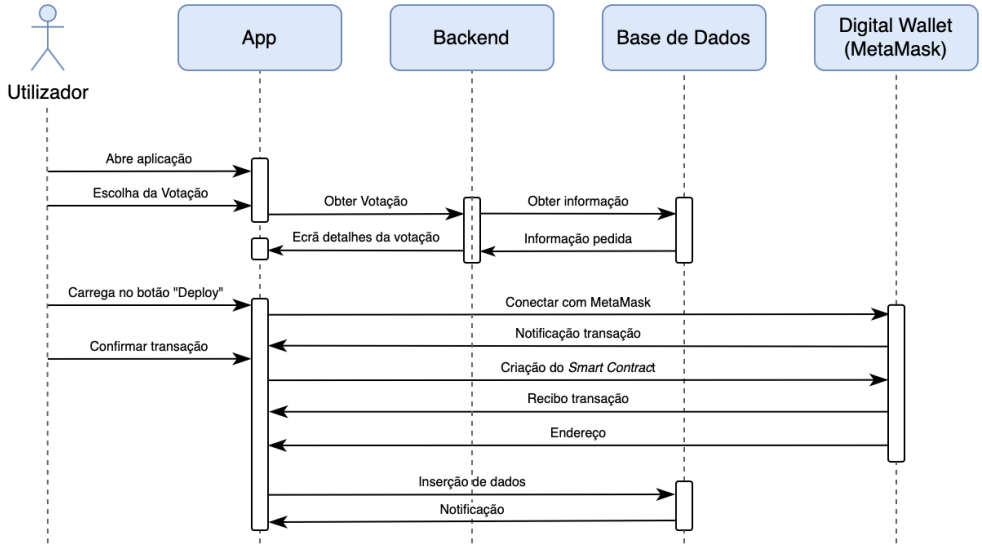


Figura 4.42: DS8 - Diagrama de sequência dos casos de uso 8

A Figura 4.43 ilustra o diagrama de sequência relativo ao Cenário de Casos de Uso 9 da aplicação *BackOffice* correspondente a adicionar um eleitor a uma votação.

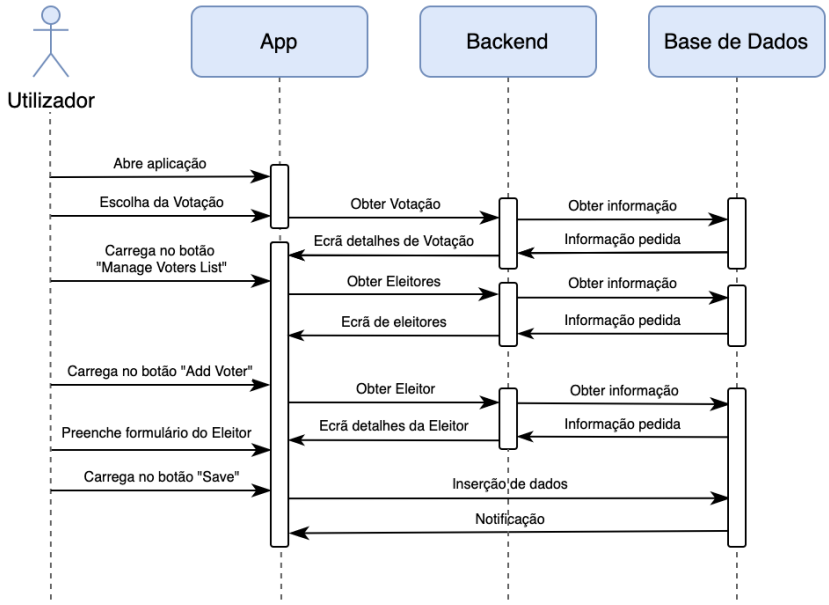


Figura 4.43: DS9 - Diagrama de sequência dos casos de uso 9

A Figura 4.44 ilustra o diagrama de sequência relativo ao Cenário de Casos de Uso 10 da aplicação *BackOffice* correspondente a aprovar o recenseamento de um eleitor.

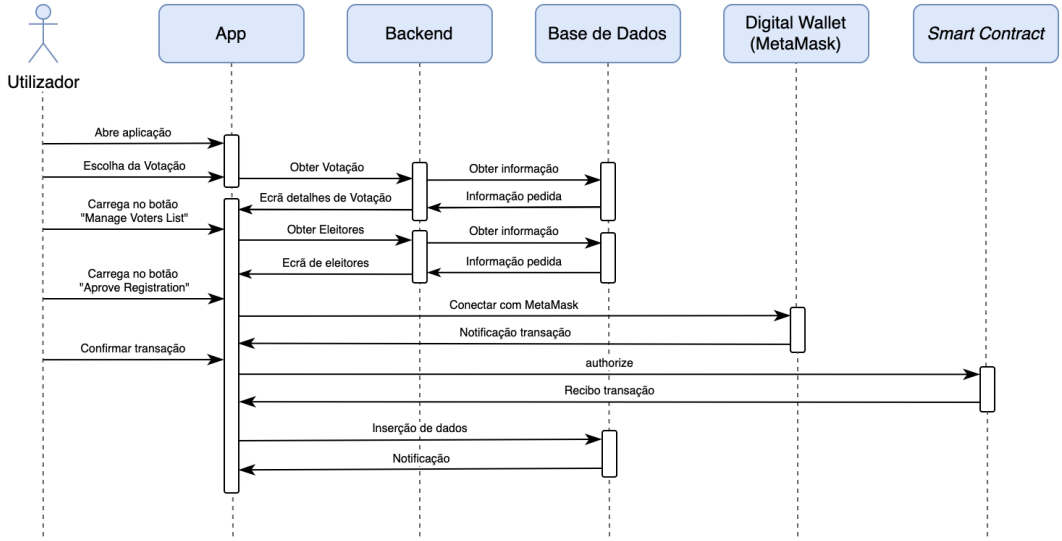


Figura 4.44: DS10 - Diagrama de sequência dos casos de uso 10

A Figura 4.45 ilustra o diagrama de sequência relativo ao Cenário de Casos de Uso 11 da aplicação *BackOffice* correspondente a iniciar uma votação.

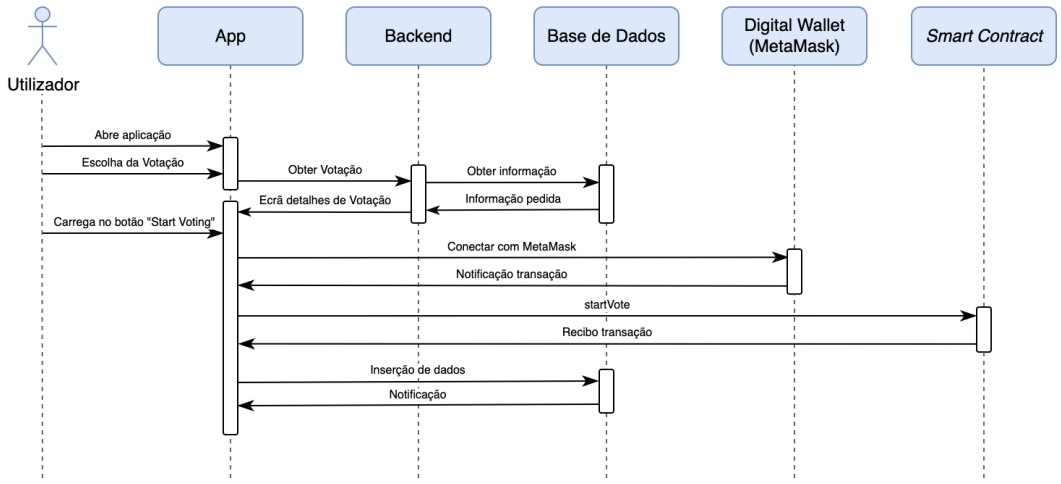


Figura 4.45: DS11 - Diagrama de sequência dos casos de uso 11

A Figura 4.46 ilustra o diagrama de sequência relativo ao Cenário de Casos de Uso 12 da aplicação *BackOffice* correspondente a terminar uma votação.

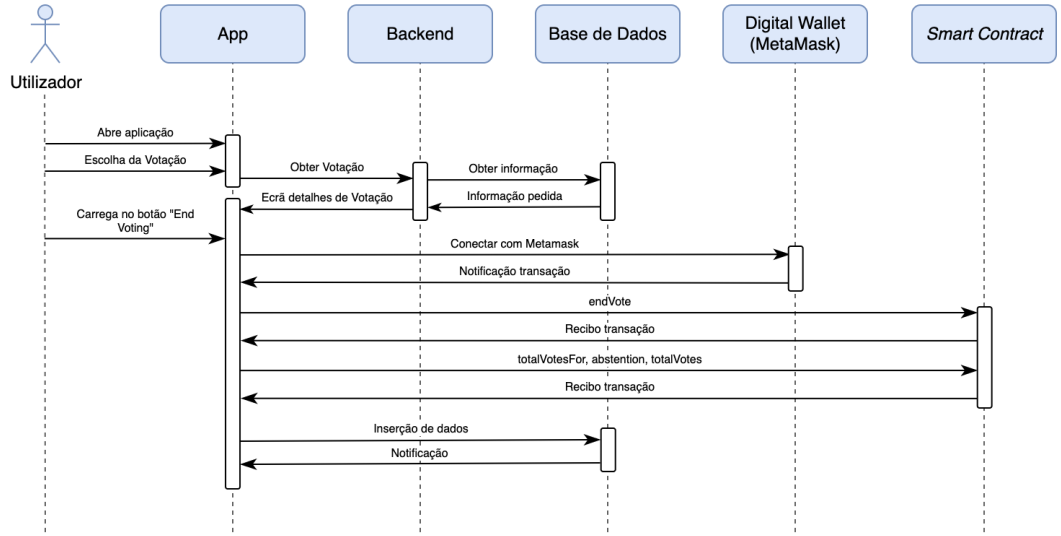


Figura 4.46: DS12 - Diagrama de sequência dos casos de uso 12

A Figura 4.47 ilustra o diagrama de sequência relativo ao Cenário de Casos de Uso 13 da aplicação *BackOffice* correspondente a consultar o histórico de votações.

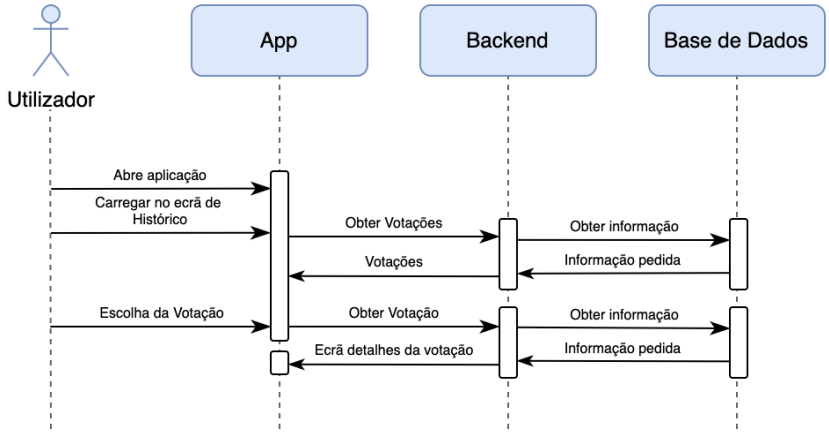


Figura 4.47: DS13 - Diagrama de sequência dos casos de uso 13

A Figura 4.48 ilustra o diagrama de sequência relativo ao Cenário de Casos de Uso 14 da aplicação *BackOffice* correspondente a consultar os resultados de uma determinada votação.

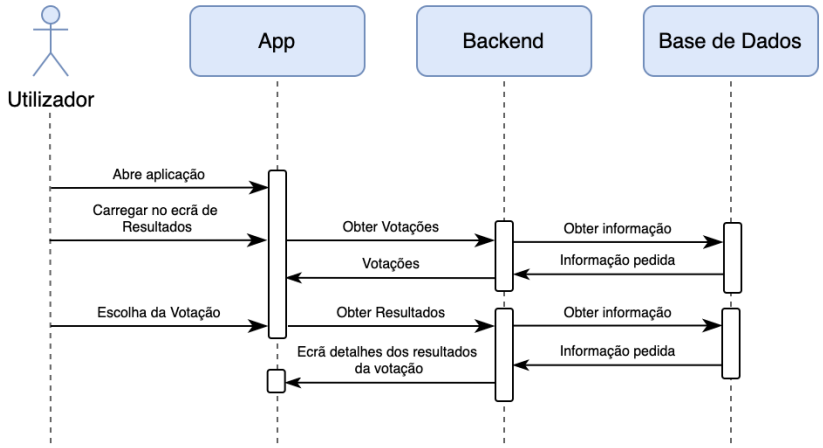


Figura 4.48: DS14 - Diagrama de sequência dos casos de uso 14

A Figura 4.49 ilustra o diagrama de sequência relativo ao Cenário de Casos de Uso 15 da aplicação *BackOffice* correspondente a inserir um contrato na aplicação *Backoffice*.

