INSTITUTO POLITÉCNICO
DE VIANA DO CASTELO

TECNOLOGIAS BLOCKCHAIN & IOT COMO FACILITADOR DA ECONOMIA CIRCULAR NA CADEIA DE VALOR DOS PRODUTOS ELÉTRICOS E ELETRÔNICOS

Leonardo Fernandes Magalhães

Escola Superior de Tecnologia e Gestão

INSTITUTO POLITÉCNICO
DE VIANA DO CASTELO

Leonardo Fernandes Magalhães

TECNOLOGIAS BLOCKCHAIN & IOT COMO FACILITADORES DA ECONOMIA CIRCULAR NA CADEIA DE VALOR DOS PRODUTOS ELÉCTRICOS E ELETRÓNICOS

Nome do Curso de Mestrado
Engenharia Informática

Trabalho efectuado sob a orientação do
Professor Doutor António Miguel Rosado da Cruz e
Professor doutor Sérgio Ivan Lopes

Dezembro de 2023

Resumo

O lixo eletrónico tem um impacto ambiental crescente à medida que o consumo de aparelhos eletrónicos continua a evoluir. Além disso, a produção desses dispositivos resulta no aumento do consumo de recursos naturais, e na criação de vários compostos perigosos, que por norma não são tratados adequadamente.

Circularizar o atual modelo linear utilizado na indústria dos Equipamentos Elétricos e Eletrónicos (EEE) é uma alternativa que pode ser usada para abordar tais questões. A recuperação de produtos que se encontram no final de vida, e a reintrodução dos seus componentes ou matérias-primas (como semicondutores, placas de circuito, metais brutos, etc.) na cadeia de valor são exemplos disso, ajudando a criar este processo mais sustentável. Com isso em mente, para alcançar uma abordagem mais sustentável, este trabalho propõe a implementação de um mecanismo de rastreabilidade em toda a cadeia de valor de EEE, para seguir passo a passo um produto e as suas atividades em todo o seu ciclo de vida. O mecanismo de rastreabilidade fornece informações de rastreio em tempo real sobre um produto, registo histórico acerca do mesmo, e otimiza os processos na cadeia de valor. Isso pode ser alcançado utilizando a tecnologia *blockchain* (BCT) integrada com tecnologia Internet das Coisas (IoT).

Por um lado, uma *blockchain* permite implementar uma base de dados distribuída que permite aos participantes armazenar informações com segurança, e em tempo real. Ao mesmo tempo, fornece um mecanismo de consenso para transações *peer-to-peer*, eliminando a necessidade de uma entidade intermediária para processar e preservar os dados das transações.

Por outro lado, a tecnologia IoT facilita a conexão entre o mundo físico e o mundo digital. Este trabalho visa aproveitar as vantagens disponibilizadas pela BCT e juntá-las com a tecnologia IoT, para avançar a economia circular digital na cadeia de valor de EEE. Integrar a BCT e a tecnologia IoT na cadeia de valor de EEE pode impulsionar a otimização e eficiência das operações, reduzindo erros humanos na gestão de *stock*, e fornecendo rastreabilidade do produto ao longo das suas principais etapas da cadeia de valor: 1) coleta de lixo eletrónico; 2) armazenamento de lixo eletrónico; 3) triagem manual, desmontagem, fragmentação; 4) separação mecânica; e 5) recuperação.

Abstract

Electronic waste (e-waste) is waste produced with outdated materials and has a rising environmental impact as home appliance and electronics consumption continues to expand. Additionally, the creation of such devices results in an increase in the consumption of natural resources and the generation of several poisonous and dangerous compounds, which are typically not handled appropriately.

Circularizing the linear model now used in the Electric and Electronic Equipment (EEE) value chain is one strategy that might be used to address such issues. The recovery of End-of-Life products and the reintroduction of their parts, components, or raw materials (such as semiconductors, circuit boards, raw metals, etc.) into the value chain are examples of this, which help to create a value chain that is more sustainable. Having that in mind, to accomplish a more sustainable approach, this work proposes the implementation of a traceability mechanism throughout the EEE value chain, to follow step by step an item or activity in the value chain. The traceability mechanism provides real-time traceability information about an item, offers a historical record about an object throughout the value chain, and optimizes processes in the supply chain. It can be achieved by using Blockchain Technology (BCT) integrated with Internet of Things (IoT) technology.

On one hand, Blockchain uses distributed ledger technology for implementing a distributed database that allows participants to securely store information in real-time. At the same time, it provides a consensus mechanism for peer-to-peer transactions, therefore, eliminating the requirement for a middleman to process and preserve transaction data.

On the other hand, IoT facilitate in the connection between the physical and the digital worlds. This work aims to take the advantages delivered by BCT and join it with IoT technology for advancing digital circular economy in the EEE value chain. By integrating BCT and IoT technology in an EEE value chain can boost operation optimization and efficiency, decreasing human errors in inventory management and giving product traceability throughout its core value chain stages: 1) e-waste collection; 2) e-waste storage; 3) manual sorting, dismantling, shredding; 4) mechanical separation; and 5) recovery.

Agradecimentos

O caminho percorrido para concluir esta dissertação foi bastante longo e conturbado, no entanto, são as dificuldades encontradas ao longo deste percurso que me dão o prazer e orgulho de ter finalmente terminado esta etapa. Ao longo de todo o meu caminho académico, seja na licenciatura ou no mestrado, sem dúvida que há inúmeras pessoas a agradecer por esta fantástica fase da minha vida, no entanto, não há secção suficiente para agradecer à minha mãe e aos meus irmãos pelo constante apoio e incentivo, sem o qual, não estaria, certamente, a concluir esta etapa.

Agora, sim, chegou o momento de falar sobre o meu percurso académico. Primeiramente, quero agradecer a todos os docentes que me acompanharam, tanto na licenciatura como no mestrado, e um especial obrigado aos meus orientadores, o Dr. Sérgio Lopes e o Dr. Miguel Cruz, pelo acompanhamento nesta dissertação, pelas suas sugestões e orientação. Em segundo lugar, um obrigado a todos os meus verdadeiros amigos que estiveram sempre comigo desde que me conheceram, com quem certamente já tive a felicidade de compartilhar grandes momentos com cada um deles.

Por fim, mas não menos importante, uma palavra de agradecimento à minha namorada, pela sua paciência e incentivo, qual sem dúvida também foi um pilar neste último ano.

Sem mais demoras, um grande obrigado a todos vocês, e no caso de me ter esquecido de alguém peço imensa desculpa.

Leonardo Fernandes Magalhães

"Everybody should learn to program a computer,
because it teaches you how to think"

Steve Jobs

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas

API Application Programming Interface. vii, ix, 3, 7–10, 49–51, 53, 58, 63, 73, 85, 86

auto-ID Automatic identification. 28

B2B Business to Business. 23

BCT Tecnologia blockchain. 2, 3, 17, 18, 26, 27, 31, 32, 76

BPMN Modelo e Notação de Processos de Negócios. vii, 13–15, 33

CE Economia Circular. 2, 4, 14–16, 30, 76

CPU Unidade de Processamento Central. 12, 73

CRUD Create, Retrieve, Update, Delete. 53, 57

DOM Document Object Model. 10

DSR Design Science Research. 5

EEE Equipamentos Elétrico e Eletrónico. vii, 1–4, 6, 13, 15, 17–19, 26, 29–31, 33, 35, 36, 38, 44, 76

EoL End-of-Life. 30

ERP Enterprise Resource Planning. 3, 18, 32, 36

GPS Global Positioning System. 11, 18, 28, 29

HTTP Hypertext Transfer Protocol. 7, 8, 10, 57

ID Identifier. 18, 19, 46, 48, 55, 57, 67, 71

IoT Internet of Things. 3, 4, 16–18, 27–32, 54, 77

IPFS InterPlanetary File System. 17

JSON JavaScript Object Notation. 8, 10

JWT Json Web Token. 8, 48, 50

LE Economia Linear. 12, 14, 15, 29

LPWAN Low Power Wide Area Network. 29

MRP Manufacturing Resource Planning. 36

NFC Near Field Communications. 18, 29

P2P Peer-to-Peer. 19

pBFT Practical Byzantine Fault Tolerance. 31

PE Produto elétrico. 1, 2, 12, 14, 16, 17, 36, 69

PoS Proof-of-Stake. 23, 31

PoW Proof-of-Work. 22, 23, 31

REACH Registo, Avaliação, Autorização e Restrição de Produtos Químicos. 2

REST Representational State Transfer. ix, 7–10, 48, 49, 53–55, 57, 58, 62, 73, 84

RFID Identificação por Radiofrequência. 18, 28

RTLS Real-time Locating System. 28

Tx Transação. vii, 18, 21–23, 26–28, 31, 54, 71–73, 76

UE União Europeia. 2

UML Unified Modeling Language. 45

URI Uniform Resource Identifier. 19

WSL Windows Subsystem for Linux. 9

Capítulo 1

Introdução

Os problemas levantados devido às alterações climáticas são uma das maiores preocupações que a humanidade atualmente enfrenta. Estas alterações ambientais são causadas por diversos fatores, tais como a desflorestação, a enorme utilização e queima de combustíveis fósseis, entre outras atividades humanas com impacto direto sobre o ambiente. Grande parte dos diversos tipos de indústria contribuem diretamente para este impacto ambiental que, por consequência, leva ao aquecimento global. O sector dos EEE é um dos maiores culpados deste problema. Esta indústria é uma das que mais contribui para este aumento, devido principalmente ao crescimento da exploração de matérias-primas, atividades logísticas e transformação industrial.

A indústria de produtos elétricos produz lixo elétrico (*e-waste*) a um ritmo sem precedentes, sendo que grande parte não chega a canais próprios de reciclagem, acabando em lixeiras e aterros [4]. Este sector inclui o fabrico de vários tipos de produtos, tais como telemóveis, televisões, micro-ondas e computadores que podem ser constituídos por outros tipos de produto, como, por exemplo, processadores ou memórias (componentes macro), e outros componentes de dimensões muito reduzidas (componentes micro). Para construir tais produtos são necessários vários recursos naturais, desde os mais escassos até aos mais abundantes [5].

O crescimento da procura por dispositivos mais pequenos (exemplo: *smartphones*) também levou ao aumento da necessidade de matéria-primas. De 2004 a 2014 a produção de ferro, cobalto e lítio aumento entre 125% a 180% [5]. Diferentes matérias-primas e minerais são utilizados para construir diferentes tipos de componentes e produtos elétricos. De 83 elementos estáveis presentes na tabela periódica, mais de metade dos mesmos são encontrados em *smartphones* [6]. A maioria dos elementos utilizados num Produto elétrico (PE) são facilmente encontrados (tais como ferro e alumínio), no entanto, outros estão em risco de escassez. De 17 elementos raros presentes na tabela periódica, 16 deles

são encontrados no fabrico de *smartphones* [6, 7], sendo que alguns desses materiais estão presentes na lista de matérias-primas críticas[1]. Além disso, alguns EEE também necessitam de minerais de conflito, cuja designação está relacionada com a sua área de origem, onde por muitas vezes os direitos humanos não são respeitados, o que leva com que estes produtos por vezes sejam comercializados por grupos armados ou de forma ilegal. Para combater estes problemas a União Europeia (UE) estabeleceu novas leis[2] com o objetivo de garantir que a importação de tais minerais são efetuadas apenas por fontes fidedignas. Por fim, também é importante mencionar as substâncias de preocupação, visto que contribuem para um significativo impacto social e ambiental. O uso destas substâncias é restrito no mercado europeu segundo a regulamentação Registo, Avaliação, Autorização e Restrição de Produtos Químicos (REACH)[3].

Considerando o impacto ambiental que a extração intensiva destes recursos e a transformação dos mesmos acarretam, uma das abordagens para reduzir os danos causados por esta indústria é tentar circularizar este setor, uma das possíveis abordagens para reduzir os danos ambientais que esta indústria acarreta é o estudo da Economia Circular (CE) neste setor. Este modelo económico é baseado num modelo que promete substituir o tradicional modelo linear, prometendo reduzir, reutilizar, reciclar e recuperar materiais, componentes ou matérias-primas nos processos de produção e consumo [8].

Para avaliar o impacto ambiental de um PE, é crucial registar todas as atividades envolvidas na cadeia de valor e rastrear todos os itens relevantes, como qualquer lote de matérias-primas, componentes elétricos, material elétrico, placas de circuito, circuitos integrados, etc. Para alcançar esse objetivo será necessária uma plataforma de rastreamento para este setor. Atualmente, a Tecnologia blockchain (BCT), já é utilizada para fins de rastreabilidade em diversas áreas [9]. Numa *blockchain*, os dados são armazenados em blocos por ordem cronológica, de forma permanente e imutável, fornecendo transparência na cadeia de valor [9].

