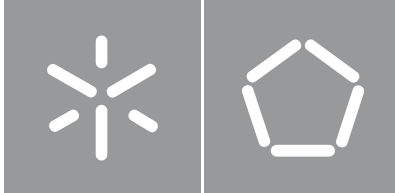


Universidade do Minho
Escola de Engenharia

Tiago Araújo Alves

Geração de NFTs para Certificação de Documentos



Universidade do Minho
Escola de Engenharia

Tiago Araújo Alves

Geração de NFTs para Certificação de Documentos

Dissertação de Mestrado
Mestrado em Engenharia Informática

Trabalho efetuado sob a orientação de
José Carlos Bacelar Ferreira Junqueira Almeida
Helena Fernández López

Direitos de Autor e Condições de Utilização do Trabalho por Terceiros

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

Licença concedida aos utilizadores deste trabalho:



CC BY

<https://creativecommons.org/licenses/by/4.0/>

Agradecimentos

Depois de finalizar esta etapa importante na minha vida, não poderia deixar de agradecer a todas as pessoas que me acompanharam e ajudaram de várias formas durante a realização desta dissertação.

Em primeiro lugar, gostaria de expressar os meus sinceros agradecimentos, ao Professor José Carlos Bacelar e à Professora Helena Fernández por todo o apoio que me deram durante a realização de todas as etapas da dissertação e por estarem sempre dispostos a ajudar em qualquer dúvida ou incerteza. Sem dúvida alguma tiveram um papel fundamental no desenvolvimento desta dissertação.

Quero deixar um profundo agradecimento à minha família e namorada, por toda a ajuda que me deram nesta fase da minha vida e por serem o meu suporte. Agradeço também por todos os conselhos que me deram e por lidarem sempre comigo da melhor forma e me fazerem feliz mesmo nos dias menos bons.

Deixo também um agradecimento especial aos meus amigos que fizeram este caminho a meu lado, agradeço por todos os momentos e por todas as memórias que ficam desta fase das nossas vidas.

Por último, gostaria de expressar um enorme agradecimento à *dstelecom* e ao João Faria pela confiança depositada e por todo o apoio que me deram durante o desenvolvimento da dissertação.

Declaração de Integridade

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

Universidade do Minho, Braga, novembro 2023

Tiago Araújo Alves

Abstract

Non-Fungible Tokens have emerged as a trend in recent times and have aroused interest among developers and companies due to their potential in a wide range of application areas. This dissertation arose from the initiative of the company dstelecom, and its main objective is to study the use of NFTs in the context of document certification and the development of a software module that issues certificates based on NFTs, thus providing a viable alternative to the methods currently used to issue certificates.

In order to meet the objectives of the dissertation, it was divided into several stages, starting with the study of blockchain and NFT technologies, application examples and related work. Since creating and managing NFTs requires a blockchain base, one of the important decisions was choosing the blockchain platform that would support the creation and maintenance of NFTs.

Once the technological decisions had been made, the next step was to implement a software module which, in the future, could be used as a tool for managing and issuing certificates for documents based on NFTs.

Keywords Blockchain, NFTs, Hyperledger Fabric, Digital Certificates, Smart Contract

Resumo

Os Non-Fungible Tokens emergiram como uma tendência nos tempos recentes o que despertou interesse entre desenvolvedores e empresas, pelas suas potencialidades nas mais diversas áreas de aplicação. Esta dissertação surgiu da iniciativa da empresa *dstelecom*, sendo o seu principal objetivo estudar a utilização de NFTs no contexto de certificação de documentos e o desenvolvimento de um módulo de *software* que emita certificados baseados em NFTs, proporcionando assim uma alternativa viável aos métodos utilizados atualmente para emissão de certificados.

Para que os objetivos da dissertação fossem cumpridos, a mesma foi dividida em várias etapas, começando pelo estudo das tecnologias *blockchain* e NFT, exemplos de aplicação e trabalhos relacionados. Uma vez que, para criar e gerir NFTs é necessário uma *blockchain* base, uma das decisões importantes foi a escolha da plataforma *blockchain* que iria suportar a criação e manutenção dos NFTs.

Depois de tomadas as decisões tecnológicas, o passo seguinte centrou-se na implementação de um módulo de *software* que, no futuro, poderá ser utilizado como uma ferramenta de gestão e emissão de certificados de documentos baseados em NFTs.

Palavras-chave Blockchain, NFTs, Hyperledger Fabric, Certificados Digitais, Smart Contract

Conteúdo

1	Introdução	1
1.1	Contexto e Motivação	1
1.2	Objetivos principais	1
1.3	Estrutura do documento	2
2	Estado da arte	3
2.1	Blockchain	3
2.1.1	Principais caraterísticas	4
2.1.2	Tipos de Blockchain	4
2.2	Tokens	5
2.3	Non-Fungible Tokens	6
2.3.1	Definição	6
2.3.2	Origem	6
2.3.3	Token standards	7
2.3.4	Geração de NFTs	9
2.3.5	Estrutura de um NFT	10
2.3.6	Aplicações de NFTs	11
2.3.7	Riscos e Desafios	13
2.4	Trabalhos relacionados	15
3	O problema e os seus desafios	18
3.1	Descrição da proposta	18
3.1.1	Proposta	18
3.2	Utilização de <i>NFTs</i> no caso de estudo	19
3.2.1	Blockchain	20
3.3	Hyperledger Iroha	21

3.4	Hyperledger Fabric	22
3.4.1	Rede Blockchain do Fabric	23
3.4.2	Modelo	25
3.4.3	Fluxo de transação	26
4	Modelação do Sistema	27
4.1	Levantamento de Requisitos	27
4.1.1	Requisitos funcionais	27
4.1.2	Requisitos não funcionais	34
4.2	Diagrama de casos de uso	34
4.3	Arquitetura Aplicacional	35
5	Desenvolvimento Aplicacional	37
5.1	Base de Dados	37
5.1.1	MongoDB	37
5.1.2	Coleções	38
5.2	Blockchain	42
5.2.1	Criação da rede	42
5.2.2	Conexão ao canal da rede	43
5.2.3	Wallet	44
5.2.4	Smart Contract	46
5.2.5	Interações com a rede blockchain	47
5.2.6	Estrutura dos NFTs	49
5.3	Assinatura Digital	53
5.3.1	Processo de Assinatura	54
5.4	Construção da API de dados	55
5.4.1	Estrutura da API	55
5.4.2	Autenticação e permissões	57
5.4.3	Rotas Implementadas	59
5.5	Interface Gráfica	62
5.5.1	Páginas	62
5.5.2	Utilizadores	64
5.5.3	Certificador	65

5.5.4	Administrador	67
5.6	<i>Deployment</i>	69
5.6.1	Blockchain	69
5.6.2	Aplicação	70
6	Conclusões e trabalho futuro	71
6.1	Conclusões	71
6.2	Perspetiva de trabalho futuro	72

Lista de Figuras

1	Sequência de blocos numa blockchain	3
2	Dados de uma NFT dentro e fora da rede (extraído de [Teshirogi, 2021]).	11
3	Esquema JSON dos metadados em NFTCert	16
4	Diagrama de casos de uso da plataforma	35
5	Esquema das tecnologias da plataforma	36
6	Ciclo de vida de uma carteira.	45
7	Estado da Blockchain	49
8	Estrutura dos NFTs da plataforma	50
9	Comparação entre protocolo HTTP e IPFS de [Daniel and Tschorsch, 2022]	51
10	Exemplo de página adicionada no documento assinado	54
11	Barra de Pesquisa do Administrador	63
12	Barra de Pesquisa do Utilizador	63
13	Página Inicial e Início de Sessão	64
14	Página de Certificar Documento - Utilizador	65
15	Página de Certificados	65
16	Página de Certificar Documento	66
17	Página de Sucesso após certificação	66
18	Página de Gestão de Utilizadores	67
19	Página de Gestão de Certificadores	68
20	Página de Registo de novos utilizadores	68
21	Página de Certificados do Administrador	69

Listagens

1	Estrutura da coleção 'Utilizador'	38
2	Encriptação da password	39
3	Verificação da password	39
4	Estrutura da coleção 'Solicitacao'	39
5	Estrutura da coleção 'Role'	40
6	Estrutura da coleção 'NFT'	40
7	Conexão à Base de Dados	41
8	Conexão à Rede Kaleido - rede.js	43
9	Conexão à Rede Kaleido - kaleido.js	43
10	Estrutura de uma identidade da carteira.	45
11	Chaincode - ERC721	46
12	Conexão à Rede Kaleido - kaleido.js	47
13	Transação - Criação de NFT	48
14	Transação - Revogação de NFT	48
15	Transação - Leitura do NFT	48
16	Conexão ao Pinata	52
17	Assinatura Digital - certificador.js	54
18	Assinatura Digital - SignPDF.js	55
19	Exemplo de uma chamada à API	62

Acrónimos

APIs Application Programming Interfaces.

BSON Binary JSON.

CA Certificate Authority.

DeFi Decentralized Finance.

ERC Ethereum Request for Comments.

HTTP Hypertext Transfer Protocol.

IPFS InterPlanetary File System.

JSON JavaScript Object Notation.

JWT JSON Web Token.

MSP Membership Service Provider.

NFT Non-Fungible Token.

P2P Peer-to-Peer.

PDF Portable Document Format.

PKCS#12 Public Key Cryptography Standards 12.

PKI Public Key Infrastructure.

SDKs Software Development Kits.

SQL Structured Query Language.

URI Uniform Resource Identifier.

URL Uniform Resource Locator.

YAC Yet-Another-Consensus.

Capítulo 1

Introdução

Nesta secção, o contexto e motivação do tema serão abordados, bem como a apresentação dos principais objetivos da dissertação. Por último, será apresentada a estrutura da dissertação.

1.1 Contexto e Motivação

Ao longo dos últimos tempos, os **Non-Fungible Token (NFT)** têm vindo a adquirir bastante popularidade e, consequentemente, têm gerado curiosidade entre desenvolvedores e empresas acerca da sua utilidade. Devido a isso, novas aplicações para estes *tokens* têm vindo a ser exploradas [Rehman et al., 2021].

Desta forma, a empresa **dstelecom**, como forma de inovação e expansão dos seus conhecimentos e exploração de novas áreas, propôs a investigação e criação de uma plataforma capaz de gerar e consultar certificados de documentos baseados em *NFTs*. Desta forma, é necessário estudar a viabilidade do uso de *NFTs* para esses certificados e, posteriormente, decidir qual plataforma *blockchain* será mais adequada para o caso de estudo.

1.2 Objetivos principais

Esta dissertação tem como principal objetivo o desenvolvimento de um módulo de software que emita *NFTs* e que, de acordo com as suas características as mesmas, possam representar certificados de documentos e sejam uma alternativa viável aos certificados existentes na atualidade.

A nível tecnológico, pretende-se estudar os *NFTs*, bem como alguns exemplos da sua aplicação e a viabilidade da sua utilização para certificados de documentos.

Será também efetuado um estudo sobre a *blockchain* e a escolha de qual plataforma será mais

adequada para o desenvolvimento da dissertação, principalmente a nível de funcionalidade e de custos, uma vez que a *dstelecom* pretende que o projeto seja desenvolvido com ferramentas *open-source* para o desenvolvimento e geração dos *NFTs*. Caso se verifique a inviabilidade na utilização deste tipo de *tokens* neste caso de estudo, será necessário proceder com uma alternativa.

1.3 Estrutura do documento

A dissertação encontra-se dividida em seis capítulos.

- **Introdução** - O primeiro capítulo aborda o contexto, motivação e principais objetivos da dissertação.
- **Estado da arte** - Neste capítulo é efetuado um estudo sobre *blockchain* e *NFTs*, a sua definição, origem, bem como alguns exemplos da sua aplicação. Serão também abordados alguns exemplos de trabalhos relacionados.
- **Problema e seus desafios** - O terceiro capítulo descreve o problema e algumas decisões tomadas para alcançar os objetivos pretendidos. Para além disso, aborda algumas questões relacionadas com a utilização de *NFTs* na dissertação e são apresentadas plataformas *blockchain* que se adequam aos requisitos impostos. Por último, é tomada a decisão sobre a *blockchain* utilizada na dissertação.
- **Modelação do Sistema** - Este capítulo relata o levantamento de requisitos, apresenta o diagrama de casos de uso e descreve a arquitetura aplicacional da aplicação desenvolvida.
- **Desenvolvimento Aplicacional** - Neste capítulo é descrito o processo de implementação da aplicação desenvolvida para que os objetivos da dissertação fossem alcançados.
- **Conclusão** - No capítulo final é feita uma conclusão ao trabalho realizado e são apresentados alguns pontos para trabalho futuro.

Capítulo 2

Estado da arte

Este capítulo aborda alguns conceitos e fundamentos sobre *tokens* em contexto de *blockchain*, além da sua utilidade e representação neste tipo de redes distribuídas. De forma mais aprofundada, revê e analisa *NFTs*, a sua origem, as suas características, a sua utilidade e aplicações no mundo real, bem como alguns desafios presentes com a sua aplicação em casos reais, e como podem ser superados.

2.1 Blockchain

Uma *Blockchain* é uma lista ligada de registos, denominados de *blocos*, em que os mesmos são conectados uns aos outros por meios criptográficos [Abaci and Ulku, 2022] dado que cada bloco aponta para o bloco imediatamente anterior através duma referência, que é o valor de *hash* do bloco anterior, formando dessa forma, uma cadeia irreversível de blocos ligados uns aos outros [Zheng et al., 2018]. Um aspeto importante desta tecnologia é a forma como se determina qual será o próximo bloco a ser publicado, sendo isto resolvido através da implementação de um modelo de consenso, como por exemplo o *proof of work* ou o *proof of stake*, entre outros [Yaga et al., 2019]. Na Figura 1, é possível observar a sequência de blocos interligados por códigos de *hash*, que formam uma *blockchain*¹.

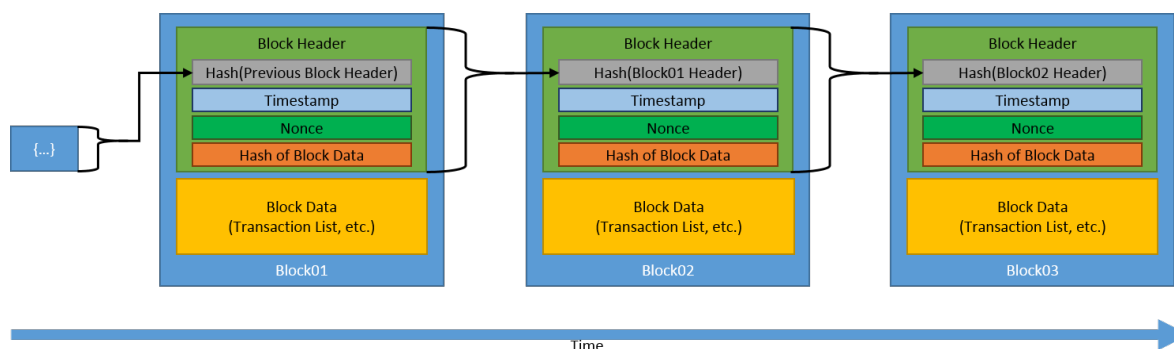


Figura 1: Sequência de blocos numa blockchain

¹ <https://www.nist.gov/blockchain>

2.1.1 Principais características

- **Irreversível** - Numa *blockchain* nenhum participante pode alterar uma transação depois da mesma ter sido arquivada na rede distribuída. Para corrigir um erro, é necessário adicionar uma nova transação para o reverter.

Uma vez que é um *arquivo partilhado*, ou seja, todos os participantes têm permissão para aceder ao registo de transações, estas são registadas apenas uma vez evitando assim esforços de duplicação.²

- **Descentralização** - as transações não precisam de validação de uma entidade central, numa *blockchain* uma transação pode ser efetuada entre dois nodos sem a autenticação dum agente central.
- **Persistência** - todas as transações efetuadas precisam de ser confirmadas e registadas em blocos distribuídos pela rede, estes fatores tornam a rede praticamente inalterável. Em adição, cada bloco transmitido precisa de ser validado por outros nodos e as transações são verificadas, desta forma, qualquer fraude ou tentativa de falsificação pode ser detetada de forma simples [Zheng et al., 2018]. Caso algum utilizador tente alterar um registo antigo, a rede é capaz de detetá-lo através dos valores de *hash* dos blocos, esta particularidade torna a *blockchain*, de certa forma, inviolável e persistente. [Abaci and Ulku, 2022].

2.1.2 Tipos de Blockchain

Segundo [Sheth and Dattani, 2019], pode-se classificar as *blockchains* nas seguintes categorias:

- **Permissioned blockchain** - Também conhecidas como blockchains privadas, atuam como um ecossistema fechado em que existe controlo de acesso para preservar a rede contra utilizadores desconhecidos. Os utilizadores necessitam de permissão para efetuarem transições e acederem à rede. Pertence a uma unidade privada ou uma organização que é a autoridade central e é responsável pelas permissões.
- **Permissionless ou Public blockchain** - é aberta para qualquer utilizador que deseje participar e interagir. É totalmente descentralizada. Ao contrário das *Permissioned blockchains* não necessita de controlo de acesso.

² <https://www.ibm.com/topics/what-is-blockchain>

- **Consortium blockchain** - Várias organizações podem partilhar responsabilidades de manutenção da *blockchain*. As organizações previamente selecionadas decidem quem pode efetuar transações ou aceder aos dados.

2.2 Tokens

No contexto da *blockchain*, *tokens* são representações de ativos digitais, ou seja, representam algo com valor atual ou futuro, e são construídos sobre *blockchains*, tendo sido originalmente aplicados pela *blockchain* da *Ethereum* [Ardavanis, 2022].

Ativos digitais *tokenizados* nas redes *blockchain*, representam ativos palpáveis como por exemplo o ouro, ou ativos não palpáveis como direitos ou a posse de algum objeto [Posavec et al., 2022], ou até mesmo outras criptomoedas, com exceção da criptomoeda nativa. Por exemplo, na rede da *Ethereum*, a "*Ether*" é a criptomoeda nativa e é utilizada para facilitar as transações nesta *blockchain*.

Tudo o que é preciso para a criação de um *token* é seguir um modelo padrão na *blockchain*, que possibilita a criação de *tokens*. Esta funcionalidade é possível através do uso de *smart contracts* [Arsov et al., 2017], que são programas computacionais que podem ser acionados sem a interação humana [Abaci and Ulku, 2022].

No contexto da *blockchain*, "*Tokenization*" é o nome dado ao processo de conversão de algo com valor num *token* digital utilizado numa aplicação da *blockchain*.

Numa *blockchain*, os *tokens* são diferenciados entre *native tokens* e *application tokens*. Os *native tokens* são inerentes à *blockchain* e são fundamentais para o seu funcionamento, enquanto que os *application tokens* são utilizados por aplicações sobre a *blockchain*, no entanto são ambos importantes para as plataformas *blockchain*. [Posavec et al., 2022]

A maioria dos *tokens* mencionados anteriormente são fungíveis, ou seja podem ser substituídos ou trocados por algo com o mesmo valor. Como por exemplo, uma moeda de um euro pode ser substituída por outra com o mesmo valor, isto também se verifica com *Bitcoin*. O valor ou a funcionalidade de uma *Bitcoin* presente numa carteira não difere de outra presente noutra carteira, são idênticas e substituíveis por conceção. Esta característica não se verifica com NFTs. [Ardavanis, 2022]

2.3 Non-Fungible Tokens

2.3.1 Definição

São unidades de dados guardadas numa *blockchain* e, como o nome indica, são não-fungíveis, ou seja, diferem uns dos outros no seu valor, características e funcionalidade. Cada *NFT* é uma entidade única e representa apenas um objeto específico. Na sua essência são representações digitais únicas e indivisíveis de ativos, sendo que, a cada momento, tem apenas um proprietário oficial. [Ardavanis, 2022].