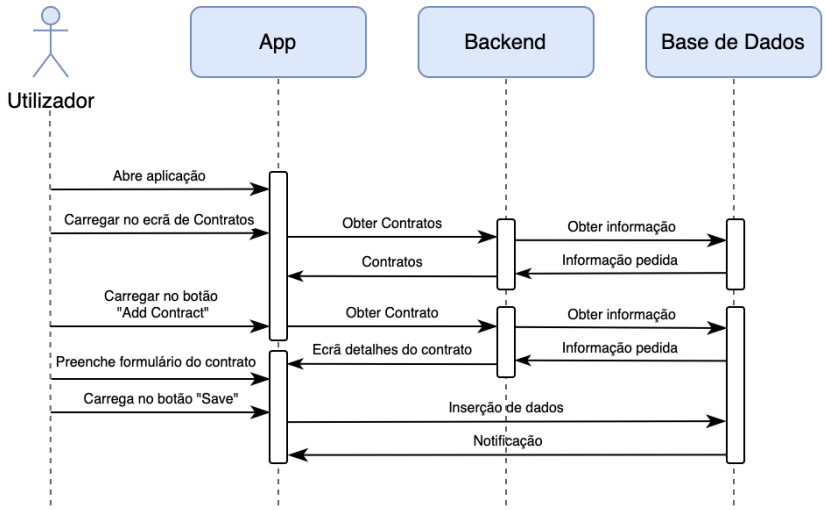


Figura 4.49: DS15 - Diagrama de sequência dos casos de uso 15

A Figura 4.50 ilustra o diagrama de sequência relativo ao Cenário de Casos de Uso 16 da aplicação *BackOffice* correspondente a efetuar o *download* dos componentes de um contrato.

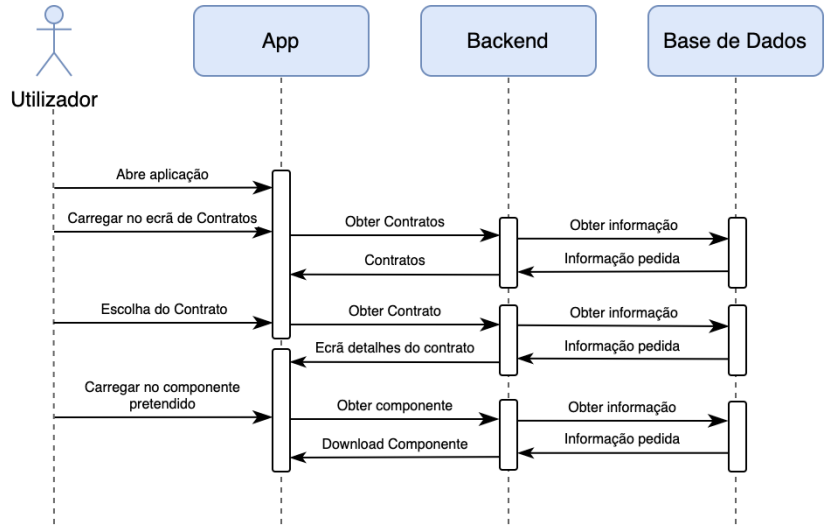


Figura 4.50: DS16 - Diagrama de sequência dos casos de uso 16

Matriz de rastreabilidade entre os casos de uso e diagramas de sequência A matriz de rastreabilidade interligando os casos de uso da aplicação *web* com os diagramas de sequência, está exposta na Tabela 4.29.

Tabela 4.29: Matriz de rastreabilidade entre casos de uso e diagramas de sequência

CU. Id	DS1	DS2	DS3	DS4	DS5	DS6
CCU1	X					
CCU2	X					
CCU3	X					
CCU4		X				
CCU5	X					
CCU6			X			
CCU7			X			
CCU8						
CCU9			X			
CCU10			X			
CCU11				X		
CCU12				X		
CCU13					X	
CCU14					X	
CCU15						X
CCU16						X

Para realizar o *deploy* (Figura 4.51), é necessário compilar a priori os candidatos associados à respetiva votação e verificar se o dispositivo contém a extensão do MetaMask. Posto isto, é possível traduzir o ficheiro meta do *smart contract*, que se encontra em dados binários, para texto e proceder-se à interação com o mesmo.

Dentro do bloco *javascript* com o nome "DeployContract" está o código presente no Anexo B.2, onde é iniciado de antemão o *provider* a ser utilizado, neste caso a rede de testes Ropsten, bem como as variáveis necessárias para a interação com o *smart contract* advindas dos *inputs* deste bloco, sendo elas o endereço do *smart contract* e a lista de candidatos associados a esta votação.

Seguidamente, é solicitado o acesso à carteira digital do utilizador e conectado o *signer* à sua conta do MetaMask, pelo que é definido uma instância do tipo *ContractFactory*, através do ABI do *smart contract*, do seu *bytecode* e do *signer*.

Em seguida é realizado o *deploy* do *smart contract*, tendo em consideração a sua lista de candidatos, onde é despoletada a função *Constructor* no *smart contract*, que será analisada mais tarde. Depois da transação ser executada, o utilizador recebe uma notificação que o *deploy* do *smart contract* foi realizado com sucesso, informando qual é o seu endereço e o mesmo é guardado na base de dados.

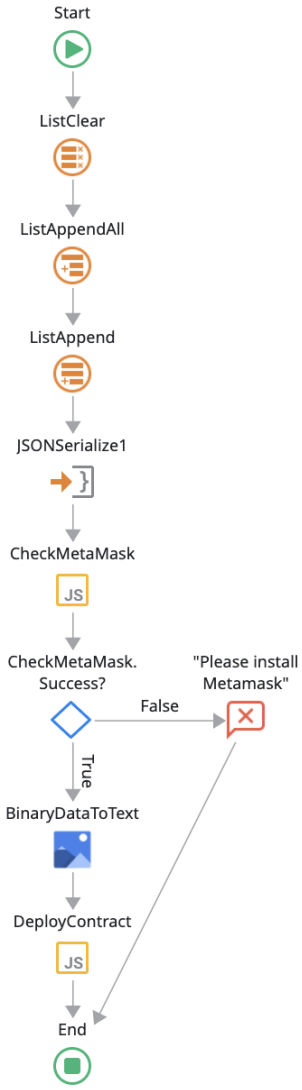


Figura 4.51: Função de *Deploy* do *smart contract*

Tendo como exemplo o cenário de iniciar uma votação (Figura 4.52), em primeiro lugar, é verificado se o dispositivo contém a extensão do MetaMask, sendo que se o mesmo não tiver, este recebe uma notificação para proceder à sua instalação. Após traduzir o ficheiro meta do *smart contract*, que se encontra em dados binários, para texto, procede-se interação com o *smart contract* propriamente dita.

Dentro do bloco *javascript* com o nome "IniciarVotacao" está o código presente no Anexo B.3, onde é iniciado de antemão o *provider* a ser utilizado e as variáveis necessárias para a interação com o *smart contract* advindas dos *inputs* deste bloco, sendo elas o endereço do *smart contract* e o ABI.

À semelhança da função anterior (Vote) descrita na secção 4.2.2 e exposta no Anexo B.1, é solicitado o acesso à carteira digital do utilizador e conectado o *signer*

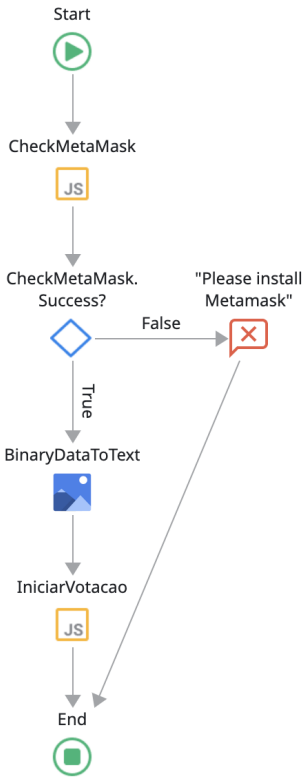


Figura 4.52: Função de Iniciar Votação

à sua conta do MetaMask, e definido o objeto do *smart contract* usando seu endereço de *smart contract*, ABI e *signer*.

Neste momento, é accionada a função denominada de *startVote* dentro do *smart contract*, explicada a posteriori, e o utilizador recebe a notificação que a votação foi iniciada, sendo essa informação também guardada na base de dados.

4.2.4 Block The Vote - *Smart Contract*

Este sistema contém o *smart contract* que interliga as aplicações *mobile* e *web* com a *blockchain* utilizada. Os atores intervenientes (gestor de votação e eleitor) podem essencialmente adicionar eleitores, iniciar e terminar uma votação, votar e obter os resultados de uma votação.

Requisitos

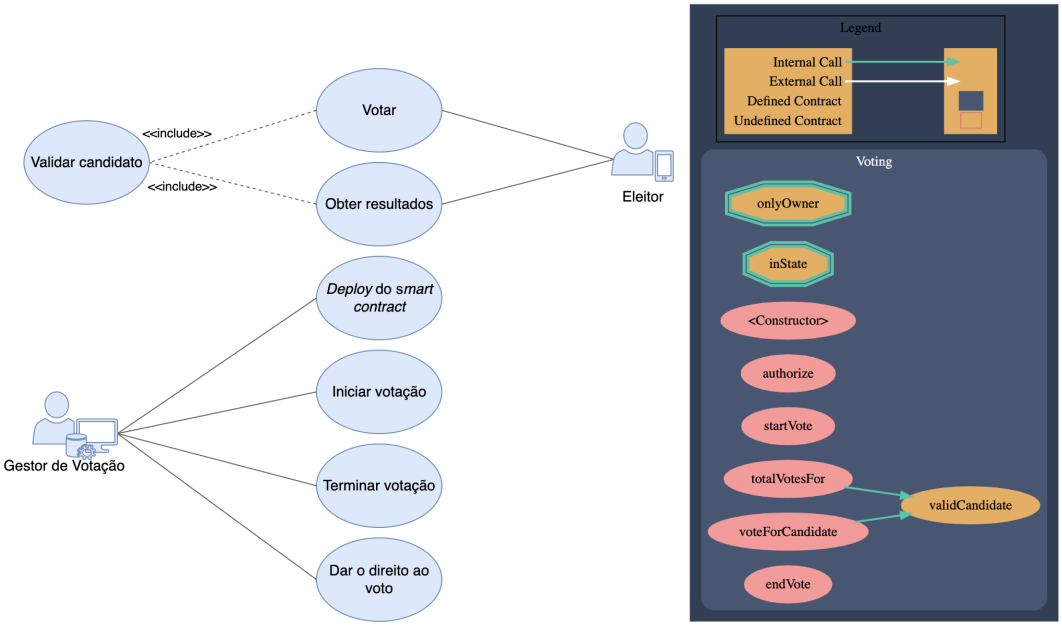
Na seguinte tabela, Tabela 4.30, estão expostos os requisitos aos quais o *smart contract* deve atender, em que F e NF representam um requisito do tipo Funcional e Não Funcional, respetivamente.

Tabela 4.30: Requisitos do *smart contract* Block The Vote

Id	Tipo	Descrição
1.F.S	F	Deverá permitir apenas ao gestor de votação iniciar uma votação, somente após esta ter sido criada.
2.F.S	F	Deverá permitir apenas ao gestor de votação terminar uma votação, somente quando esta estiver a decorrer.
3.F.S	F	Deverá permitir apenas ao gestor de votação dar a um eleitor a permissão de votar, somente após uma votação ter sido criada.
4.F.S	F	Deverá permitir que os eleitores votem somente num candidato.
5.F.S	F	Deverá permitir que os eleitores votem somente uma vez.
6.F.S	F	Deverá permitir a solicitação dos resultados de uma votação.
7.F.S	F	Deverá associar a pessoa que fez deploy do contrato como o seu dono.
8.F.S	F	Deverá validar um candidato.
9.F.S	F	Deverá aceitar uma lista de candidatos, aquando do seu deploy.
10.F.S	F	Deverá fazer a contagem dos votos de uma votação.

Cenários

De seguida, serão expostos diversos cenários tal como o respetivo diagrama de casos de uso (Figura 4.53a), resultantes das interações do gestor de votação e eleitor com este sistema, através das funções expostas na Figura 4.53b.



(a) Diagrama de casos de uso

(b) Diagrama das funções

Figura 4.53: Diagrama de casos de uso (4.53a) e funções (4.53b) do *Smart Contract*

A Tabela 4.31 retrata o Cenário de Casos de Uso 1 do *smart contract* correspondente a efetuar o *deploy* de um contrato.

Tabela 4.31: Cenário 1: fazer *deploy* do contrato

Id	CCU1: *Deploy* do contrato
Descrição	O Utilizador pretende fazer *deploy* do contrato
Ator	Gestor de Votação
Cenário	1. O gestor de votação (app *BackOffice*) chama a função *constructor*; 2. O *smart contract* inicializa um *array* com a lista dos candidatos, tendo em conta os candidatos submetidos; 3. O *smart contract* define o endereço que assinou a transação como o dono (*owner*) do *smart contract*; 4. O *smart contract* define o estado da votação como "*Created*".

A Tabela 4.32 retrata o Cenário de Casos de Uso 2 do *smart contract* correspondente a dar o direito ao voto a certos eleitores.