[1] Resiliência de matérias-primas: traçando um caminho para maior segurança e sustentabilidade, Comissão Europeia, Bruxelas, setembro de 2020, https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52020DC0474

[2] Regulamento de Minerais de Conflito, Comissão Europeia, Bruxelas, janeiro de 2021, https://ec.europa.eu/trade/policy/in-focus/conflict-minerals-regulation/

[3] Regulamento (CE) n.º 1907/2006 do Parlamento Europeu e do Conselho, de 18 de dezembro de 2006, relativo ao REACH, que institui a Agência Europeia dos Produtos Químicos, altera a Diretiva 1999/45/CE e revoga Regulamento do Conselho (CEE) n.º 793/93 e Regulamento da Comissão (CE) n.º 1488/94, bem como a Directiva do Conselho 76/769/CEE e as Directivas da Comissão 91/155/CEE, 93/67/CEE, 93/105/CE e 2000/21/EC, https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A02006R1907-20140410

1.1 Objetivos

Tendo em conta os problemas associados com a gestão do lixo eletrónico, tornou-se evidente que há uma necessidade crescente de soluções mais sustentáveis e eficientes para lidar com esse problema ambiental. Em particular, a falta de transparência e rastreabilidade na cadeia de valor de equipamentos eletrónicos e elétricos tem sido apontada como uma das principais barreiras para uma gestão adequada deste tipo de resíduos. Portanto, o objetivo desta pesquisa é explorar o potencial da BCT e Internet of Things (IoT) para rumar a uma cadeia de valor mais sustentável, e melhorar a gestão dos seus resíduos. Os objetivos e passos definidos nesta dissertação para propor um protótipo são:

- Fazer a revisão da leitura acerca da cadeia de valor de EEE: 1) Analisar a problemática do lixo eletrónico; 2) Estudar e verificar os benefícios que a BCT e a tecnologia IoT trazem para resolver os problemas encontrados; 3) Estudar outros tipos de sistema de rastreabilidade propostos para esta indústria.

- Propor um mecanismo de rastreabilidade assente em *blockchain* e *IoT* para monitorizar os produtos ao longo da cadeia de valor de EEE, a fim de otimizar processos e melhorar a transparência. 1) Levantar os requisitos cruciais para o primeiro protótipo deste sistema de rastreabilidade; 2) Diagrama conceptual de dados, para estruturar corretamente todos os dados que serão necessários guardar; 3) Diagrama de domínio para definir todas as classes necessárias para o desenvolvimento do protótipo; 4) Diagrama de atividades dos principais casos de uso para saber todo o fluxo de atividades de um determinado processo; 5) Arquitetura do sistema para definir todos os módulos e camadas que vão ser necessários para desenvolver o sistema.

- Desenvolver o protótipo do sistema de rastreabilidade, o mais configurável possível, tendo em conta os seguintes pontos: 1) Uma aplicação web, que fornece a capacidade de configurar a plataforma, e que permite a utilizadores de uma determinada organização registar atividades da cadeia de valor; 2) Aplicação móvel para as empresas criarem atividades da cadeia de valor através de sensores incorporados encontrados nestes dispositivos; 3) Na camada de serviços (API) fornecer rotas com permissões para integrações, com o objetivo de fornecer a possibilidade de integrar os dados de sistemas externos (Enterprise Resource Planning (ERP)).

- Testar as principais rotas da camada de serviços (API) e avaliar a eficácia do mecanismo proposto por meio de um estudo de caso, demonstrando como a tecnologia *blockchain* e *IoT* pode ser utilizada para melhorar a gestão de lixo eletrónico na cadeia de valor de EEE.

- Avaliar o desempenho da aplicação, e caso seja necessário propor soluções futuras para melhorar o desempenho do sistema. Por fim, verificar que tipo de módulos ou funcionalidades possam ser interessantes para evoluir este primeiro protótipo do sistema.

1.2 Contribuições

Esta dissertação foi feita a par com uma bolsa de investigação disponibilizada pelo Instituto Politécnico de Viana do Castelo, tendo contribuído para o avanço do conhecimento sobre a gestão sustentável de lixo eletrónico na cadeia de valor de EEE por meio do estudo da aplicação das tecnologias *blockchain* e IoT. Em particular, esta pesquisa apresenta as seguintes contribuições:

- Publicação do artigo "*A Review on Adopting Blockchain and IoT Technologies for Fostering the Circular Economy in the Electrical and Electronic Equipment Value Chain*" [1] no jornal *Sustainability* do MDPI[4] que contribuiu na: 1) Análise da cadeia de valor linear de EEE e proposta de um modelo de negócio de CE para a circularização desta cadeia de valor; 2) Revisão do estado da arte em torno da rastreabilidade na cadeia de valor de EEE; 3) Revisão do estado da arte para a identificação de facilitadores tecnológicos que suportam CE na cadeia de valor de EEE.

- Participação na segunda edição do *Symposium of Applied Science for Young Researchers* (SASYR)[5], onde este estudo ganhou o prémio de melhor investigação do Instituto Politécnico de Viana do Castelo[6].

- A apresentação de um mecanismo de rastreabilidade baseado em *blockchain* e IoT, e todo o código-fonte do protótipo do mesmo.

Assim, estas contribuições representam uma importante contribuição para a literatura acerca da gestão sustentável de lixo eletrónico na cadeia de valor de EEE, e podem ser úteis para pesquisadores, empresas e governos interessados em melhorar a gestão deste problema ambiental.

[4] https://www.mdpi.com/journal/sustainability
[5] http://sasyr.ipb.pt/sasyr_2022.html
[6] https://www.ipvc.pt/

1.3 Metodologia de Investigação Utilizada

Para prosseguir com esta pesquisa para a revisão de leitura, foi utilizada a metodologia Design Science Research (DSR). Esta metodologia de pesquisa é comummente utilizada entre investigadores científicos de sistemas de informação, para resolver problemas identificados, ou criar soluções inovadoras em problemas já resolvidos por meio da implementação de métodos, modelos, e até mesmo inovações sociais. Em 2007, [10] desenvolveu uma metodologia estruturada em seis etapas, que envolve a identificação e motivação do problema, definição dos objetivos, conceção e desenvolvimento, demonstração, avaliação e comunicação:

1. A identificação do problema baseia-se na pesquisa de soluções existentes já propostas acerca de um determinado tema, para obter um melhor compreensão do mesmo, encontrando conhecimentos sólidos do mesmo para posteriormente propor novas soluções ou novas alternativas para o tema em questão.

2. A definição dos objetivos para o problema em questão deve ser apresentada de forma quantitativa ou qualitativa, descrevendo de maneira racional e fundamentada como o problema será abordado, e porque é que esta solução terá melhores resultados do que as já existentes.

3. A etapa de conceção e desenvolvimento é onde a arquitetura da solução ganhará forma. É também nesta etapa que as principais funcionalidades devem ser completamente descritas e implementadas, a fim de serem testadas.

4. Esta etapa resume-se a demonstrar como a solução apresentada funciona, seja ela simulada ou experimentada com utilizadores reais.

5. Avaliar os resultados e a precisão da solução apresentada é crucial, pois irá decidir se a solução apresentada funciona efetivamente, ou se quaisquer outros aspetos relevantes devem ser melhorados para apresentar uma solução viável. A partir disto, o processo pode voltar para a segunda ou terceira etapa, dependendo dos aspetos em que a solução falhou ou não. Por vezes pode ser apenas uma questão de usabilidade, ou pode ser que, no final, a solução apresentada não mostre viabilidade suficiente para resolver o problema existente, o que levará a uma redefinição completa dos objetivos.

6. A última atividade depende da publicação da solução apresentada numa conferência, revista ou outro ambiente de pesquisa apropriado.

1.4 Estrutura da Dissertação

A estrutura desta dissertação foi definida segundo o caminho sequencial mais lógico possível, facilitando a leitura e compreensão da mesma. O capítulo 1 introduz o tema e a problemática desenvolvida neste documento, assim como outros pontos. A seguir é apresentado um novo capítulo 2 que descreve alguns dos conceitos e ferramentas mais técnicas que vão ser mencionadas ao longo do documento.

No capítulo 3 é efetuada a revisão de leitura, onde tanto é levantado o estado de arte acerca da cadeia de valor de EEE, assim como são estudadas algumas das tecnologias que serão implementadas no protótipo deste trabalho.

Com base no estudo feito no capítulo 3, foi conceptualizada a arquitetura da solução proposta nesta dissertação, a qual é apresentada no capítulo 4, e onde se pode consultar os casos de uso do sistema, a descrição dos mesmos, o diagrama de domínio e a arquitetura.

Após a conceptualização da solução, no 5 são demonstrados os passos e alguma lógica do desenvolvimento dos vários componentes do sistema. Posteriormente, no capítulo 6 é feita a demonstração do objetivo final do sistema (a construção da rastreabilidade do sistema), e uma análise e discussão dos resultados obtidos ao utilizar o sistema desenvolvido.

Por fim, no capítulo 7 são tiradas algumas conclusões acerca do trabalho desenvolvido e ideias futuras para implementar no sistema.

Capítulo 2

Conceitos, Ferramentas, Linguagens & *Frameworks*

Esta secção tem como objetivo apresentar alguns conceitos mencionados ao longo deste documento, introduzir algumas definições cruciais para a melhor compreensão do mesmo, como a explicação das ferramentas, linguagens de programação e *frameworks* que foram utilizadas durante o desenvolvimento do protótipo referente ao caso sob estudo.

2.1 Conceitos

2.1.1 HTTP

O Hypertext Transfer Protocol (HTTP) define um conjunto de métodos que permitem indicar a ação desejada sempre que é efetuado um pedido para executar um determinado recurso. Estes métodos são conhecidos como verbos HTTP, embora estes estejam diretamente conectados com uma REST API, já que a mesma também os possui (GET, POST, PUT, PATCH & DELETE) [11].

2.1.2 REST API

As APIs são mecanismos que permitem que vários componentes ou diferentes sistemas comuniquem entre si, podendo afirmar que funcionam como um meio intermediário entre aplicações. Estes mecanismos são cruciais no desenvolvimento de *software* já que permitem a integração de diversos componentes de um sistema, ou a integração de um novo sistema com um já existente. Existem diferentes tipos de arquiteturas API, no entanto, a arquitetura que foi utilizada neste projeto foi a REST. Esta arquitetura permite

interagir com comunicações HTTP e com os métodos descritos na secção anterior (secção 2.1.1). Quando se utiliza uma API REST, duas partes têm que estar envolvidas, já que uma é responsável por fazer o pedido, e a outra é responsável por responder ao pedido com os recursos pedidos, ou com alguma mensagem/código de erro [11].

2.1.3 JWT

O Json Web Token (JWT)[1] é um método padrão no desenvolvimento de *software* para a autenticação entre duas partes, este método utiliza um *token* assinado por meio de uma determinada função de encriptação, e contém objetos JavaScript Object Notation (JSON) que permitem guardar dados no processo de autenticação. Assim o *token* é composto por três partes, 1) o cabeçalho que contém o tipo de algoritmo para encriptar os dados e identifica o *token* como um *token* JWT; 2) o corpo do pedido que contém um objeto com dados específicos para autenticação; 3) a assinatura digital que garante que o *token* não foi alterado em nenhum momento.

2.1.4 *Frontend & Backend*

Estes dois conceitos são bastante mencionados ao longo deste documento, uma vez que são conceitos muito importantes no desenvolvimento de aplicações. Resumidamente, o *frontend* refere-se a todas as aplicações do lado do cliente num sistema, isto é, tudo que o cliente/utilizador interage diretamente. Por outro lado, o *backend* representa o lado do servidor de um sistema e é a parte que o cliente/utilizador não consegue visualizar. As responsabilidades de aplicações *backend* incluem gerir e armazenar dados, como também toda a lógica de negócio do sistema.

Tanto o *frontend* como o *backend* necessitam de comunicar entre si para um sistema funcionar da maneira esperada, no entanto, não significa que um não pode funcionar sem o outro. Por norma, o *frontend* necessita de uma aplicação *backend* para funcionar corretamente, no entanto, uma aplicação *backend* não necessita de uma aplicação *frontend* para funcionar.

[1] `https://jwt.io/`

2.2 Ferramentas

2.2.1 WSL

A ferramenta Windows Subsystem for Linux (WSL)[2] permite executar a maioria das ferramentas da linha de comandos, recursos e aplicações de um ambiente GNU/Linux (no caso deste projeto o *Ubunto* [3]) diretamente no *windows*, sem a necessidade da sobrecarga de uma máquina virtual tradicional.

2.2.2 *Docker & Docker Compose*

A ferramenta *docker*[4] é uma tecnologia para desenvolver, publicar e gerir aplicações em contentores, que podem ser executados em qualquer ambiente. Um contentor *docker* virtualiza um único sistema operativo com os recursos mínimos para as aplicações serem executadas. As maiores vantagens desta tecnologia é o facto dos contentores poderem ser executados em qualquer máquina, visto que o ambiente é sempre o mesmo.

O *docker compose* é uma ferramenta utilizada para gerir os vários contentores *docker*, permitindo compactar todas as configurações *docker* num único ficheiro, de forma a executar todos os contentores necessários ao mesmo tempo.

2.2.3 *Fablo & Fablo REST*

O *Fablo* [5] é uma ferramenta que permite criar uma *blockchain hyperledger fabric* e gerar todos os seus componentes em contentores *docker*. Esta ferramenta suporta os diferentes tipos de protocolos de consenso do *hyperledger fabric*, assim como a gestão dos diferentes módulos que o mesmo fornece, como as organizações da *blockchain*, a definição dos canais da rede, os *smart contracts* também conhecidos como *chaincodes*, etc.

O *Fablo REST* é uma sub-ferramenta do *Fablo* que permite a criação automática de uma REST API para cada organização da *blockchain*, permitindo lidar com os diferentes canais em que a mesma está inserida, e os consequentes *smart contracts*.

[2] https://learn.microsoft.com/en-us/windows/wsl/install
[3] https://ubuntu.com/
[4] https://www.docker.com/
[5] https://labs.hyperledger.org/labs/fablo.html

2.2.4 *NodeJS & ExpressJS*

O *NodeJS* é uma ferramenta multi-plataforma que permite a execução de código *javascript* no *backend*, ou seja, do lado do servidor. O *ExpressJS* é uma biblioteca desenvolvida para *NodeJS* que permite a criação de servidores HTTP, o que faz com que esta ferramenta seja ideal para a criação da REST API deste projeto.