O proprietário de um *NFT* pode trocá-lo ou vendê-lo, transferindo a posse do *token* da sua conta para a conta do comprador, no entanto a cada momento, o *token* tem apenas um proprietário. A posse de um *NFT* não pode ser partilhada Musamih et al. [2022].

Desta forma, os *NFTs* podem ser utilizados para representar a posse de objetos únicos, permitindo desta forma a *tokenização* de artefactos como arte, objetos colecionáveis ou até mesmo imobiliária. [Posavec et al., 2022]. Estes *tokens* contêm um identificador e referências para conteúdo digital como música, vídeo, imagem, entre outros e o seu valor é calculado em termos de criptomoedas.[Rehman et al., 2021]

2.3.2 Origem

O conceito de *NFT* surgiu em meados de 2012 através de Yoni Assia, co-fundador e CEO da corretora *eToro*, com as *Coloured Coins*, construídas sobre a *blockchain* da *Bitcoin*, que ao contrário de *Bitcoins*, deveriam ser únicas, identificáveis, raras e não-replicáveis. No entanto, apenas quando a *Ethereum blockchain* e os seus *smart contracts* surgiram é que se tornou possível *tokenizar* ativos de forma eficiente.

Desta forma, em 2017 a primeira coleção de *NFTs*, denominada de *CryptoPunks*, foi criada através da utilização do **Ethereum Request for Comments (ERC)-20 token standard** [Ardavanis, 2022].

O *ERC-20 standard* foi originalmente criado para o desenvolvimento de *tokens* fungíveis na rede da *Ethereum*, não sendo adequado para a criação de *tokens* únicos e não fungíveis, levando desta forma à necessidade de criação do *ERC-721 token standard*, que foi a primeira estrutura universal para os contratos *NFT* [Ardavanis, 2022]. Desta forma, no *Standard ERC-721 da Ethereum*, publicado em 2018, o termo "Non-Fungible-Token", foi criado oficialmente. Isto tornou possível a representação de vasto universo de "ativos" como a posse de algum objeto ou cargo, coleções virtuais e ativos sem valor monetário como responsabilidades ou cargos.

Com todo o interesse gerado à volta dos *NFTs*, outras *blockchains* começaram a desenvolver os seus

próprios NFTs [Posavec et al., 2022], sendo que atualmente, grande parte das redes *blockchain* suportam este tipo de *tokens*.

2.3.3 Token standards

Um *smart contract* é código executável que corre numa *blockchain* para ajudar a executar e fazer cumprir os termos de um acordo, sendo o seu principal objetivo a execução, de forma automática, dos termos desse mesmo acordo, quando as condições especificadas são atendidas [Alharby and Van Moorsel, 2017]. O código de um contrato pode ser invocado por utilizadores da *blockchain* e o código invocado tem ações na *blockchain* podendo as mesmas ser de consulta ou de registo de novas informações.

Um *smart contract* para funcionar como desejado numa determinada *blockchain* deve cumprir um conjunto de regras, que são os *smart contract standards*. Os *token standards* são conjuntos de regras, condições e funções que determinam como um *token* criptográfico funciona, e são um subconjunto dos *smart contract standards*.

Nas *blockchains* que suportam *smart contracts*, os *token standards* funcionam como um guia para a criação, emissão e implementação de *tokens* nessas *blockchains* ³.

De seguida, enumeram-se alguns dos *token standards* mais populares:

- **ERC-20** - é o *token standard* mais utilizado e mais genérico para a criação e implementação de tokens fungíveis na *blockchain* da *Ethereum* [Posavec et al., 2022].

Este *token standard* define um conjunto de regras às quais todos os *tokens* fungíveis devem seguir.

De seguida encontram-se apresentadas as principais regras que um *ERC-20* deve implementar ⁴.

Funções

- **totalSupply()** - Retorna a quantidade de tokens existentes.
- **balanceOf(account)** - Retorna a quantidade de tokens que uma conta (*account*) possui.
- **transfer(recipient, amount)** - Move uma quantidade (*amount*) de tokens da conta do chamador para o destinatário (*recipient*) e retorna um booleano que indica se a operação teve sucesso. Emite um evento *Transfer*, apresentado em baixo.
- **allowance(owner, spender)** - Retorna o número restante de tokens que o gastador poderá gastar em nome do *owner* por meio de *transferFrom*. Este valor é alterado quando *approve* ou *transferFrom* são chamadas.

³ <https://crypto.com/university/what-are-token-standards>

⁴ <https://docs.openzeppelin.com/contracts/2.x/api/token/erc20#ERC20>

- **approve(spender, amount)** - Define a quantidade permitida ao *spender* sobre os *tokens* do chamador, retornando um booleano que indica se a operação teve sucesso. Emite um evento *Approval*, apresentado em baixo.
- **transferFrom(sender, recipient, amount)** - Move uma quantidade de *tokens* do remetente (*sender*) para o destinatário (*recipient*) através do mecanismo de permissão (*allowance*), retornando um booleano que indica se a operação teve sucesso. Emite um evento *Transfer*.

Eventos

- **Transfer(from, to, value)** - Emitido quando *tokens* de valor são movidos de uma conta para outra.
- **Approval(owner, spender, value)** - Emitido quando a permissão de um *spender* para um *owner* é definida por uma chamada a *approve*. Sendo *value* a nova permissão.
- **ERC-721** - é o *token standard* utilizado para criação e implementação de NFTs na *blockchain* da *Ethereum*⁵. Como diz respeito a *tokens* não fungíveis possui novas funcionalidades para respeitar as características deste tipo de *tokens*.

De seguida apresentam-se as principais regras que um *ERC-721* deve implementar⁶.

Funções

- **balanceOf(owner)** - Obtém a quantidade de *NFTs* que na conta do *owner*.
- **ownerOf(tokenId)** - Retorna o proprietário de um *NFT* especificado por *tokenId*.
- **safeTransferFrom(from, to, tokenId)** - Transfere de forma segura, através dum mecanismo de segurança, um *NFT* (*tokenId*) de uma conta para outra. O *tokenId* deve ser possuído pelo remetente (*from*). Se o chamador não for o remetente (*from*), deve ser atribuída permissão para mover este *NFT* por *approve* ou *setApprovalForAll*.
- **transferFrom(from, to, tokenId)** - Transfere a posse de um dado *tokenId* para outro endereço. Tal como *safeTransferFrom*, se o chamador não for o remetente, deve ser atribuída permissão para mover o *NFT*. No entanto, a utilização deste método é desaconselhada, devendo ser utilizado *safeTransferFrom* sempre que possível.
- **approve(to, tokenId)** - Autoriza que outro endereço transfira um dado *tokenId*. Pode apenas ser chamada pelo dono do *token* ou por um operador aprovado.

⁵ <https://crypto.com/university/what-are-token-standards>

⁶ <https://docs.openzeppelin.com/contracts/4.x/api/token/erc721#IERC721-Transfer-address-address-uint256->

- **getApproved(tokenId)** - Obtém o endereço aprovado para um dado *tokenId*.
- **setApprovalForAll(operator, approved)** - Define ou remove a aprovação de um dado operador. O operador tem permissão para transferir todos os tokens do remetente em seu nome.
- **isApprovedForAll(owner, operator)** - Indica se um operador é aprovado por um dado *owner*.

Eventos

- **Transfer(from, to, tokenId)** - Emitido quando um *tokenId* é transferido de *from* para *to*.
- **Approval(owner, approved, tokenId)** - Emitido quando um *owner* dá permissão a outro endereço para gerir o *tokenId*.
- **ApprovalForAll(owner, operator, approved)** - Emitido quando um *owner* ativa ou desativa a permissão ao *operator* de gerir todos os seus ativos.
- **ERC-777** - permite a construção de funcionalidades extra e recursos avançados para os *tokens* permanecendo compatível com versões anteriores do *ERC-20* [di Angelo and Salzer, 2020]. É um *token standard* para tokens fungíveis que melhora o *ERC-20*.
- **ERC-1155** - permite trocas e agrupamento de transações de forma mais eficiente e dessa forma economiza custos. Este padrão permite a criação de *tokens* fungíveis e não fungíveis ⁷.

2.3.4 Geração de NFTs

O nome dado ao processo de criação de um *NFT* é conhecido por "*minting*". Quando um *NFT* é criado, uma chave única é gerada e, ao mesmo tempo, os dados relacionados com a posse do *NFT* são arquivados na *blockchain*. Isto significa que apenas a pessoa com controlo sobre a chave digital única pode ser o verdadeiro dono desse mesmo *token*. Qualquer cópia dos dados representados por um *NFT*, não tem o identificador único associado a si, tornando fácil diferenciar o ficheiro original das cópias. [Ardavanis, 2022].

⁷ <https://ethereum.org/en/developers/docs/standards/tokens/>

2.3.5 Estrutura de um NFT

Nesta secção encontram-se explicadas e definidas as principais componentes dos NFTs.

- A **blockchain** é o principal componente de um *NFT* uma vez que contém e mantém as características únicas destes *tokens*, que os distinguem entre si.
- Cada NFT pertence a um *smart contract* com um **endereço único**. Isto é necessário para os utilizadores estarem cientes da autenticidade do contrato do *NFT* com que interagem, ou seja, este endereço é essencial para garantir que um utilizador está a interagir com o *smart contract* correto.
- O criador do *smart contract* é identificado pelo seu endereço único, que é denominado de **Unique Creator Address** [Musamih et al., 2022].
- Como referenciado anteriormente, cada *NFT* é identificado por uma **chave única** que é gerada durante o seu processo de criação e esta não pode ser duplicada.
- O **histórico** de transações de um *NFT* é guardado de forma permanente e imutável na respetiva *blockchain*, este fator torna possível seguir e rastrear os *tokens*. A informação sobre o proprietário atual e os anteriores é guardada.
- Os **metadados** de cada *NFT* podem ser guardados na rede *blockchain* ou não, dependendo do tamanho dos dados envolvidos. Caso os metadados sejam guardados fora da rede, um *token URI* do conteúdo externo é guardado na *blockchain* para torná-lo imutável [Musamih et al., 2022]. Ou seja, este *URI* é um atributo que indica a localização das informações de conteúdo (metadados) associadas a um *NFT*. Os metadados de um *NFT* podem conter o nome do conteúdo, a sua descrição, o **URL** que indica a localização dos dados de conteúdo que estão associados ao *NFT*, entre outros.

É importante referir que, uma vez que o tamanho dos dados que podem ser registados na *blockchain* não é grande, os metadados e os dados de conteúdo são normalmente geridos fora da *blockchain* (*off-chain*) [Teshirogi, 2021]. A Figura 2 esquematiza a informação relatada.

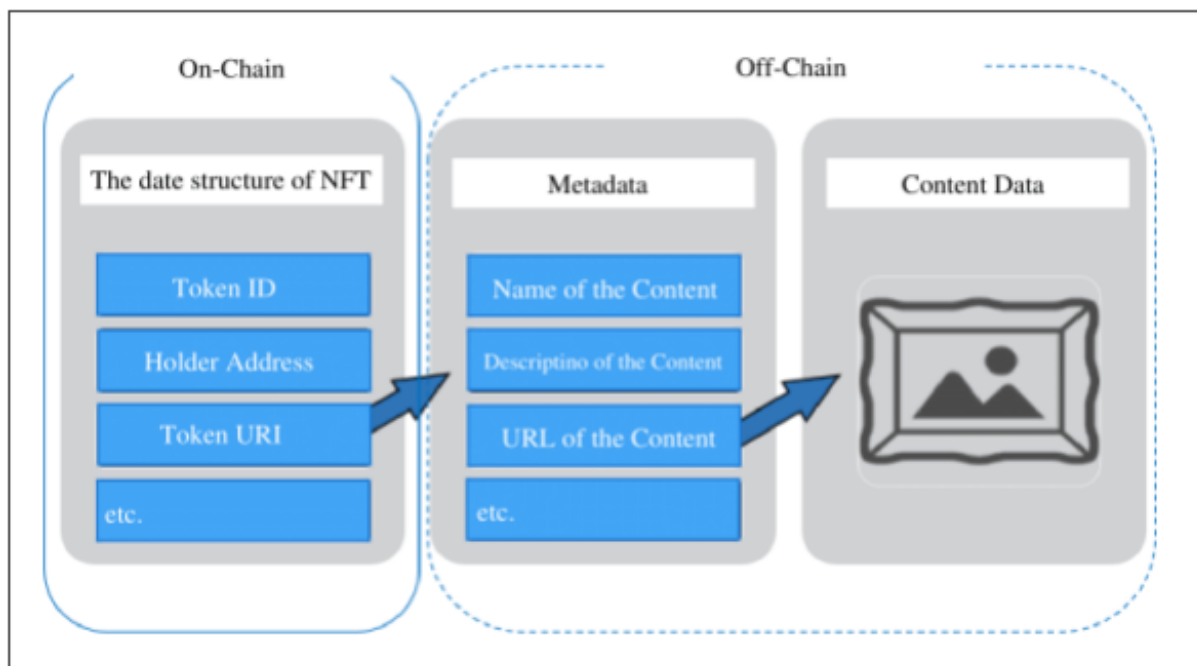


Figura 2: Dados de uma NFT dentro e fora da rede (extraído de [Teshirogi, 2021]).

2.3.6 Aplicações de NFTs

As características destes *tokens* atraíram a atenção de várias áreas e apesar de estarem, cada vez mais, a surgir novas aplicações para estes *tokens*, os mesmos já são utilizados em várias áreas com vários propósitos diferentes.

Esta secção discute várias aplicações de *NFTs* atuais e emergentes e como estes estão a moldar o futuro de várias áreas.

Arte digital

Todo o tipo de arte seja física ou digital apresenta risco de ser falsificada ou roubada. A utilização de *NFTs* torna muito difícil ou até mesmo erradica a possibilidade de falsificação de arte digital, uma vez que apesar de ser possível fazer cópias de conteúdo digital, os *NFTs* asseguram que cada exemplar pertence apenas ao seu comprador e cada um deles é único e insubstituível. Este fator atraiu uma enorme atenção e curiosidade por parte de colecionadores. [Rehman et al., 2021] O interesse para os artistas também aumentou, uma vez que no processo de criação de um *NFT* como conteúdo artístico, os criadores da arte podem atribuir o preço à sua arte e, para além disso, podem definir direitos e regras para vendas da arte, codificadas no *smart contract* e deste modo evitam comissões altas das galerias e museus. [Posavec et al., 2022]. Por isso, a *tokenização* da arte por meio de *NFTs* permite, desta forma, que os artistas não

só, lucrem mais com as suas vendas como também recebam uma percentagem cada vez que o conteúdo é vendido a novos compradores, o que dantes era imprático uma vez que era difícil rastrear a arte e a sua posse [Rehman et al., 2021].

O mesmo acontece com outro tipo de artigos colecionáveis, em que a utilização desta tecnologia se revela útil para a proteção e troca dos mesmos, sejam eles imóveis virtuais, cartas de coleção, diamantes, entre outros [Posavec et al., 2022].

Moda

Várias marcas e empresas perdem grandes quantidades monetárias para falsificações dos seus produtos, isto pode ser prevenido com o uso de NFTs [Rehman et al., 2021], devido às suas características de unicidade e exclusividade analisadas previamente.

Os produtos fabricados por empresas de moda luxuosas têm uma coisa em comum com *NFTs*, que é a sua exclusividade. Como sabemos, a raridade é um fator importante nas indústrias de luxo e é também algo presente no mercado de *NFTs* [Kirjavainen et al., 2022]. Em consequência disto, algumas empresas já começaram a associar *NFTs* a produtos físicos de forma a distinguir os seus direitos de posse e reter a sua exclusividade [Rehman et al., 2021].

Bilheteira de eventos

Na compra de um ingresso para eventos, sejam eles desportivos, culturais ou de entretenimento, existe risco do bilhete ser falsificado ou de não ser válido.

Resumidamente, a utilização de *NFTs* neste contexto apresenta várias vantagens, como a transparência, uma vez que bilhetes baseados em *NFT* desde a sua criação até à sua venda toda a informação está de forma transparente publicada na *blockchain* permitindo desta forma que todos possam provar a autenticidade dos bilhetes, ajuda também no controlo de futuras vendas, através dos *smart contracts* [YuanJiang and Zhou, 2022], podendo desta forma eliminar a revenda ilegal de bilhetes. Para além disso, como cada *NFT* é único e possui características específicas, a sua utilização neste âmbito pode servir não só como comprovativo para assistir a algum evento como também pode melhorar a experiência dos consumidores pois pode ser guardado para sempre como uma lembrança. [YuanJiang and Zhou, 2022]

Indústria dos vídeo-jogos

A comunidade dos vídeo-jogos e os seus desenvolvedores mostraram bastante interesse em *NFTs*, uma vez que as suas características podem fornecer direitos de posse sobre objetos dentro de um jogo, promo-

vendo o aspeto económico do mesmo. Isto revela interesse, uma vez que permite aos jogadores criarem ou comprarem objetos num jogo e posteriormente venderem e lucrarem com eles [Posavec et al., 2022]. Como acontece com os artistas e a sua arte digital, os desenvolvedores podem também receber uma percentagem por cada venda de um objeto no jogo [Rehman et al., 2021].

Certificados e documentos

Como referenciado previamente, *NFTs* são na sua essência certificados digitais em forma de um *token* numa *blockchain*. Consequentemente, é esperado que documentos reais emitidos por instituições beneficiem desta tecnologia [Ardavanis, 2022]. A presença de documentos reais na forma de *NFTs* permite uma verificação rápida e automática da sua autenticidade, podendo desta forma erradicar a carga de trabalho associada à manutenção desses registos [Rehman et al., 2021].

Além disso, a emissão dos certificados na *blockchain* torna-os resistentes a adulteração, o que reduz a probabilidade de aparecerem documentos fraudulentos [Rehman et al., 2021].

Atualmente existem vários exemplos e casos de utilização de *NFTs* para este propósito, como é o caso do projeto *NFTCert*, que desenvolve uma plataforma de emissão de certificados académicos [Zhao and Si, 2021]. Outro exemplo é a utilização de *NFTs* em certificados digitais de vacinação contra a *Covid-19*, na República de San Marino [Posavec et al., 2022]. Mais um exemplo da utilização de *NFTs* para certificação encontra-se na *Duke Pratt School of Engineering*, em que os alunos que completaram uma sequência de cursos sobre fundamentos da tecnologia *blockchain* receberam certificados de conclusão na forma de *NFTs* que foram gerados e distribuídos via *blockchain* na rede *Polygon*⁸.

2.3.7 Riscos e Desafios

Os *NFTs*, como qualquer outra tecnologia recente, acarreta alguns desafios relacionados ao seu desenvolvimento e utilização. Esta secção aborda alguns desafios da aplicação de *NFTs*.

Velocidade de confirmação

As aplicações que utilizam *NFTs* dependem da *blockchain* subjacente e um desempenho lento da mesma torna também lenta a velocidade de confirmação dos *NFTs*. Isto deve-se principalmente aos protocolos de mineração da *blockchain* utilizados para o mecanismo de consenso *proof of work*. Para superar este desafio, a tendência é que a maioria dos mecanismos de *proof of work* mudem para protocolos de mineração mais rápidos e baratos [Posavec et al., 2022].

⁸ <https://pratt.duke.edu/about/news/duke-engineerings-fintech-program-sends-certificates-coursera-students-nfts>

Consumo energético

Um dos desafios mais evidentes para aplicações da *blockchain* é o grande consumo de energia que as mesmas acarretam. Os gastos de energia com o mecanismo *proof of work* são insustentáveis e as transações de *NFTs* consomem ainda mais energia, uma vez que necessitam de operações mais complexas com contratos que requerem mais recursos computacionais e armazenamento.

Os esforços para superar este desafio estão focados na transição para outro mecanismo de consenso. Contudo, a transição deste mecanismo, mantendo todos os princípios de segurança e integridade, é complexa e requer esforços e investimentos consideráveis [Posavec et al., 2022].