Tabela 4.32: Cenário 2: dar direito ao voto

Id	CCU2: Dar direito ao voto
Descrição	O utilizador pretende dar o direito ao voto a certos eleitores
Ator	Gestor de Votação
Cenário	1. O gestor de votação (app *BackOffice*) chama a função *authorize*; 2. O *smart contract* verifica se a votação está no estado "*Created*"; 3. O *smart contract* verifica se foi o *owner* que chamou a função; 4. O *smart contract* verifica se o eleitor já votou; 5. O *smart contract* atribui um peso ao eleitor; 6. O *smart contract* incrementa a variável *totalvotes*.

A Tabela 4.33 retrata o Cenário de Casos de Uso 3 do *smart contract* correspondente a iniciar uma votação.

Tabela 4.33: Cenário 3: iniciar uma votação

Id	CCU3: Iniciar uma votação
Descrição	O utilizador pretende iniciar uma votação
Ator	Gestor de Votação
Cenário	1. O gestor de votação (app *BackOffice*) chama a função *startVote*; 2. O *smart contract* verifica se a votação está no estado "*Created*"; 3. O *smart contract* verifica se foi o *owner* que chamou a função; 4. O *smart contract* altera o estado da votação para "*Voting*"; 5. O *smart contract* inicializa a variável *abstention* como o número de eleitores esperados.

A Tabela 4.34 retrata o Cenário de Casos de Uso 4 do *smart contract* correspondente a votar numa votação.

Tabela 4.34: Cenário 4: votar

Id	CCU4: Votar
Descrição	O utilizador pretende votar
Ator	Eleitor
Cenário	1. O eleitor (app *Mobile*) chama a função *voteForCandidate*; 2. O *smart contract* verifica se a votação está no estado "*Voting*"; 3. O *smart contract* verifica se o eleitor já votou; 4. O *smart contract* verifica se o candidato pertence à lista de candidatos previamente definida; 5. O *smart contract* define que o eleitor já votou; 6. O *smart contract* incrementa o número de votos de candidato; 7. O *smart contract* decrementa variável *abstention*.

A Tabela 4.35 retrata o Cenário de Casos de Uso 5 do *smart contract* correspondente a terminar uma votação.

Tabela 4.35: Cenário 5: terminar uma votação

Id	CCU5: Terminar uma votação
Descrição	O utilizador pretende terminar uma votação
Ator	Gestor de Votação
Cenário	1. O gestor de votação (app *BackOffice*) chama a função *endVote*; 2. O *smart contract* verifica se a votação está no estado "*Voting*"; 3. O *smart contract* verifica se foi o *owner* que chamou a função; 4. O *smart contract* altera o estado da votação para "*Ended*".

A Tabela 4.36 retrata o Cenário de Casos de Uso 6 do *smart contract* correspondente a obter os resultados de uma votação.

Tabela 4.36: Cenário 6: obter os resultados

Id	CCU6: Obter os resultados
Descrição	O utilizador pretende obter os resultados de uma votação
Ator	Gestor de Votação e Eleitor
Cenário	1. O gestor de votação (app *BackOffice*) chama a função *totalVotesFor* para cada candidato; 2. O *smart contract* verifica se a votação está no estado "*Ended*"; 3. O *smart contract* verifica se o candidato pertence à lista de candidatos previamente definida; 4. O *smart contract* retorna o número de votos desse candidato; 5. A aplicação *BackOffice* chama a variável *abstention*; 6. O *smart contract* retorna o o valor da variável *abstention*.

Matriz de rastreabilidade entre requisitos e os casos de uso

A seguinte tabela, Tabela 4.37, ilustra a matriz de rastreabilidade entre os requisitos do *smart contract* com os casos de uso respetíveis.

Tabela 4.37: Matriz de rastreabilidade entre requisitos e casos de uso

Req. Id	CCU1	CCU2	CCU3	CCU4	CCU5	CCU6
1.F.S			X			
2.F.S					X	
3.F.S		X				
4.F.S				X		
5.F.S				X		
6.F.S						X
7.F.S	X					
8.F.S				X		X
9.F.S	X					
10.F.S						X

Diagramas de Estados

Os diagramas de estados (DE) representativos dos casos de uso do sistema estão evidenciados nas seguintes figuras.

A Figura 4.54 ilustra o diagrama de estados relativo à função *Constructor* do *smart contract*.

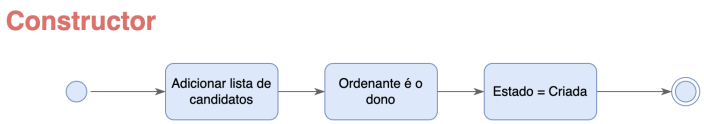


Figura 4.54: DE1 - Diagrama de estados da função *Constructor*

A Figura 4.55 ilustra o diagrama de estados relativo à função *Authorize* do *smart contract*.

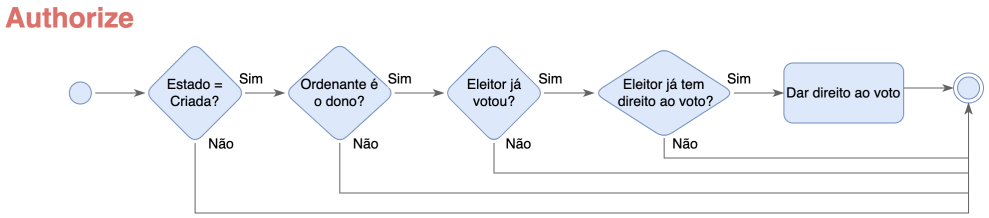


Figura 4.55: DE2 - Diagrama de estados da função *Authorize*

A Figura 4.56 ilustra o diagrama de estados relativo à função *StartVote* do *smart contract*.

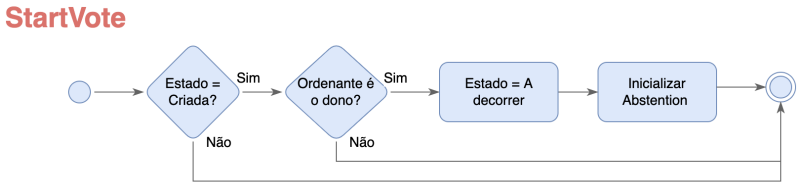


Figura 4.56: DE3 - Diagrama de estados da função *StartVote*

A Figura 4.57 ilustra o diagrama de estados relativo à função *VoteForCandidate* do *smart contract*.

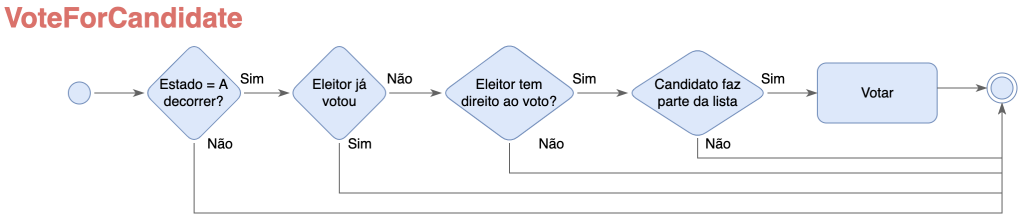


Figura 4.57: DE4 - Diagrama de estados da função *VoteForCandidate*

A Figura 4.58 ilustra o diagrama de estados relativo à função *EndVote* do *smart contract*.

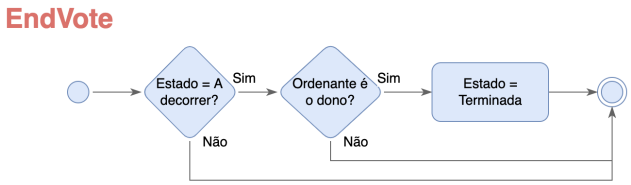


Figura 4.58: DE5 - Diagrama de estados da função *EndVote*

A Figura 4.59 ilustra o diagrama de estados relativo à função *TotalVotesFor* do *smart contract*.

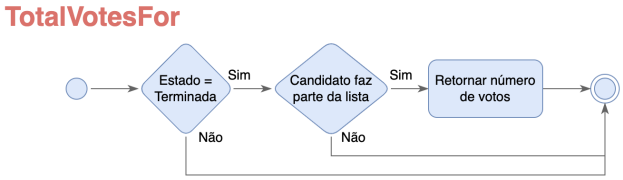


Figura 4.59: DE6 - Diagrama de estados da função *TotalVotesFor*

Matriz de rastreabilidade entre os casos de uso e diagramas de estados
A matriz de rastreabilidade interligando os casos de uso do *smart contract* com os respetivos diagramas de estados, está exposta na Tabela 4.38.

Tabela 4.38: Matriz de rastreabilidade entre casos de uso e diagramas de estados

CU. Id	DE1	DE2	DE3	DE4	DE5	DE6
CCU1	X					
CCU2		X				
CCU3			X			
CCU4				X		
CCU5					X	
CCU6						X

O *smart contract* utilizado foi concebido na linguagem de programação *Solidity*, recorrendo ao Remix IDE [76], uma aplicação *open source* designada a facilitar a escrita de *smart contracts* em *Solidity*, através do *browser*. Adicionalmente, esta aplicação permite compilar o *smart contract* desenvolvido, efetuar o seu *deploy* e, posteriormente, toda a interação com o mesmo.

No contexto deste projecto, o Remix IDE foi utilizado unicamente para criar o *smart contract* e gerar o ficheiro meta correspondente, advindo do processo de compilação, para futuramente ser inserido na aplicação *BackOffice*.

É importante referir ainda que, a produção do código pertencente ao *smart contract*, foi baseada tendo em consideração o contributo da comunidade *open source* presente em [77, 78, 79, 80].

Ao longo de todo o *smart contract*, situado no Anexo C, foram utilizados *modifiers* para assegurar que certas funções são executadas somente pelo gestor de votação classificado como dono, e apenas em certos momentos do processo de votação. Tendo como exemplo a função *endVote*, esta só pode ser chamada pelo gestor de votação e depois da votação estar a decorrer.

Capítulo 5

Validação da Solução

Este capítulo consiste na avaliação do produto e discussão de resultados, obtido tanto pela confirmação de funcionamento como a realização de testes de software perante a solução desenvolvida e análise da sua utilidade.

5.1 Testes de *Software*

A fase de testes é fundamental na elaboração de um projeto, uma vez que permite verificar as funcionalidades implementadas e descobrir possíveis erros que possam surgir. Consequentemente, a forma como estes testes são organizados e realizados permite à equipa de desenvolvimento melhorar a solução existente, garantindo a qualidade do produto final. Posto isto, foram realizados diversos testes, desde o início do projeto, para averiguar o funcionamento das aplicações a serem desenvolvidas e a habilidade do sistema satisfazer todos os requisitos propostos.

Os testes de unidade permitem testar separadamente cada componente de software desenvolvido em ambas as aplicações (*web* e *mobile*) e no sistema do *smart contract*, pelo que foi o tipo de testes efetuado mais regularmente ao longo de todo o processo de desenvolvimento de *software* [81]. Foram, ainda, realizados alguns testes para verificar a correta inserção de dados na base de dados, bem como todas as outras operações CRUD.

Adicionalmente, foram efetuados testes de integração com o objetivo de verificar se todas as partes envolventes do sistema funcionam de forma conjunta, determinando possíveis erros [81]. Como no presente projeto as componentes constituintes

do sistemas são a vertente *web*, a vertente *mobile* e a vertente do *smart contract*, foram realizados testes de integração para tentar determinar:

- se as informações disponibilizadas pelo gestor de votação e pelo candidato na aplicação *web* eram corretamente apresentado na aplicação *mobile*;
- se o conteúdo inserido na aplicação *mobile* por parte do eleitor eram devidamente visualizadas na aplicação *web*;
- se as interações realizadas pelo gestor de votação, a partir da aplicação *web*, com o *smart contract*, como por exemplo efetuar o *deploy* do *smart contract*, iniciar ou terminar uma votação, aconteciam conforme o expectável;
- se as interações realizadas pelo eleitor, a partir da aplicação *mobile*, com o *smart contract*, como por exemplo votar, aconteciam conforme o expectável;
- se tanto os utilizadores da aplicação *web* (gestor de votação) e *mobile* (eleitor) conseguiam conectar-se com sucesso às suas carteiras digitais.

As Figura 5.1, 5.2, 5.3 e 5.4 ilustram alguns dos testes realizados nomeadamente teste de conexão da componente *web* e *mobile* com a *Digital Wallet* (MetaMask), e com o *smart contract* através das funções *startVote* e *voteForCandidate* respetivamente.