2.2.5 *MongoDB & GridFS*

O *MongoDB Atlas*[6] é um serviço de base de dados na nuvem (*cloud*). Este serviço simplifica a implementação, a gestão e a escalabilidade de uma base de dados *MongoDB*. Além disso, este serviço também fornece escalabilidade automática e medidas de segurança robustas.

O *MongoDB*[7] é uma base de dados NoSQL orientada a documentos, que fornece flexibilidade, escalabilidade e desempenho. Esta base de dados armazena documentos semelhantes a objetos JSON, permitindo definir relacionamentos hierárquicos complexos e fácil adaptação a estruturas de dados em evolução. Este modelo combinado com os recursos de consulta e indexação de alta desempenho fornecidos por esta base de dados, permite o armazenamento e leitura de dados de forma bastante eficiente. Para além destas características, o *MongoDB* também fornece o *GridFS*[8], um sistema de armazenamento de ficheiros desenhado para guardar e ler arquivos grandes numa base de dados *MongoDB*.

2.3 Linguagens & *Frameworks*

2.3.1 *React*

React[9] é uma *framework frontend javascript* que permite construir interfaces para o utilizador. Esta *framework* utiliza componentes (blocos reutilizáveis), para criar aplicações *web* dinâmicas. O *React* atualiza e renderiza componentes com eficiência devido ao uso de uma Document Object Model (DOM) virtual. Com um ecossistema próspero e ampla adoção, o *React* é uma escolha popular para criar aplicações *web* interativas e escaláveis.

[6] https://www.mongodb.com/atlas/database
[7] https://www.mongodb.com/
[8] https://www.mongodb.com/docs/manual/core/gridfs/
[9] https://react.dev/

2.3.2 React Native

O *React Native*[10] é uma *framework* que estende os recursos do *React* para permitir a criação de aplicações móveis nativas utilizando *javascript*. Com esta *framework*, os programadores podem aproveitar o seu conhecimento em *React* para desenvolver aplicações móveis multi-plataforma, que funcionam perfeitamente em dispositivos *iOS* e *Android*.

Semelhante ao *React*, o *React Native* também segue uma arquitetura assente em componentes. Estes componentes são compilados em código nativo, criando interfaces que funcionam como se fossem verdadeiramente ecrãs nativos. Esta abordagem elimina a necessidade do desenvolvimento de múltiplas aplicações nativas para diferentes sistemas operativos. Uma das principais vantagens do *React Native* é a sua capacidade de utilizar recursos nativos dos dispositivos móveis, como a câmara, Global Positioning System (GPS), notificações, etc.

2.3.3 Golang

A linguagem de programação *Go* é uma linguagem que se concentra na simplicidade e eficiência, é uma escolha moderna para aplicações *backend* de alto desempenho, visto que compila de forma bastante rápida. Assim, dentro das linguagens disponíveis para desenvolver *smart contracts* para a *framework hyperledger fabric* esta é a mais indicada devido à sua alta desempenho.

[10] `https://reactnative.dev/`

Capítulo 3

Revisão de Literatura

3.1 Cadeia de valor de Produtos Elétricos & Eletrónicos

Atualmente, a cadeia de valor de PE é uma das maiores a nível mundial, envolvendo diversas empresas à volta do mundo com diferente processos industriais, desde a extração de matérias-primas ao consumidor final. Algumas destas empresas são responsáveis por produzir um produto final, no entanto, a maioria delas apenas fabrica algum tipo de componente intermediário para outro produto, como placas de circuito, *chips*, Unidade de Processamento Central (CPU), entre outros componentes eletrónicos. Estas atividades (extração de recursos naturais, transporte, consumo de energia na produção, etc.) acarretam um significativo uso de recursos naturais que, por sua vez, traduzem-se consequências ambientais [12].

De acordo com Foundation, o modelo económico atualmente predominante é o Economia Linear (LE). Desde a terceira revolução industrial, o pensamento "linear" tem levado ao crescimento económico em diversas partes do mundo. Este modelo de negócio é assente no intensivo uso de materiais e no baixo custo de mão de obra. Por um lado a automação, o baixo preço das matérias e a redução dos custos de mão de obra são as razões pelo qual atividades de reciclagem e reutilização têm sido negligenciadas. Por outro lado, a busca de mão de obra cada vez mais barata muitas das vezes tem como consequência distâncias de transporte mais longas e ineficazes.

Embora a LE tenha sido um modelo com grande sucesso no último século, o mesmo levantou várias questões, uma vez que utiliza recursos de forma insustentável, que por consequência leva à produção de grandes quantidades de lixo eletrónico [14]. Tendo isto em vista, o crescimento populacional também vai exigir cada vez mais recursos, para acompanhar a demanda por tal crescimento. Além das questões ambientais, também existe uma preocupação relativa aos recursos não renováveis, visto que estes estão a

tornar-se cada vez mais escassos (alguns metais, minerais e combustíveis fósseis). Atualmente, os preços destes recursos tem-se tornado imprevisíveis devido ao aumento dos mesmos, causando custos das atividades mais elevados ao longo da cadeia de valor, que por sua vez traduz-se num preço superior para o consumidor final [14].

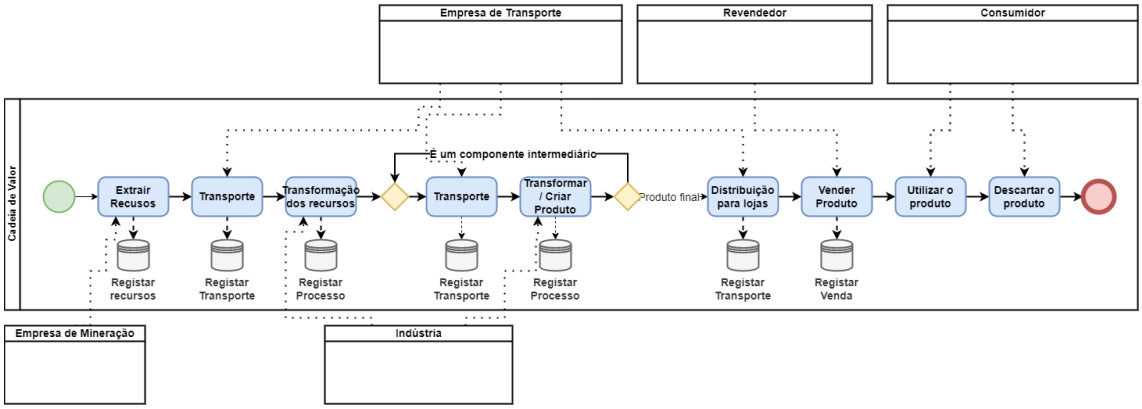

Figura 3.1: BPMN do modelo linear da cadeia de valor de EEE (adaptado de [1])

Como foi mencionado anteriormente, esta cadeia de valor envolve inúmeros participantes de todo o mundo. Na figura 3.1 é apresentado um modelo de processo de negócio genérico e interorganizacional da cadeia de valor de EEE, com alto nível de abstração [1]. Tendo isto em conta, a *pool* principal representa as atividades envolvidas na cadeia de valor, desde a extração de recursos naturais até ao descarte do produto pelo consumidor final. Todos os participantes envolvidos na cadeia de valor são representados como participantes externos. Qualquer participante responsável por realizar uma ou mais atividades no BPMN é representado por uma "mensagem"(seta tracejada) do participante para a atividade correspondente. Uma atividade, a este nível interorganizacional, representa um processo da empresa (participante) responsável pela sua execução, sendo que estes processos podem ser complexos e pouco amigáveis ao meio ambiente.

Conforme se pode observar na figura 3.1, o processo de negócio começa pela extração e tratamento inicial de vários tipos de matérias-primas (primeira atividade do processo) pelas empresas de mineração, como, por exemplo, estanho, silício, cobalto, ferro ou cobre. Posteriormente, estes recursos são transportados para refinação ou fusão de metais, conforme representado na segunda e terceira atividade na figura 3.1. A seguir, os recursos refinados são transportados para empresas responsáveis por criar os componentes elétricos (quarta e quinta atividades na Figura 3.1), como chips, transístores, etc. Algumas destas empresas produzem produtos finais para o consumidor, enquanto outras produzem componentes que serão utilizados por outra empresa para produzirem outro produto (este ciclo é representado entre os dois *gateways* presentes no processo de negócio da figura 3.1).

Após o produto final ser produzido, o mesmo é distribuído para lojas para venderem o produto para o consumidor ou cliente final. Por fim, no fim do ciclo de vida deste produto, o seu utilizador pode descartá-lo, produzindo assim lixo eletrónico. No BPMN representado na figura 3.1, os principais participantes na cadeia de valor são:

- Empresas de Mineração - Este participante representa as empresas responsáveis pela exploração de minas de onde são retiradas as matérias-primas necessárias para esta cadeia de valor.
- Empresas de Logística - Este participante representa as empresas responsáveis pelo transporte e distribuição dos produtos entre empresas à volta do mundo.
- Indústria - Este participante representa as empresas responsáveis pela produção de novos tipos de produtos, como a refinação de matéria-primas, a produção de componentes intermediários ou produtos finais.
- Revendedor - Este participante representa as empresas responsáveis por distribuir produtos pelas lojas para chegarem ao consumidor final.
- Consumidor - Este participante representa o consumidor final de um determinado produto, que após o adquirir e o utilizar dispões do mesmo, segundo o LE representado na figura 3.1.

Este setor contribuí bastante para as alterações climáticas. Assim, é necessária uma abordagem mais sustentável para minimizar esse impacto ambiental. Quando o objetivo é rastrear um determinado produto na cadeia de valor, é necessário guardar todas as atividades da mesma. Esta informação acerca das atividades é representada pelos armazenamentos de dados na figura 3.1.

3.2 Circularização do Atual Modelo Económico

Devido à escassez de alguns recursos naturais, ao crescente uso de PE, aos recentes avançados tecnológicos e, principalmente, ao impacto que o lixo eletrónico não tratado provoca no ambiente, a sociedade é, hoje em dia, desafiada com a gestão do fim de vida de produtos, para economizar recursos e reduzir o impacto ambiental [15].

A CE garante a circulação de recursos naturais ou componentes para reutilizá-los em vez de os descartar. Para atingir os objetivos deste modelo económico, é necessário, entre outras variáveis, a cooperação de todos os participantes na cadeia de valor. Para além

disto, é preciso melhorar o *design* de um produto, tendo em vista um *design* que permita substituir facilmente um componente de um produto [16].

O conceito de CE como um modelo regenerativo, geralmente, é visto como uma necessidade rumo à sustentabilidade, com vista em reduzir o desperdício de recursos e maximizar a sua eficiência, mantendo os componentes ou matérias-primas o maior tempo possível dentro da cadeia de valor. Isto pode ser alcançado por meio de métodos de reciclagem, reutilização ou reparação de produtos e/ou matérias-primas [17].

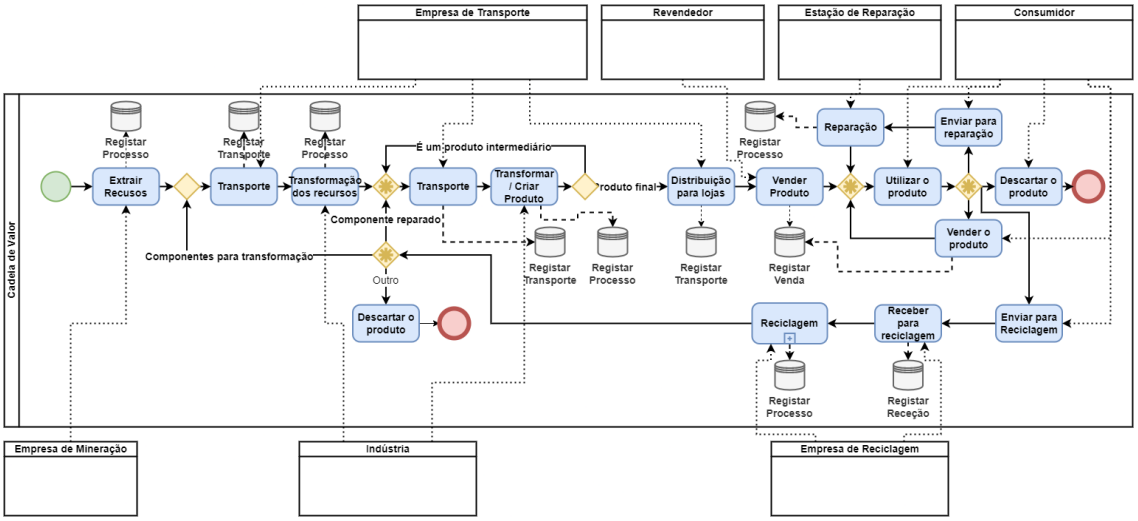

Figura 3.2: Proposta BPMN para facilitar a economia circular na cadeia de valor de EEE (adaptado de [1])

A CE é definida como um modelo de negócio que pode substituir a LE, reduzindo, reutilizando e reciclando matérias-primas ou componentes que podem ser novamente utilizados em processos de produção e consumo. Na figura 3.2 é proposto um BPMN, com alto nível de abstração, para a circularização do atual modelo linear da cadeia de valor de EEE (representado na figura 3.1) [1]. Para circularizar o modelo de negócio foi necessário adicionar novos participantes ou adicionar novas responsabilidades aos participantes já existentes, nomeadamente:

- Consumidor - Os consumidores desempenham um papel bastante importante na circularização do modelo de negócio, já que podem ser responsáveis por i) recuperar um produto e reutilizá-lo; ii) vendê-lo para outro consumidor; ou iii) mandá-lo para vias próprias de reciclagem de lixo eletrónico.

- Empresa de recolha - É um novo tipo de empresa de transporte, responsável pela coleta de lixo eletrónico com destino a empresas de reciclagem.