Links quebrados

As aplicações de *NFTs* apoiam-se numa *blockchain* para estabelecerem direitos de posse de ativos digitais, no entanto apesar destes registos não poderem ser alterados estes apenas armazenam um *link* para o ativo real, que se encontra num *URL* convencional onde foi guardado aquando da sua criação. Como qualquer outro *URL*, pode deixar de funcionar por várias razões, o que pode fazer com que um *link* para um ativo digital aponte para uma página indisponível.

Algumas das plataformas *NFT* estão a implementar o **InterPlanetary File System (IPFS)**, ou seja, um sistema de arquivos baseado na *web Peer-to-Peer (P2P)*, e um protocolo que ajuda a garantir a distribuição dos arquivos em vários *hosts*, como tentativa de resolução do problema mencionado. Este sistema permite que o conteúdo seja acedido desde que alguém na rede possua uma cópia. No entanto não resolve completamente o problema de *links* quebrados [Posavec et al., 2022].

Cibersegurança

O avanço tecnológico contribui para uma maior eficiência no comércio de ativos digitais, no entanto também acarreta alguns riscos indesejáveis. Com o aumento do número de transações de *NFTs*, os riscos de fraude e de cibersegurança associados a estas operações também têm aumentado, existindo a possibilidade de utilizadores mal-intencionados imitarem criadores de *NFTs* famosos e venderem essas falsificações. A imitação de *NFTs* populares ou o roubo de direitos de autor são alguns dos principais riscos associados a tokens não fungíveis em termos de cibersegurança e fraude [Rehman et al., 2021].

Smart contracts

Um dos principais riscos associados a *NFTs* é o desenvolvimento e segurança dos *smart contracts*.

Recentemente a *Poly Network*, um famoso protocolo de finanças descentralizadas **DeFi**, foi alvo de um ataque informático, em que o roubo de *NFTs* causou uma perda de 600 milhões de dólares, tendo sido destacadas falhas graves na segurança de *smart contracts*.

Uma das principais causas que podem levar a vulnerabilidades nos *smart contracts* é a inexistência de padrões de segurança claros com validação em todo o setor, no caso do *Solidity*, que é a linguagem de implementação de *smart contracts* na rede da *Ethereum* [Rehman et al., 2021].

2.4 Trabalhos relacionados

Nesta secção um conjunto de trabalhos relacionados publicados serão revistos, prestando atenção particularmente a casos de estudo de utilização da *Blockchain* e *NFTs* para certificados de documentos.

NFTCert

Como referido na secção anterior, em [Zhao and Si, 2021] os autores propõem uma estrutura para emissão e gestão de certificados escolares baseados em *NFTs*, denominada de *NFTCert*, com o objetivo de prevenir que entidades fraudulentas emitam certificados ilegítimos ou falsos. Esta proposta é baseada numa *blockchain* privada em que apenas instituições educacionais verificadas podem ter permissão para aceder. Para emitirem certificados baseados em *NFTs* para os alunos, as instituições e universidades verificadas juntam-se à *blockchain* privada, e participam na mesma como nodos para manter a sua integridade e para gerar confiança entre os utilizadores.

A plataforma funciona da seguinte forma:

1. As instituições verificadas primeiro geram os certificados baseado em *NFT* e aguardam pela conclusão do pagamento por parte do aluno.
2. Após efetuar o pagamento, o certificado (*NFT*) será enviado para a carteira digital do aluno.
3. O aluno pode partilhar o endereço da sua carteira digital e qualquer entidade pode facilmente verificar a autenticidade do certificado.
4. As instituições que estão conectadas com a *blockchain* privada podem aceder aos certificados (*NFTs*) armazenados.

Para além disto, o sistema proposto *NFTCert* encontra-se dividido em quatro componentes fundamentais.

- **Esquema** - baseado no padrão **ERC-721** mencionado previamente, descreve os dados e campos necessários para o certificado. De acordo com o padrão *ERC-721*, um esquema **JSON** de metadados é definido para representar o certificado. O esquema é apresentado na Figura 3.

```
{
  "title": "Certificate Metadata",
  "properties": {
    "certification_authority_name": String,
    "degree_title": String,
    "degree_conferral_date": String,
    "certificate_URI": String,
    "hash": SHA256
  }
}
```

Figura 3: Esquema JSON dos metadados em NFTCert

O campo *hash* será preenchido com uma encriptação *SHA256* do nome, dia e local de nascimento e nacionalidade do aluno, nome da instituição educacional, título e data de atribuição do grau educacional e o *URI* do certificado.

- **Processo de Minting** - cria uma assinatura verificável, popula o campo hash mencionado no ponto anterior e emite os certificados na *blockchain*.

A imagem do certificado é guardada num servidor *Web* ou *IPFS*, e o *URI* do certificado no esquema JSON aponta para a imagem do certificado associada ao *NFT*.

- **Processo de verificação** - demonstra como é possível verificar a autenticidade e a posse do certificado através das informações dos alunos e das informações do certificado (NFT). Os passos para o processo de verificação são os seguintes:

1. Instituições extraem do currículo do aluno, o nome, dia e local de nascimento e nacionalidade.
2. Instituições extraem dos *metadados* do certificado, outras informações relacionadas com o certificado.
3. A assinatura *hash* é calculada através do algoritmo de *hash* criptográfico *SHA256* e de seguida é comparado com a assinatura *hash* presente nos metadados do certificado. Caso sejam iguais, o certificado é autêntico e válido, caso contrário alguma coisa está errada e o certificado poderá ser inválido.

Para além disto, uma vez que os *NFTs* permitem a verificação do seu histórico e proprietário, é

possível verificar de forma simples se o antigo proprietário é efetivamente o endereço publicado pela instituição educacional numa página oficial ou governamental.

- **Processo de revogação** - quando emitem um certificado de forma incorreta, as instituições podem substituir a imagem do certificado guardada no seu servidor por uma declaração de anulação. As instituições podem confirmar a validade acedendo à informação do *URI* do certificado.

Desta forma, é possível concluir que este projeto poderá servir de exemplo para a plataforma em desenvolvimento, uma vez que a sua estrutura e todos os seus processos estão muito bem definidos. Para além disso, apresenta semelhanças ao que é esperado do projeto desta dissertação.

Blockcerts

Outro exemplo da aplicação de certificados e registos oficiais baseados em *blockchain* é o *Blockcerts*⁹, que é uma plataforma *open-source* para criação de aplicações de emissão e verificação de registos oficiais na *blockchain*, podendo incluir certificados, credenciais académicas, licenças profissionais, entre outros.

Esta plataforma pode ser utilizada por outros projetos e contém componentes para criação, emissão, visualização e verificação de certificados através de qualquer *blockchain*, sendo todas estas partes essenciais para um ecossistema completo.

Certificado de vacinação de San Marino

Como mencionado em [Posavec et al., 2022], na República de San Marino, foi aprovada a utilização de *NFTs* nos certificados digitais do *Covid-19* com a ajuda da *VeChain*, uma das *blockchains* públicas mais populares. Estes certificados contêm dois códigos *QR*, em que um desses códigos encontra-se de acordo com os requisitos da União Europeia, enquanto o outro pode ser verificado por qualquer pessoa em qualquer parte do mundo. Qualquer dispositivo pode ler o código e é direcionado para uma página web onde a validade do certificado pode ser verificada. Isto é permitido através da vinculação a um *NFT* que, como referido anteriormente, é um certificado único e exclusivo de autenticidade digital. E ao ser registado na *blockchain* pública *VeChainThor* garante a sua imutabilidade e acessibilidade. Esta solução inovadora permite a verificação destes certificados fora da União Europeia¹⁰.

Através da análise dos projetos mencionados é possível concluir que a utilização de *blockchain* e *NFTs* para certificados digitais já é uma realidade.

⁹ <https://www.blockcerts.org/about.html>

¹⁰ <https://www.dnv.com/news/san-marino-approves-national-green-pass-allowing-citizens-and-residents-to-move-freely-203736>

Capítulo 3

0 problema e os seus desafios

Este capítulo relata as principais funcionalidades levantadas para a aplicação a desenvolver e aborda também algumas decisões efetuadas para o desenvolvimento da mesma.

3.1 Descrição da proposta

Inicialmente, o principal objetivo da dissertação era o desenvolvimento de uma aplicação capaz de gerar e consultar certificados de moradias baseados em NFTs, de acordo com as suas características de ligação de fibra ótica. Neste caso, a informação relativa às moradias encontrar-se-ia numa base de dados disponibilizada pela *dstelecom* e seria apenas necessário o desenvolvimento do mecanismo para a criação dos certificados das habitações.

No entanto, o objetivo do projeto mudou do desenvolvimento de uma ferramenta mais específica, focada apenas na certificação das moradias, de acordo com características da infraestrutura de conectividade disponível, para uma mais generalizada, para permitir a certificação de qualquer tipo de documento.

Desta forma, poderá ser utilizada para vários propósitos e por várias entidades diferentes.

3.1.1 Proposta

Assim sendo, a aplicação desenvolvida é um motor de certificação de documentos baseados em *NFTs* e terá o seguinte *input* e *output*:

- **Input** - A aplicação receberá como *input* um **documento não certificado** e o motor tratará do processo de certificação do documento inserido.
- **Output** - Após efetuar o processo de certificação do documento dado como *input*, o *output* gerado pela aplicação será o **documento certificado**.

Para além de funcionar como motor de emissão de certificados digitais, a aplicação terá de possibilitar também a consulta dos certificados gerados. Desta forma a aplicação terá duas funcionalidades fundamentais:

- **Emissão de certificados digitais** - Os utilizadores poderão inserir documentos que requerem certificação.
- **Consulta dos certificados digitais** - Os utilizadores poderão consultar os documentos certificados pela aplicação.

3.2 Utilização de NFTs no caso de estudo

Uma vez que se pretende desenvolver uma aplicação que emita certificados digitais baseados em NFTs é importante estudar se a utilização desta tecnologia emergente é viável para este propósito.

Como mencionado no segundo capítulo, atualmente já existem vários exemplos de aplicação de *NFTs* para certificados digitais seja em projetos universitários, como também por instituições que emitiram certificados digitais baseados em *tokens* não fungíveis.

De acordo com as suas características de unicidade, autenticidade e indivisibilidade, será expectável que a utilização de NFTs para certificados digitais seja adequada, uma vez que cada *NFT* atua na sua essência como um certificado digital que comprova a posse de um determinado ativo.

No entanto, para certificados digitais serem úteis, válidos e autênticos é importante que os mesmos tenham em consideração e superem alguns riscos associados à sua falsificação e segurança.

A falsificação de documentos é um problema grave e que pode ser controlado com a utilização de certificados baseados em *NFTs*. Como referenciado previamente, a emissão de certificados e outros documentos na *blockchain* torna-os resistentes a adulteração e, desta forma, reduz a probabilidade de existirem documentos fraudulentos. Para além disso, existe a possibilidade de alguém mal intencionado emitir algum documento em nome de alguma instituição. No entanto, utilizando certificados baseados em *NFTs*, uma vez que as suas atividades de *minting* e de transferência são acessíveis publicamente [Zhao and Si, 2021], é possível observar o endereço do emissor do *NFT* podendo, desta forma, ser comprovado quem realmente emitiu o *token*.

Segundo as pesquisas efetuadas nos capítulos anteriores, confirma-se que os *NFTs* têm apenas um proprietário oficial e uma vez que estes *tokens* se encontram suportados por uma *blockchain*, é praticamente impossível alterar os registos de posse dos *NFTs* ou duplicá-los, desta forma, utilizando certificados digitais baseados em *NFTs* é possível assegurar a unicidade e a posse única dos mesmos [Zhao and Si,

2021]. Inclusive, os *tokens* não fungíveis são verificáveis, ou seja, a sua autenticidade pode ser verificada por qualquer pessoa com acesso à *blockchain* e desta forma facilitar o processo verificação da validade de um certificado digital.

Por último, visto que os *NFTs* existem como registos permanentes na *blockchain* e são imutáveis, podem garantir confiança na validade de certificados digitais.

Uma das questões que se levanta é o facto dos *NFTs* serem transferíveis. Se eles representarem certificados digitais, a ideia de serem transferíveis não faz sentido, uma vez que não seria apropriado que um certificado pudesse ser transferido de um proprietário para outro. No entanto, é possível tornar um *NFT* intransferível, através da personalização do *smart contract ERC-721*, garantindo, desta forma, que os *NFTs* permaneçam com os proprietários originais.

Devido às razões apresentadas, a aplicação desta tecnologia na plataforma de geração de certificados digitais prosseguirá.

Desta forma, após a produção do documento certificado por parte do motor de geração de certificados, para cada *output* será gerado um *NFT* que representará o documento certificado produzido.

3.2.1 Blockchain

Uma das decisões mais importantes a tomar para o desenvolvimento da aplicação é a escolha da *blockchain* a utilizar para publicar os *NFTs*. Para tomar a melhor decisão para o presente caso de estudo, foi necessário estudar os seus requisitos e necessidades a ter em conta.

Dado que os *NFTs* a emitir na blockchain representarão certificados digitais reais, será necessário ter especial atenção a questões de segurança e proteção de dados. Em blockchains privadas, o acesso à rede é restrito e os dados guardados nas mesmas encontram-se mais protegidos de acessos não autorizados. Desta forma, oferecem maiores níveis de segurança e proteção de dados. Para além disso, a plataforma será gerida pela *dstelecom* e, por isso, é importante que a empresa tenha controlo sobre a rede, incluindo quem poderá aceder e efetuar transações.

A nível de desempenho, numa blockchain pública como a *Ethereum* ou *Bitcoin*, os mecanismos de consenso utilizados podem ser lentos e utilizar uma grande quantidade de poder computacional. Por outro lado, numa blockchain privada, o mecanismo de consenso pode ser customizado para se adequar melhor às necessidades da *dstelecom* e consequentemente melhorar o desempenho. Desta forma, a utilização de uma blockchain privada será a escolha mais adequada para o caso de estudo. De acordo com os requisitos definidos em colaboração com a *dstelecom*, é importante escolher uma plataforma *open-source* para a geração dos *NFTs* e às transações efetuadas na rede. Deste modo, é importante

que a opção escolhida respeite estas características, ou seja, uma *blockchain privada, open-source* e sem custos associados a transações.

Desta forma, foram procuradas tecnologias que atendessem estas características.

3.3 Hyperledger Iroha

De acordo com os requisitos gerais mencionados em 3.2.1, é possível afirmar que o *Hyperledger Iroha*¹ é uma boa escolha, uma vez que:

- é uma plataforma blockchain privada, para uso geral, que pode ser utilizada para gerir ativos digitais, identidade e dados serializados, tendo utilidade para vários exemplos de aplicação.
- para além da interação com o sistema ter permissões, as consultas à informação possuem também permissões, desta forma, o acesso a todos os dados pode ser controlado.
- permite que utilizadores executem funções comuns, como criação e transferência de ativos digitais, utilizando comandos previamente construídos que se encontram no sistema. Este fator elimina a necessidade de escrever *smart contracts* complexos e difíceis de testar, permitindo aos desenvolvedores concluir tarefas simples com mais rapidez e com menos riscos. Os comandos integrados do *Iroha* são uma grande vantagem ao compararmos com outras plataformas, uma vez que torna simples a realização de tarefas comuns, como criar, registar e transferir ativos digitais.
- inclui recursos como suporte a várias linguagens de programação, como Python, JavaScript e Java.
- possui um novo algoritmo de consenso, denominado de **Yet-Another-Consensus (YAC)**, que apresenta elevado desempenho e permite a submissão de transições com latência reduzida.
- não possui uma criptomoeda nativa, usualmente não possui taxas de transação integradas.
- possibilita a configuração das taxas associadas às transferências na rede, ou seja, o custo associado às transações na rede depende da implementação e configuração da rede.

¹ <https://iroha.readthedocs.io/en/main/>

3.4 Hyperledger Fabric

No entanto, para além do *Iroha* existe uma outra plataforma do projeto *Hyperledger* denominada de *Hyperledger Fabric* ² que se revela igualmente adequada para este caso de estudo, de acordo com os requisitos mencionados. Visto que:

- permite inovação, versatilidade e otimização para uma vasta gama de casos de uso, visto que possui uma arquitetura extremamente modular e configurável.
- é uma plataforma blockchain *open-source* projetada para casos de uso corporativos e é particularmente adequada para a criação de redes blockchain privadas. Permite a criação de canais privados entre participantes específicos, onde eles podem compartilhar e transacionar informações confidenciais.
- oferece suporte a várias linguagens de programação, incluindo *Go*, *Java* e *Node.js*, o que oferece flexibilidade em termos de escolha da linguagem que melhor atende aos requisitos do projeto. O que significa que grande parte dos desenvolvedores já possui as habilidades necessárias para desenvolver *smart contracts*, não sendo necessário nenhum treino adicional para aprender uma nova linguagem. Para além disso, possui um vasto conjunto de **Application Programming Interfaces (APIs)** e **Software Development Kits (SDKs)** que facilitam o desenvolvimento e a implementação de *smart contracts* na plataforma.
- um fator diferenciador da plataforma é o suporte a protocolos de consenso conectáveis que permitem que a plataforma seja personalizada com mais eficiência para atender a casos de estudo específicos. Pode utilizar protocolos de consenso que não requerem uma criptomoeda nativa para incentivar a mineração com custo associado ou para alimentar a execução de *smart contracts*.

Depois de efetuada a análise das tecnologias mencionadas, verificou-se que o *Hyperledger Fabric* se destaca uma vez que é uma ferramenta mais estabelecida e apresenta uma comunidade maior de utilizadores, sendo isto uma enorme vantagem porque devido a este fator existe mais apoio e documentação para ajudar no desenvolvimento do projeto.

De seguida, serão apresentadas as principais características do *Fabric* ³.

² <https://hyperledger-fabric.readthedocs.io/en/latest/whatis.html>

³ <https://hyperledger-fabric.readthedocs.io/en/release-2.2/>

3.4.1 Rede Blockchain do Fabric

Uma rede *blockchain* é uma infraestrutura técnica que fornece, a aplicações, serviços de *ledger* e contratos inteligentes. Os contratos inteligentes são utilizados para gerar transações que posteriormente são distribuídas pelos *peer nodes* da rede, onde são registados de forma permanente na sua cópia do *ledger*. Os utilizadores de aplicações poderão ser administradores da rede *blockchain* ou utilizadores finais através de aplicações cliente.

Na maioria dos casos, várias organizações reúnem-se como um consórcio para formar a rede e as suas permissões são determinadas por um conjunto de políticas acordadas pelo consórcio quando a rede é configurada, no entanto, a configuração inicial poderá ser futuramente alterada caso as organizações participantes concordem. Para além disso, as redes podem conter vários canais e os componentes de rede podem ser associados a vários canais. Por exemplo, os nós de uma organização podem participar em dois canais diferentes.

Ledger

Um *ledger* é composto por duas partes distintas mas relacionadas: *blockchain* e a base de dados de estado ou estado global. Ao contrário de outras *ledgers*, *blockchains* são imutáveis, que significa que, uma vez que um bloco é adicionado à corrente, este não pode ser alterado. Por outro lado, o estado global é uma base de dados que contém o valor atual do conjunto de pares chave-valor que foram adicionados, alterados ou removidos pelo conjunto de transações validadas e confirmadas na *blockchain*. Cada *peer* num canal mantém a sua própria cópia do *ledger* que é consistente com a cópia de todos os outros *peers* por meio de um processo denominado de consenso.

Organizações

As organizações também conhecidas por membros são convidadas para se juntarem à rede *blockchain*, por um *blockchain network provider*. Uma organização junta-se à rede adicionando o seu **Membership Service Provider (MSP)** à rede. O MSP define como outros membros da rede podem verificar se as assinaturas, sobre as transações, foram geradas por uma identidade válida, emitida por aquela organização. Os direitos de acesso específicos das identidades dentro de um MSP são geridos por políticas que também são acordadas quando a organização se conecta à rede.

Membership Service Provider

O MSP é um componente abstrato do sistema que fornece credenciais aos clientes e *peers* para que estes participem na rede. Os clientes utilizam essas credenciais para autenticar as suas transações e os *peers* utilizam essas credenciais para autenticar os resultados do processamento da transação.