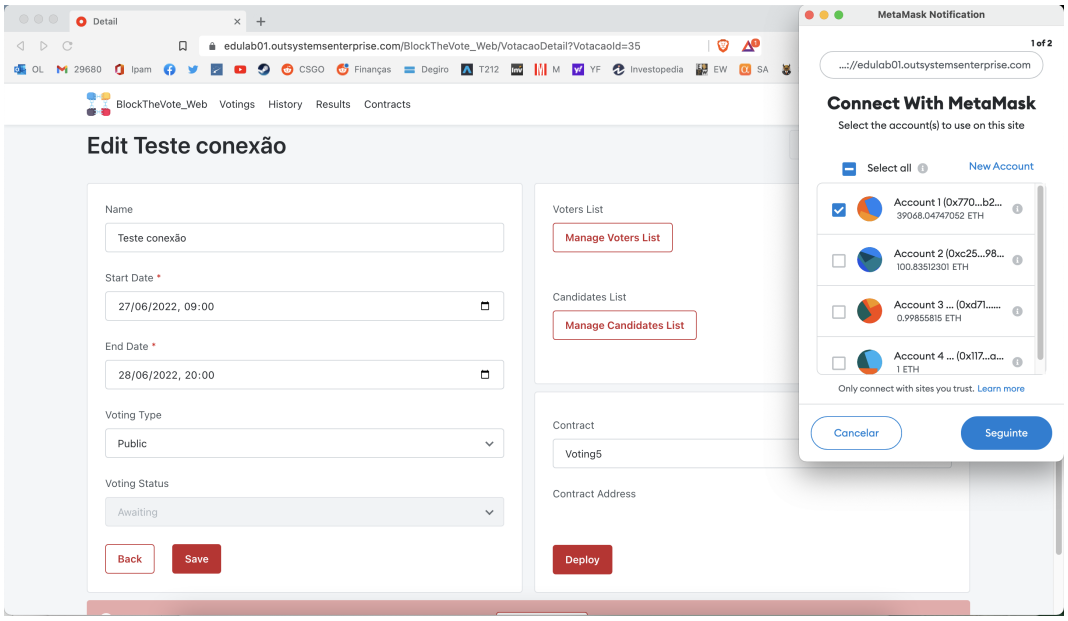


Figura 5.1: Teste de conexão da componente *web* com a carteira digital (MetaMask)

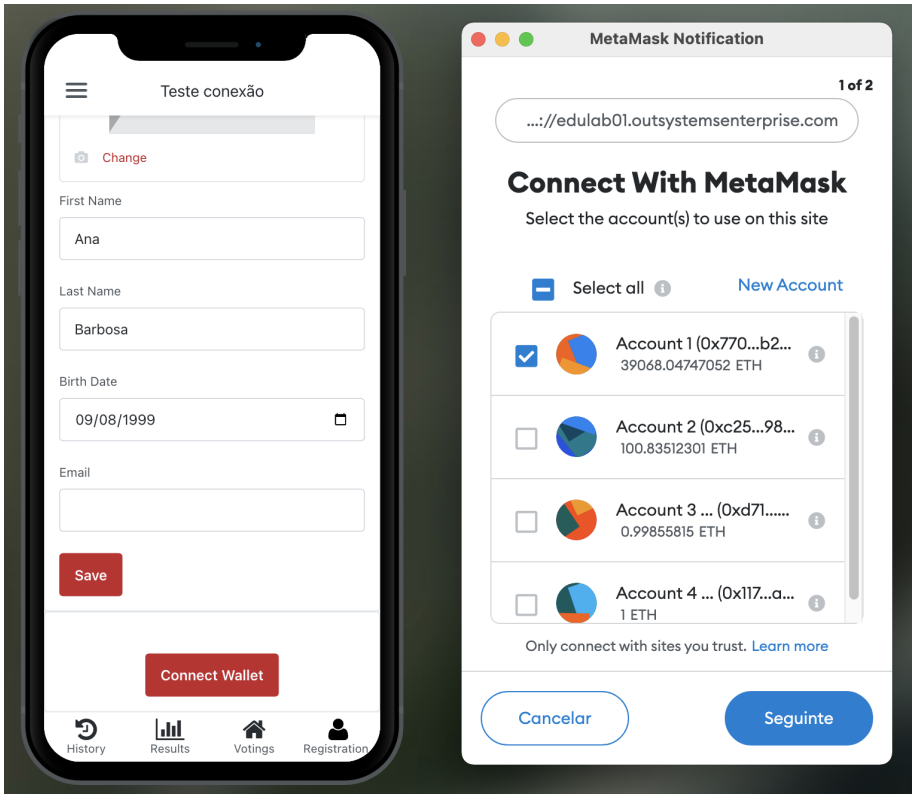


Figura 5.2: Teste de conexão da componente *mobile* com a carteira digital (MetaMask)

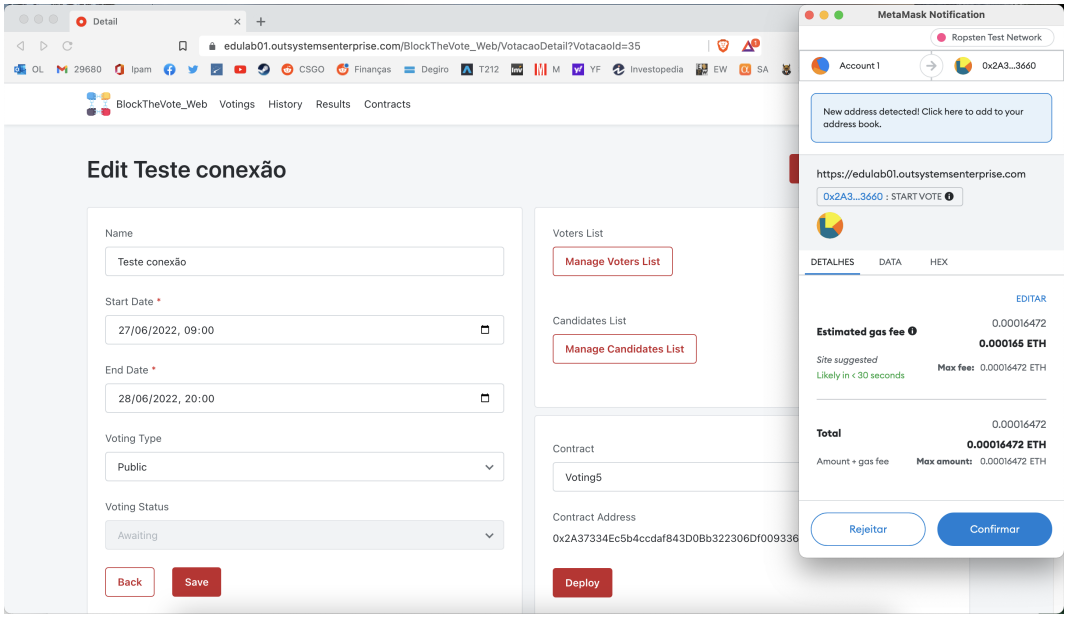


Figura 5.3: Teste de conexão da componente *web* com o *smart contract*

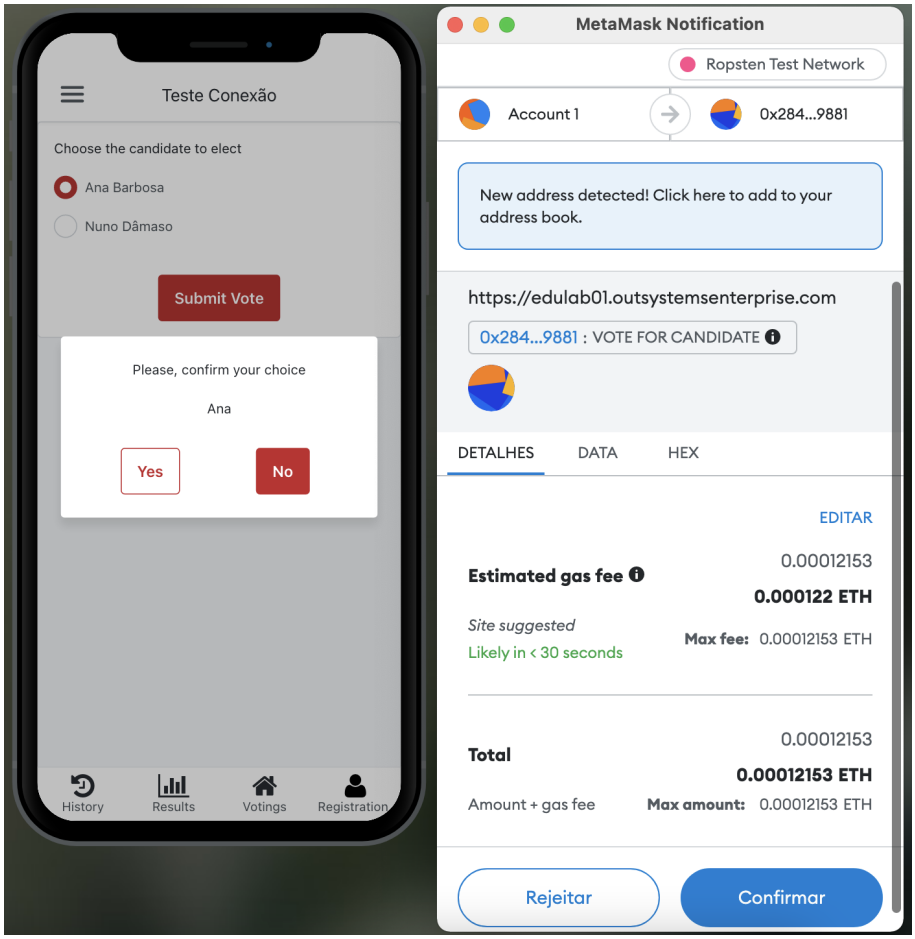


Figura 5.4: Teste de conexão da componente *mobile* com o *smart contract*

Por outro lado, os testes de sistema, realizados antes da aplicação ser disponibilizada ao utilizador final, consistem na verificação global do sistema e dos seus intervenientes, bem como a sua comunicação com a base de dados, garantindo que o projeto está em conformidade com os requisitos previamente impostos [81]. Aqui, foram simulados múltiplos cenários de processos de votação com a finalidade de analisar o sistema como um todo.

Por último, os testes de usabilidade permitem, através da participação dos utilizadores finais da aplicação, verificar se o sistema satisfaz os requisitos pré-definidos e se o seu funcionamento vai ao encontro ao expetável [81]. Para este projeto em específico, foi concebido um questionário de usabilidade da aplicação desenvolvida, exibido na Figura 5.5, de modo a analisar o desempenho da mesma e a sua aceitação por parte dos utilizadores. Este inquérito foi, posteriormente, disponibilizado a algumas pessoas, entre elas alguns colaboradores da empresa.

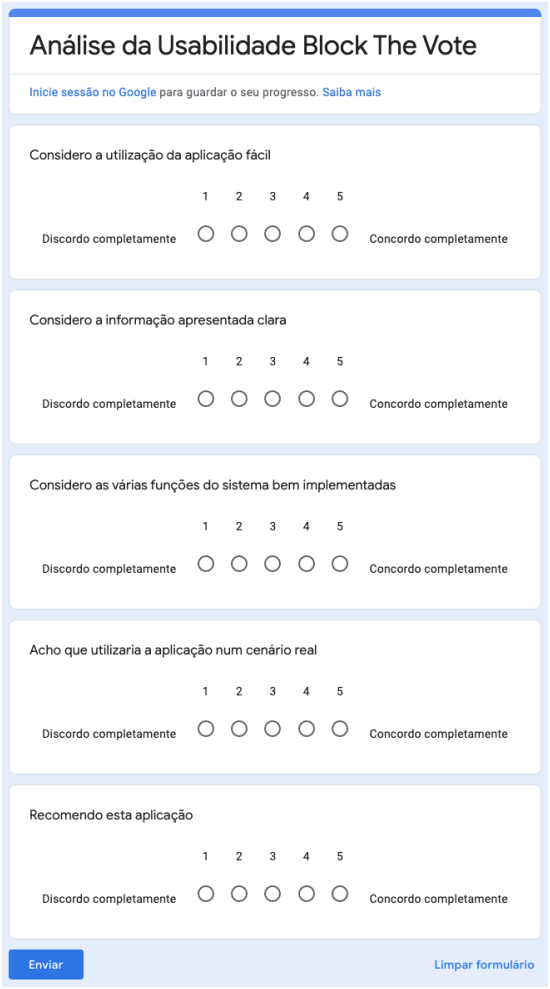


Figura 5.5: Questionário de usabilidade da aplicação Block The Vote

O inquérito concebido através do Google Forms recorre à escala de Likert, visto que a mesma é uma escala psicométrica utilizada como meio de análise em inquéritos de pesquisa de opinião.

Desta forma, as respostas neste tipo de questionário refletem a opinião e nível de concordância dos inquiridos face a umas questões pré-definidas.

Posto isto, são utilizados 5 níveis de resposta, sendo eles:

- 1 – Discordo completamente;
- 2 – Discordo;
- 3 – Não concordo nem discordo;
- 4 – Concordo;
- 5 – Concordo completamente.

5.2 Avaliação da Solução

Como referido anteriormente, a avaliação da solução deu-se através de um questionário de usabilidade (Figura 5.5), ao qual responderam 33 inquiridos. Os resultados advindos deste inquérito encontram-se apresentados de seguida.