- Empresa de reciclagem - Este tipo de empresa é responsável pela limpeza e desmantelamento dos produtos, verificando se algum tipo de componente pode ser reaproveitado ou reciclados. O processo de reciclagem de um PE pode ser bastante complexo devido ao grande número de micro-componentes e materiais envolvidos.

- Empresa de reparação - As empresas de reparação estão preparadas para receber e reparar produtos (ou componentes) de consumidores. As empresas de reparação podem substituir ou reparar qualquer componente que deixou de funcionar. Esta facilidade de reparação também está diretamente ligada com o *design* do produto, que deve ser concebida de modo a facilitar a substituição de qualquer componente.

A CE também precisa da participação ativa dos governos para, entre outras medidas, promover a disponibilização de pontos de recolha para este tipo de produtos e, de alguma forma, incentivar a participação dos consumidores no modelo circular por meio da reciclagem dos produtos. Para além disto, os governos também podem criar leis que impeçam ou desacelerem o descarte de PE, principalmente eletrodomésticos de grande porte [1].

3.3 Outras Abordagens para Rastrear PE

Os mecanismos de rastreabilidade possuem a capacidade de seguir passo a passo um produto ao longo da sua cadeia de valor e compreender a sua história, por meio de uma etiqueta IoT associada a esse mesmo produto. O rastreamento de produtos traz transparência e pode ser uma ferramenta de gestão de risco, já que permite seguir cada componente em todas as fases da cadeia de valor e todos os seus movimentos, fornece um registo histórico da rastreabilidade dos componentes e oferece a capacidade de responder de forma mais rápida a potenciais erros na cadeia de valor. Para além disto, um sistema de rastreamento pode ser benéfico para melhorar a relação entre fabricantes, parceiros de negócio e clientes, oferecendo transparência do processo e produtos [18, 19]. Para cumprir com regulamentos, especialmente garantias, devoluções e custos, é benéfico para os fabricantes de PE que tenham plataformas de rastreabilidade.

O autor Richardson, fala acerca de uma indústria 4.0 *smart factory* onde os sistemas estão interligados e são capazes de receber e enviar dados de outros sistemas externos. Os autores também afirmam que existem sistemas/máquinas que são capazes de auxiliar a implementação de rastreabilidade em cada etapa da produção, desde a receção de mercadorias até à monitorização de toda a cadeia de produção. Hoje em dia, com a globalização das cadeias de valor, a necessidade da rastreabilidade tornou-se cada vez mais importante, permitindo seguir um produto ao longo da sua cadeia de valor [9].

Na cadeia de valor de PE também é necessário obter informação sobre os produtos e os seus componentes, garantindo assim a sua autenticidade para evitar falsificações. A rastreabilidade também pode ser vista como um passo para maior transparência nas cadeias de valor [9].

Embora as plataformas de rastreabilidade tragam diversos benefícios para uma cadeia de valor, os custos também são fatores determinantes, uma vez que coletar dados de todas as etapas do ciclo de vida de um produto é bastante dispendioso.

Os autores Li et al. apresentaram um caso de estudo dos resíduos de televisões LCD, para propor um modelo conceptual de dados para uma gestão mais sustentável dos resíduos eletrónicos. O modelo conceptual proposto permite registar dados acerca de um determinado produto e a informação relativa a vários processos da cadeia de valor (informações do fabricante, dados de rastreamento, informações legais, informações económicas e ecológicas, etc.).

No artigo [21], os autores apresentam um sistema na nuvem para gerir resíduos elétricos através de uma aplicação web (utilizando uma base de dados centralizada para gerir os dados), que permite rastrear um PE ao longo do seu ciclo de vida. Esta plataforma suporta diversas funcionalidades, tais como registar um novo produto (seja um produto intermédio ou um produto final), procurar por produtos, gerir um produto (editar, adicionar componentes, trocar um determinado componente, entre outros), desmantelar um produto, etc. Em cada fase do ciclo de vida do produto, o lote do mesmo pode ser lido automaticamente por uma etiqueta IoT, permitindo que a informação relativa a esse produto seja atualizada de forma automatizada na base de dados.

Os autores Dasaklis et al. propuseram uma plataforma de rastreabilidade e auditabilidade para atividades de logística reversa de EEE, com um foco especial em *smartphones*. A plataforma assente num tipo de *blockchain* privada, fornece uma implementação funcional de vários *smart contracts*. Esta solução oferece automação na recolha, manipulação, análise de dados e eventos de rastreabilidade nos processos de logística reversa. A BCT não é a mais adequada para armazenar uma grande quantidade de dados, com isto em vista o autor utilizou uma tecnologia alternativa de armazenamento de dados, o InterPlanetary File System (IPFS), para guardar dados críticos fora da *blockchain*, de forma a melhorar a escalabilidade e integridade dos mesmos. Cada participante armazena no IPFS todos os dados importantes das operações da cadeia de valor numa 'tabela de conteúdos' e, em seguida, as *hashes* de todos os registos individuais são armazenados na *blockchain*. Assim, em vez de guardar todos os dados diretamente na *blockchain*, são apenas guardadas as *hashes* desses mesmos dados [22].

No artigo [23] foi apresentada uma prova de conceito genérica de rastreabilidade baseada em *blockchain* para várias cadeias de valor. Nesta solução, para demonstrar o

potencial da BCT no contexto de rastreabilidade, foi desenvolvida uma solução através da plataforma *Microsoft Blockchain Workbench*. Os autores também exploram diversas aplicações de plataformas de rastreamento em diferentes cadeias de valor, assim como também descrevem várias ferramentas IoT utilizadas para rastreabilidade, como etiquetas Identificação por Radiofrequência (RFID), códigos QR, códigos de barras, Near Field Communications (NFC), GPS, entre outras.

Outra proposta foi feita no artigo [24], onde os autores apresentaram uma a combinação da tecnologia IoT e a BCT para melhorar a circularização na cadeia de valor de EEE. Estas tecnologias permitem a criação de cadeias de valor rastreáveis, que como consequência pode ajudar a prologar a vida útil de um produto, manter os materiais na cadeia de valor por um período de tempo maior, criando assim maior eficiência de recursos. Como escolha da sua prova de conceito, o autor procurou uma *blockchain* que permita vários tipos de permissões, visto que assim é possível introduzir mecanismos de controlo de acesso, o que acaba por ser importante para aceitar transações de utilizadores com diferentes tipos de permissões. Entre várias soluções disponíveis, os autores selecionaram o *Hyperledger Fabric*, uma vez que permite a criação de uma *blockchain* com vários canais (podem ter diferentes tipos de permissões) no mesmo nó. Em conjunto com os benefícios que a implementação da BCT traz, o uso de etiquetas IoT também podem ajudar neste tipo de sistemas de rastreabilidade, visto que é possível ler dados de um produto, como o seu Identifier (ID), disponibilidade, localização, meta-data, entre outros dados, de forma automatizada [24].

Com uma proposta diferente de como utilizar a BCT num sistema de rastreabilidade, os autores Kuhn et al. apresentaram uma solução de rastreabilidade no setor de produção de sistemas elétricos para automóveis que é assente no uso de BCT junto de etiquetas IoT. O processo de fabricação deste tipo de produtos é complexo, chegando a envolver milhares de componentes que precisam de ser geridos por diferentes tipos de sistemas, como ERP ou outras aplicações. A importância da rastreabilidade neste setor vem a par com a condução autónoma, especialmente com produtos críticos para a segurança. A rastreabilidade pode ajudar a identificar os componentes desses produtos no caso de falhas ou problemas no processo de produção. Os autores para esta solução decidiram utilizar a *blockchain Ethereum*, no entanto, em vez de utilizar uma solução de blockchain convencional (como a integração de meta-dados do produto como dados de um Tx na *blockchain*), eles propuseram uma abordagem utilizando *tokens ERC 1155*[1] (*tokens* nativos da *blockchain Ethereum*). O padrão *ERC 1155* introduziu a capacidade de gerir diferentes tipos de objetos virtuais utilizando o mesmo *smart contract*. Visto que vários produtos deste

[1] https://eips.ethereum.org/EIPS/eip-1155

sector são únicos, será necessário criar diferentes *tokens* para vincular cada um ao seu determinado produto (através do ID do produto). Assim, o uso do padrão *ERC 1155* neste caso de uso permite a criação de vários *tokens*, necessários para representar diferente produtos, sendo também possível vincular ao *token* um Uniform Resource Identifier (URI) que pode contar mais informações acerca do produto.

Na indústria de EEE existe a preocupação relativamente à introdução de peças eletrónicas falsificadas. Segundo DiMase et al., a rastreabilidade não garante a qualidade dos componentes quando estes saem da cadeia de valor, independentemente da eficácia e dos procedimentos implementados. Embora seja quase impossível evitar completamente que componentes falsificados entrem na cadeia de valor, é possível reduzir esta ocorrência implementando "políticas baseadas em risco", e comprando apenas materiais ou produtos de fornecedores registados e autorizados.

3.4 Tecnologia Blockchain

Ao contrário dos tradicionais sistemas de gestão de base de dados centralizados, onde todos os nós (*nodes*) estão conectados a um único ponto central, uma *blockchain* é um sistema distribuído composto por vários nós, que trabalham em conjunto entre si. Cada nó está de forma indireta conectado a outro, no entanto, nenhum deles está diretamente conectado a todos os outros presentes na *blockchain*. Esta base de dados distribuída permite aos seus participantes armazenar e compartilhar dados em tempo real, de forma segura e transparente [27].

Os primeiros passos desta tecnologia foram dados em 1990, por um grupo de investigadores [28]. Originalmente, o propósito da mesma era carimbar a data e hora dos diferentes tipos de documentos digitais, tais como texto, áudio, fotos e vídeos, permitindo certificar a data em que um documento foi criado ou modificado de forma imutável. Após uns anos, os mesmo autores introduziram o conceito de árvores de dispersão (*Merkle Trees*), possibilitando o aumento de informação armazenada num único bloco [29].

Foi apenas em 2008 que a primeira *blockchain* foi concebida para criar um sistema de pagamento digital, sem nenhum tipo de entidade intermediária (bancos ou outros serviços centralizados). Esta tecnologia, baseada num sistema Peer-to-Peer (P2P), permitiu que quaisquer utilizadores efetuassem transações entre eles de forma direta e imutável [2].

Em 2014, [30] apresentou um novo tipo de *blockchain*, já que a *Bitcoin* demonstrou várias limitações ao nível de programabilidade dentro da sua infraestrutura. *Ethereum* é uma *blockchain* desenvolvida com uma linguagem de programação integrada, oferecendo uma camada que permite o desenvolvimento de programas inteligentes (*smart contracts*)

que apenas são executados quando um determinado número de regras são compridas. Isto possibilitou o desenvolvimento de diversos tipos de aplicações descentralizadas, mantendo todas as características que uma *blockchain* fornece [30].

Entretanto, até aos dias de hoje, inúmeras *blockchains* surgiram focadas em diferentes tipos de soluções, por exemplo, *IBM Blockchain, Hyperledger Sawtooth, Ethereum, Hyperledger Fabric & Quorum* agora consideradas *Blockchain 2.0* [31]. Estas tecnologias vieram introduzir novas capacidades relativamente às iterações anteriores, no entanto, com a rápida evolução desta área, o conceito de *blockchain 3.0* possibilita a utilização desta tecnologia em diversos ramos, tais como arte, saúde, ciência, gestão de cadeias de valor, etc.

A *blockchain* é uma tecnologia idealizada para armazenar e registar dados de forma transparente, imutável e viável. Para tal, cada bloco armazena um conjunto de dados ou transações, enquanto que a *ledger* conecta todos os blocos em ordem. Assim, a *blockchain* não mantém apenas o conjunto de dados atuais, como também todo o histórico completo de transações efetuadas no sistema [32]. As seguintes subsecções descrevem os quatro principais conceitos e componentes da *blockchain*, ou seja, *distributed ledger tecnology*, tipos de permissão, contratos inteligentes e protocolos de consenso.

3.4.1 *Distributed Ledger Tecnology*

Um dos componentes mais importantes de uma *blockchain* são os blocos, como se pode verificar na figura 3.3, um bloco é composto por pelo seu corpo (*body*) e cabeçalho (*header*). O cabeçalho de um bloco em composto pelos atributos seguintes [33]:

- A versão do bloco indica o conjunto de regras para um determinado bloco.
- A *Root Hash* representa uma *hash* de 256 *bits* da raiz da *Merkle Tree*.
- O campo *Prev Hash* é a *Root Hash* do bloco anterior da cadeia.
- O valor do *timestamp* corresponde à data e hora da criação do bloco;
- O campo *nBits* é o parâmetro de dificuldade da *hash* em formato compacto;
- *Nonce* significa "número utilizado apenas uma única vez", que é um número aleatório associado à *Prev Hash*, *Timestamp* e *Root Hash*, este é utilizado para resolver o desafio matemático para a criação de blocos.