Peer

Um *peer* é uma entidade da rede que mantém uma cópia do *ledger* e invoca funções do *chaincode* para realizar operações de leitura e escrita no *ledger*. Os *peers* são propriedade dos membros da rede e são mantidos por estes. O *endpoint* de uma organização é um *Peer*.

Ordering Service

Um conjunto definido de nós que ordena as transações num bloco e, de seguida, distribui os blocos aos *peers* conectados para validação e confirmação. Este serviço existe independentemente dos processos dos *peers* e ordena as transações por ordem de chegada para todos os canais da rede.

Canal

Um canal é uma abstração de uma *blockchain* privada que permite o isolamento e confidencialidade dos dados. Um *ledger* específico do canal é partilhado entre os *peers* do canal e as partes de uma transação têm de ser autenticadas num canal para interagirem com ele. Os canais são definidos por um bloco de configuração.

Certificate Authority

O *Hyperledger Fabric* **Certificate Authority (CA)** é o componente padrão da autoridade de certificação, que emite certificados baseados em **Public Key Infrastructure (PKI)** para organizações membros da rede e seus utilizadores. A CA emite um certificado raiz para cada membro e um certificado de inscrição para cada utilizador autorizado.

Endorsement

Refere-se ao processo em que *peer nodes* específicos executam uma transação e retornam uma resposta de proposta à aplicação cliente. A resposta da proposta inclui a mensagem de resposta da execução do *chaincode*, resultados e eventos, bem como uma assinatura para servir como prova da execução

do *chaincode* por parte do *peer*. As aplicações *chaincode* têm políticas de *endorsement* em que os *endorsement peers* são especificados.

3.4.2 Modelo

Nesta secção serão descritos os principais recursos incorporados no *Hyperledger Fabric* que fazem cumprir a sua promessa de uma solução de *blockchain* corporativa, abrangente e personalizável.

Assets - o *Fabric* fornece a capacidade de alterar ativos recorrendo a transações no *chaincode*. Estes são representados como uma coleção de pares chave-valor com alterações de estado registadas como transações numa *ledger* de canal.

Chaincode - é um *software* que define ativos e as instruções de transação para os modificar. Aplica as regras de leitura e alteração dos pares chave-valor ou outras informações da base de dados de estado. As funções do *chaincode* são executadas na base de dados de estado atual do *ledger* e são iniciadas através de uma proposta de transação. A execução do *chaincode* resulta num conjunto de pares chave-valor que pode ser submetido à rede e aplicado ao *ledger* em todos os *peers*.

Recursos do Ledger - O *ledger* é o registo sequenciado e inviolável de todas as transições de estado no *fabric*. As transições de estado são resultado de invocações ao *chaincode* efetuadas pelos participantes. Cada transação resulta num conjunto pares chave-valor de ativo que são submetidos no *ledger* como criações, atualizações ou remoções. O *ledger* é composto, por uma *blockchain* para armazenar o registo sequenciado e imutável em blocos, e por uma base de dados de estado para manter o estado atual do *fabric*. Existe um *ledger* por canal e cada *peer* mantém uma cópia do *ledger* para cada canal do qual é membro.

Security & Membership Services - o *Hyperledger Fabric* sustenta uma rede de transações onde todos os participantes possuem identidades conhecidas. A infraestrutura *Public Key* é utilizada para gerar certificados criptográficos associados a organizações, componentes da rede e a utilizadores finais ou aplicações cliente. Desta forma, o controlo de acesso a dados pode ser manipulado e governado na rede amplamente ou a nível dos canais. A privacidade do *Fabric* juntamente com a existência de canais e os seus recursos, ajuda a abordar cenários em que a privacidade e confidencialidade são preocupações primárias.

Consensus - O consenso é definido como a verificação completa da precisão de um conjunto de transações que compõem um bloco e é fundamental em todo o fluxo de transação. O consenso é alcançado quando o pedido e os resultados das transações de um bloco atendem às verificações explícitas nos critérios da política.

3.4.3 Fluxo de transação

Este fluxo assume que um canal está configurado e ativo, que um utilizador está registado e inscrito na CA da organização e recebeu de volta material criptográfico necessário, utilizado para autenticação na rede, e para além disso que o *chaincode* está instalado nos *peers* e instalado no canal.

O fluxo de uma transação funciona da seguinte forma:

1. O utilizador inicia a transação.
2. Os *endorsing peers* verificam a assinatura e executam a transação.
3. As respostas da proposta são inspecionadas.
4. Os *endorsements* são reunidos numa transação.
5. A transação é validada e submetida.
6. A *ledger* é atualizada.

Capítulo 4

Modelação do Sistema

Nesta capítulo serão apresentados os requisitos da aplicação desenvolvida e serão apresentados e especificados os casos de uso da mesma. Para além disso, será apresentada também a arquitetura aplicacional da mesma.

4.1 Levantamento de Requisitos

O processo de levantamento de requisitos tem um papel fundamental no sucesso de qualquer projeto, uma vez que envolve a identificação, compreensão e documentação dos requisitos que guiarão o desenvolvimento.

A aplicação desenvolvida pretende criar uma solução para a assinatura digital de documentos. O levantamento de requisitos desempenha um papel crítico para garantir que a aplicação se encontra de acordo com as expectativas da *dstelecom* e alinhada com os objetivos do projeto de dissertação.

O processo de levantamento de requisitos foi efetuado, maioritariamente, através de reuniões com o supervisor externo. Estas reuniões consistiam em discussões sobre as funcionalidades pretendidas para a plataforma, para que a mesma tivesse utilidade e fosse de encontro aos objetivos pretendidos para o tema da dissertação.

De seguida, serão apresentados os requisitos da aplicação.

4.1.1 Requisitos funcionais

Nesta secção serão especificados os requisitos funcionais. Inicialmente serão apresentados os requisitos funcionais comuns a todos os utilizadores e, de seguida, serão apresentados os requisitos funcionais específicos de cada tipo de utilizador na plataforma.

1. **Requisito Funcional:** Iniciar e Terminar sessão

Descrição: A plataforma deve fornecer a funcionalidade de iniciar e terminar sessão para todos os tipos de utilizadores, de forma a permitir o acesso à plataforma por meio de autenticação segura e encerramento da sua sessão quando desejarem.

Contexto: Esta funcionalidade é fundamental para assegurar a segurança e controlo de acesso dos utilizadores à plataforma. Ao fornecer esta funcionalidade, cada utilizador, registado por um administrador, será capaz de se autenticar na aplicação através das suas credenciais, fornecidas pelo administrador. Para além disso, a funcionalidade de terminar sessão permitirá que os utilizadores encerrem a sua sessão de forma segura podendo evitar acessos não autorizados à sua conta.

Critérios de Aceitação:

- A plataforma deve apresentar uma página de início de sessão onde os utilizadores possam inserir as suas credenciais.
- As credenciais fornecidas pelos utilizadores devem ser verificadas para autenticação na plataforma.
- Em caso de sucesso na autenticação, os utilizadores devem ser redirecionados para a página principal da aplicação, caso contrário será demonstrado o erro que causou o insucesso.
- A plataforma deve apresentar um botão claro e visível para os utilizadores poderem encerrar a sua sessão a qualquer momento.
- Depois de terminarem a sessão, os utilizadores não devem ter permissão para aceder à plataforma sem iniciarem sessão novamente.

2. **Requisito Funcional:** Consultar documentos certificados

Descrição: A plataforma deve permitir que os utilizadores possam consultar os documentos certificados pela mesma. Os documentos disponíveis para consulta variam de acordo com permissões e de acordo com o tipo de utilizador.

Contexto: A consulta de documentos certificados é fundamental para que os utilizadores possam visualizar e verificar a validade dos documentos certificados pela plataforma. Esta é uma das principais funcionalidades desta plataforma. Garantir que cada utilizador possa aceder apenas aos documentos apropriados é essencial para preservar a confidencialidade e segurança dos dados.

Critérios de Aceitação:

- Todos os utilizadores devem poder consultar documentos certificados na plataforma.
- Os administradores devem ter acesso a todos os documentos certificados na plataforma independentemente do seu certificador ou permissões de acesso.
- Os certificadores devem poder consultar todos os documentos gerados por si, como também os documentos para os quais eles têm permissões de acesso.
- Os utilizadores comuns apenas podem consultar os documentos para os quais têm permissões de acesso.
- A plataforma deve fornecer uma interface intuitiva de pesquisa para que os utilizadores possam localizar os documentos pretendidos.
- A plataforma deve garantir que a segurança nos acessos aos documentos se mantenha.

Requisitos Funcionais exclusivos do Administrador

1. Requisito Funcional: Registrar utilizador

Descrição: A plataforma deve permitir aos administradores registarem novos utilizadores. Esta funcionalidade é exclusiva para os administradores fazendo com que os mesmos tenham controlo total de novos acessos à aplicação.

Contexto: Esta funcionalidade é essencial para garantir que novos utilizadores são registados apenas por administradores da plataforma, desta forma garantindo que apenas utilizadores autorizados tenham acesso à mesma.

Critérios de Aceitação:

- A plataforma deve apresentar uma interface para os administradores registarem novos utilizadores.
- Ao registar um novo utilizador, os administradores devem fornecer as informações necessárias do novo utilizador, como nome de utilizador, email, senha e tipo de utilizador (utilizador normal ou certificador).
- As informações fornecidas deverão ser verificadas para garantir a autenticidade e segurança da plataforma.

2. Requisito Funcional: Remover utilizador

Descrição: A plataforma deve permitir aos administradores a remoção de utilizadores existentes na mesma. Esta funcionalidade é exclusiva para os administradores, de forma a garantir a segurança na plataforma e permitir que estes controlem a gestão de utilizadores na mesma.

Contexto: O poder de remover utilizadores da plataforma é essencial para assegurar que apenas utilizadores autorizados e ativos possam ter acesso à plataforma. A atribuição desta funcionalidade apenas aos administradores, permite que estes possam controlar a plataforma de forma eficaz através da remoção de utilizadores desnecessários, inativos ou que apresentem riscos para a segurança da plataforma.

Critérios de Aceitação:

- A plataforma deve fornecer uma opção específica e intuitiva para os administradores removerem utilizadores existentes.
- Os utilizadores devem ter capacidade de pesquisar e detetar o utilizador que desejam remover com base em informações como o nome ou email do mesmo.
- Após a confirmação, a aplicação deve remover o utilizador, rejeitando o seu acesso à mesma.
- Os registos de todos os utilizadores não devem ser apagados permanentemente, para garantir a possibilidade de verificação de quem emitiu cada documento gerado pela plataforma.

3. **Requisito Funcional:** Atribuir e Retirar função de certificador

Descrição: A plataforma deve possibilitar que os administradores atribuam e retirem a função de certificador a um utilizador existente. Esta funcionalidade é exclusiva para os administradores, dando-lhes controlo sobre quais utilizadores podem ou não certificar documentos na plataforma.

Contexto: O controlo das permissões de certificação de documentos é uma parte crucial da plataforma. Desta forma, a possibilidade de atribuir e retirar estas permissões é fundamental para o funcionamento adequado da plataforma. Ao fornecer esta funcionalidade apenas aos administradores, a plataforma assegura que somente utilizadores autorizados possam exercer funções de certificação de documentos, garantindo assim a autenticidade do sistema e segurança dos dados.

Critérios de Aceitação:

- A plataforma deve fornecer opções específicas e intuitivas para os administradores atribuírem e retirarem a função de certificador a um utilizador.
- Aquando da atribuição da função de certificador, a plataforma deve conceder as permissões necessárias ao utilizador para desempenhar o papel de certificador.

- Ao retirar a função de certificador de um utilizador, a plataforma deve retirar as permissões associadas à certificação de documentos.
- As alterações efetuadas relacionadas com funções de certificador deverão ser imediatamente refletidas nos acessos dos utilizadores na plataforma.

4. **Requisito Funcional:** Revogar documento certificado

Descrição: A plataforma deve permitir que apenas os administradores possam revogar documentos gerados por certificadores.

Contexto: Esta funcionalidade é essencial para assegurar a integridade e a confiabilidade dos documentos certificados na plataforma. Ao conceder esta função apenas aos administradores a plataforma é capaz de garantir que apenas decisões autorizadas são tomadas neste sentido, assegurando o controlo sobre a revogação de documentos.

Critérios de Aceitação:

- A plataforma deve possuir uma interface intuitiva para a revogação dos certificados.
- A plataforma deve possibilitar a identificação e seleção de documentos que os administradores pretendam revogar.
- A revogação do documento deverá ser registada no sistema.
- Depois de ser revogado, um documento deverá ser considerado inválido e inacessível para os utilizadores da aplicação.

5. **Requisito Funcional:** Listar Utilizadores

Descrição: A plataforma deve fornecer a funcionalidade de listar todos os utilizadores registados na aplicação, de forma a permitir aos administradores visualizar todos os utilizadores registados.

Contexto: A listagem de utilizadores é uma funcionalidade essencial para garantir a gestão eficiente e o controlo de acessos à plataforma. Ao disponibilizar essa funcionalidade, os administradores poderão visualizar e pesquisar os utilizadores.

Critérios de Aceitação: A plataforma deve permitir aos administradores aceder à lista de utilizadores através de uma interface intuitiva. Os utilizadores devem ser apresentados numa tabela ou lista, exibindo informações relevantes, como nome e email. Deve ser possível pesquisar e filtrar os utilizadores com base em critérios específicos, como nome ou email. A listagem de utilizadores deve ser atualizada em tempo real, refletindo qualquer alteração ou novo registo realizado na aplicação.

6. **Requisito Funcional:** Listar Certificadores

Descrição: A plataforma deve fornecer a funcionalidade de listar todos os certificadores registrados na aplicação, de forma a permitir aos administradores visualizar todos os certificadores registrados.

Contexto: A listagem de certificadores é uma funcionalidade essencial para garantir a gestão eficiente, o controlo de acessos e o controlo de registos de novos documentos na plataforma. Ao disponibilizar essa funcionalidade, os administradores poderão visualizar e pesquisar os certificadores.

Critérios de Aceitação: A plataforma deve permitir aos administradores aceder á lista de certificadores através de uma interface intuitiva. Os certificadores devem ser apresentados numa tabela ou lista, exibindo informações relevantes, como nome e email. Deve ser possível pesquisar e filtrar os certificadores com base em critérios específicos, como nome ou email. A listagem de certificadores deve ser atualizada em tempo real, refletindo qualquer alteração ou novo registo realizado na aplicação.

7. **Requisito Funcional:** Listar Solicitações para função de certificador

Descrição: A plataforma deve possuir a funcionalidade de listar todas as solicitações de permissão de certificação efetuadas por utilizadores, de forma a permitir aos administradores visualizarem todas as solicitações efetuadas.

Contexto: A listagem de solicitações é uma funcionalidade essencial para permitir aos administradores visualizar novas solicitações e tomar decisões acerca das mesmas. Ao disponibilizar esta funcionalidade, os administradores poderão pesquisar e visualizar novas solicitações.

Critérios de Aceitação: A plataforma deve permitir aos administradores aceder á lista de solicitações através de uma interface intuitiva. As solicitações devem ser apresentados numa tabela ou lista, exibindo informações relevantes sobre os utilizadores que as fizeram.

Requisitos Funcionais exclusivos do Utilizador comum

1. **Requisito Funcional:** Solicitar função de certificador

Descrição: A plataforma deve permitir que os utilizadores comuns possam solicitar a permissão para certificarem documentos. Esta funcionalidade oferece aos utilizadores a possibilidade de demonstrarem o seu interesse em certificar documentos na plataforma.

Contexto: Permitir que utilizadores comuns peçam permissão para certificar documentos na

plataforma, possibilita a cada utilizador tornar-se certificador. A aplicação ao fornecer esta funcionalidade permite que a comunidade de certificadores possa aumentar ao longo do tempo conforme os administradores a gerirem.

CrITÉrios de Aceitação:

- A plataforma deve possuir uma opção intuitiva para que os utilizadores efetuem os pedidos para se tornarem certificadores.
- As solicitações efetuadas pelos utilizadores devem ser guardadas no sistema, incluindo informações relevantes, como o nome ou email do solicitador.
- Os administradores devem receber de forma imediata as novas solicitações que são efetuadas.
- Caso uma solicitação seja aceite, o utilizador deverá receber a permissão de certificador e poderá começar a certificar documentos na plataforma.

Requisitos Funcionais exclusivos do Certificador

1. Requisito Funcional: Certificar documentos

Descrição: A plataforma deverá permitir que os certificadores certifiquem documentos. Esta funcionalidade permite aos certificadores fazerem upload de documentos na plataforma e certificarem os mesmos através da criação de um NFT na blockchain construída.

Contexto: A funcionalidade de certificação de documentos é fundamental para fazer cumprir o propósito da dissertação, que é a geração de NFTs para documentos digitais. A certificação de documentos é efetuada através a criação de um token não fungível para cada documento selecionado. Ao atribuir esta responsabilidade apenas aos certificadores, a plataforma assegura que os documentos certificados são apenas gerados por utilizadores autorizados, garantindo maior controlo sobre a certificação de documentos.

CrITÉrios de Aceitação:

- A plataforma deve possuir uma interface apelativa para a certificação de documentos.
- Os certificadores deverão poder escolher os documentos que pretendem certificar.
- A certificação do documento deve ser registada no sistema, com a criação de um NFT para o mesmo.

- Os documentos certificados devem receber uma marca que os identifique como certificados pela plataforma.

4.1.2 Requisitos não funcionais

Nesta secção serão especificados os requisitos não funcionais da plataforma em desenvolvimento.

1. O sistema deverá permitir uma autenticação segura a cada utilizador.
2. O sistema deverá impedir o acesso a utilizadores não autorizados.
3. A plataforma deverá operar em ambiente *Web*.
4. A plataforma deverá ser responsiva para se ajustar a vários dispositivos.
5. O sistema deverá gerir as permissões dos diferentes tipos de utilizador.
6. O sistema deverá dar acessos conforme as permissões do utilizador.
7. O sistema deverá apresentar apenas o conteúdo a que o utilizador tem acesso.

4.2 Diagrama de casos de uso

Com o objetivo de resumir e fornecer uma forma de visualização dos requisitos mencionados na secção anterior, foi desenvolvido um diagrama de casos de uso. Neste diagrama encontram-se explicitadas as funcionalidades dos três tipos de utilizadores da aplicação.