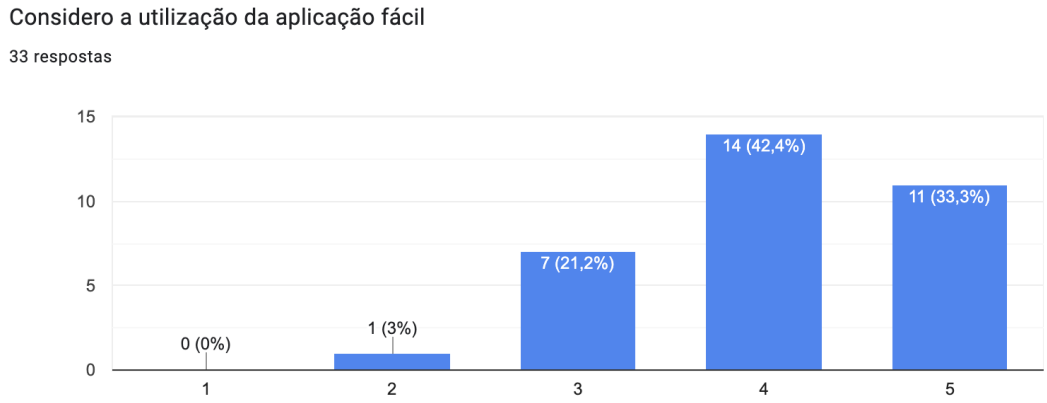


Figura 5.6: Gráfico 1 – Facilidade de utilização

Relativamente à facilidade de utilização da aplicação (Figura 5.6), verifica-se que onze dos inquiridos consideram a aplicação muito fácil de utilizar, catorze consideram fácil, sete pessoas consideram que a mesma não era nem fácil, nem difícil, e uma pessoa considerou difícil.

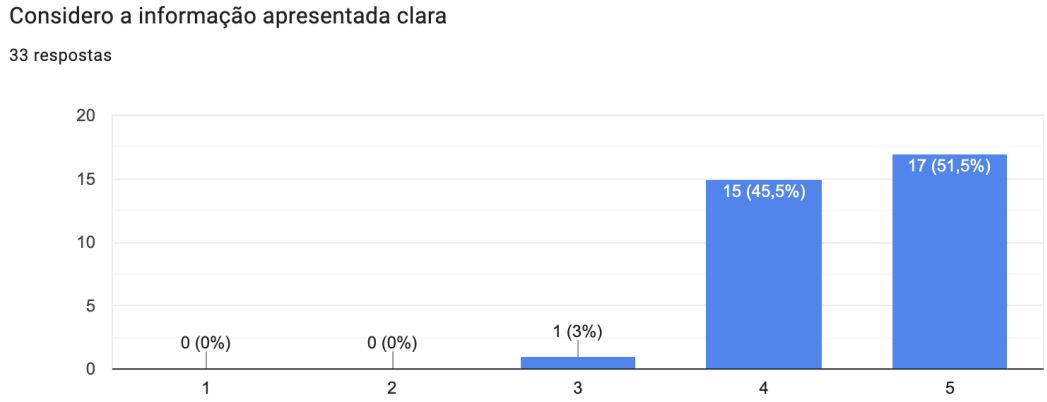


Figura 5.7: Gráfico 2 – Clareza da informação

No que diz respeito à clareza da informação apresentada na aplicação (Figura 5.7), a resposta mais comum (dezassete pessoas) apontava para a aplicação ser muito clara, seguindo-se de quinze elementos afirmarem que a mesma era clara e existiu ainda uma pessoa que considerou nem como clara nem confusa.

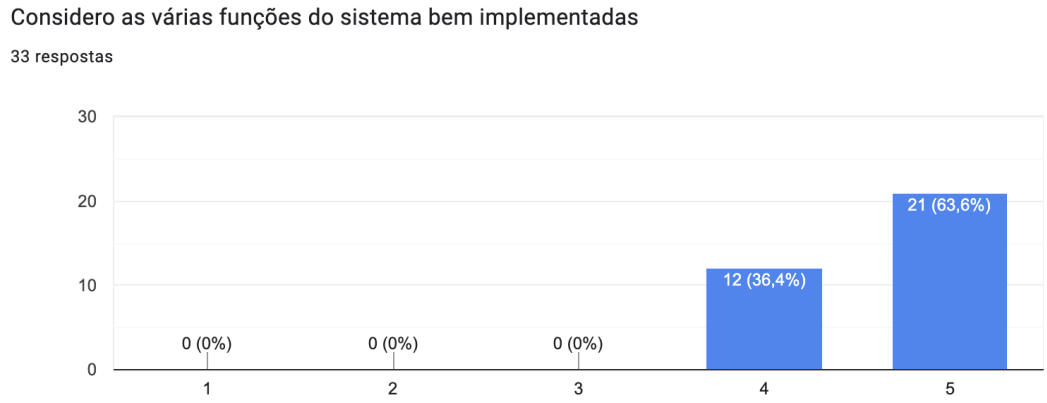


Figura 5.8: Gráfico 3 – Implementação de funcionalidades

Quanto à implementação das funcionalidades presentes na aplicação (Figura 5.8), a maioria dos inquiridos (vinte e um) considerou que as mesmas estavam muito bem implementadas, pelo que os restantes elementos (doze) qualificaram as funcionalidades como bem implementadas.

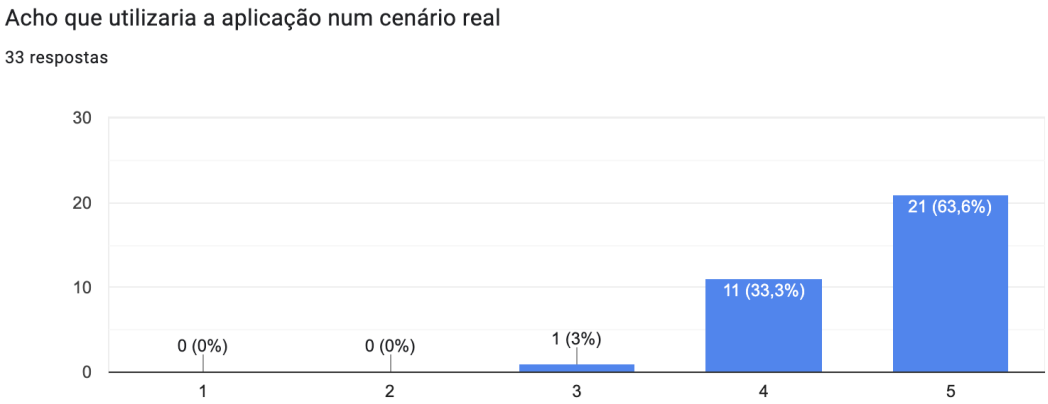


Figura 5.9: Gráfico 4 – Utilização num cenário real

Face à possibilidade de utilização da aplicação num cenário real (Figura 5.9), a grande maioria (vinte e um) afirma que utilizaria muito a aplicação, onze pessoas afirmam que utilizariam a aplicação e apenas um individuo manteve-se neutro perante a questão.

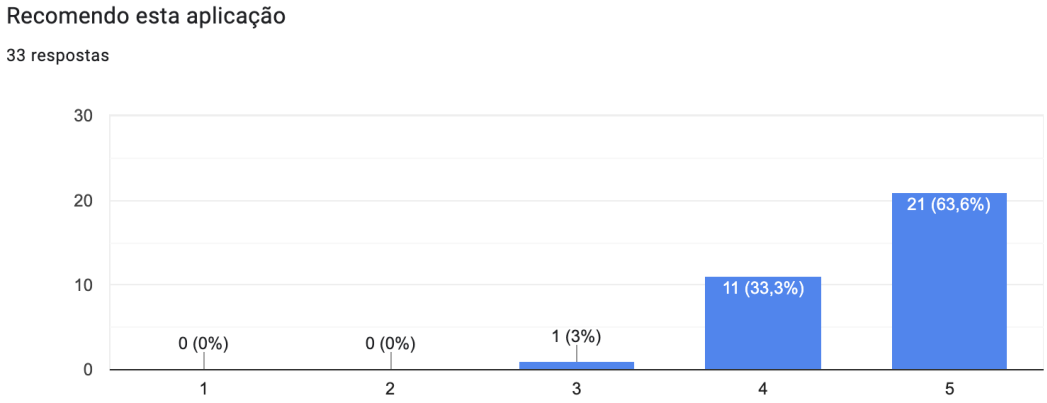


Figura 5.10: Gráfico 5 - Percentagem de pessoas que recomendariam a aplicação

Por último, em relação à recomendação aplicação de modo geral (Figura 5.10), vinte e um elementos dos inquiridos concordam completamente com esta afirmação, onze pessoas concordam e um indivíduo não tem uma opinião definida.

Na generalidade, os resultados obtidos a partir do questionário de usabilidade foram positivos, na medida em que as respostas mais comuns em todas as questões apontavam para os dois níveis de concordância mais altos (concordo e concordo completamente).

Não obstante verificou-se uma resposta negativa perante a facilidade da aplicação, o que é perfeitamente normal tendo em conta a temática em que está aplicação está inserida e os recurso por ela utilizados.

Ainda assim, pode concluir-se que a maioria dos inquiridos considera a aplicação de fácil utilização, em que a informação e as funcionalidades por ela apresentada são respetivamente claras e bem implementadas. Adicionalmente, alegam que a utilizariam num cenário real, facto este que aliado aos fatores anteriores impulsiona a sua decisão em recomendar esta aplicação a outros utilizadores.

A solução obtida almeja impulsionar a adoção da Web3, na medida em que vai ao encontro dos seus princípios de descentralização, através de uma aplicação descentralizada resultante do culminar da temática *blockchain* e sistemas de *e-voting* (questão de investigação nº1).

Para isso, a aplicação desenvolvida tem de ser capaz de atender a certas funcionalidades como auxiliar o processo de recenseamento e autenticação dos eleitores, bem como gerir a lista de candidatos, o *deploy* de *smart contracts* e, consequentemente, a interação com os mesmos, a votação propriamente dita, e a contagem/apresentação dos seus resultados (questão de investigação nº2).

A fim de responder à questão de investigação nº3, recorreu-se a certas ferramentas para permitir a interação com uma *blockchain*, neste caso a rede de testes Ropsten

pertencente ao ecossistema Ethereum. A aplicação *open source* Remix IDE possibilitou a conceção do *smart contract* utilizado na linguagem *Solidity* e a obtenção do ficheiro meta correspondente, advindo do processo de compilação, para futuramente ser inserido na aplicação *BackOffice*. A comunicação com o *smart contract* é feita através da inserção em todos os módulos aplicacionais de um ficheiro *javascript* denominado de *EthersJS* que proporciona a comunicação com uma *blockchain*. Adicionalmente, foi utilizada a extensão do *browser* MetaMask, uma carteira digital capaz de facilitar esta comunicação entre a aplicação descentralizada e a *blockchain*, sem ser necessário recorrer a um *software* especializado para esse efeito.

Capítulo 6

Conclusões

Este capítulo retrata o culminar de todo o processo de investigação e desenvolvimento. Aqui, é feita uma retrospetiva crítica, passando pela identificação de limitações encontradas ao longo do projeto, bem como possíveis implementações futuros com a finalidade de aprimorar o presente projeto, contribuindo, assim, para o avanço tecnológico e científico.

Face à evolução da Internet e da tecnologia ao longo dos últimos anos, surge o conceito de *blockchain*, isto é, uma rede de blocos protegidos criptograficamente, que ficou mundialmente conhecido com a criação da criptomoeda Bitcoin. Esta tecnologia, na qual está baseada a Web3, pode ser aplicada a uma vasta área temática, tentando combater as suas fragilidades e auxiliar o seu funcionamento e eficiência através das características que uma *blockchain* tem para oferecer.

Paralelamente, os sistemas de *e-voting*, ou seja, os sistemas de votação eletrónica, estão presentes, ainda que não existam muitos casos de implementações permanentes, pois as comunidades demonstram a falta de confiança e preocupação face à segurança dos sistemas de votação eletrónica.

Resultante da combinação da temática *blockchain* com sistemas de *e-voting*, surge a aplicação desenvolvida que retrata todo o processo de votação, passando pelas etapas de recenseamento, autenticação, votação e contagem, e recorrendo a uma *blockchain* para armazenar e manter a integridade e auxiliar os processos previamente identificados.

É importante evidenciar que, o facto da aplicação em questão ter sido desenvolvida num curto espaço de tempo, deve-se à utilização da plataforma de *low-code* Outsystems, que possibilitou uma solução bastante funcional e automatizada, permitindo a interligação de todos os componentes e recursos necessários para o bom funcionamento do projeto.

É essencial referir que a linguagem de conceção do *smart contract* foi a linguagem *Solidity*, todavia poderia ter sido usada outra opção. Nessa eventualidade, a presente aplicação estaria apta para lidar com as alterações, uma vez que todo o processo de votação é independente da linguagem utilizada no desenvolvimento do *smart contract*, dependendo somente o ABI e *bytecode* resultante da sua compilação.

Para analisar a usabilidade da aplicação, desenvolveu-se um questionário composto por cinco questões avaliadas através de uma escala de cinco níveis que, posteriormente, foi distribuído por trinta e três utilizadores que tiveram a oportunidade de testar a aplicação obtida. As questões abordadas no questionário remetiam para a facilidade de utilização, clareza da informação apresentada, implementação das funcionalidades, utilização num cenário concreto e real e finalmente recomendação da aplicação por parte dos utilizadores inquiridos.