Ao incluir a *Root Hash* do bloco anterior no cabeçalho de um bloco, uma *blockchain* estrutura-se em lista ligada, sendo assim definida a arquitetura da cadeia (*chain*) entre

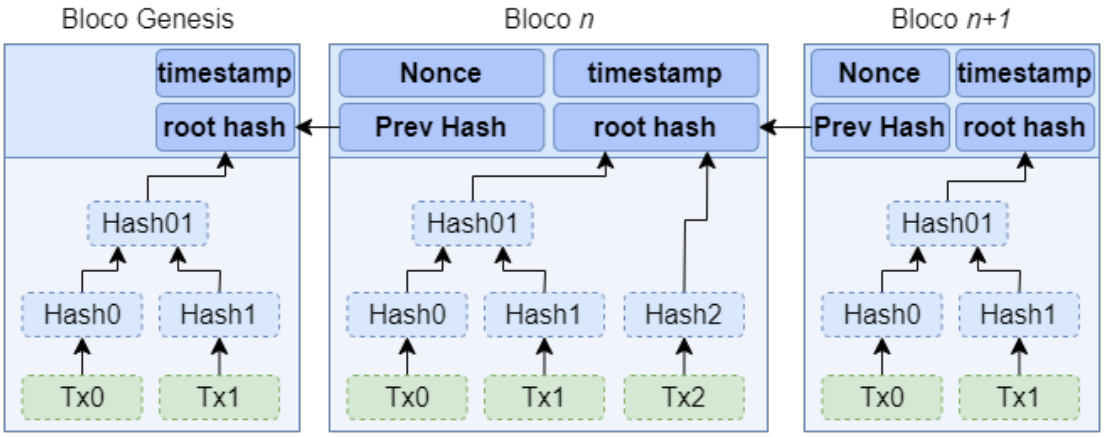

Figura 3.3: Estrutura de um bloco e da *blockchain*.
Fonte: Adaptado de [2]

blocos. Ao primeiro bloco de uma *blockchain* define-se de "bloco génese", sendo que este não possui o atributo *Prev Hash*, já que não existe nenhum bloco anterior ao primeiro. O corpo (*body*) de um bloco regista os dados de transações em forma de *Merkle Tree*. Uma *Merkle Tree* é uma árvore binária onde cada um dos seus nós, pode representar outra árvore binária. Esta estrutura de dados pode ser classificada hierarquicamente em componentes, tais como a "raiz", "nós internos" e "folhas". Visto que a *hash* da raiz está localizada no cabeçalho de um bloco, o resto dos nós contém os dados que apontam para a *hash* dos seus nós "filho", que por sua vez estão presentes no corpo do bloco [33, 34]. Os 'nós folha' da *Merkle Tree* (representada na figura 3.4) de um bloco possuem informação acerca das transações validadas pela *blockchain*, sendo que não possuem nenhum tipo de "nó pai" e têm um único "nó filho". Por outro lado, os "nós internos" são o resultado da *hash* dos seus dois "nós pai" e o seu respetivo "nó filho", onde o "nó filho" é utilizado na seguinte função *hash*, iterando a *Markle Tree* recursivamente até ao último nó da árvore, o "nó raiz" (nó que não tenha qualquer tipo de "nó filho" e tem os seus dois respetivos nós predecessores) [3]. Esta estrutura de dados foi aplicada na arquitetura de um bloco, visto que este método permite minimizar os custos de armazenamento, já que assim não existe a necessidade de armazenar todos os dados de um bloco, para garantir a integridade dos dados e a validação da *blockchain* [2].

Normalmente, uma Tx é apenas uma troca de bens ou serviços, podendo ser uma transferência monetária ou não. No caso da primeira *blockchain*, a *Bitcoin*, as transações são apenas monetárias, sendo que um determinado valor da criptomoeda nativa da rede é enviada de um determinado endereço (endereço fonte) para outro (endereço destino), ao assinar digitalmente uma *hash* da Tx anterior e da chave pública (*public key*) do endereço

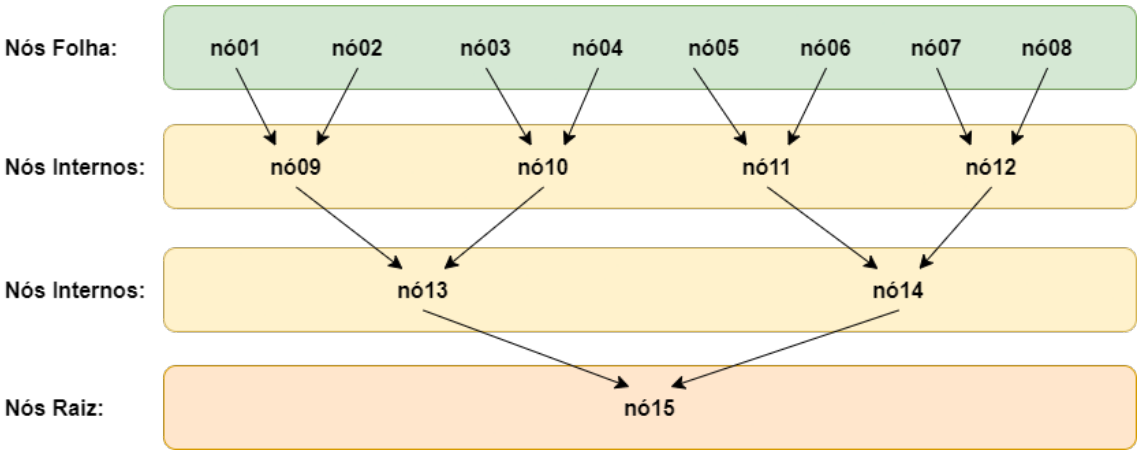

Figura 3.4: Estrutura de uma *Merkle Tree*.
Fonte: Adaptado de [3]

destino (novo proprietário) no fim da Tx [2]. Os dados ao serem armazenados numa Tx são encriptados através de um algoritmo matemático que fornece como saída os dados em forma binária de tamanho fixo, mesmo que apenas seja alterado um caratere nos dados de entrada, a *hash* gerada vai ser diferente, tornando este mecanismo quase indecifrável [35].

Nos dias de hoje, a maioria das soluções *blockchain* permitem que outro tipo de troca de ativos sejam registados numa Tx, já que uma Tx representa uma modificação de dados armazenados, em forma de uma operação. Deste modo, esta tecnologia pode ser aplicada em diversas áreas, dado que uma Tx pode não apenas envolver operações financeiras, como também registos de documentos, registo de sensores, entre outros. Num sistema de rastreamento, as transações podem ser vistas como uma ferramenta para guardar os dados acerca do ciclo de vida de um produto. Numa *blockchain* este tipo de Tx serão geridas através de contratos inteligentes [33], como explicado na secção 3.4.4.

3.4.2 Protocolos de consenso (*Consensus protocols*)

Como foi mencionado anteriormente, numa *blockchain* não existe nenhum tipo de autoridade central que verifique se todos os nós da rede são iguais em toda a rede. Em alternativa, as transações armazenadas numa *ledger* são verificadas por um protocolo de consenso. Estes mecanismos garantem que todos os nós da *blockchain* estão de acordo no mesmo bloco de transações, e que o último bloco foi adicionado corretamente à *ledger*, certificando que os dados estão sincronizados em todos os nós da rede [36].

O algoritmo de consenso Proof-of-Work (PoW) foi adaptado à tecnologia *blockchain* a par com a *Bitcoin*, de forma a garantir o consenso da rede e a veracidade dos blocos

adicionados à *ledger*. Este algoritmo agrupa as Tx na forma de blocos. Os chamados "mineiros" verificam o bloco de transações ao resolver um problema matemático (*Nonce*). O primeiro "mineiro" a resolver primeiro é recompensado (o chamado incentivo por participar no processo de consenso da *blockchain*). As *blockchains* que adotam este protocolo são bastante seguras, já que para atacar a rede é necessário um grande poder de processamento, além disso, sempre que o parâmetro *nBits* sobe, mais difícil é para resolver o problema. Por outro lado, este tipo de consenso tem pouca desempenho, e embora seja algo muito difícil de acontecer, a rede está sujeita a um ataque 51%, etc. [2, 37].

Em alternativa ao protocolo PoW, foi desenvolvido o algoritmo Proof-of-Stake (PoS), que troca o poder computacional do PoW, pelo conceito de *staked coins*. Este modelo baseia-se na quantidade de moedas *staked* num nó da rede para criar o próximo bloco da *blockchain*. Tendo isto em conta, quanto mais moedas um nó tem, maior é a probabilidade desse nó ser escolhido para adicionar o novo bloco à *blockchain* e receber a respetiva recompensa. Apesar deste algoritmo ser teoricamente mais suscetível a um ataque 51%[37], a consequência da tentativa de efetuar este ataque, traduz-se na perda de todas as moedas *staked* do nó. Esta perda de fundos pelo nó atacante, reflete que embora o protocolo PoS seja mais exposto a este tipo de ataques, as consequências são mais severas [37].

Contudo, existem outros tipos de mecanismos de consenso otimizados para diferentes tipos de *blockchain*. Alguns destes algoritmos mais populares e que de facto foram aplicados em vários projetos *blockchain* são *Proof-of-Elapsed Time* [38], *Proof-of-Authority* [39], *practical Byzantine Fault Tolerance* [39], entre outros [40].

3.4.3 Tipos de permissões

Blockchains podem ter diferentes tipos de propriedades, dependendo principalmente de três tipos [41]:

- *Blockchains* públicas são redes abertas, onde qualquer pessoa pode participar, este tipo de *blockchains* são verdadeiramente descentralizadas, no sentido de que nenhum nó da rede em particular controla uma parte, ou até mesmo toda a rede.
- *Blockchains* privadas colocam determinadas restrições nos papéis dos participantes na rede, sendo que os nós também necessitam de permissão para ingressar e realizar transações.
- Em *blockchains* de consórcio os nós podem ter diferentes níveis de autorização e são normalmente utilizadas em cenários Business to Business (B2B) parcialmente descentralizados, onde os dados podem ser públicos ou restritos.

Quantos mais nós uma rede tem, por norma mais descentralizada esta é, e maior é a garantia de imutabilidade. No entanto, um número maior de nós diminui a eficiência da rede para consenso [36]. *Blockchains* também podem ser categorizadas pelo seu protocolo de consenso, o qual pode permitir ou não a definição de diferentes permissões a diferentes tipos de utilizadores. Por norma, redes públicas são abertas para toda a gente, logo são sem definição de permissões. Qualquer outra *blockchain*, com restrições na participação de um nó, cai no tipo com permissão [41]. Outro aspeto das *blockchains* é o seu critério de disponibilidade para fins de leitura, podendo serem abertas ou fechadas. Assim a tabela 3.1 representa a combinação dos diversos aspetos de uma *blockchain*.

	Tipos de Permissão	**Ler**	**Escrever**	***Commit***
Aberto	Sem permissão pública	Todos	Todos	Todos
	Com permissão pública	Todos	Participantes Autorizados	Todos ou um grupo de participantes
Fechado	Consórcio	Participantes Autorizados	Participantes Autorizados	Todos ou um grupo de participantes
	Com permissão privada	Participantes Privados	Operador da rede	Operador da rede

Tabela 3.1: Permissões e consenso de uma *blockchain*

3.4.4 Contratos Inteligentes (*Smart Contracts*)

Os *smart contracts* são programas digitais que fornecem uma forma automatizada e segura de garantir que as partes presentes nesse contrato cumpram as suas obrigações definidas. Estes são armazenados e executados na *blockchain*, o que significa que são descentralizados e imutáveis. Isto garante que as condições estabelecidas no contrato sejam cumpridas de forma precisa e sem a necessidade de uma parte intermediária. Além disto, os *smart contracts* permitem que as partes acedam a dados do contrato em tempo real. Assim, estes são amplamente utilizados em vários setores, incluindo finanças, plataformas de rastreabilidade, comércio eletrónico e em projetos de criptomoedas [42].

Como um simples exemplo de *smart contract*, representado no excerto de código 1, podemos considerar que a "organização 1" e a "organização 2" são dois participantes numa *blockchain*, onde existe um "lote X" de um "produto Y" que a "organização 1" pode criar e enviar para a "organização 2". Um *smart contract* pode ser publicado na *blockchain* que pode permitir: a) uma função "registaLotProduto" que regista o lote

(identificador do lote e a quantidade) para um determinado produto (o produto apenas contém o seu identificador), caso o produto não esteja registado, é registado tanto o produto como o lote do mesmo, caso o produto já se encontre registado, apenas é registado o lote desse produto; b) uma função "transportLotProduto" que recebe como parâmetro o identificador do lote e envia esse lote da "organização 1" para a "organização 2".

```
contract LoteProduto {
    struct Produto {
        address proprietario;
        string id;
        mapping(string => uint) lotes;
        mapping(string => address) transferenciaLote;
    }
    mapping(string => Produto) produtos;
    address public organizacao1;
    address public organizacao2;

    function registaLotProduto(string memory _idProduto,
        string memory _idLote, uint _quantidade) public {
            Produto storage produto = produtos[_idProduto];
            if (produto.proprietario == address(0)) {
                produto.proprietario = msg.sender;
                produto.id = _idProduto;
            }
            require(produto.proprietario == msg.sender,
                "Não é o proprietário deste produto.");
            produto.lotes[_idLote] = _quantidade;
    }

    function transporteLoteProduto(string memory _idLote) public {
        Produto storage produto = produtos[_idLote];
        require(produto.transferenciaLote[_idLote] == address(0),
            "Este lote já foi transferido.");
        require(produto.proprietario == organizacao1,
            "Não é o proprietário deste produto.");
        produto.transferenciaLote[_idLote] = organizacao2;
        produto.proprietario = organizacao2;
```

```
    }
}
```

Listing 1: Exemplo de um *smart contract*.

Assim, o contrato permite-nos expressar a lógica de negócio em código. Um contrato devidamente bem escrito deve ter em conta todos os possíveis desfechos, por exemplo, na função "transporteLoteProduto": a) a quantidade a transportar não deve ser negativa; b) a organização origem e a organização destino devem ser também verificadas se de facto são essas organizações a fazer a devida Tx; c) entre outro tipo de verificações necessárias para garantir a segurança e viabilidade do *smart contract*. Um contrato é determinístico, ou seja, os mesmos dados de entrada sempre produzem o mesmo tipo de dados de saída. Caso se escreva um *smart contract* não determinista, quando este é chamado será executado por todos os nós da rede e poderá retornar resultados aleatórios diferentes, isto pode levar à rede não chegar a um consenso sobre a execução desse contrato. Numa *blockchain* desenvolvida adequadamente, escrever *smart contracts* não determinísticos é praticamente impossível, já que qualquer tentativa de publicar tal contrato na rede será recusada [43].