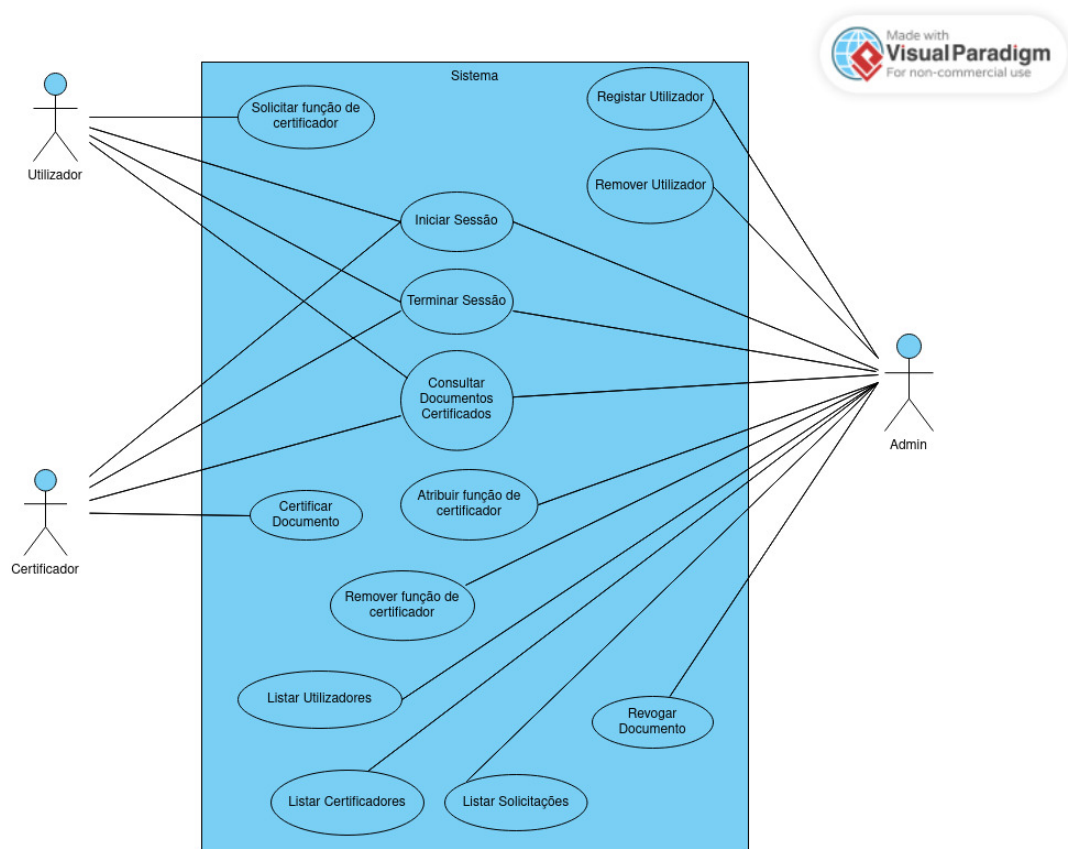


Figura 4: Diagrama de casos de uso da plataforma

Através da análise do diagrama apresentado na Figura 4 pode-se concluir que utilizadores com as permissões de administrador estarão responsáveis pela gestão da plataforma, dos seus utilizadores e certificados gerados. Os certificadores serão responsáveis pela criação de novos documentos certificados. Os utilizadores comuns poderão apenas consultar documentos, podendo vir a certificar documentos caso lhes seja fornecida a permissão.

4.3 Arquitetura Aplicacional

Na Figura 5 encontra-se o esquema das tecnologias utilizadas para o desenvolvimento aplicação.

A aplicação divide-se em quatro partes principais, o cliente (browser), a interface, a API e as plataformas de armazenamento de dados.

A interface foi desenvolvida utilizando *React Js* para execução de pedidos à API e estruturação das páginas da interface do utilizador. Sendo esta camada desenvolvida com o auxílio da *framework bootstrap* para o desenvolvimento das páginas da interface do utilizador.

A API foi desenvolvida com recurso a *Node Js + Express*, para definição das rotas necessárias com as ações nas plataformas de armazenamento de dados. Nesta camada foi também utilizada a API da plataforma *Pinata*, para guardar e providenciar o conteúdo digital dos NFTs, uma vez que, permite o armazenamento do conteúdo necessário em *IPFS*, 5.2.6.

Para além disso, foi utilizada uma rede blockchain *Hyperledger Fabric*, criada e gerida pela plataforma *Kaleido*, para a criação e consulta dos *NFTs* gerados pela plataforma, bem como para o registo de novos utilizadores na respetiva rede.

Para o armazenamento de informações respetivas aos utilizadores do sistema, bem como alguns dados relacionados com os *NFTs* gerados pela plataforma, foi utilizada uma base de dados em *mongoDB*.

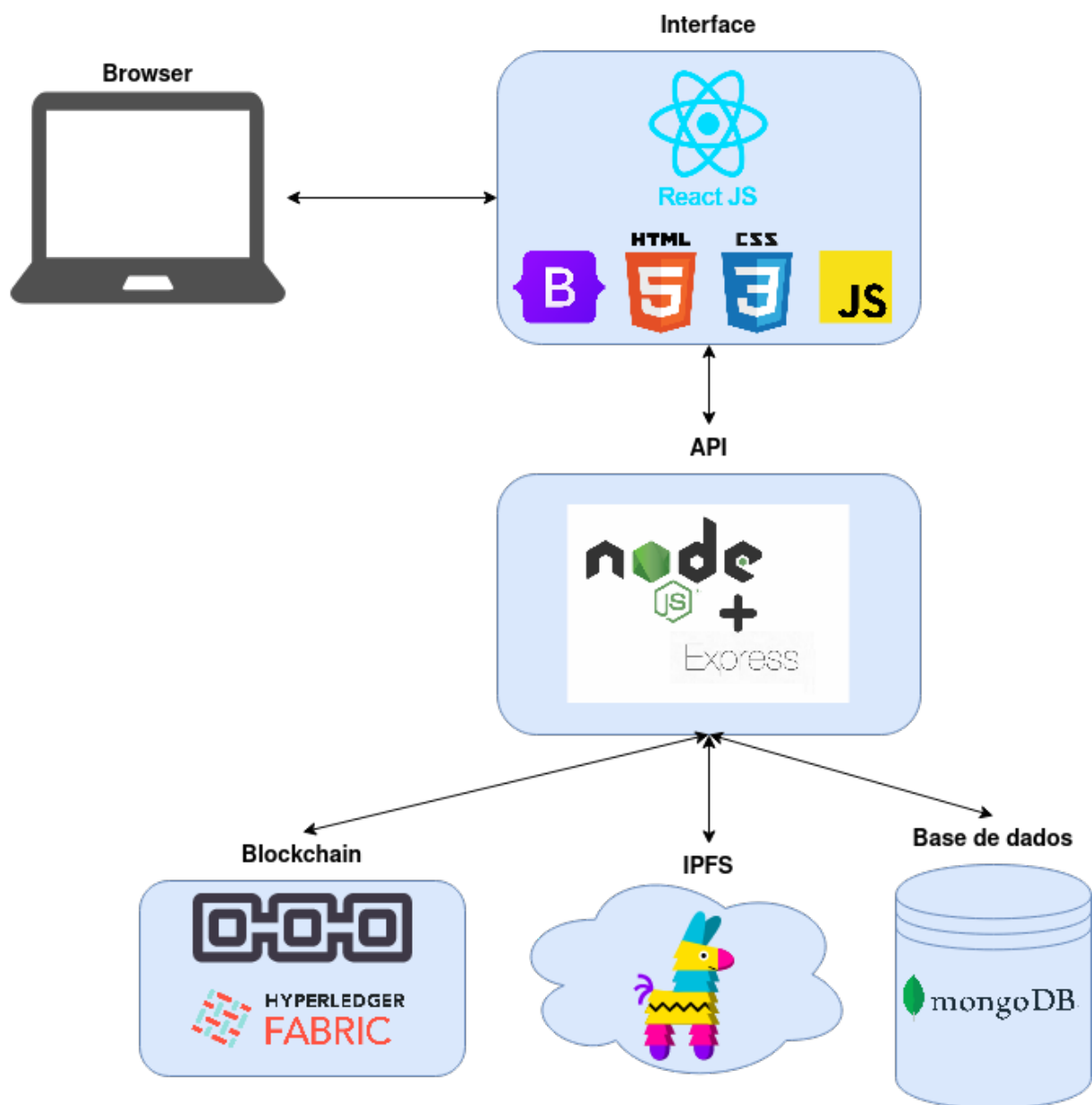


Figura 5: Esquema das tecnologias da plataforma

Capítulo 5

Desenvolvimento Aplicacional

O capítulo de desenvolvimento da API descreverá todo o processo de implementação do servidor de dados necessário para fazer cumprir todos os requisitos mencionados no capítulo anterior.

Neste capítulo serão explicitadas todas as fases do desenvolvimento da API, desde a implementação da base de dados, da rede *blockchain* e a construção das rotas implementadas. Para além disso será explicado o processo de autenticação e alguns mecanismos de segurança utilizados no desenvolvimento da API.

5.1 Base de Dados

Depois da análise e estudo dos requisitos funcionais, surgiu a necessidade de criar uma nova base de dados para permitir a consulta, atualização e registo de dados necessários na plataforma.

Esta base de dados foi criada com o objetivo de guardar os dados relativos a utilizadores da plataforma e também para armazenar alguns dados relacionados com a geração de NFTs no sistema.

5.1.1 MongoDB

O *MongoDB* é uma base de dados documental open-source. Ao contrário de base de dados relacionais (**Structured Query Language (SQL)**), que armazenam dados em tabelas relacionadas entre si, cada registo numa base de dados MongoDB é um documento descrito em **Binary JSON (BSON)**, ou seja uma representação binária dos dados, sendo que estes dados são obtidos em formato *JSON* pelas aplicações. As bases de dados documentais são extremamente flexíveis permitindo alterações na estrutura de documentos e armazenamento de documentos parcialmente completos, para além disso, o MongoDB, facilita aos desenvolvedores o armazenamento de dados estruturados ou não estruturados e é capaz de lidar com grande volume de dados ¹.

¹ <https://www.mongodb.com/why-use-mongodb>

Deste modo, conclui-se que as suas características justificam a sua utilização na aplicação, e uma vez que não se verifica a necessidade de utilização de uma base de dados relacional decidiu-se utilizar uma base de dados *MongoDB*.

5.1.2 Coleções

Uma base de dados MongoDB pode ser constituída por várias coleções, que são agrupamentos de documentos e são o equivalente a tabelas em bases de dados relacionais ².

Utilizadores

De acordo com os requisitos necessários para a plataforma, verificou-se a necessidade da criação de uma coleção de dados com as informações necessárias de cada utilizador.

```
{
  "_id": {},
  "username": "",
  "email": "",
  "password": "",
  "roles": [],
  "disabled": bool,
  "tokens": []
}
```

Listagem 1: Estrutura da coleção 'Utilizador'.

Através da estrutura apresentada em ¹, é possível observar que a coleção de utilizadores guarda as seguintes informações:

- *id* - Identificador único de cada utilizador.
- *username* - Nome do utilizador na plataforma.
- *email* - Email do utilizador registado na plataforma.
- *password* - Password do utilizador.

² <https://www.mongodb.com/why-use-mongodb>

- *roles* - Representa a role do utilizador na plataforma.
- *disabled* - Informação que indica se o utilizador está ou não ativo.
- *tokens* - Lista de *NFTs* que o utilizador tem permissão de visualização.

Por questões de segurança a palavra passe não é guardada diretamente na base de dados mas sim uma encriptação da mesma, usando a função da biblioteca *bcryptjs* do Javascript:

```
bcrypt.hashSync(pass, n°);
```

Listagem 2: Encriptação da password

Para verificação da palavra passe, no início de sessão de um utilizador, a senha inserida no processo de *login* é encriptada e comparada com a encriptação guardada na base de dados da seguinte forma:

```
bcrypt.compareSync(senhaInserida, senhaBD);
```

Listagem 3: Verificação da password

Solicitações

Para armazenar as informações relativas aos pedidos de permissão de certificação de documentos na plataforma, verificou-se a necessidade de criação de uma coleção de dados.

```
{
  "_id": {},
  "username": "",
  "evaluated": bool
}
```

Listagem 4: Estrutura da coleção 'Solicitacao'.

De acordo com a estrutura apresentada, verifica-se que a coleção de solicitações armazena as seguintes informações:

- *id* - Identificador único da solicitação.
- *username* - Username do utilizador que efetuou a solicitação.
- *evaluated* - Booleano que indica se a solicitação já foi ou não avaliada.

Roles

Como referenciado ao longo deste documento, a plataforma possui três categorias de utilizadores, ou seja, em consequência disto foi criada uma coleção que guarda as informações de cada categorias da plataforma. Ou seja, esta coleção terá apenas três documentos, um para cada categoria.

Esta ação é necessária pois são os identificadores das categorias que são associados aos utilizadores e não o nome das categorias em si.

```
{
  "_id": {},
  "name": ""
}
```

Listagem 5: Estrutura da coleção 'Role'.

Através da estrutura apresentada, observa-se que a coleção de *roles* armazena as seguintes informações:

- *id* - Identificador único da role.
- *name* - Denominação da *role* na plataforma.

NFTs

Cada vez que é gerado um novo *token* aquando da certificação de um documento pela ação de um certificador, existem informações que serão guardadas na base de dados. Desta forma, verificou-se a necessidade de criação de uma coleção para guardar estas informações.

```
{
  "_id": {},
  "name": "",
  "valido": bool
}
```

Listagem 6: Estrutura da coleção 'NFT'.

Segundo a estrutura apresentada, pode-se verificar que esta coleção guarda as seguintes informações acerca de um *NFT*:

- *id* - Identificador único do NFT, coincidente com o *Token Id* na blockchain.
- *name* - Nome do documento certificado.
- *valido* - Booleano que indica se o documento é válido ou não.

Conexão à Base de Dados

Para efetuar a conexão à base de dados são utilizadas as credenciais presentes no ficheiro de configuração. O excerto de código 7 demonstra como é feita a conexão.

```
connection_URL =
  `mongodb://${dbConfig.HOST}:${dbConfig.PORT}/${dbConfig.DB}`

db.mongoose
  .connect(connection_URL, {
    useNewUrlParser: true,
    useUnifiedTopology: true
  })
  .then(() => {
    console.log("Successfully connect to MongoDB.");
    initial();
  })
  .catch(err => {
    console.error("Connection error", err);
    process.exit();
  });
```

Listagem 7: Conexão à Base de Dados

Povoamento

Ao iniciar a aplicação, existem dados que devem ser imediatamente adicionados à base de dados. Desta forma, quando se inicia a aplicação a coleção das categorias é povoada com as três entradas correspondentes a cada uma das categorias. Para além disso, o utilizador correspondente ao administrador da aplicação é automaticamente criado. Para isto, é inserida uma entrada na base de dados com os dados referentes a este utilizador. O mesmo é também registado na *blockchain* e, para além disto, é efetuado o registo de uma identidade para o mesmo na *wallet* de identidades do sistema, abordado em 5.2.3.

Depois destes dados serem inseridos na base de dados, a aplicação já se encontra pronta para funcionar como o esperado.

5.2 Blockchain

Nesta secção são apresentadas as decisões tomadas para a criação da rede *blockchain* utilizada pela plataforma em desenvolvimento, bem como, a arquitetura da rede.

5.2.1 Criação da rede

Para a criação da rede *blockchain Hyperledger Fabric*, optou-se por utilizar uma plataforma que oferece a rede *blockchain* como um serviço. Numa primeira fase foram exploradas as plataformas da *Amazon Managed Blockchain*³ e a *IBM Blockchain Platform*⁴. No entanto, devido aos custos associados optou-se por descartar estas opções. Numa segunda pesquisa foi encontrada a plataforma do *Kaleido*⁵, que é também um fornecedor de serviços *blockchain* que oferece um período de experimentação gratuito. O *Kaleido* possui integração com *Hyperledger Fabric*, entre outras tecnologias *blockchain*, e permite a criação da rede de forma fácil e intuitiva. Esta plataforma permite lançar redes *blockchain*, personalizadas e escaláveis de forma rápida e eficaz retirando muito do esforço e complexidade necessários para criar uma rede do zero.

O *Kaleido* simplifica imenso a experiência geral com o *Fabric*, uma vez que abstrai as complexidades subjacentes associadas a autoridades de certificação, gestão da rede, ciclos de vida do *chaincode* e gestão de canais.

Apesar dos serviços fornecidos por esta plataforma serem pagos, a versão gratuita fornece os recursos necessários para o desenvolvimento de uma rede *blockchain* protótipo a ser utilizada pela aplicação em desenvolvimento.

Desta forma, foi utilizado o plano de demonstração gratuita, que apenas permite a criação de uma rede com um máximo de uma organização, dois canais e dois nodos (1 *orderer* e 1 *peer*). Estes recursos são suficientes para o desenvolvimento da aplicação protótipo, uma vez que a aplicação apenas necessita de interagir com um canal e as transações podem ser efetuadas todas pelo mesmo *peer* e ordenadas pelo *orderer*.

Para a criação da rede e instalação do *chaincode* no respetivo canal, foi seguida a documentação oficial da plataforma⁶.

³ <https://aws.amazon.com/pt/managed-blockchain/>

⁴ <https://www.ibm.com/products/blockchain-platform-hyperledger-fabric>

⁵ <https://www.kaleido.io/>

⁶ <https://docs.kaleido.io/using-kaleido/quick-start-fabric/first-blockchain/>

5.2.2 Conexão ao canal da rede

Uma vez que a rede se encontra criada e pronta a ser utilizada, foi necessário preparar a aplicação em desenvolvimento para que a mesma se conectasse à rede criada através do *Kaleido*. Para isso foram necessárias as seguintes linhas de código.

```
const KaleidoClient = require('./kaleido');
const { Gateway } = require('fabric-network');
const Client = require("fabric-common");

const chaincodeName = 'nft_app_v1';
```

Listagem 8: Conexão à Rede Kaleido - rede.js

```
const API_KEY = 'u0lraamuo2-9uo7hqv4P7z7sLPo/GDgZas1eXH5DyPB/ItrQWAWtuQ=';
const API_URL = "https://console-eu.kaleido.io/api/v1";

const membershipID_nodes = 'e0hv36d6pn';
const env_id = 'e0t60tftjp';
```

Listagem 9: Conexão à Rede Kaleido - kaleido.js

- A variável *chaincodeName* contém o nome do *chaincode* presente na rede.
- A *API_KEY* é necessária para a aplicação ter acesso aos recursos criados no perfil do *Kaleido*.
- A *API_URL* contém o *url* base para as chamadas à API do *Kaleido*.
- O *env_id* guarda o identificador do ambiente, do perfil do *Kaleido*, onde se encontra a rede. O *membershipId_nodes* guarda o identificador da organização que possui os nodos na rede.

Ambos necessários para a criação e conexão de identidades de utilizadores da rede.

A adaptação do código foi baseada num repositório do *github* da própria plataforma ⁷, que demonstra a conexão de uma aplicação à rede do *Kaleido* e executa transações nessa mesma rede. Do projeto exemplo presente no repositório mencionado foram utilizados e adaptados os ficheiros presentes na diretoria `"/nodejs/lib"`.

⁷ <https://github.com/kaleido-io/kaleido-fabric-samples/tree/master/node.js>

5.2.3 Wallet

A *wallet* é composta pelo conjunto de identidades de utilizador do sistema. Os direitos de acesso a recursos do canal são determinados através da combinação da identidade do utilizador com um *MSP*. Os *MSPs* (Membership Service Providers) presentes num canal têm o papel de determinar os direitos das identidades criadas sobre os recursos do canal.

Cada utilizador pode possuir diferentes identidades para se conectar a diferentes canais de uma rede, podendo o mesmo ter diferentes permissões em cada canal. No entanto, neste caso de estudo é apenas utilizado um canal na rede e cada utilizador possui apenas uma entidade para se conectar ao canal ⁸.

Quando a aplicação é executada por um utilizador, a identidade correspondente ao mesmo é selecionada para este se conectar ao respetivo canal da rede *blockchain*.

Para além disso, quando um novo utilizador é registado na plataforma uma nova identidade é adicionada à carteira do sistema e posteriormente utilizada para as ações desse utilizador no canal da rede.

Existem vários tipos de carteira dependendo do local onde as mesmas armazenam as suas identidades. Os três diferentes tipos de armazenamento para as carteiras são os seguintes:

- **File System** - este é o local utilizado mais frequentemente usado para armazenar carteiras. Os sistemas de arquivos são uma boa escolha padrão porque são fáceis de entender e podem ser montados em rede.
- **In-memory** - neste caso a carteira é armazenada na memória local da aplicação. Este tipo de carteiras é geralmente utilizado quando uma aplicação está a ser executada em ambiente restrito sem acesso a sistema de arquivos. Este tipo de carteira é volátil uma vez que as identidades serão perdidas depois da aplicação terminar normalmente ou parar subitamente.
- **CouchDB** - neste caso a carteira é armazenada numa base de dados *CouchDB*, sendo esta a forma mais rara de armazenamento de carteiras, no entanto, o armazenamento em base de dados permite a utilização de mecanismos de *back-up* e recuperação de dados.