O foco dos resultados advindos deste inquérito incidia sobre os dois níveis superiores, isto é, concordo e concordo plenamente, pelo que é possível afirmar que a aplicação satisfaz as necessidades dos utilizadores, sendo viável a sua implementação numa situação em concreto.

Em suma, é possível afirmar que o projeto desenvolvido satisfez os objetivos previamente definidos, bem como respondeu às questões levantadas inicialmente. A aplicação em questão, Block The Vote, é considerada um produto funcional com uma potencial implementação no auxílio de um processo de votação.

Limitações e Trabalho Futuro

Tal como no desenvolvimento de qualquer projeto, existem particularidades que podem ser melhoradas provenientes de limitações que surgem ao longo da elaboração do mesmo, seja ao nível da tecnologia, razões temporais, etc.. Tendo como visão principal a construção de uma aplicação o mais completa possível que retratasse todo o processo de uma votação e face ao limite temporal existente, optou-se pela não implementação de certas funcionalidades.

Relativamente à componente *web* e *mobile*, sempre que é executada uma transação, como por exemplo o *deploy* de um *smart contract*, iniciar ou terminar uma votação e a ação de votar, o utilizar deverá esperar que a transação seja concluída, situação esta que pode ser observada na carteira digital MetaMask, antes de proceder, para garantir que as ações que ocorrem no *background* na aplicação sejam devidamente executadas. Perante esta condição, poderia ser criado uma API na qual o sistema iria aceder, aquando o fim da transação. Esta API claramente estaria

responsável por executar uma série de ações diferentes dependendo da transação executada previamente.

Atualmente, o processo de recenseamento é feito manualmente, na medida em que o eleitor insere as suas informações para a posteriori serem aprovadas e o mesmo fazer parte da lista de eleitores elegíveis dentro do *smart contract*. Todavia, o ideal seria quando o eleitor clicasse numa votação à qual pretendia participar, a aplicação comunicaria com algum sistema que devolve-se as informações deste eleitor através de uma identificação única caso o mesmo estivesse elegível, como por exemplo uma autenticação governamental recorrendo ao cartão de cidadão.

De forma de assegurar o anonimato do voto e do respetivo eleitor, seria possível encriptar diretamente o parâmetro do candidato. Assim sendo, no momento do *deploy* do *smart contract*, o parâmetro a submeter não seria o nome do candidato, mas sim a sua chave privada. Quando o eleitor votasse, recorrendo à função *voteForCandidate*, o parâmetro a submeter não seria novamente o nome do candidato, mas sim a respetiva chave pública, garantindo a autenticidade e o não repudio. Como alternativa, podia recorrer-se a *ZK-Rollups* (*Zero-Knowledge Rollups*), ou seja, mecanismos de validação de transações fora da cadeia principal que implementa *Zero Knowledge Proofs* (ZKPs), que representa uma maneira de provar o conhecimento de algo sem revelar o quê. Aplicado ao presente projeto, o eleitor seria capaz de mostrar que votou, sem revelar em que candidato.

Ao nível do *smart contract* uma das possíveis implementações para aprimorar o seu funcionamento e o para o próprio processo de votação ser um pouco mais independente da intervenção humana, seria automatizar o término da votação. Este facto tornar-se-ia possível se, aquando do *deploy* do *smart contract* relativo a uma votação específica, para além de serem transmitidos os candidatos, também seria transmitida a informação com a duração ou data de término da votação. Posto isto, a função *endVote* incorporada no *smart contract* seria despoletada quando a duração da votação chegasse ao fim.

Apesar de em qualquer uma das transações realizadas durante o processo de votação, não envolver nenhuma quantidade de Ether (uma vez que a rede de testes utilizada faz parte do ecossistema Ethereum, cuja criptomoeda associada é o Ether) a ser transacionado, para as transações serem validadas e inseridas no *ledger*, as mesmas estão sujeitas a *gas fees*, isto é, taxas pagas aos *miners* como recompensa durante o processo de *mining*. Para o custo destas taxas não recair sobre o eleitor, o gestor de votação poderia transferir uma quantidade de Ether aquando do cenário de aprovar recenseamento ou o eleitor poderia pedir um reembolso após a votação terminar, durante um momento específico para essa finalidade. Esta funcionalidade, símbolo do conceito *gasless* ou *meta transactions*, seria implementada alterando o código do *smart contract* ou através da utilização de um plataforma própria para este fim como a Gas Station Network.

Para permitir o bom funcionamento e o acesso a todos os recursos utilizados, optou-se por tornar a aplicação móvel uma *Progressive Web App* (PWA), pois a mesma corre num browser permitindo a interação com a extensão MetaMask sem qualquer incidência.

Referências

[1] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," in *Conference on the Theory and Application of Cryptography*, pp. 437–455, Springer, 1990. [Citado nas páginas 1 e 16]

[2] "Conheça a história da tecnologoia blockchain: Quando ela começou?." Available at `https://101blockchains.com/pt/historia-da-tecnologia-blockchain/` (Last accessed 30/06/2022). [Citado na página 1]

[3] L. Ribeiro, "Introdução à blockchain e contratos inteligentes: Apostila para iniciante relatório técnico do ine," 2019. [Citado nas páginas 2, 16, 20, 21, 22, 23, 25 e 26]

[4] E. E. L. Taveira, "Uma visão sobre a tecnologia blockchain: Domínios de aplicação e especificidades na cadeia de abasteciment," 2020. [Citado nas páginas vii, 2, 17, 18, 20, 22, 23 e 25]

[5] "What is web3 and why is it important? | ethereum.org." Available at `https://ethereum.org/en/web3/` (Last accessed 30/06/2022). [Citado na página 2]

[6] "Do it lean | outsystems apps you can lean on." Available at `https://www.doitlean.com/` (Last accessed 30/06/2022). [Citado na página 2]

[7] "Do it lean | our history." Available at `https://www.doitlean.com/our-history` (Last accessed 30/06/June 2022). [Citado nas páginas 2 e 3]

[8] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *Management Information Systems Quarterly*, vol. 28, pp. 75–105, 03 2004. [Citado nas páginas vii, xi, 3, 34 e 36]

[9] M. P. Bax, "Design science: filosofia da pesquisa em ciência da informação e tecnologia," *Ciência da Informação*, vol. 42, pp. 298–312, 2013. [Citado nas páginas 3, 33, 34 e 35]

[10] S. Risnanto, Y. Bin, A. Rahim, and N. S. Herman, "E-voting readiness mapping for general election implementation," *Journal of Theoretical and Applied Information Technology*, vol. 31, p. 20, 2020. [Citado nas páginas xi, 8, 9, 11, 12, 14, 15 e 16]

[11] A. Babu and V. D. Dhore, "Electronic polling agent using blockchain: a new approach," in *IC-BCT 2019*, pp. 69–77, Springer, 2020. [Citado nas páginas 8 e 9]

[12] "Focus on e-voting —." Available at `https://aceproject.org/ace-en/focus/e-voting/default` (Last accessed 25/06/2022). [Citado na página 8]

[13] "If e-voting is currently being used, is it taking place in controlled or uncontrolled environment? | international idea." Available at `https://www.idea.int/data-tools/question-view/744` (Last accessed 25/06/2022). [Citado nas páginas vii e 8]

[14] O. Cetinkaya and D. Cetinkaya, "Verification and validation issues in electronic voting," *The Electronic Journal of e-Government*, vol. 5, pp. 117–126, 2007. [Citado nas páginas vii, 9 e 10]

[15] R. Anane, R. Freeland, and G. Theodoropoulos, "e-voting requirements and implementation," *Proceedings - The 9th IEEE International Conference on E-Commerce Technology; The 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services, CEC/EEE 2007*, pp. 382–389, 2007. [Citado nas páginas 9, 10 e 11]

[16] K.-H. Wang, S. K. Mondal, K. Chan, and X. Xie, "A review of contemporary e-voting: Requirements, technology, systems and usability," vol. 1, 2017. [Citado nas páginas 9 e 10]

[17] L. Mitrou, D. Gritzalis, and S. Katsikas, "Revisiting legal and regulatory requirements for secure e-voting," *IFIP Advances in Information and Communication Technology*, vol. 86, pp. 469–480, 2002. [Citado na página 10]

[18] "Introducing electronic voting: Essential considerations," *International IDEA resources on Electoral Processes*, 2011. [Citado na página 11]

[19] A.-G. Tsahkna, "E-voting: lessons from estonia," [Citado nas páginas vii, 12, 13 e 14]

[20] A. H. Trechsel, "Report for the council of europe internet voting in estonia a comparative analysis of four elections since 2005," 2010. [Citado nas páginas vii, 12 e 13]

[21] J. I. Pegorini, A. C. C. Souza, A. R. Ortoncelli, R. T. Pagno, and N. C. Will, "Security and threats in the brazilian e-voting system: A documentary case study based on public security tests," [Citado na página 15]

[22] "Philippines: Implementing e-counting | national democratic institute." Available at `https://www.ndi.org/e-voting-guide/philippines-CS/implementing-e-counting` (Last accessed 25/06/2022). [Citado na página 15]

[23] "Philippines: Implementing e-counting | national democratic institute." Available at `https://www.ndi.org/e-voting-guide/philippines-CS/implementing-e-counting` (Last accessed 25/06/2022). [Citado na página 15]

[24] P. Fuchs, "Blockchain: Everything you need to know about how this remarkable technology will impact you, your organization and society," 2019. Available at `https://www.marshmclennan.com/content/dam/mmc-web/insights/publications/2019/jan/gl-2019-blockchain-101-overview-mercer.pdf` (Last accessed 25/06/2022). [Citado na página 16]

[25] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," [Citado na página 16]

[26] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," pp. 557–564, Institute of Electrical and Electronics Engineers Inc., 9 2017. [Citado nas páginas vii, 16, 17, 19, 20, 21 e 25]

[27] "Oecd blockchain primer," Available at `https://www.oecd.org/finance/OECD-Blockchain-Primer.pdf` (Last accessed 25/06/2022). [Citado nas páginas xi, 16, 17, 18, 19, 21, 22 e 24]

[28] A. Anderberg, E. Andonova, M. Bellia, L. Calès, A. Inamorato dos Santos, I. Kounelis, I. Nai Fovino, M. Petracco Giudici, E. Papanagiotou, M. Sobolewski, F. Rossetti, and L. Spirito, *Blockchain now and tomorrow : assessing multidimensional impacts of distributed ledger technologies*. Publications Office, 2019. [Citado nas páginas vii, 17, 18, 19, 22, 23, 24, 25 e 26]

[29] T. A. Borrego, "Tecnologia blockchain-potencial de aplicação no âmbito dos processos de negócio das cadeias de abastecimento," 2019. [Citado nas páginas vii, 18, 19, 20, 21, 22, 23, 24, 25 e 26]

[30] "The story of the dao — its history and consequences | by samuel falkon | the startup | medium." Available at `https://medium.com/swlh/the-story-of-the-dao-its-history-and-consequences-71e6a8a551ee` (Last accessed 25/06/2022). [Citado na página 18]

[31] "The dao: What was the dao hack? | gemini." Available at `https://www.gemini.com/cryptopedia/the-dao-hack-makerdao` (Last accessed 25/06/2022). [Citado na página 18]

[32] "Hard fork (blockchain) definition." Available at `https://www.investopedia.com/terms/h/hard-fork.asp` (Last accessed 25/06/2022). [Citado na página 19]

[33] "Hard and soft forks - what is the difference between them? | analytics steps." Available at https://www.analyticssteps.com/blogs/hard-and-soft-forks-what-difference-between-them (Last accessed 25/06/2022). [Citado nas páginas vii e 19]

[34] "How does bitcoin mining work? what is crypto mining?." Available at https://www.investopedia.com/tech/how-does-bitcoin-mining-work/#toc-what-you-need-to-mine-bitcoins (Last accessed 25/06/2022). [Citado na página 24]

[35] Y. E. M. Aliaga, V. C. Leal, A. U. de Lucena, and M. A. A. Henriques, "Avaliação de mecanismos de consenso para blockchains em busca de nova estratégia mais eficiente e segura," in *Anais do XVIII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, pp. 33–40, SBC, 2018. [Citado na página 25]

[36] "Proof-of-stake (pos) | ethereum.org." Available at https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/ (Last accessed 25/06/2022). [Citado na página 25]

[37] U. Jafar, M. J. A. Aziz, and Z. Shukur, "Blockchain for electronic voting system—review and open research challenges," *Sensors*, vol. 21, no. 17, p. 5874, 2021. [Citado nas páginas vii, 26, 27, 28 e 30]

[38] "Secure decentralized application development - follow my vote." Available at https://followmyvote.com/ (Last accessed 25/06/2022). [Citado na página 27]

[39] M. Chaieb, S. Yousfi, P. Lafourcade, and R. Robbana, "Verify-your-vote: A verifiable blockchain-based online voting protocol," in *European, Mediterranean, and Middle Eastern Conference on Information Systems*, pp. 16–30, Springer, 2019. [Citado nas páginas xi, 27, 28, 29, 30 e 31]