3.4.5 *Hyperledger Fabric*

A BCT tem sido utilizada em diferentes domínios, incluindo logística reversa e cadeias de valor [24, 23]. No entanto, foram encontrados poucos estudos acerca do uso de BCT na cadeia de valor de EEE. Esta utilização seria útil e traria muitos benefícios, como conformidade, transparência, imutabilidade dos dados, etc. [44].

Um dos objetivos desta dissertação é identificar qual a *blockchain ledger* mais adequada para desenvolver o protótipo de um sistema de rastreamento na cadeia de valor de EEE. A *Hyperledger*[2] é uma comunidade *open source* que procura o desenvolvimento de *frameworks*, ferramentas e bibliotecas estáveis para implementações de blockchain empresariais. Uma das ferramentas disponibilizadas é o *Hyperledger Fabric*. Uma característica notável desta *framework* é a funcionalidade de "canais" (*channels*), onde os nós, apenas podem compartilhar e obter informações nos canais em que têm permissão. Esta arquitetura permite a partilha de dados entre nós conectados a um determinado canal, protegendo informações privadas de outros nós fora do canal. O uso desta solução *blockchain*, traz várias vantagens, como:

[2] https://www.hyperledger.org/

- Rede com permissões (*permissioned*) – O *Hyperledger Fabric* permite implementar soluções *blockchain* que não são publicamente acessíveis. Diferentes permissões podem ser atribuídas a diferentes participantes, e apenas os participantes autorizados podem aceder à *ledger* e agir consoante as permissões atribuídas.

- Transações confidenciais – Os dados manipulados numa Tx são compartilhados e visíveis apenas entre as entidades envolvidas na Tx.

- Arquitetura acoplável – Os protocolos de consenso da *Hyperledger Fabric* são acopláveis, o que significa que é possível modificar/configurar diferentes implementações de protocolos de consenso.

- Fácil de começar – Esta *framework* suporta diferentes linguagens de programação conhecidas, ao contrário de outras soluções *blockchain*, como *Ethereum* que apenas suporta a linguagem de programação *Solidity* para a execução de *smart contracts* na sua rede.

Os dados numa *blockchain* estão disponíveis de forma transparente para os participantes da rede, devido às características desta tecnologia mencionadas anteriormente. Estas características podem melhorar os processos de rastreamento de um produto de uma determinada cadeia de valor [45]. A BCT permite armazenar dados acerca da origem de um produto e/ou componente, processos, entidades, e todas as transações relacionadas. Estes dados são rastreáveis e verificáveis pelos participantes da rede. Assim, esta tecnologia pode levar à aplicação de critérios de sustentabilidade para materiais, produtos, fornecedores, atividades da cadeia de valor, entre outros aspetos, bem como um design de logística mais sustentável [24].

3.5 Tecnologias *IoT*

Os sistemas de rastreabilidade são cada vez mais procurados em diversos setores industriais, não apenas para atividades logísticas, como também para atividades de produção. Estes sistemas permitem a identificação e o rastreamento de itens ao longo da cadeia de valor, desde o fabricante até ao cliente. As tecnologias IoT oferecem ferramentas que possibilitam a conexão entre o mundo físico e o mundo digital. Estas ferramentas podem ser compreendidas como dispositivos inteligentes com acesso à *web*, que podem recolher facilmente dados do ambiente ao seu redor, ou fornecer de forma fácil e interativa informações acerca de um produto ou material. Ao conectar estes dispositivos a um IoT *gateway*, é possível partilhar os dados arrecadados pelos dispositivos IoT para a *cloud*.

A utilização de dispositivos IoT de baixo custo para rastrear, é uma abordagem comum em aplicações que integrarem tecnologias de identificação automática, as aplicações de rastreabilidade não são exceção [46]. No caso das cadeias de valor de alimentos [47] e farmacêuticos [48], é necessária uma atenção especial em relação à rastreabilidade do produto, já que esta tipologia de produtos requer também a aquisição de diferentes parâmetros ambientais, como temperatura e humidade relativa. Estas variáveis de ambiente convém serem recolhidas não apenas durante a etapa de produção, mas também durante a distribuição, aumentando assim o controlo de segurança e qualidade do produto (por exemplo: em qualquer etapa da cadeia de valor, estes parâmetros ambientais podem ser consultados antes da conclusão de uma Tx).

A introdução de tecnologias de informação revolucionaram a forma com que as empresas gerem os seus processos de fabricação. O uso destas em sistemas de rastreabilidade digital veio oferecer inúmeros benefícios, entre eles, uma maior segurança, precisão, e eficiência dos dados obtidos em todos os processos da cadeia de valor [49]. *Automatic identification (auto-ID)* oferece a capacidade de rastrear objetos ao longo do seu ciclo de vida através da leitura e transferência de dados com baixa ou nenhuma intervenção humana. Embora as tecnologias auto-ID permitam o rastreamento automático, as tecnologias IoT mais comuns em cadeias de valor são o código de barras, RFID, Real-time Locating System (RTLS) e GPS. Estas tecnologias têm características distintas e podem ser adotadas para diferentes fins:

- Código de barras: consiste numa etiqueta impressa com linhas pretas e espaços brancos que podem ser lidos por um *scanner* ótico. Por norma, os códigos de barras são utilizados para facilitar e acelerar a identificação de itens. Atualmente, sugiram novas iterações desta tecnologia, como os códigos QR e *datamatrix*, que possibilitam armazenar mais dados. Por outro lado, este tipo de tecnologia deteriora-se ao longo do tempo e são apenas etiquetas de leitura.

- Um dispositivo RFID é uma etiqueta que contém chips e antenas que permitem responder aos sinais enviados pelos leitores destas etiquetas. esta tecnologia permite com que várias etiquetas sejam lidas simultaneamente, e estas podem ser reescritas e encriptadas para segurança adicional.

- RTLS: fornece informações acerca da localização em tempo real de um determinado produto. Além da localização, este tipo de dispositivo também pode fornecer dados sobre velocidade, temperatura e outras variáveis de ambiente em que o produto se encontra. Esta tecnologia pode ser encontrada em diversas aplicações, como, por exemplo, para o rastreamento de veículos e outros ativos.

- GPS: fornece a capacidade de rastrear remotamente qualquer objeto no mundo em tempo real. Esta tecnologia é uma mais-valia para, por exemplo, rastrear veículos e lotes em atividades logísticas.

Para além das tecnologias de comunicação IoT mencionadas anteriormente, existem outras que podem também ser adotadas para rastreabilidade, como NFC, Low Power Wide Area Network (LPWAN), *SigFox*, *NB-IoT* e Bluetooth, que são discutidas e comparadas em detalhes pelos autores Alves et al.. Por norma, alguns destes dispositivos IoT geram dados complementares (como temperatura, humidade, velocidade, localização, etc.) que podem ser utilizados para melhorar a rastreabilidade de uma determinada cadeia de valor. Além disso, o uso de *smartphones* como leitores móveis facilita o desenvolvimento de aplicações que garantem a integração de serviços de rastreabilidade através de uma arquitetura orientada a serviços.

Uma cadeia de valor é composta por diversas atividades e sub-processos interligados que requerem rastreamento, para ser possível evitar futuros possíveis problemas, tais como problemas operacionais, problemas na qualidade de produto, etc. A falha em alguma dessas atividades pode impactar significativamente a cadeia de valor e levar ao aumento dos custos. Para lidar com essa complexidade, as tecnologias de identificação automática são consideradas mais-valias tecnológicas, que podem otimizar a gestão da cadeia de valor em tempo real, e promover a circularidade do ciclo de vida de um produto. Assim, a adoção de tecnologias de identificação automática no processo de rastreabilidade na cadeia de valor de EEE pode oferecer vários benefícios, incluindo: i) Transparência, permitindo a identificação de problemas; ii) Otimização logística, melhora o transporte por otimização de rotas; iii) Eficiência operacional, visto que as a leitura desta etiquetas são menos suscetíveis a erros do que humanos, resultando numa redução de custos operacionais.

3.6 Discussão

Esta revisão de leitura teve como objetivo examinar a adoção das tecnologias *blockchain* e IoT para impulsionar a implementação da economia circular na cadeia de valor de EEE. O primeiro dos dois conjuntos de questões da pesquisa que queremos responder é:

1) Como os processos de negócios da cadeia de valor EEE podem ser circularizados, para reduzir o seu impacto ambiental global?

O atual LE na indústria de EEE necessita de caminhar para um paradigma mais sustentável, que reduza o uso intensivo de recursos naturais, o consumo de energia e a produção

de resíduos. O modelo de CE aborda essas questões circularizando a cadeia de valor, mantendo os recursos na cadeia de valor o maior tempo possível, ao longo de ciclos de processos CE contínuos. O modelo de negócio circular proposto para a cadeia de valor de EEE, representado na fig. 3.2 (subsecção 3.2), fornece uma visão sobre como a indústria EEE pode ser movida para um modelo de CE.

O modelo proposto tem algumas atividades novas na cadeia de valor, como atividades que permitem a reciclagem e a reutilização de produtos End-of-Life (EoL), proporcionando a possibilidade de prolongar o uso de matérias-primas e componentes eletrónicos na cadeia de valor, reintroduzindo-os novamente na cadeia de valor para a produção de novos produtos. O modelo de CE proposto traz, por um lado, as vantagens de reaproveitar recursos naturais, reciclar alguns componentes de produtos mais antigos, entre outros benefícios. Por outro lado, o modelo coloca alguns desafios a ultrapassar, i) é necessária a participação de vários intervenientes da cadeia de valor; ii) evitar a introdução de produtos falsificados na cadeia de valor; iii) A fase de recuperação de materiais pode ser desafiadora uma vez que lida com lixo eletrónico de diferentes tipos de EEE [7].

As plataformas de rastreabilidade são um requisito essencial no modelo de CE, pois permitem rastrear um produto até aos seus componentes e materiais constituintes, com as atividades que construíram cada um deles. Isto fornece ao consumidor final informações sobre todo o caminho de produção de um produto. Na secção 3.3, algumas plataformas de rastreabilidade na cadeia de valor de EEE foram identificadas e resumidas. Isto levanta a segunda questão da pesquisa a ser respondida neste estudo:

2) Como as tecnologias *Blockchain* e IoT podem ser utilizadas para a rastreabilidade na cadeia de valor de EEE e ajudar na promoção da adoção da CE ?

As tecnologias *Blockchain* e IoT podem alterar a maneira como as empresas operam ao longo das atividades/sub-processos da cadeia de valor, fornecendo a capacidade de rastrear um objeto ao longo da cadeia. Uma *blockchain*, neste contexto, refere-se apenas a uma forma distribuída de registar e compartilhar dados. Conforme mencionado na subsecção 3.4, uma *blockchain* é uma *ledger* distribuída que pode e tem sido usada, em criptomoedas, mas está acima dos problemas éticos, financeiros e outros levantados por estes.

Estas duas tecnologias combinadas podem oferecer automação no registo de dados rastreáveis e na identificação de produtos, componentes e materiais. Estas também podem oferecer facilidade na rastreabilidade de produtos e materiais, imutabilidade de dados, transparência nas atividades da cadeia de valor e no uso dos produtos ao longo da cadeia de valor. Embora ambas as tecnologias forneçam vários benefícios, como os mencionados na secção 3.4, também podem acarretar alguns desafios. Os principais desafios técnicos

levantados pela BCT são [51], [52]:

- Segurança: as *blockchains* públicas (utilizadas em criptomoedas) não permitem que os utilizadores tenham permissões diferentes. Além disso, este tipo de *blockchains* estão normalmente assentes em protocolos de consenso PoS ou PoW, o que pode levar a que estas redes sofram um ataque 51%, no caso desta percentagem ou mais de nós se coordenarem para atacar a rede. Normalmente, as *blockchains* privadas e de consórcio suportam canais privados e utilizadores com determinadas permissões, permitindo que eles desempenhem diferentes papéis na rede. Isto fornece melhores mecanismos de segurança e transações privadas, embora os mecanismos de consenso associados possam ter menor tolerância a ataques.

- Escalabilidade: Particularmente em *blockchains* públicas, este parâmetro pode ser problemático, visto que, embora quantos mais mineiros estejam conectados à rede maior é a descentralização e a segurança, contudo o aumento repentino de transações pode levar ao atraso de concluir transações, reduzindo assim a desempenho da rede, ameaçando a escalabilidade.

- Custo de transações: Este é mais um problema relacionado com *blockchains* públicas, já que estas dependem de recompensar os mineiros pelo seu poder de participação no mecanismo de consenso. Por norma, as *blockchains* públicas são associadas a criptomoedas, que por sua via são utilizadas como método de pagamento da recompensa dos mineiros, aumentando assim os custo de Tx. No caso de *blockchains* privadas esses custos podem ser insignificantes.

- Consumo de energia: Algumas *blockchains*, tais como aquelas que utilizam o protocolo de consenso PoW, são intensivas no uso de computação e, por norma, têm uma grande pegada de carbono devido à energia consumida ao tentar confirmar transações. Embora o protocolo PoW acarrete esse problema, atualmente, já existem outras opções para algoritmos de consenso que não dependem do consumo de energia, como o algoritmo PoS e o Practical Byzantine Fault Tolerance (pBFT), bem como aqueles disponíveis no *Hyperledger Fabric* .