Uma vez que grande parte dos projetos desenvolvidos em *Hyperledger Fabric* utilizam o armazenamento da carteira em **sistema de ficheiros**, promovendo assim maior ajuda por parte dos ambientes de pesquisa, este foi o tipo de armazenamento escolhido para a carteira da aplicação em desenvolvimento.

⁸ <https://hyperledger-fabric.readthedocs.io/en/release-2.2/developapps/wallet.html#scenario>

Uma carteira pode armazenar várias identidades com certificados emitidos por uma *Certificate Authority*, que é uma entidade confiável que dentro de várias outras funções têm a função de registrar novas identidades.

Cada identidade tem uma estrutura padrão, apresentada em 10, que é composta por um certificado *X.509* contendo uma chave pública, uma chave privada e alguns metadados do *Fabric*, como o identificador do *MSP*.

```
const identity: X509Identity = {
  credentials: {
    certificate: certificatePEM,
    privateKey: privateKeyPEM,
  },
  mspId: 'Org1MSP',
  type: 'X.509',
};
```

Listagem 10: Estrutura de uma identidade da carteira.

Qualquer um dos tipos de carteira mencionados implementa uma interface *Wallet* comum que disponibiliza um conjunto de *APIs* para gerir as identidades. Desta forma as aplicações podem ser desenvolvidas de forma independente do tipo de armazenamento da carteira subjacente.

As carteiras podem ser criadas e abertas e novas identidades podem ser adicionadas, removidas ou consultadas ⁹.

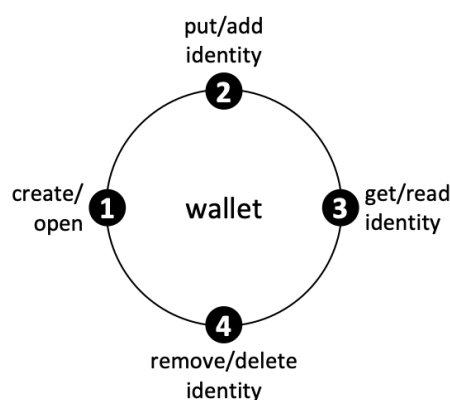


Figura 6: Ciclo de vida de uma carteira.

⁹ <https://hyperledger-fabric.readthedocs.io/en/release-2.2/developapps/wallet.html#scenario>

No caso desta dissertação, ao iniciar a aplicação uma nova carteira é criada. Novas identidades são adicionadas quando é efetuado o registo de algum utilizador na plataforma. As identidades são utilizadas quando o utilizador autenticado pretende efetuar alguma ação no canal da rede.

5.2.4 Smart Contract

De acordo com os token standards apresentados em 2.3.3 conclui-se que o mais adequado para a criação de *Non-Fungible Tokens* é o *ERC-721*.

O *Smart Contract* utilizado na implementação da plataforma em desenvolvimento foi o *ERC-721 token smart contract* providenciado pelo repositório *fabric-samples* do *Hyperledger Fabric* ¹⁰, que é um repositório aberto para utilização por desenvolvedores e utilizadores desta tecnologia.

No *Hyperledger Fabric*, os contratos têm a denominação de *chaincode* e a versão utilizada neste projeto foi a versão em *javascript* presente no repositório mencionado.

No pequeno excerto em 11 é apresentada uma parte do contrato utilizado.

```
const { Contract } = require('fabric-contract-api');
...
class TokenERC721Contract extends Contract {
  ...
  async TokenURI(ctx, tokenId) {
    // Check contract options are already set first to execute the
    // function
    await this.CheckInitialized(ctx);
    const nft = await this._readNFT(ctx, tokenId);
    return nft.tokenURI;
  }

  async MintWithTokenURI(ctx, tokenId, tokenURI) { ... }
  ...
  async Burn(ctx, tokenId) { ... }
  ...
}
module.exports = TokenERC721Contract;
```

Listagem 11: Chaincode - ERC721

- O método *MintWithTokenURI* permite a criação de novos *NFTs*.

¹⁰ <https://github.com/hyperledger/fabric-samples/tree/main/token-erc-721>

- O método *Burn* permite a revogação de *tokens* que foram criados na *blockchain*.
- O método *TokenURI* permite a consulta de *NFTs*.

5.2.5 Interações com a rede blockchain

Conexão à rede

```
async function connectToNetwork(userId){
    const kclient = new KaleidoClient(userId);
    await kclient.init();
    const gateway = new Gateway();
    const identity = await kclient.wallet.getUser(userId);

    await gateway.connect(kclient.config, {
        wallet: kclient.wallet.wallet,
        identity: userId,
        clientTlsIdentity: userId,
        discovery: { enabled: useDiscovery === 'true', asLocalhost:
            false }
    });
    const network = await gateway.getNetwork(kclient.channel.name);
    const contract = network.getContract(chaincodeName);
    let networkConfig = {
        contract: contract,
        network: network,
        gateway: gateway
    };
    return networkConfig;
}
```

Listagem 12: Conexão à Rede Kaleido - kaleido.js

A conexão à rede do *Kaleido* é feita como apresentada na Listagem 12.

1. Inicialmente, cria-se um cliente do *Kaleido* para o utilizador que tem a sua sessão iniciada. De seguida cria-se uma identidade na *wallet* para o utilizador, caso seja a primeira conexão à rede desse utilizador. Caso contrário obtém-se a identidade desse utilizador presente na *wallet*.
2. De seguida é feita a conexão à rede através do *gateway* do *fabric*. O *fabric gateway* é um serviço,

introduzido nos *peers* do Hyperledger Fabric v2.4, que fornece uma API simples para o envio de transações para uma rede Fabric ¹¹.

3. Depois da conexão estar estabelecida é possível obter o canal da rede e o contrato que serão utilizados pela plataforma.
4. Nesta fase, o utilizador já pode efetuar transações na rede através da invocação de métodos do contrato.

Tipos de interações com rede

As interações com a rede *blockchain* necessárias para se fazerem cumprir os requisitos da aplicação são as seguintes:

- **Criação de NFTs**

```
//CONEXÃO À BLOCKCHAIN
let networkObj = await network.connectToNetwork(username);

//INVOCAÇÃO À BLOCKCHAIN PARA GERAR O NFT
let resposta= await network.invoke(networkObj,false,
    'MintWithTokenURI',[nft_ID, ipfsHash]);
```

Listagem 13: Transação - Criação de NFT

- **Revogação de NFTs**

```
//CONEXÃO À BLOCKCHAIN
let networkObj = await network.connectToNetwork(username);

//INVOCAÇÃO À BLOCKCHAIN PARA REVOGAR O NFT
let resposta = await network.invoke(networkObj,false,'Burn',token_Id);
```

Listagem 14: Transação - Revogação de NFT

- **Consulta dos NFTs**

```
//CONEXÃO À BLOCKCHAIN
let networkObj = await network.connectToNetwork(username);

//INVOCAÇÃO À BLOCKCHAIN PARA BUSCAR O URI do ID CORRESPONDENTE
```

¹¹ <https://hyperledger-fabric.readthedocs.io/en/latest/gateway.html>

```
let uri= await network.invoke(networkObj,false,'TokenURI',[tuple.id]);
```

Listagem 15: Transação - Leitura do NFT

Como é possível observar através dos excertos de código apresentados nas Listagens 13, 14 e 15, para cada interação com a rede é necessário, primeiramente, estabelecer uma conexão com a rede e o respectivo canal e de seguida efetuar a invocação do respetivo método presente no contrato colocado no canal da rede *blockchain*.

O *Hyperledger Explorer* é uma ferramenta de código aberto que fornece uma interface *web* que permite a visualização, consulta e análise de dados de uma rede *blockchain* alimentada pelo *Hyperledger Fabric*. Esta ferramenta é hospedada pelo projeto *Hyperledger* e foi desenvolvida pela *Linux Foundation*.

Com recurso ao *Hyperledger Explorer* é possível visualizar e analisar o estado da rede *blockchain* utilizada.

Analisando as imagens retiradas do *Explorer* e apresentadas na Figura 7, pode-se verificar o estado da *blockchain* antes e depois de uma transação ser efetuada. Observando-se então que após uma transação na rede *blockchain*, foi adicionado um novo bloco à mesma.



Figura 7: Estado da Blockchain

5.2.6 Estrutura dos NFTs

O principal objetivo da aplicação é a criação de *NFTs* para certificados de documentos e, como mencionado no capítulo 2, apesar de algumas informações serem publicadas na *blockchain* aquando da geração

de um novo *token*, existem outros dados que não serão inseridos na rede mas que correspondem ao *NFT* gerado.

Desta forma aquando da certificação de um documento na plataforma e respetiva criação de um novo *NFT*, as informações que ficarão guardadas na rede serão apenas as seguintes:

- **Token ID** - Identificador único do token.
- **Token URI** - URI do conteúdo digital do documento. De notar que o URI vincula o conteúdo do documento (i.e. inclui o hash criptográfico do seu conteúdo).
- **Owner** - Identificador do proprietário do NFT que será o utilizador que o gerou.

O conteúdo que será armazenado fora do canal da rede será o seguinte:

- **Content Data** - O conteúdo digital do documento, ou seja o **PDF** do documento agora certificado, será afixado num servidor *IPFS* da plataforma *Pinata*, que disponibilizará este conteúdo através de um URI.
- **BD** - Para além disso, como mencionado previamente neste capítulo, em 5.1.2, as informações relativas aos *NFTs* que serão guardadas na base de dados *mongodb* são o identificador do *NFT*, o nome do documento e uma propriedade que indica a validade do documento.

Na Figura 8 é apresentado o esquema da estrutura dos *NFTs* criados na plataforma.

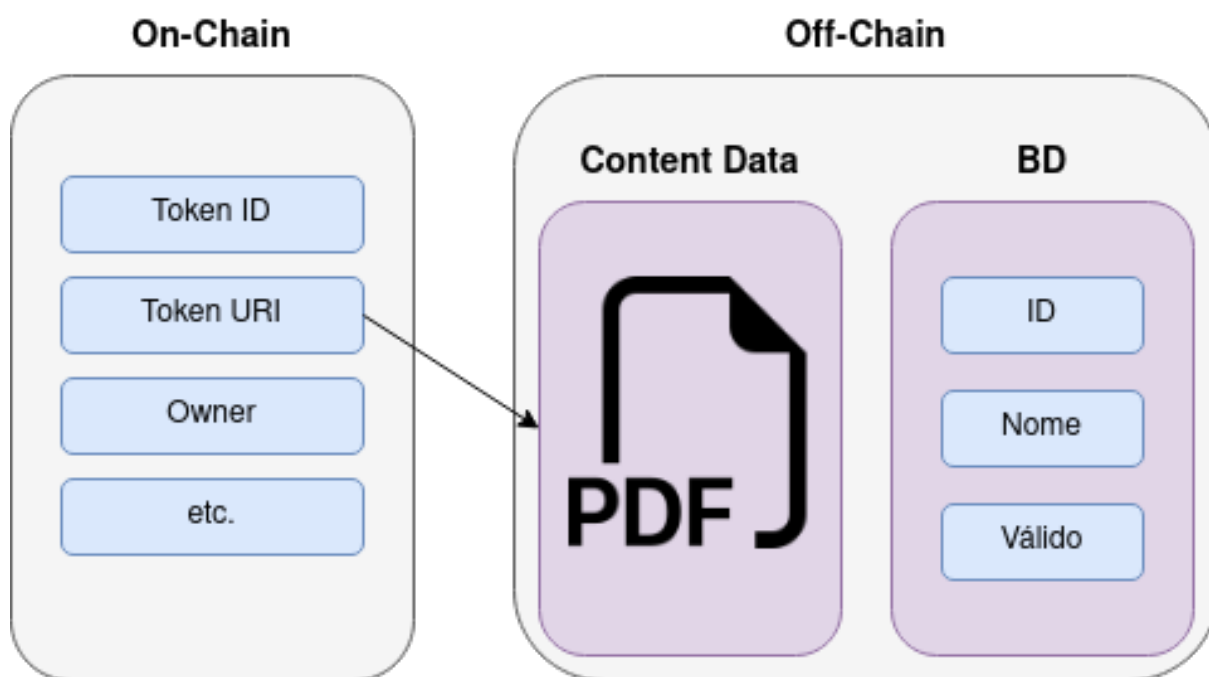


Figura 8: Estrutura dos NFTs da plataforma

IPFS

O *Interplanetary File System*, é um protocolo de partilha de ficheiros *peer-to-peer* e tem como objetivo melhorar a eficiência da *Web* e torná-la mais descentralizada e resiliente. O *IPFS* utiliza endereçamento baseado em conteúdo, onde o conteúdo não é endereçado por meio de um local, mas sim pelo seu conteúdo. A forma como o *IPFS* guarda e endereça dados de forma descentralizada visa permitir o armazenamento eficiente de dados. Por meio de *IPFS* é possível a partilha de ficheiros de forma descentralizada aumentando a resistência à censura do seu conteúdo. Esta tecnologia é utilizada como um serviço de armazenamento que complementa *blockchains*, permitindo diferentes aplicações sobre o *IPFS*. Devido às suas características de endereçamento baseado no conteúdo, este concentra-se principalmente em conteúdo imutável. A imagem apresentada 9 ilustra o comportamento de uma solução centralizada, **Hypertext Transfer Protocol**, e de um **IPFS** ¹².

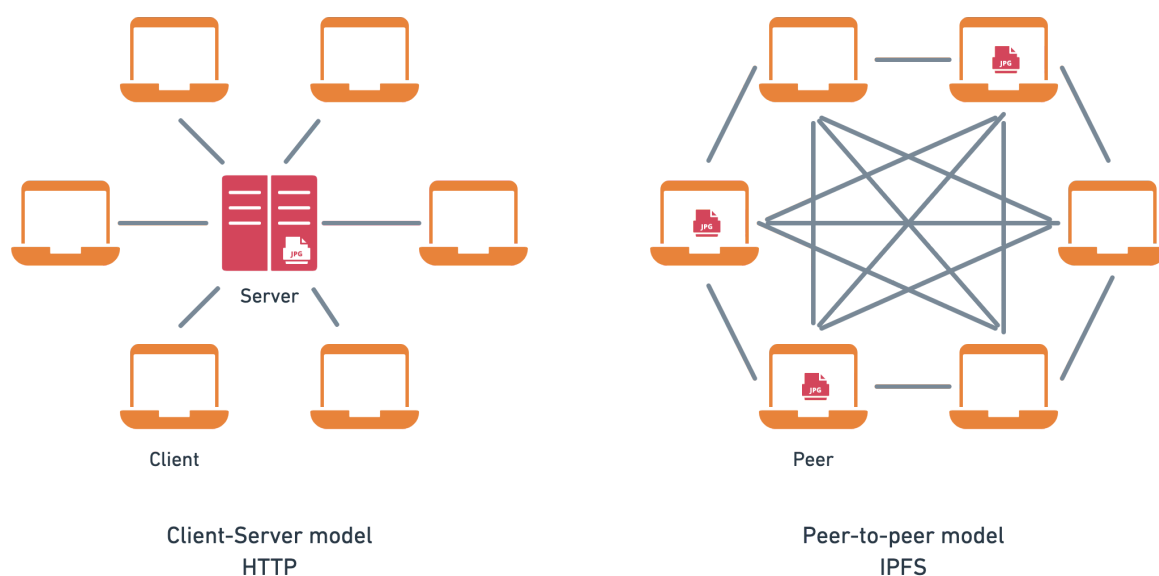


Figura 9: Comparação entre protocolo HTTP e IPFS de [Daniel and Tschorsch, 2022]

Devido às características de resiliência, descentralização e disponibilidade mencionadas, prevenindo quaisquer alterações nos ficheiros, justifica-se a utilização de *IPFS* para guardar o conteúdo digital dos *NFTs* gerados pela plataforma e desta forma, nesta dissertação foi utilizado um *IPFS* providenciado pelo *Pinata*.

¹² <https://moralis.io/ipfs-nft-how-to-use-ipfs-for-nft-metadata/>

Pinata

O *Pinata* é um fornecedor de serviço *IPFS* que permite aos seus utilizadores manter, gerir e partilhar ficheiros nos seus servidores.

Para desenvolvedores, o *Pinata* disponibiliza uma *API* e *SDKs* para o desenvolvimento de aplicações utilizando a sua tecnologia, fazendo a maior parte do trabalho necessário na fixação de conteúdo digital em *IPFS*. Não havendo necessidade por parte dos utilizadores de criar e gerir nodos *IPFS*.

Na aplicação foi utilizado o plano grátis do *Pinata* que tem algumas limitações, no entanto, para o protótipo final desta dissertação revela-se suficiente.

Conexão à API do Pinata

1. O primeiro passo para conectar a aplicação ao *Pinata* foi fazer o registo na página *web*, tendo sido utilizados os dados pessoais do autor deste documento.
2. O passo seguinte foi a criação de uma *API Key* através da interface *web*.
3. De seguida, foi necessário importar as credenciais geradas no passo anterior a *API Key* e a *Secret API Key* necessárias para efetuar pedidos à *API* do *Pinata* para publicar o conteúdo digital no *IPFS* disponibilizado.
4. Nesta fase, para efetuar pedidos à *API* do *Pinata* apenas é necessário incluir nos *headers* dos pedidos as credenciais importadas. O excerto de código, apresentado na Listagem 16, demonstra como são efetuados os pedidos para afixar novo conteúdo digital no *IPFS* aquando da geração de um *NFT* na plataforma.

```
const pinataOptions = {
  method: 'POST',
  url: 'https://api.pinata.cloud/pinning/pinFileToIPFS',
  maxContentLength: Infinity,
  maxBodyLength: Infinity,
  headers: {
    'Content-Type': `multipart/form-data; ...`,
    'pinata_api_key': pinataApiKey,
    'pinata_secret_api_key': pinataSecretApiKey,
  },
  data: formData,
};
```

```

    axios(pinataOptions)
      .then(async response => {
        //URI ONDE O FICHEIRO FOI COLOCADO - IPFS - PINATA
        let ipfsHash = response.data.IpfsHash
        ...
      })

```

Listagem 16: Conexão ao Pinata

Como é possível observar no código apresentado em 16, no cabeçalho do pedido são enviadas as chaves necessárias para o pedido à API do *Pinata* ser válido. O ficheiro a afixar encontra-se no *formData* enviado no pedido. Após o pedido ser enviado, o documento é finalmente afixado em *IPFS*, ficando disponível para consulta através de um *URI*.

O URI do conteúdo digital do documento é único e inalterável. Este contém o *Unique Content Identifier* (CID) utilizado para referenciar o conteúdo digital no *IPFS*. Uma vez que o conteúdo é afixado no servidor, o seu CID é único e não pode ser alterado, garantindo desta forma a segurança e integridade dos dados armazenados.

5.3 Assinatura Digital

De forma a complementar e tornar mais segura a certificação de documentos baseados em *NFTs*, decidiu-se incorporar na solução proposta um mecanismo com dois níveis de segurança para aumentar a segurança da certificação de documentos na plataforma.

Desta forma, para além da criação de um *NFT* para o documento certificado na plataforma é também necessário que contenha uma assinatura digital válida, de forma a garantir as vantagens de ambas as soluções.

Assim sendo, quando um novo documento PDF é selecionado para ser certificado na plataforma, verifica-se se o mesmo possui alguma assinatura. Caso o documento selecionado esteja assinado, o processo de criação do *NFT* prosseguirá normalmente. Caso contrário, a plataforma assinará o documento digital e acrescentará uma página no início do documento PDF com algumas informações relativas ao processo efetuado.

Certified in the blockchain by DSTelecom
With Token ID: 64dd00c06000e9bd0177daa6
dstelecom

The content of the certified file can be
checked on the following pages.

Figura 10: Exemplo de página adicionada no documento assinado

A página apresentada na Figura 10 é apenas acrescentada aos documentos que são assinados digitalmente pela plataforma, uma vez que, caso fosse adicionada a um documento PDF previamente assinado, iria invalidar a assinatura já presente no documento.