[40] Y. Abuidris, R. Kumar, and W. Wenyong, "A survey of blockchain based on e-voting systems," in *Proceedings of the 2019 2nd International Conference on Blockchain Technology and Applications*, pp. 99–104, 2019. [Citado nas páginas 27 e 28]

[41] J. D. S. A. S. Monteiro, "Blockchain-based decentralized application for electronic voting using an electronic id," 2019. [Citado nas páginas 27, 28, 29 e 30]

[42] R. Osgood, "The future of democracy: Blockchain voting," *COMP116: Information security*, pp. 1–21, 2016. [Citado nas páginas 27 e 28]

[43] B. F. Braz, "A proposal for the use of blockchain in the portuguese voting system," 2021. [Citado na página 28]

[44] J. Cucurull, A. Rodríguez-Pérez, T. Finogina, and J. Puiggali, "Blockchain-based internet voting: systems' compliance with international standards," in *International Conference on Business Information Systems*, pp. 300–312, Springer, 2018. [Citado na página 28]

[45] N. Gailly, P. Jovanovic, B. Ford, J. Lukasiewicz, and L. Gammar, "Agora: bringing our voting systems into the 21st century," 2018. Available at `https://static1.squarespace.com/static/5b0be2f4e2ccd12e7e8a9be9/t/5f37eed8cedac41642edb534/1597501378925/Agora_Whitepaper.pdf` (Last accessed 25/06/2022). [Citado nas páginas vii, 28 e 29]

[46] "Tivi | secure, verifiable online voting solution with end to end integrity - smartmatic." Available at `https://www.smartmatic.com/elections/remote-voting/tivi/` (Last accessed 25/06/2022). [Citado na página 29]

[47] "Online voting successfully solving the challenges," Available at `https://www.smartmatic.com/fileadmin/user_upload/Whitepaper_Online_voting_Mar2022.pdf` (Last accessed 25/06/2022). [Citado nas páginas 29 e 30]

[48] P. McCorry, S. F. Shahandashti, and F. Hao, "A smart contract for boardroom voting with maximum voter privacy," in *International conference on financial cryptography and data security*, pp. 357–375, Springer, 2017. [Citado nas páginas vii e 30]

[49] J. R. Venable, J. Pries-Heje, and R. L. Baskerville, "Choosing a design science research methodology," 2017. [Citado na página 33]

[50] M. Pimentel and D. Filippo, "Re@d-revista de educação a distância e elearning design science research: pesquisa científica atrelada ao design de artefatos," [Citado na página 34]

[51] M. C. Padua, M. J. V. Jorente, and A. Semedo, "Design da informação e ações comunicacionais em websites de museus," in *Anais do 9º CIDI-Congresso Internacional de Design da Informação, edição 2019 e do 9º CONGIC-Congresso Nacional de Iniciação Científica em Design da Informação*, 2019. [Citado nas páginas 34 e 35]

[52] A. B. Brendel, P. Zapadka, and L. Kolbe, "Design science research in green is - analyzing the past to guide future research," *Research Papers*, 11 2018. [Citado na página 35]

[53] S. Cronholm and H. Göbel, "Empirical grounding of design science research methodology," *Lecture Notes in Computer Science (including subseries Lecture*

Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 9073, pp. 471–478, 2015. [Citado na página 35]

[54] P. Salza, P. Musmarra, and F. Ferrucci, "Agile methodologies in education: A review," *Agile and Lean Concepts for Teaching and Learning*, pp. 25–45, 2019. [Citado nas páginas 36, 37, 38, 39 e 40]

[55] L. Williams, "Agile software development methodologies and practices," in *Advances in computers*, vol. 80, pp. 1–44, Elsevier, 2010. [Citado nas páginas 36, 39 e 40]

[56] "Manifesto for agile software development." Available at `http://agilemanifesto.org/iso/en/manifesto.html` (Last accessed 15/06/2022). [Citado na página 36]

[57] "What is agile software development? - k&c." Available at `https://kruschecompany.com/agile-software-development/` (Last accessed 15/06/2022). [Citado nas páginas vii e 37]

[58] S. Al-Saqqa, S. Sawalha, and H. Abdelnabi, "Agile software development: Methodologies and trends," *International Journal of Interactive Mobile Technologies*, vol. 14, pp. 246–270, 2020. [Citado nas páginas vii, xi, 38, 39 e 40]

[59] "Scrum guide | scrum guides." Available at `https://scrumguides.org/scrum-guide.html` (Last accessed 15/06/2022). [Citado na página 39]

[60] A. Srivastava, S. Bhardwaj, and S. Saraswat, "Scrum model for agile methodology," *Proceeding - IEEE International Conference on Computing, Communication and Automation, ICCCA 2017*, vol. 2017-January, pp. 864–869, 12 2017. [Citado na página 39]

[61] "Developing with outsystems | evaluation guide | outsystems." Available at `https://www.outsystems.com/evaluation-guide/developing-with-outsystems/` (Last accessed 15/06/2022). [Citado nas páginas vii, 41 e 42]

[62] "Outsystems development and management tools | evaluation guide | outsystems." Available at `https://www.outsystems.com/evaluation-guide/development-and-management-tools/` (Last accessed 15/06/2022). [Citado na página 42]

[63] "Outsystems platform services | evaluation guide | outsystems." Available at `https://www.outsystems.com/evaluation-guide/platform-services/` (Last accessed 15/06/2022). [Citado na página 42]

[64] "Outsystems - ebiz solutions, llc." Available at `https://www.thinkebiz.net/low-code-no-code/outsystems/` (Last accessed 15/06/2022). [Citado nas páginas vii e 43]

[65] "The architecture canvas - outsystems best practices." Available at `https://success.outsystems.com/Documentation/Best_Practices/Architecture/Designing_the_Architecture_of_Your_OutSystems_Applications/The_Architecture_Canvas` (Last accessed 15/06/2022). [Citado nas páginas vii e 44]

[66] S. Karmali, "Desenvolvimento de aplicação móvel para submissão/revisão de despesas recorrendo a metodologias ágeis de desenvolvimento e à plataforma low-code outsystems," [Citado na página 44]

[67] "The architecture canvas < designing apps using an architecture framework - training | outsystems." Available at `https://www.outsystems.com/training/lesson/2303/the-architecture-canvas` (Last accessed 15/06/2022). [Citado na página 44]

[68] "Validating your application architecture - outsystems best practices." Available at `https://success.outsystems.com/Documentation/Best_Practices/Architecture/Designing_the_Architecture_of_Your_OutSystems_Applications/Validating_your_application_architecture` (Last accessed 15/06/2022). [Citado nas páginas vii, 44 e 45]

[69] "The architecture design process < designing apps using an architecture framework - training | outsystems." Available at `https://www.outsystems.com/training/lesson/2304/the-architecture-design-process` (Last accessed 15/06/2022). [Citado na página 45]

[70] "Welcome to outsystems documentation - outsystems." Available at `https://success.outsystems.com/Documentation/Best_Practices/Architecture/From_architecture_to_development` (Last accessed 15/06/2022). [Citado na página 45]

[71] "What is ethers.js — ethers.js 4.0.0 documentation." Available at `https://docs.ethers.io/v4/` (Last accessed 15/06/2022). [Citado na página 47]

[72] "Announcing ethers.js — a web3 alternative | by l4 | l4 blog | medium." Available at `https://medium.com/l4-media/announcing-ethers-js-a-web3-alternative-6f134fdd06f3` (Last accessed 15/06/2022). [Citado na página 47]

[73] "Web3.js vs ethers.js - guide to eth javascript libraries » moralis » the ultimate web3 development platform." Available at `https://moralis.io/`

web3-js-vs-ethers-js-guide-to-eth-javascript-libraries/ (Last accessed 15/06/2022). [Citado na página 47]

[74] "Infura explained - what is infura? - moralis academy." Available at https://academy.moralis.io/blog/infura-explained-what-is-infura (Last accessed 15/06/2022). [Citado na página 47]

[75] "Alchemy vs. infura: Which node provider is best? - logrocket blog." Available at https://blog.logrocket.com/alchemy-vs-infura-which-node-provider-best/ (Last accessed 15/06/2022). [Citado na página 47]

[76] "Remix - ethereum ide." Available at https://remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js (Last accessed 15/06/2022). [Citado na página 93]

[77] "Solidity by example — solidity 0.4.24 documentation." Available at https://docs.soliditylang.org/en/v0.4.24/solidity-by-example.html (Last accessed 15/06/2022). [Citado na página 93]

[78] "Full stack hello world voting ethereum dapp tutorial — part 1 | by mahesh murthy | medium." Available at https://medium.com/@mvmurthy/full-stack-hello-world-voting-ethereum-dapp-tutorial-part-1-40d2d0d807c2 (Last accessed 15/06/2022). [Citado na página 93]

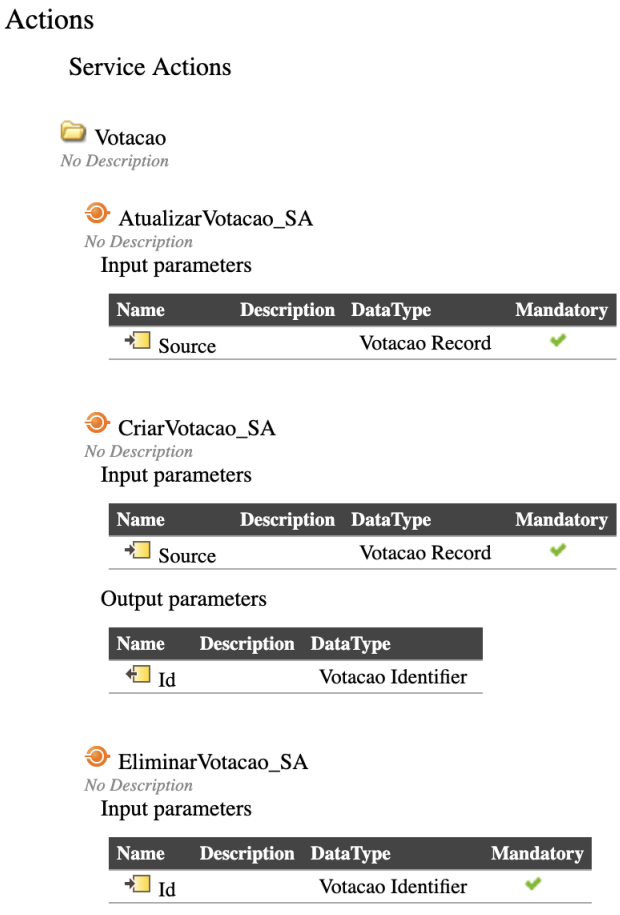
[79] "Voting on a blockchain: Solidity contract codes explained | jackson ng." Available at https://jacksonng.org/voting-blockchain-2 (Last accessed 15/06/2022). [Citado na página 93]

[80] "How to code ethereum election smart contract - live coding - youtube." Available at https://www.youtube.com/watch?v=ucszgKGFnwc (Last accessed 15/06/2022). [Citado na página 93]

[81] R. N. C. Neto, "Suporte a testes automáticos em aplicações web geradas com a outsystems platform," 2013. [Citado nas páginas 95 e 98]

Anexo A

Documentação das API's do Sistema Backend


Actions

Service Actions

Votacao

No Description

AtualizarVotacao_SA

No Description

Input parameters

Name	Description	DataType	Mandatory
Source		Votacao Record	✔

CriarVotacao_SA

No Description

Input parameters

Name	Description	DataType	Mandatory
Source		Votacao Record	✔

Output parameters

Name	Description	DataType
Id		Votacao Identifier

EliminarVotacao_SA

No Description

Input parameters

Name	Description	DataType	Mandatory
Id		Votacao Identifier	✔

Figura A.1: Documentação do Sistema Backend (I)

ListarRecenseamentos

No Description

Input parameters

Name	Description	DataType	Mandatory
UserId		User Identifier	✔

Output parameters

Name	Description	DataType
Votacao		Votacao List

TerminarVotacao_SA

No Description

Input parameters

Name	Description	DataType	Mandatory
VotacaoId		Votacao Identifier	✔
ResultadoVotacaoList		ResultadoVotacao List	✔
TotalVotes		Integer	✔

Candidato

No Description

AtualizarCandidato_SA

No Description

Input parameters

Name	Description	DataType	Mandatory
Source		Candidato Record	✔

CriarCandidato_SA

No Description

Input parameters

Name	Description	DataType	Mandatory
Source		Candidato Record	✔

Output parameters

Name	Description	DataType
Id		Candidato Identifier

Figura A.2: Documentação do Sistema Backend (II)