Ao implementar uma solução *blockchain* na cadeia de valor de EEE, certas especificações ou propriedades são mais adequadas. Como todos os participantes da cadeia de valor podem criar um consórcio, a escolha ideal é uma *blockchain* privada [24]. Além disso, apesar das vantagens que oferecem, as tecnologias IoT apresentam dificuldades para sua implementação numa cadeia de valor para garantir a rastreabilidade. Existem vários desafios associados à sua implantação, incluindo aspetos técnicos e não técnicos. Desafios não técnicos compreendem fatores que dificultam a adoção dessas tecnologias [53]:

- Os proprietários de empresa, normalmente, não conseguem acompanhar o surgimento de novas tecnologias e podem hesitar na sua adoção, não apenas devido à ausência de padrões e boas práticas no setor, mas também em relação às incertezas do mercado, que normalmente têm um grande impacto nos primeiros a chegar.

- Os sistemas ERP tradicionais e amplamente utilizados não suportam a integração de novas tecnologias, o que aumenta o custo de adaptação dessas ferramentas e restringem a sua funcionalidade.

- A integração destas tecnologias exige que os operadores da cadeia de valor adquiram novas competências técnicas, e uma compreensão das atividades relacionadas com o negócio da cadeia de valor a nível macro.

Por outro lado, existem alguns desafios técnicos associados à adoção da tecnologia IoT, incluindo:

- Escalabilidade: É um dos principais desafios na implementação de tecnologias IoT com a BCT para rastreabilidade, uma vez que a cadeia de valor de um produto necessita de se manter competitiva através de várias mudanças e melhoria contínua.

- Alcançar a interoperabilidade entre dispositivos heterogéneos dentro da rede é outro desafio técnico significativo. Para promover a integração entre redes IoT tão diversas é essencial padronizar esses processos.

Capítulo 4

Conceptualização da Solução

A rastreabilidade numa cadeia de valor é crucial para o fácil acesso de dados acerca de um determinado produto, ou até para a identificação de algum problema relacionado com um item nessa mesma cadeia, como foi mencionado na secção 3.3. O sector de EEE não é diferente, neste capítulo vão ser apresentadas as várias etapas da modelação do sistema de rastreamento proposto nesta dissertação. Considerando o modelo circular apresentado na fig. 3.2 (secção 3.2), será necessário modelar uma solução que permita o rastreamento de produtos internacionalmente (já que o fabrico deste tipo de produtos envolve várias empresas à volta do mundo). Tendo isto em mente, foi necessário identificar as principais atividades da cadeia de valor EEE cruciais para o rastreio de um produto. As principais atividades identificadas no BPMN representado na fig. 3.2 são:

- Registar - Atividade na plataforma de rastreamento que permite aos participantes registar um lote.
- Produção - Atividade que permite criar um lote (lote de saída) a partir de um ou vários lotes (lotes de entrada).
- Transporte - Atividade que permite a um participante da cadeia de valor registar a saída de *stock* de um lote, com destino a outro participante da cadeia de valor.
- Receção - Permite registar a receção de um lote proveniente de outro participante da cadeia de valor.
- Venda - Permite registar na plataforma a atividade de venda de um determinado produto para um consumidor final.
- Reciclagem - Atividade que ao desmantelar um produto (lote de entrada) cria lotes provenientes do desmantelamento (lotes de saída).

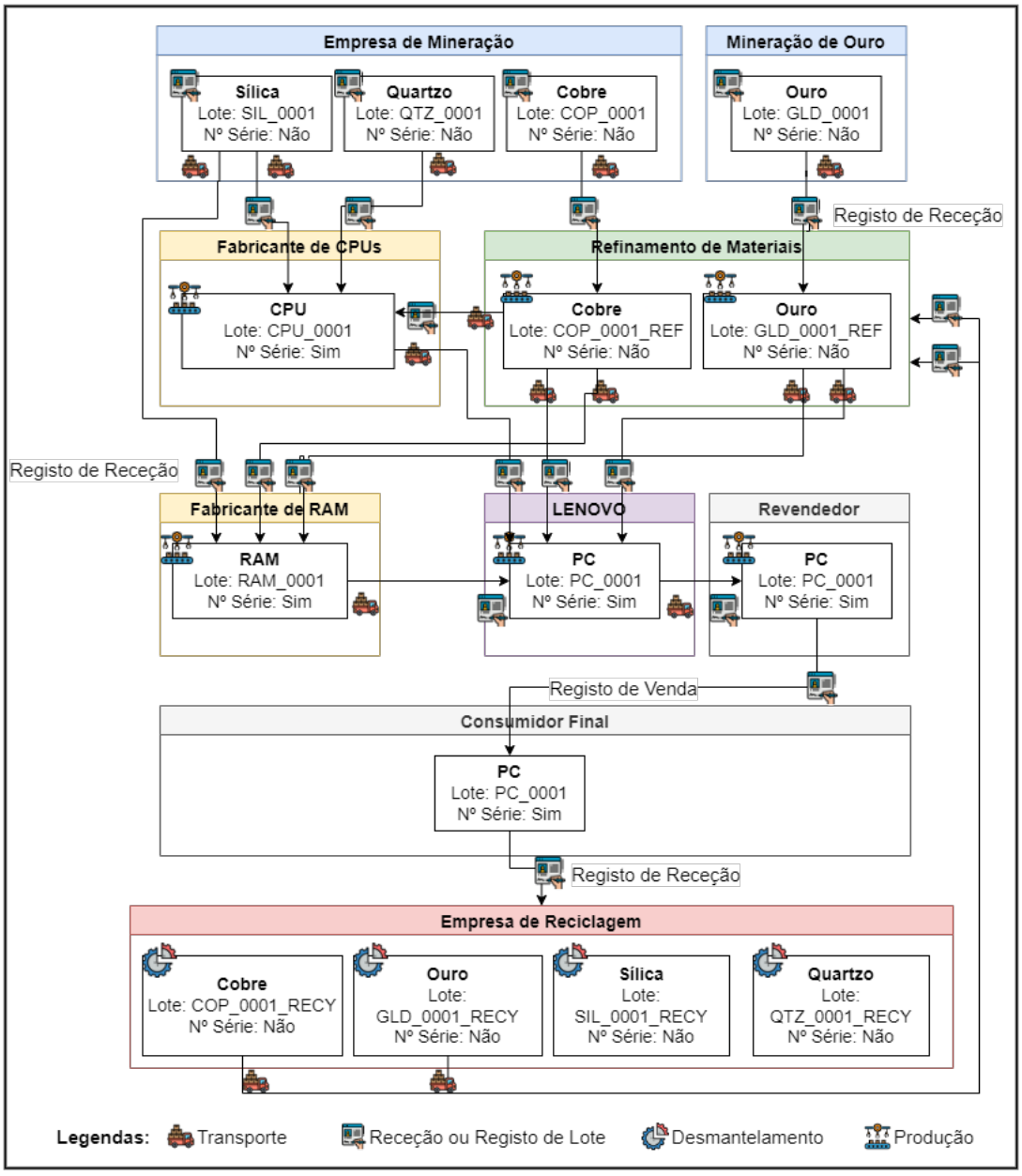

Figura 4.1: Exemplo genérico do fluxo de lotes e atividades a registar na plataforma de rastreabilidade

Após levantar as atividades da cadeia de valor que necessitam de ser registadas na plataforma de rastreabilidade a desenvolver, na fig. 4.1 é apresentado um exemplo genérico de como vai funcionar o fluxo de lotes no protótipo do sistema modelado neste capítulo. Nesta figura é dado o exemplo da produção de um computador (lote: "PC_0001"), desde a extração de minerais, refinação dos mesmos, fabrico de componentes intermédios, até à produção do computador, sendo posteriormente vendido, e no fim descartado, onde vai entrar no processo de reciclagem, de forma reintroduzir as suas matérias-primas de novo na

cadeia de valor. Este processo começa pelo registo de diferentes lotes de matérias-primas por duas "empresas de mineração" (lotes: "SIL_0001", "QTZ_0001", "COP_0001" e "GLD_0001"), a seguir estes lotes vão ser enviados, por uma atividade de transporte, para uma empresa responsável pela sua refinação. No caso da empresa "Refinamento de Materiais", a mesma recebe (atividade de receção) os lotes "COP_0001" e "GLD_0001", que serão introduzidos como lotes de entrada em duas atividades de produção diferentes para produzir (lotes de saída) os lotes "COP_0001_REF" e "GLD_0001_REF". No caso das empresas "Fabricante de CPUs" e "Fabricante de RAM", ambas recebem matérias-primas, ou lotes refinados para a fabricação dos seus respetivos produtos (produtos/componentes intermediários), que a seguir serão enviados para a empresa "Lenovo". É nesta empresa que o produto final é montado ou fabricado, por um processo de fabrico que terá diversos lotes de entrada para produzir o lote de saída "PC_0001". Este produto vai ser distribuído para um determinado revendedor, onde vai ser adquirido pelo consumidor final (registo de venda). Contando que esse consumidor descarte o produto, uma empresa de reciclagem poderá receber esse produto para o reintroduzir da melhor alternativa na cadeia de valor (atividade de desmantelamento). Esta atividade tem em conta o produto a reciclar como lote de entrada ("PC_0001") para separar os componentes ou recursos naturais que possam ser reintroduzidos de novo na cadeia de valor. Assim, considerando este genérico fluxo da cadeia de valor de EEE, o protótipo da plataforma a ser desenvolvida tem como objetivo registar todas a informações acerca dos lotes e as respetivas atividades representadas na fig. 4.1, a fim de ser possível consultar a rastreabilidade de um determinado produto, a qualquer momento.

4.1 Casos de Uso

O objetivo desta secção é apresentar o diagrama de caso de uso para o protótipo proposto, que como foi mencionado na secção anterior, é uma aplicação *web* que permite a múltiplas organizações registar informação de diversas atividades da cadeia de valor de EEE. O diagrama de caso de uso descreve os requisitos funcionais do sistema, incluindo os principais atores e as suas interações com o sistema. Na fig. 4.2 é apresentado o diagrama, onde estão representados os principais atores da aplicação:

- Administrador da Plataforma: Este ator é responsável por gerir e parametrizar todos os aspetos da plataforma, como adicionar organizações à mesma, configurar unidades de medida e unidades de custo, etc.
- Administrador da Organização: Tem a possibilidade de gerir todos os parâmetros da sua organização, utilizadores, papéis dos utilizadores na organização, etc.

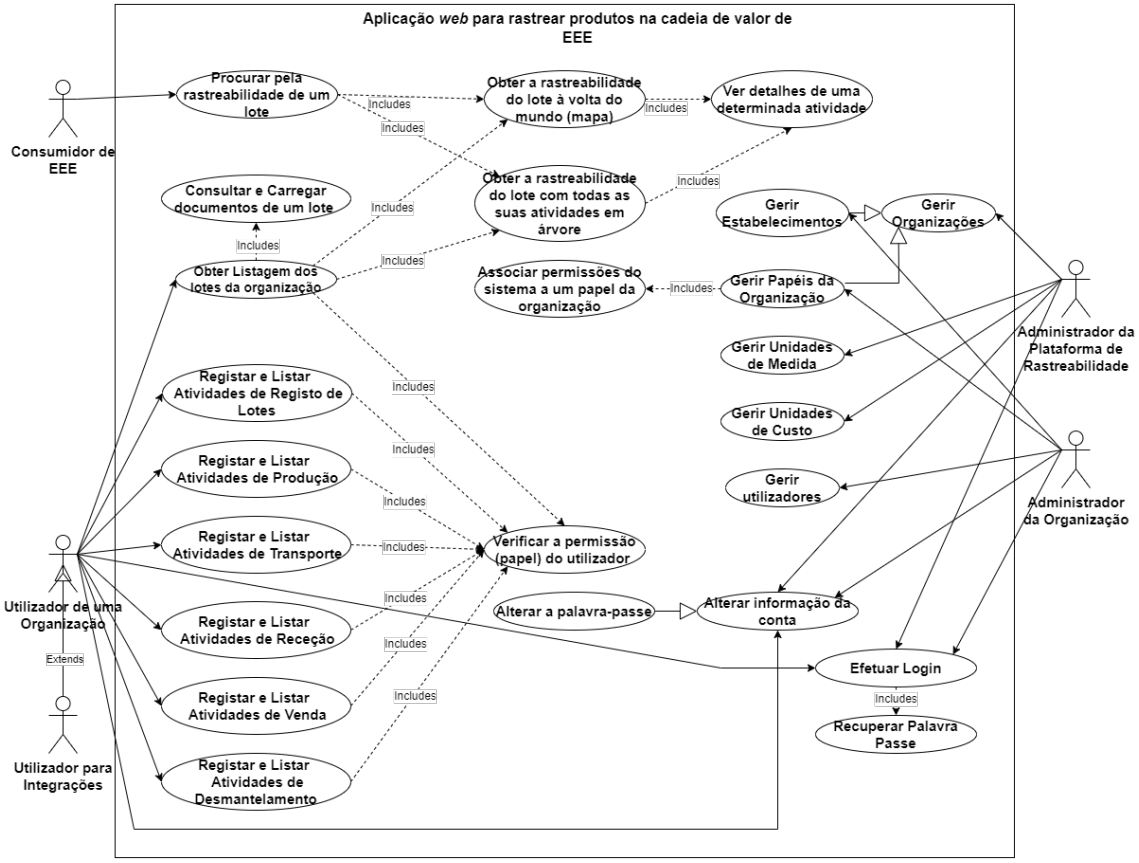

Figura 4.2: Diagrama de caso de uso do sistema proposto

- Utilizador da Organização: Refere-se a qualquer outro tipo de utilizador numa organização, cada utilizador pode ter diferentes papéis dentro da mesma, sendo que a cada papel que o utilizador desempenha tem certas permissões associadas. Assim, é possível criar um papel "Produção"com permissões apenas para gerir atividades de produção, permitindo aos utilizadores com esse papel criar e ver atividades de produção (assim é possível segmentar tipos de utilizadores para diferentes etapas dentro da organização).