5.3.1 Processo de Assinatura

Para o caso em que o documento é assinado digitalmente na plataforma foi necessário criar um certificado auto-assinado através da ferramenta *OpenSSL*. Os comandos utilizados foram os seguintes:

- `openssl req -x509 -newkey rsa:2048 -nodes -keyout privateKey.pem -out certificate.pem -days 365`
- comando utilizado para gerar um certificado auto-assinado.
- `openssl pkcs12 -export -out keyStore.p12 -inkey privateKey.pem -in certificate.pem`
- comando utilizado para criar um armazenamento de chaves **Public Key Cryptography Standards 12 (PKCS#12)** contendo a chave privada e o certificado associado.

Estes comandos, em conjunto, geram um certificado auto-assinado e, de seguida, o armazenam junto com a sua chave privada num armazenamento de chaves *PKCS#12*, que será utilizado na assinatura de documentos na plataforma. A assinatura é feita através da biblioteca *node-signpdf*, como apresentado nas Listagens 17 e 18.

```
//Private key + certificate para assinar documentos
const privKeyPATH = path.join(__dirname, '..', 'keys', 'keyStore.p12');

//Assinatura digital
```

```
const signedFile = await signPDF(signedFilePath,privKeyPATH);
```

Listagem 17: Assinatura Digital - certificador.js

```
const signer = require("node-signpdf").default;

async function signPDF(pdfFile, certFile) {
  const pdfDoc = fs.readFileSync(pdfFile);
  const certificate = fs.readFileSync(certFile);
  let newPDF = await _addPlaceholder(pdfDoc);

  const signedPDF = await signer.sign(newPDF, certificate);
  return signedPDF;
}
```

Listagem 18: Assinatura Digital - SignPDF.js

Esta solução apenas poderá ser temporária e utilizada em protótipo, uma vez que certificados auto-assinados não possuem a confiança e validação fornecidas por certificados emitidos por autoridades de certificação reconhecidas. Desta forma, não são adequados em cenários onde os utilizadores esperam um elevado nível de confiança e segurança. Neste caso, obter um certificado de uma autoridade de certificação confiável é essencial para garantir que os utilizadores reconheçam e confiem no certificado.

5.4 Construção da API de dados

Depois da construção da base de dados e da rede *blockchain* a utilizar pela API, será abordada a construção da API de dados que irá fornecer todos os dados e funcionalidades necessárias para a aplicação em desenvolvimento.

A API foi construída utilizando *Node.js* recorrendo à *framework Express*.

5.4.1 Estrutura da API

A API apresenta a seguinte estrutura:

Cada componente da estrutura apresentada tem um papel importante na API desenvolvida, que serão explicados de seguida:

- **app:**

- **config** - Nesta diretoria são guardadas as credenciais de acesso à base de dados, necessárias para a conexão à mesma.
 - **controllers** - Nesta diretoria encontra-se o controlador de autenticação que possui os métodos necessários para o registo, início e fim de sessão de utilizadores da plataforma.
 - **keys** - Nesta diretoria são armazenadas as chaves pública e privada da *dstelecom* utilizadas para assinar digitalmente os ficheiros certificados pela plataforma.
 - **middlewares** - Os ficheiros *javascript* com as funções executadas em *middleware*, ou seja os métodos invocados e executados antes dos pedidos serem enviados à *API*, encontram-se nesta diretoria. Por exemplo, o método de verificação da existência do *JWTToken* e os métodos de verificação de *roles*.
 - **models_mongodb** - Nesta diretoria são definidos os modelos da base de dados necessários para a aplicação.
 - **routes** - Nesta diretoria encontram-se os *routers* da *API*, ou seja, os ficheiros *javascript* onde estão implementadas todas as rotas desta *API*.
- **fabric** - Nesta pasta encontram-se todos os ficheiros *javascript* necessários para a conexão dos utilizadores da plataforma à rede *blockchain* e invocação de métodos do contrato no canal da rede. Para além disso, é nesta diretoria que se encontra a *wallet* com as identidades dos utilizadores na rede.
 - **digital_signatures** - Nesta diretoria encontram-se os ficheiros *javascript* necessários para assinar digitalmente os documentos PDF a certificar.
 - **exports** - Os documentos, após serem devidamente manipulados pelo sistema, são armazenados temporariamente nesta diretoria. No fim de cada pedido de certificação de um documento esta diretoria é esvaziada.
 - **uploads** - Os documentos carregados para certificação no sistema são guardados temporariamente nesta diretoria. No fim de cada pedido de certificação de um documento esta diretoria é esvaziada.
 - **node_modules** - Nesta pasta encontram-se todos os módulos que o *Node.js* necessita para a aplicação.

- **server.js** - Este é o ficheiro utilizado para inicializar a aplicação e neste são importados todos os ficheiros da pasta *routes* e são definidos os prefixos para as rotas associadas aos respetivos ficheiros. Adicionalmente, neste ficheiro são definidas algumas opções a utilizar por toda a aplicação. Para além disso é definida a porta que ficará à escuta de pedidos. Neste ficheiro é também estabelecida a conexão à base de dados *mongodb*.

5.4.2 Autenticação e permissões

Json Web Token (JWT)

Uma vez que a aplicação contém múltiplas entidades, era importante utilizar um mecanismo de autenticação seguro. O mecanismo escolhido foi **JSON Web Token (JWT)**, que é um padrão frequentemente utilizado no processo de autenticação de utilizadores de uma aplicação. Este é um padrão aberto que define uma forma compacta e independente de transmitir informações de forma segura entre partes, na forma de um objeto *JSON*. Os *JWTokens* são confiáveis e verificáveis visto que são assinados digitalmente, com recurso a um segredo ou a um par de chaves pública e privada, e para além disso é possível garantir que a informação do *JWT* não foi adulterada. No caso da presente aplicação, na criação de *JWTokens* de sessão é utilizado um segredo, também é utilizado o identificador do utilizador e um tempo de expiração para o token.

Os *JWTokens* são compostos por três partes separadas por pontos: Cabeçalho, Dados e Assinatura. Cada uma das partes é encriptada e um *JWT* geralmente se aparenta com o seguinte: *xxxxx.yyyyyy.zzzzz*.

O cenário mais comum para a utilização de *JWT* é em casos de autorização de uma aplicação. Uma vez que após um utilizador se autenticar na aplicação, cada pedido desse utilizador irá incluir o *JWT*, permitindo ao utilizador aceder a rotas, serviços e recursos que são permitidos com aquele *token* em específico ¹³.

A utilização de *JWTokens* na aplicação funciona da seguinte forma:

- Quando um utilizador inicia sessão na plataforma, é criado um *JWTToken* para o mesmo, e este é guardado na memória local.
- Nos pedidos à API efetuados pelo utilizador este *JWTToken* é enviado nos seus *headers* tornando possível a sua verificação.
- A aplicação faz a verificação do *JWTToken* e processa os pedidos efetuados.

¹³ <https://jwt.io/introduction>

- Quando o utilizador termina sessão o *JWTToken* é removido da memória local.

Permissões

A aplicação possui três categorias de entidades diferentes com diferentes permissões de acesso, e devido a isso foi necessário definir as permissões de acesso de cada rota.

Desta forma, foram definidas funções de verificação utilizadas para garantir que apenas utilizadores autorizados acedem a uma determinada rota. Estas são funções de *middleware*, ou seja, agem como uma ponte entre a solicitação recebida do cliente e a resposta final enviada de volta ao cliente, uma vez que são funções executadas antes da chamada da função principal de uma rota.

As funções de verificação implementadas são as seguintes:

- **verifyToken** - função que verifica a existência do *JWT* no cabeçalho do pedido. Caso se verifique a existência do *token* no pedido, a função principal da rota será executada, caso contrário será retornado um erro de autorização.
- **isAdmin** - função que verifica se o utilizador que efetuou a solicitação é administrador. Caso se verifique que o utilizador é administrador o pedido prosseguirá, caso contrário será devolvido um erro de autorização.
- **isCertificador** - função que verifica se o utilizador que efetuou a solicitação é certificador. Caso se verifique que o utilizador é certificador o pedido prosseguirá, caso contrário será devolvido um erro de autorização.
- **checkDuplicateUsernameOrEmail** - função que verifica se o nome de utilizador ou email inseridos já se encontram em uso. Caso se verifique a unicidade do nome de utilizador e email inseridos o pedido prossegue caso contrário será devolvido um erro no registo de um novo utilizador.

As rotas que apenas utilizadores autenticados podem ter acesso têm a função *verifyToken* na sua camada de *middleware*. Por sua vez, as rotas que apenas administradores podem ter acesso têm a função *isAdmin* no seu *middleware*. E por último, as rotas que apenas certificadores podem ter acesso têm a função *isCertificador* no seu *middleware*.

Através da utilização de **JSON Web Token (JWT)** para autenticação e as funções de verificação na camada de *middleware* dos pedidos, garante-se que as rotas estão protegidas, ou seja, que apenas utilizadores autorizados acedem às rotas.

5.4.3 Rotas Implementadas

Nesta secção serão apresentadas todas as rotas de acesso definidas na aplicação e será explicado o papel de cada uma delas.

A API tem vários prefixos principais estando cada um deles relacionado a diferentes partes da mesma.

- **Router com prefixo '/certificador'**

```
1. router.post("/atribuirPermissoes",
    [authJwt.verifyToken, authJwt.isCertificador],
    async(req,res) => {...});
```

Adiciona o identificador do novo certificado à lista de certificados com permissão de acesso, dos utilizadores seleccionados.

```
2. router.post("/gerarCertificado",
    [authJwt.verifyToken, authJwt.isCertificador],
    upload.single('file'),
    async (req,res) => {...});
```

Gera um *NFT* na *blockchain* para o certificado carregado, adiciona a página de marca, cria a assinatura digital, afixa o documento em *IPFS* e cria uma nova entrada na base de dados.

- **Router com prefixo '/certificados'**

```
1. router.get("/validos",
    [authJwt.verifyToken, authJwt.isAdmin],
    async(req,res) => {...});
```

Devolve a lista de tuplos (nome, ID,URI) de todos os certificados válidos.

```
2. router.get("/certUser", authJwt.verifyToken, async(req,res)
    => {...});
```

Devolve a lista de tuplos (nome, ID,URI) dos certificados válidos, cujo utilizador autenticado tem permissão de acesso.

Em ambos os casos a verificação da validade dos certificados é feita em dois passos:

- (a) Verificação da validade do certificado na base de dados.
- (b) Verificação da validade do *NFT* gerado na *blockchain*, através da invocação do método 'TokenURI' do *chaincode* que faz a leitura do *NFT* através do seu identificador e retorna o *URI* do mesmo em caso de sucesso.

Caso não se verifiquem os dois passos mencionados significa que o certificado não é válido.

```
3. router.post("/revogar",
  [authJwt.verifyToken, authJwt.isAdmin],
  async(req,res) => {...});
```

Atualiza a validade de um determinado certificado na base de dados e invoca o método *Burn* do *chaincode* da rede para destruir o *NFT* associado ao documento.

• Rotas de autenticação

```
1. app.post("/api/auth/signup", [...],controller.signup);
```

Efetua as validações de autenticação. Verifica a validade dos dados recebidos como input e encripta a palavra passe recebida. Por fim, em caso de sucesso, regista um novo utilizador na base de dados e também na rede *blockchain*.

```
2. app.post("/api/auth/signin", controller.signin);
```

Verifica as credenciais recebidas como input e inicia a sessão do utilizador, criando e devolvendo um *token* de autenticação para o utilizador.

```
3. app.post("/api/auth/signout", controller.signout);
```

Destroi a sessão do utilizador que seleciona a opção de término de sessão na aplicação.

• Rotas de utilizador

```
1. app.get("/user/roles",authJwt.verifyToken,async(req,res)=> {...});
```

Devolve a *role* do utilizador autenticado.

```
2. app.get("/users",
  [authJwt.verifyToken,authJwt.isAdmin],
  async(req,res) => {...});
```

Devolve a lista de utilizadores comuns.

```
3. app.get("/usersCert",
  [authJwt.verifyToken,authJwt.isCertificador],
  async(req,res) => {...});
```

Devolve a lista de utilizadores. A diferença deste pedido para o anterior é a função de *middleware*. Neste caso apenas certificadores poderão executar este pedido.

```
4. app.get("/certificadores",  
    [authJwt.verifyToken,authJwt.isAdmin],  
    async(req,res) => {...});
```

Devolve a lista de certificadores.

```
5. app.post("/remRoleCert",  
    [authJwt.verifyToken,authJwt.isAdmin],  
    async(req,res) => {...});
```

Remove a *role* de certificado de um utilizador.

```
6. app.post("/giveRoleCert",  
    [authJwt.verifyToken,authJwt.isAdmin],  
    async(req,res) => {...});
```

Atribui a *role* de certificado a um utilizador.

```
7. app.post("/removeUser",  
    [authJwt.verifyToken,authJwt.isAdmin],  
    async(req,res) => {...});
```

Retira as permissões de acesso à aplicação, a um utilizador.

```
8. app.post("/requestRole",[authJwt.verifyToken],  
    async (req, res) => {...});
```

Cria uma solicitação de *role* de certificado.

```
9. app.get("/pedidoFeito",authJwt.verifyToken,  
    async(req,res) => {...});
```

Devolve um *booleano* que indica se o utilizador autenticado tem ou não uma solicitação ativa.

```
10. app.get("/solicitacoes",  
    [authJwt.verifyToken,authJwt.isAdmin],  
    async(req,res) => {...});
```

Devolve a lista de solicitações de *role* certificado, não avaliadas.

```
11. app.post("/aceitarSol",  
    [authJwt.verifyToken,authJwt.isAdmin],  
    async (req, res) => {...});
```

Altera o estado da solicitação e atribui a *role* certificado a um determinado utilizador.

```
12. app.post("/rejeitarSol",
    [authJwt.verifyToken, authJwt.isAdmin],
    async(req, res) => {...});
```

Altera o estado da solicitação, no entanto, não atribui a *role* a um determinado utilizador.

Nos pedidos *POST*, as informações necessárias são devidamente enviadas, como um objeto *JSON*, no *body* dos mesmos.

5.5 Interface Gráfica

Depois do desenvolvimento da API de dados mencionada, foi necessária a implementação de um *front-end* responsável por efetuar os pedidos à API desenvolvida e interagir com os utilizadores finais da plataforma em desenvolvimento. Esta aplicação foi desenvolvida com recurso à *framework react.js*.

Os pedidos à API são feitos com recurso à biblioteca *axios* ¹⁴, da seguinte forma:

```
const handleBuscarCertificados = async () => {
    const token = localStorage.getItem('token');
    const config = {
        headers: { Authorization: `${token}` }
    };
    axios.get(`${API_URL}/certificados/certUser`, config)
        .then(response => {
            setCertificados(response.data);
            setLoading(false);
        })
        .catch(e => {
            setAlertMsg(e.response.data);
            setAlert(true);
        });
}
```

Listagem 19: Exemplo de uma chamada à API

5.5.1 Páginas

Nesta secção são apresentadas as várias páginas com que os utilizadores finais vão interagir. De acordo com o tipo de utilizador que interage com a aplicação existirão diferentes páginas, existindo páginas a

¹⁴ <https://axios-http.com/docs/intro>

que apenas administradores têm acesso e outras a que apenas certificadores podem aceder.

Através da análise das Figuras 11 e 12 pode-se verificar as diferenças existentes entre as páginas a que os utilizadores comuns e os administradores têm acesso.

Apesar dos certificadores e utilizadores terem funcionalidades diferentes, a sua barra de pesquisa é semelhante.



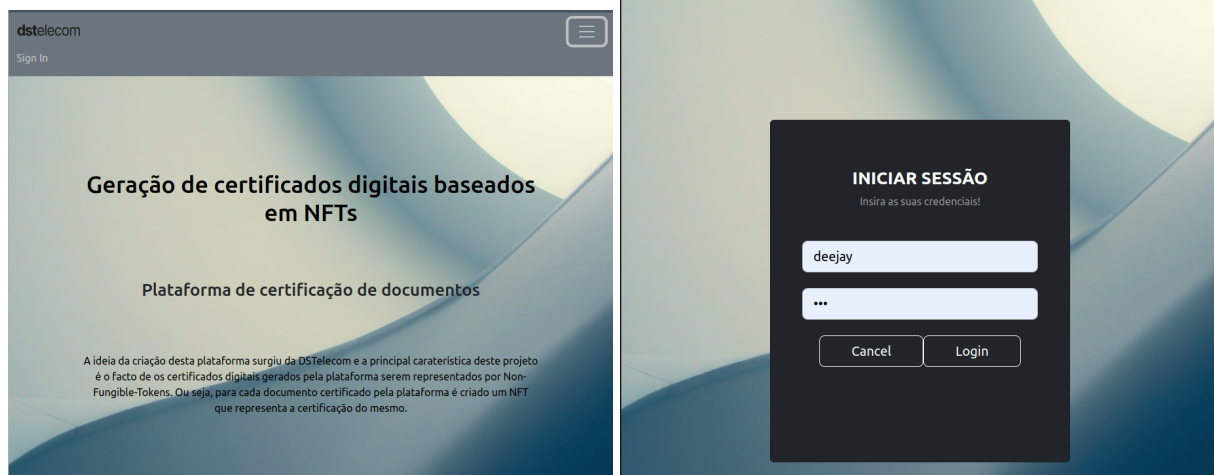
Figura 11: Barra de Pesquisa do Administrador



Figura 12: Barra de Pesquisa do Utilizador

A página inicial é apresentada na Figura 13 contém informações para os utilizadores finais da aplicação e um botão para os mesmos efetuarem o seu início de sessão.

A página de início de sessão apresentada na Figura 13, possui dois *inputs* de entrada, o nome de utilizador e a palavra passe, para os utilizadores inserirem as suas credenciais para efetuarem o início de sessão na plataforma.



(a) Página Inicial

(b) Iniciar Sessão

Figura 13: Página Inicial e Início de Sessão

5.5.2 Utilizadores

Nesta secção são apresentadas as páginas disponíveis para os utilizadores comuns. Como apresentado na Figura 12, os utilizadores têm acesso a duas páginas diferentes, sendo a aba de 'Sign Out' apenas um botão para terminar a sessão.

Uma vez que os utilizadores comuns apenas têm permissões de consulta de documentos na plataforma, a aba de 'Certificar Documento' apresentada na sua barra de pesquisa, leva à página demonstrada na Figura 14. Nesta página os utilizadores podem efetuar uma solicitação para se tornarem certificadores.

Uma vez efetuada a solicitação, a página altera até ao utilizador receber resposta para a sua solicitação.

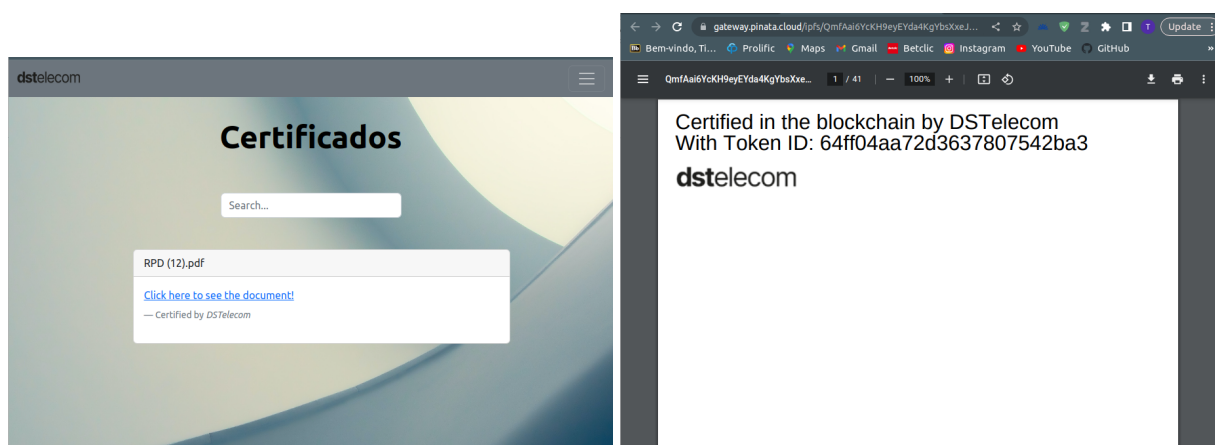


(a) Fazer solicitação

(b) Solicitação feita

Figura 14: Página de Certificar Documento - Utilizador

A página de 'Certificados' do utilizador, apresentada na Figura 15, apresenta todos os certificados válidos que o utilizador autenticado tem permissões de visualização, podendo o mesmo analisar o seu conteúdo através de um clique na hiperligação para o URI correspondente.