EliminarCandidato_SA

No Description

Input parameters

Name	Description	DataType	Mandatory
Id		Candidato Identifier	✔

Eleitor

No Description

AprovarEleitor_SA

No Description

Input parameters

Name	Description	DataType	Mandatory
EleitorId		Eleitor Identifier	✔

AtualizarEleitor_SA

No Description

Input parameters

Name	Description	DataType	Mandatory
Source		Eleitor Record	✔

CriarEleitor_SA

No Description

Input parameters

Name	Description	DataType	Mandatory
Source		Eleitor Record	✔

Output parameters

Name	Description	DataType
Id		Eleitor Identifier

EliminarEleitor_SA

No Description

Input parameters

Name	Description	DataType	Mandatory
Id		Eleitor Identifier	✔

Figura A.3: Documentação do Sistema Backend (III)

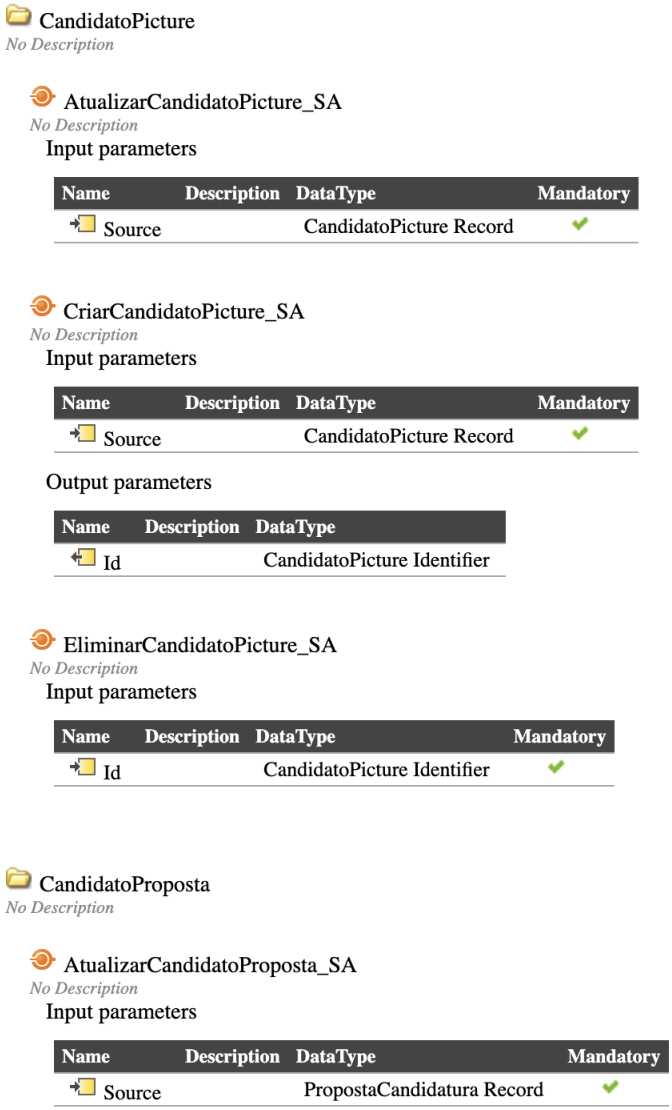


Figura A.4: Documentação do Sistema Backend (IV)

CriarCandidatoProposta_SA

No Description

Input parameters

Name	Description	DataType	Mandatory
Source		PropostaCandidatura Record	✔

Output parameters

Name	Description	DataType
Id		PropostaCandidatura Identifier

EliminarCandidatoProposta_SA

No Description

Input parameters

Name	Description	DataType	Mandatory
Id		PropostaCandidatura Identifier	✔

EleitorPicture

No Description

AtualizarEleitorPicture_SA

No Description

Input parameters

Name	Description	DataType	Mandatory
Source		EleitorPicture Record	✔

CriarEleitorPicture_SA

No Description

Input parameters

Name	Description	DataType	Mandatory
Source		EleitorPicture Record	✔

Output parameters

Name	Description	DataType
Id		EleitorPicture Identifier

Figura A.5: Documentação do Sistema Backend (V)

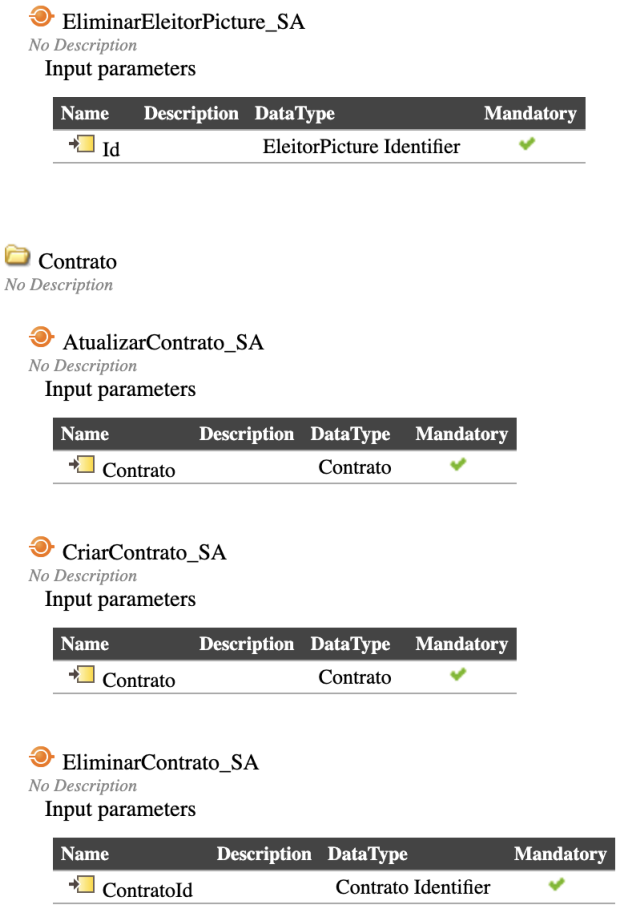


Figura A.6: Documentação do Sistema Backend (VI)

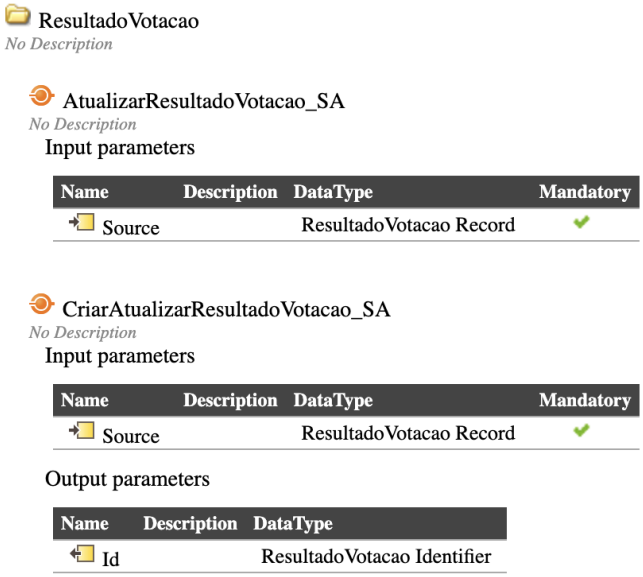


Figura A.7: Documentação do Sistema Backend (VII)

Anexo B

Código aplicação *mobile* e *web*

B.1 Código linguagem Javascript para votar

```
var provider = new ethers.providers.Web3Provider(window.ethereum,
    "ropsten");

var contractArtifactJSON = JSON.parse($parameters.ContractArtifact
    );
var VoteContractAddress = $parameters.Address;
var VoteContractABI = contractArtifactJSON.output.contracts["
    contracts/Voting4.sol"].Voting.abi;
var VoteContract;
var signer;

provider.send("eth_requestAccounts", []).then(function() {
  provider.listAccounts().then(function(accounts) {
    signer = provider.getSigner(accounts[0]);
    console.log(signer);
    VoteContract = new ethers.Contract(
      VoteContractAddress,
      VoteContractABI,
      signer
    );
    console.log(VoteContract);
    VoteContract.voteForCandidate(ethers.utils.formatBytes32String
    ($parameters.Candidate)).then(function(f) {
```

```
20          console.log(f);
21          $actions.VoteForCandidateCallback($parameters.Candidate);
22          $resolve();
23      });
24    });
25 });
```

Listagem B.1: Função de votar

B.2 Código linguagem Javascript para efetuar o *deploy* do *smart contract*

```
1  var provider = new ethers.providers.Web3Provider(window.ethereum,
       "ropsten");
2  var candidates = JSON.parse($parameters.CandidateJSONList);
3  var parsedCandidates = candidates.map(function(candidate) {return
       ethers.utils.formatBytes32String(candidate)});
4  var contractArtifactJSON = JSON.parse($parameters.ContractArtifact
       );
5  provider.send("eth_requestAccounts", []).then(function() {
6    provider.listAccounts().then(function(accounts) {
7      signer = provider.getSigner(accounts[0]);
8      var factory = new ethers.ContractFactory(
9          contractArtifactJSON.output.contracts["contracts/Voting4.
       sol"].Voting.abi,
10         contractArtifactJSON.output.contracts["contracts/Voting4.
       sol"].Voting.evm.bytecode.object ,
11         signer);
12
13     factory.deploy(parsedCandidates).then(function(contract){
14         console.info(contract.address);
15         console.info(contract.deployTransaction);
16         $actions.DeployCallback(contract.address);
17         $resolve();
18     });
19     $resolve();
20   });
21   $resolve();
22 });
```

Listagem B.2: Função de *deploy* do *smart contract*

B.3 Código linguagem Javascript para iniciar uma votação

```
var provider = new ethers.providers.Web3Provider(window.ethereum,
    "ropsten");

var contractArtifactJSON = JSON.parse($parameters.ContractArtifact
    );
var VoteContractAddress = $parameters.Address;
var VoteContractABI = contractArtifactJSON.output.contracts["
    contracts/Voting4.sol"].Voting.abi;
var VoteContract;
var signer;

provider.send("eth_requestAccounts", []).then(function() {
  provider.listAccounts().then(function(accounts) {
    signer = provider.getSigner(accounts[0]);
    console.log(signer);
    VoteContract = new ethers.Contract(
      VoteContractAddress,
      VoteContractABI,
      signer
    );
    console.log(VoteContract);

    VoteContract.startVote().then(function(f) {
        console.log("Voting started");
        console.log(f);
        $actions.IniciarVotacaoCallback()
        $resolve();
    });

    $resolve();
  });
  $resolve();
});
```

Listagem B.3: Função de iniciar votação

Anexo C

Código *smart contract*

C.1 Código linguagem Solidity do *smart contract*

```
1  // SPDX-License-Identifier: Unlicensed
2  pragma solidity >=0.7.0 <0.9.0;
3  // We have to specify what version of compiler this code will
       compile with
4
5  contract Voting {
6    /* mapping field below is equivalent to an associative array or
       hash.
7    The key of the mapping is candidate name stored as type bytes32
       and value is
8    an unsigned integer to store the vote count
9    */
10
11   mapping (bytes32 => uint256) private votesReceived;
12
13
14   //We will use an array of bytes32 to store the list of
       candidates
15
16   bytes32[] public candidateList;
17
18   struct Voter{
19         bool voted;
```

```
20          uint weight;
21      }
22
23  address public immutable owner;
24  mapping (address => Voter) public voters;
25
26  uint public totalVotes;
27  uint public abstention;
28
29  enum State { Created, Voting, Ended }
30  State public state;
31
32  modifier onlyOwner() {
33          require(msg.sender == owner);
34          _;
35      }
36
37  modifier inState(State _state) {
38          require(state == _state);
39          _;
40      }
41
42  /* This is the constructor which will be called once when you
43  deploy the contract to the blockchain. When we deploy the
      contract,
44  we will pass an array of candidates who will be contesting in
      the election
45  */
46  constructor(bytes32[] memory candidateNames)  {
47          candidateList = candidateNames;
48          owner = msg.sender;
49          state = State.Created;
50  }
51
52  function authorize(address voter) public inState(State.Created)
      onlyOwner {
53          require(!voters[voter].voted);
54          require(voters[voter].weight == 0);
55
56          voters[voter].weight = 1;
57          totalVotes +=1;
58  }
59
60  function startVote() public inState(State.Created) onlyOwner {
61          state = State.Voting;
62          abstention = totalVotes;
63  }
64
```

```
65    // This function returns the total votes a candidate has
        received so far
66    function totalVotesFor(bytes32 candidate) view public inState(
        State.Ended) returns (uint256) {
67          require(validCandidate(candidate));
68
69          return votesReceived[candidate];
70    }
71
72    // This function increments the vote count for the specified
        candidate. This is equivalent to casting a vote
73    function voteForCandidate(bytes32 candidate) inState(State.
        Voting) public {
74          require(!voters[msg.sender].voted);
75          require(voters[msg.sender].weight == 1);
76          require(validCandidate(candidate));
77
78          voters[msg.sender].voted = true;
79          votesReceived[candidate] += voters[msg.sender].weight;
80          abstention -= 1;
81    }
82
83    function validCandidate(bytes32 candidate) view public returns (
        bool) {
84          for(uint i = 0; i < candidateList.length; i++) {
85              if (candidateList[i] == candidate) {
86                  return true;
87              }
88          }
89          return false;
90    }
91
92    function endVote() public inState(State.Voting) onlyOwner {
93          state = State.Ended;
94      }
95
96 }
```

Listagem C.1: Código do *smart contract*