(a) Lista de Certificados

(b) Visualização do documento no seu URI

Figura 15: Página de Certificados

5.5.3 Certificador

Nesta secção são apresentadas as páginas disponíveis para os certificadores. De forma semelhante aos utilizadores comuns, este tipo de utilizadores também têm acesso a duas páginas distintas.

A página de 'Certificados' é semelhante á apresentada na Figura 15, no entanto os certificados apresentados serão diferentes. Os certificados que um certificador poderá visualizar são os que o mesmo certificou.

A página de 'Certificar Documento' de um certificador é apresentada na Figura 16. Nesta página, os certificadores poderão selecionar os utilizadores que terão permissões de visualização para o documento a certificar. Para além disso, os mesmos também selecionarão o documento que pretendem certificar. A certificação do documento selecionado acontecerá após um clique no botão 'Certificar Documento'.

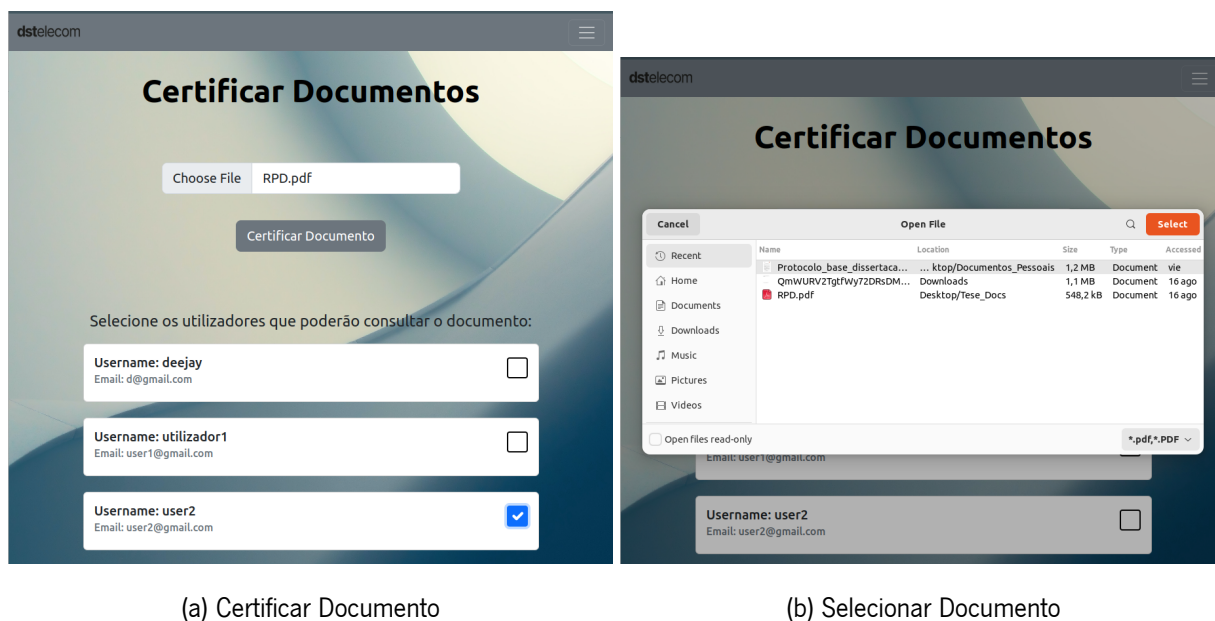


Figura 16: Página de Certificar Documento

Após o clique no botão 'Certificar Documento', se o documento tiver sido certificado com sucesso, a página sofrerá uma atualização, para a página da Figura 17. Esta página contém uma mensagem de sucesso e um botão que reencaminha para os certificados do certificador, para que o mesmo possa verificar o documento recém certificado.

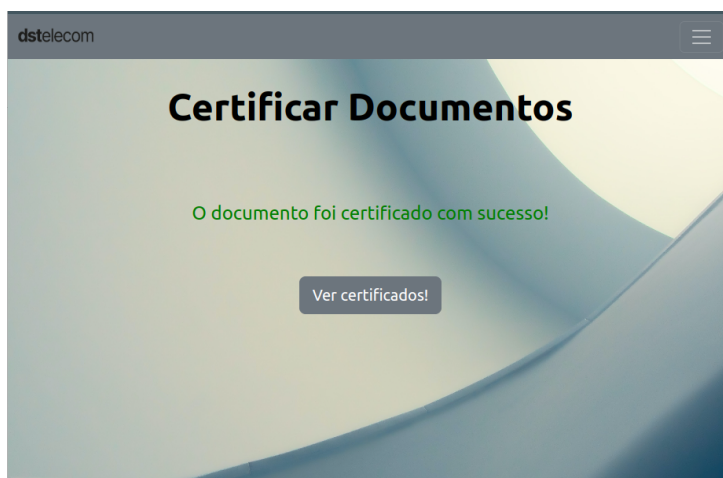


Figura 17: Página de Sucesso após certificação

5.5.4 Administrador

Nesta secção serão apresentadas as páginas disponíveis para o administrador. Como apresentado na Figura 11, os utilizadores têm acesso a quatro páginas diferentes.

Na página de gestão de 'Utilizadores', apresentada na Figura 18, serão listados todos os utilizadores comuns da aplicação, assim como as suas informações, e opções para os remover da plataforma e para dar permissões de certificador.

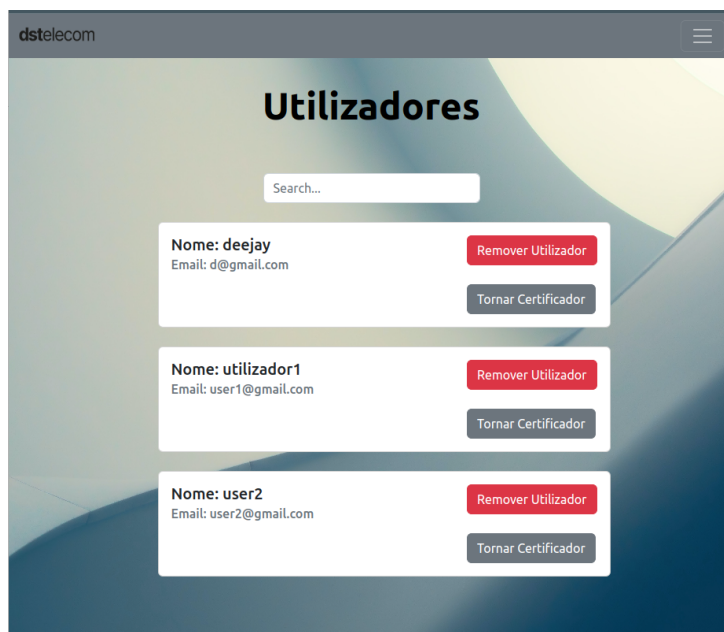
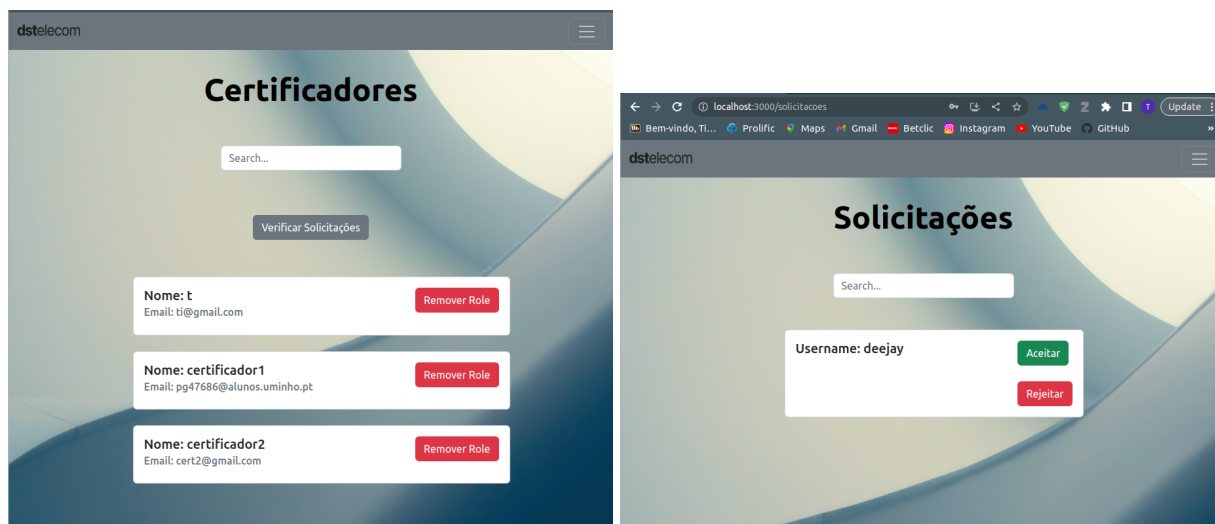


Figura 18: Página de Gestão de Utilizadores

Na página de gestão de 'Certificadores', apresentada na Figura 19, serão listados todos os certificadores da plataforma, assim como as suas informações e opções para remoção das permissões de certificador. Para além disso, nesta página existe um botão que, quando pressionado, reencaminhará para a página das solicitações de permissões de certificador, efetuadas por utilizadores comuns em 14. Na página de gestão de 'Solicitações' serão listadas as solicitações efetuadas, cada uma com duas opções, uma para aceitar e outra para rejeitar a respetiva solicitação, identificada pelo nome de utilizador que a solicitou.



(a) Lista de Certificadores

(b) Lista de Solicitações

Figura 19: Página de Gestão de Certificadores

Na página de 'Registar Novo Utilizador', presente na Figura 20, é apresentado um formulário com 4 *inputs*, para registo de um utilizador. Depois de o administrador preencher o formulário de registo e pressionar o botão "Registar", o novo utilizador com as informações dadas será registado.

Figura 20: Página de Registo de novos utilizadores

Por fim, na página de 'Certificados', apresentada na Figura 21, são listados todos os certificados válidos da aplicação, podendo o seu conteúdo ser visualizado através da hiperligação para o URI correspondente. A diferença desta página para a apresentada na Figura 15 é a opção a que apenas o

administrador tem acesso e que permite a revogação de documentos.

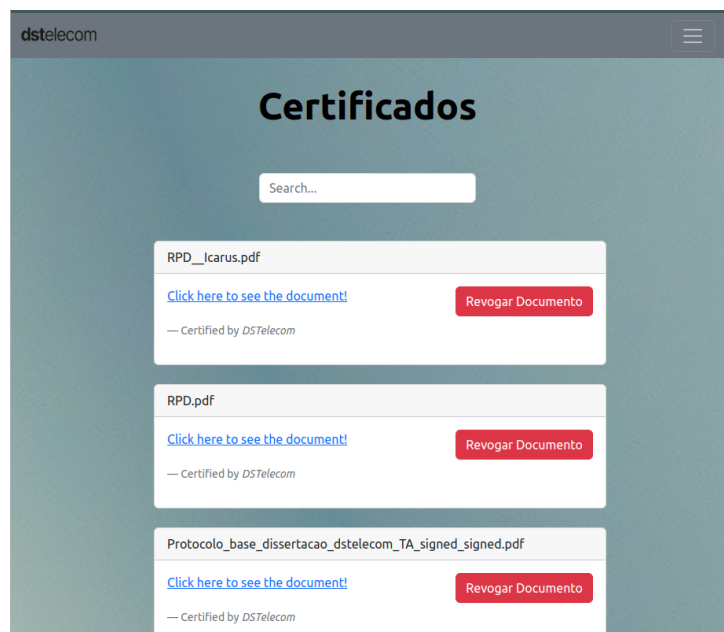


Figura 21: Página de Certificados do Administrador

5.6 Deployment

Nesta secção serão apresentadas as decisões tomadas em relação à colocação da aplicação em ambiente de produção.

5.6.1 Blockchain

Uma vez que a rede *blockchain* utilizada é disponibilizada e mantida pelo *Kaleido*, a mesma pode ser usada em produção, ou seja, a mesma já se encontra em ambiente *deployed*. No entanto, apesar da rede estar alojada na nuvem e estar disponível para ser utilizada, esta não pode ser a versão final caso o protótipo desenvolvido seja transformado num produto final, devido às restrições mencionadas em 5.2.

Desta forma, conclui-se que apesar da rede estar *deployed* e servir perfeitamente para cenários de teste, para que a mesma possa ser utilizada, na versão final do produto desenvolvido, é necessário uma alteração do plano e, consequentemente, uma remodelação da arquitetura da rede .

5.6.2 Aplicação

Uma boa opção para hospedar a aplicação desenvolvida seria através de "Cloud Hosting Providers", como a *AWS*, *Azure* ou *Google Cloud*, uma vez que estes fornecedores de nuvem oferecem soluções escaláveis que podem lidar com o crescimento da aplicação e permitem ajustar de forma simples os recursos conforme necessário. Estas plataformas oferecem também alta disponibilidade e confiabilidade.

Outra opção e, talvez, a mais indicada seria hospedar a aplicação através de um PaaS (Platform as a Service), como o *Heroku* ou *Netlify*, uma vez que, este tipo de plataformas simplificam o processo de *deploy* e gestão da aplicação. Para além disso, a maioria destas plataformas, facilitam o processo de escalonamento horizontal, tornando a aplicação facilmente escalável.

Capítulo 6

Conclusões e trabalho futuro

Neste capítulo será feita a análise do trabalho desenvolvido na dissertação e os seus resultados. Para além disso, serão apresentados alguns pontos que poderão ser melhorados num trabalho futuro.

6.1 Conclusões

A fase inicial da dissertação consistiu no estudo do tema e das diversas tecnologias envolventes que seriam importantes para a concretização dos objetivos da mesma. Desta forma, nesta primeira etapa foram estudados os conceitos de *blockchain* e NFTs, alguns dos seus exemplos de aplicação na atualidade e trabalhos relacionados.

Na fase seguinte, foi analisado o problema em detalhe e foram tomadas as principais decisões tecnológicas para o desenvolvimento da dissertação. Nesta etapa, foi discutida a utilização de NFTs no caso de estudo e foram apresentadas as razões para o qual esta tecnologia se adequa ao problema. Além disso, foram apresentadas duas plataformas *blockchain* que se adequam ao caso de estudo e foi apresentada a decisão da tecnologia utilizada para este propósito.

Para se prosseguir com o desenvolvimento da aplicação de geração de certificados de documentos, foi necessário reunir o conjunto de requisitos funcionais e não funcionais e definir a arquitetura aplicacional da solução.

Na fase do desenvolvimento da aplicação, foi necessário encontrar recursos para que as funcionalidades da aplicação fossem implementadas. Nesta fase, apesar de terem existido algumas dificuldades, foram encontradas soluções, e todos os requisitos definidos na fase anterior foram implementados com sucesso.

Resumindo, a aplicação foi desenvolvida com sucesso e os objetivos da dissertação foram cumpridos.

6.2 Perspetiva de trabalho futuro

Os objetivos para o desenvolvimento desta dissertação foram alcançados. No entanto, existem alguns pontos que poderão ser abordados em trabalhos futuros, que permitirão aumentar o nível de maturidade tecnológica da aplicação desenvolvida em direção a um produto final.

- Fazer testes de usabilidade e tornar a aplicação mais "amiga" do utilizador.
- Obter uma assinatura digital válida de entidade certificadora para que a assinatura dos documentos na plataforma seja reconhecida como válida pelos utilizadores da plataforma.
- Remodelar a rede *blockchain* para que possa ser utilizada em produção.
- Analisar o sistema de ficheiros do *Pinata* e atualizá-lo se necessário para o ambiente de produção.
- Fazer o *deploy* da aplicação na nuvem.
- Realizar testes de segurança e de desempenho na aplicação para verificar o funcionamento da aplicação em ambiente de produção.

Bibliografia

- [1] Ismet Abaci and Eyup Emre Ulku. NFT-based Asset Management System. In *2022 International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pages 697–701, October 2022. doi: 10.1109/ISMSIT56059.2022.9932702. ISSN: 2770-7962.
- [2] Maher Alharby and Aad Van Moorsel. Blockchain-based smart contracts: A systematic mapping study. *arXiv preprint arXiv:1710.06372*, 2017.
- [3] Telamon Ardavanis. Membership nfts: blockchain technology, opportunities, and implementation of utility based non-fungible-tokens. 2022.
- [4] Doc Arsov et al. Periodic table of cryptocurrencies: blockchain categorization. *Available at SSRN 3095169*, 2017.
- [5] Erik Daniel and Florian Tschorsch. Ipfs and friends: A qualitative comparison of next generation peer-to-peer data networks. *IEEE Communications Surveys & Tutorials*, 24(1):31–52, 2022. doi: 10.1109/COMST.2022.3143147.
- [6] Monika di Angelo and Gernot Salzer. Tokens, types, and standards: Identification and utilization in ethereum. In *2020 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, pages 1–10, 2020. doi: 10.1109/DAPPS49028.2020.00001.
- [7] Emma Kirjavainen et al. The future of luxury fashion brands through nfts. 2022.
- [8] Ahmad Musamih, Ahmed Dirir, Ibrar Yaqoob, Khaled Salah, Raja Jayaraman, and Deepak Puthal. NFTs in Smart Cities: Vision, Applications, and Challenges. *IEEE Consumer Electronics Magazine*, pages 1–14, 2022. ISSN 2162-2256. doi: 10.1109/MCE.2022.3217660. Conference Name: IEEE Consumer Electronics Magazine.
- [9] A. Budin Posavec, K. Aleksić-Maslać, and M. Tominac. Non-Fungible Tokens: Might Learning About Them Be Necessary? In *2022 45th Jubilee International Convention on Information, Communication*

- and Electronic Technology (MIPRO)*, pages 700–705, May 2022. doi: 10.23919/MIPRO55190.2022.9803425. ISSN: 2623-8764.
- [10] Wajiha Rehman, Hijab e Zainab, Jaweria Imran, and Narmeen Zakaria Bawany. NFTs: Applications and Challenges. In *2021 22nd International Arab Conference on Information Technology (ACIT)*, pages 1–7, December 2021. doi: 10.1109/ACIT53391.2021.9677260.
- [11] Harsh Sheth and Janvi Dattani. Overview of blockchain technology. *Asian Journal For Convergence In Technology (AJCT) ISSN-2350-1146*, 2019.
- [12] Kei Teshirogi. Mechanism of nft and legal issues related to nft transactions. 2021. URL <https://s3.amazonaws.com/documents.lexology.com/54418385-04f2-4251-a811-575f2bb41b4f.pdf>.
- [13] Dylan Yaga, Peter Mell, Nik Roby, and Karen Scarfone. Blockchain technology overview. *CoRR*, abs/1906.11078, 2019. URL <http://arxiv.org/abs/1906.11078>.
- [14] Yuan YuanJiang and Ji Ting Zhou. Ticketing system based on nft. In *2022 IEEE 24th International Workshop on Multimedia Signal Processing (MMSP)*, pages 01–05. IEEE, 2022.
- [15] Xiongfei Zhao and Yain-Whar Si. NFTCert: NFT-Based Certificates With Online Payment Gateway. In *2021 IEEE International Conference on Blockchain (Blockchain)*, pages 538–543, December 2021. doi: 10.1109/Blockchain53845.2021.00081.
- [16] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang. Blockchain challenges and opportunities: A survey. *International journal of web and grid services*, 14(4):352–375, 2018.

Este projeto foi financiado pela *dstelecom* através de uma bolsa.