- Utilizador para Integração: Este tipo de utilizador tem bastantes semelhanças com o utilizador da organização, no entanto, este tipo de utilizador permite à organização ter um utilizador específico para integração com outros sistemas, como por exemplo ERP e Manufacturing Resource Planning (MRP), também é possível apenas dar certas permissões a este utilizador (por exemplo: é possível criar um utilizador apenas com permissão para criar atividades de transporte).

- Consumidor de EEE: Ator responsável por representar qualquer pessoa que possua um PE registado nesta plataforma, podendo consultar na mesma todos os dados de

rastreio do seu produto.

Casos de uso do administrador da plataforma:

- Gerir Organizações: Permite ao utilizador consultar todas as organizações na plataforma, adicionar, editar ou eliminar uma determinada organização.
- Gerir Estabelecimentos: Este requisito é uma extensão do anterior (Gerir Organizações), permitindo ao utilizador adicionar, editar e eliminar estabelecimentos numa determinada organização.
- Gerir Perfis da Organização: Outro dos requisitos que é uma extensão do requisito "Gerir Organizações", fornece a possibilidade de definir dinamicamente diversos tipos de utilizadores com diferentes permissões na organização.
- Gerir Unidades de Medida: Fornece a possibilidade do Administrador da Plataforma parametrizar a plataforma, inserindo novas unidades de medida, editar ou remover um determinada unidade.
- Gerir Unidades de Custo: Fornece a possibilidade de o administrador da plataforma parametrizar a plataforma, inserindo novas unidades de custo, editar ou remover um determinada unidade.
- Alterar Informações da Conta: Possibilita a alteração de dados da conta do utilizador, incluindo a palavra-passe.

Casos de uso do administrador da organização:

- Gerir Estabelecimentos: Possibilita ao administrador da organização adicionar, editar e eliminar estabelecimentos numa determinada organização.
- Gerir Perfis da Organização: Permite ao utilizador definir dinamicamente diversos tipos de utilizadores com diferentes permissões na organização.
- Alterar Informações da Conta: Possibilita a alteração de dados da conta do utilizador, incluindo a palavra-passe.

Casos de uso do utilizador da organização & utilizador para integrações:

- Registar e Listar Atividades de Registo: Possibilita ao utilizador consultar e adicionar atividades de registo de lotes da sua organização.

- Registar e Listar Atividades de Produção: Possibilita ao utilizador consultar e adicionar atividades de produção na sua organização.
- Registar e Listar Atividades de Transporte: Possibilita ao utilizador consultar e adicionar atividades de transporte na sua organização.
- Registar e Listar Atividades de Receção: Possibilita ao utilizador registar receção de lotes na sua organização.
- Registar e Listar Atividades de Venda: Possibilita ao utilizador consultar e registar vendas efetuada na sua organização.
- Registar e Listar Atividades de Desmantelamento: Possibilita ao utilizador consultar e adicionar atividades de desmantelamento na sua organização.
- Alterar Informações da Conta: Possibilita a alteração de dados da conta do utilizador, incluindo a palavra-passe.

Casos de uso do consumidor de EEE:

- Procura pela rastreabilidade de um lote: Permite a um consumidor verificar a rastreabilidade do seu produto pelo seu identificador do lote. Ao fazer esta busca, o utilizador pode consultar todas as atividades (e detalhes das mesmas) da cadeia de valor por onde o seu produto passou, assim como todas as organizações.

4.1.1 Descrição dos principais casos de uso

Nesta secção são apresentados a descrição dos principais casos de uso desta plataforma de rastreabilidade, tais como o registo e consulta das diversas atividades da cadeia de valor, e a consulta da rastreabilidade de um determinado produto. Casos de uso relativos a configurações da plataforma, como gerir organizações não foram descritos visto que são assentes em consultar, registar, editar e eliminar entidades na base de dados. Assim, ao longo desta secção, é apresentado o fluxo e condicionantes de alguns dos principais casos de uso.

Caso de Uso	Registar e Listar Atividades de Registo
Atores	Utilizador de uma organização
Pré-condições	1.O utilizador estar dentro do sistema com a sua conta. 2.O utilizador ter permissões para registar e ver atividades de registo.
Pós-condições	1.Inserir uma nova atividade e consultar todos as atividades deste tipo.
Fluxo	1.O utilizador seleciona no seu menu a opção "Atividades de Registo". 2.O sistema redireciona o utilizador para a página. 3.O sistema retorna todas as atividades de registo a quais o utilizador tem permissão consultar. 4.O utilizador vê a listagem das atividades e pretende adicionar uma nova atividade. 5.O utilizador pressiona o botão "Adicionar". 6.O sistema apresenta o formulário para criar uma nova atividade ao utilizador. 7.O utilizador preenche o formulário devidamente e submete-o. 8.O sistema valida os dados preenchidos e cria uma nova atividade. 9.O sistema notifica o utilizador que a atividade foi criada com sucesso e redireciona o utilizador para a página da listagem das atividades. 10.O utilizador consulta a listagem das atividades com a nova atividade já adicionada.
Extensões	3a. Consultar detalhes da atividade; 3a.1.O utilizador clica numa determinada atividade que pretende ver o seu detalhe. 3a.2.O sistema verifica se o utilizador tem permissão para consultar os detalhes daquela atividade. 3a.3.O sistema devolve os dados dos detalhes da atividade e apresenta os dados ao utilizador. 7a. Formulário mal preenchido; 7a.1.O utilizador submete o formulário mal preenchido. 7a.2.O sistema verifica que pelo menos uma das condições do formulário não foi cumprida ou se o identificador do lote inserido já existe dentro da sua organização. 7a.3.O sistema notifica o utilizador que não foi possível criar uma nova atividade.

Tabela 4.1: Descrição do caso de uso "Registar e Listar Atividades de Registo".

Caso de Uso	Registar e Listar Atividades de Produção
Atores	Utilizador de uma organização
Casos de uso relacionados	1. Registar e Listar Atividades de Desmantelamento
Pré-condições	1.O utilizador estar dentro do sistema com a sua conta. 2.O utilizador ter permissões para registar e ver atividades de produção.
Pós-condições	1.Inserir uma nova atividade e consultar todos as atividades deste tipo. 2.Criação de um novo lote, proveniente de um conjunto de lotes.
Fluxo	1.O utilizador seleciona no seu menu a opção "Atividades de Produção". 2.O sistema redireciona o utilizador para a página. 3.O sistema retorna todas as atividades de produção, a quais o utilizador tem permissão consultar. 4.O utilizador vê a listagem das atividades e pretende adicionar uma nova atividade. 5.O utilizador pressiona o botão "Adicionar". 6.O sistema apresenta ao utilizador o formulário para criar uma nova atividade. 7.O utilizador seleciona os lotes de entrada para a processo de produção. 8.O utilizador preenche os dados para os lotes de saída. 8.O sistema valida os dados preenchidos e cria uma nova atividade de produção, assim como um novo lote. 9.O sistema notifica o utilizador que a atividade foi criada com sucesso e redireciona o utilizador para a página da listagem das atividades. 10.O utilizador consulta a listagem das atividades com a nova atividade já adicionada.
Extensões	3a. Consultar detalhes da atividade. 3a.1.O utilizador clica numa determinada atividade que pretende ver o seu detalhe. 3a.2.O sistema verifica se o utilizador tem permissão para consultar os detalhes daquela atividade. 3a.3.O sistema devolve os dados dos detalhes da atividade e apresenta os dados ao utilizador. 7a. Dados dos lotes de entrada incorretos. 7a.1.O sistema verifica se o identificador dos lotes de entrada existem no sistema, e se têm em stock a quantidade necessário para o processo de produção. 7a.3.O sistema notifica o utilizador que não foi possível criar uma nova atividade de produção 8a.Identificador de lote de saída já existente. 8a.1.O sistema verifica se o identificador dos lotes de saída já se encontram registados no sistema. 8a.1.O sistema notifica o utilizador que não foi possível criar uma nova atividade de produção

Tabela 4.2: Descrição do caso de uso "Registar e Listar Atividades de Produção".

Caso de Uso	Registar e Listar Atividades de Transporte
Atores	Utilizador de uma organização
Casos de uso relacionados	1. Registar e Listar Atividades de Receção
Pré-condições	1.O utilizador estar dentro do sistema com a sua conta. 2.O utilizador ter permissões para registar e ver atividades de transporte.
Pós-condições	1.Inserir uma nova atividade e consultar todos as atividades deste tipo. 2.Remoção da quantidade em stock do lote expedido.
Fluxo	1.O utilizador seleciona no seu menu a opção "Atividades de Transporte". 2.O sistema redireciona o utilizador para a página. 3.O sistema retorna todas as atividades de transporte, que o utilizador tem permissão de consultar. 4.O utilizador consulta a listagem das atividades, e pretende adicionar uma nova atividade. 5.O utilizador pressiona o botão "Adicionar". 6.O sistema apresenta ao utilizador o formulário para criar uma nova atividade. 7.O utilizador seleciona o lote que pretende expedir no processo de transporte. 8.O utilizador seleciona uma organização como destino final. 8.O sistema valida os dados preenchidos e cria uma nova atividade de transporte. 9.O sistema notifica o utilizador que a atividade foi criada com sucesso e redireciona o utilizador para a página da listagem das atividades. 10.O utilizador consulta a listagem das atividades com a nova atividade já adicionada.
Extensões	3a. Consultar detalhes da atividade. 3a.1.O utilizador clica numa determinada atividade que pretende ver o seu detalhe. 3a.2.O sistema verifica se o utilizador tem permissão para consultar os detalhes daquela atividade. 3a.3.O sistema devolve os dados dos detalhes da atividade e apresenta os dados ao utilizador. 7a. Dados do lote de entrada expedido incorretos. 7a.1.O sistema verifica se o identificador do lote de entrada existe no sistema, e se têm em stock a quantidade necessário para o processo de transporte. 7a.3.O sistema notifica o utilizador que não foi possível criar uma nova atividade de produção.

Tabela 4.3: Descrição do caso de uso "Registar e Listar Atividades de Transporte".

Caso de Uso	Registar e Listar Atividades de Receção
Atores	Utilizador de uma organização
Casos de uso relacionados	1. Registar e Listar Atividades de Transporte
Pré-condições	1.O utilizador estar dentro do sistema com a sua conta. 2.O utilizador ter permissões para registar e ver atividades de receção.
Pós-condições	1.Inserir uma nova atividade e consultar todos as atividades deste tipo. 2.A organização recebe um novo lote proveniente de outra organização ou atualiza o stock de um lote já existente.
Fluxo	1.O utilizador seleciona no seu menu a opção "Atividades de Receção". 2.O sistema redireciona o utilizador para a página. 3.O sistema retorna todas as atividades de transporte que a organização tem previsto receber. 4.O utilizador seleciona a atividade de transporte que corresponde à atividade de receção em curso. 5.O sistema apresenta ao utilizador o formulário de receção de lote. 6.O utilizador pode alterar ou complementar informações já pré-definidas do lote a receber. 7.O utilizador submete o formulário de receção de lote. 8.O sistema valida os dados preenchidos e cria uma nova atividade de receção. 9.O sistema altera o estado da atividade de transporte correspondente para "concluído". 10.O sistema notifica o utilizador que a atividade foi registada com sucesso e redireciona o utilizador para a página da listagem das atividades de receção já executadas. 11.O utilizador consulta a listagem das atividades de receção com a nova atividade já adicionada.
Extensões	7a. Dados do lote de receção já existentes no sistema. 7a.1.O sistema verifica se o identificador do lote de entrada existe no sistema, caso já exista, em vez de criar um novo lote, adiciona a quantidade recebida ao lote já existente. 11a. Consultar detalhes da atividade. 11a.1.O utilizador clica numa determinada atividade que pretende ver o seu detalhe. 11a.2.O sistema verifica se o utilizador tem permissão para consultar os detalhes daquela atividade. 11a.3.O sistema devolve os dados dos detalhes da atividade e apresenta os dados ao utilizador.

Tabela 4.4: Descrição do caso de uso "Registar e Listar Atividades de Receção".

Caso de Uso	Obter Listagem dos Lotes da Organização
Atores	Utilizador de uma organização
Casos de uso relacionados	1.Obter a rastreabilidade do lote à volta do mundo (mapa). 2.Obter a rastreabilidade do lote com todas as suas atividades em árvore. 3.Consultar e carregar documentos de um determinado lote.
Pré-condições	1.O utilizador estar dentro do sistema com a sua conta. 2.O utilizador ter permissões para consultar os lotes da organização.
Pós-condições	1.Consultar a informação de todos os lotes da organização. 2.Consultar detalhes de um determinado lote. 3.Consultar a rastreabilidade de um determinado lote. 4.Consultar documentos de um determinado lote.
Fluxo	1.O utilizador seleciona no seu menu a opção "Lotes". 2.O sistema redireciona o utilizador para a página de consulta de lotes 3.O sistema retorna os lotes da organização do utilizador em forma de tabela. 4.O utilizador consulta alguns dos dados do lote diretamente na tabela.
Extensões	4a. Consultar detalhes do lote. 4a.1.O utilizador clica no botão "ver detalhes" na linha da tabela de um determinado lote para consultar informação detalhada acerca do mesmo. 4a.2.O sistema verifica se o utilizador tem permissão para consultar os detalhes daquele lote. 4a.3.O sistema devolve os dados dos detalhes do lote e apresenta os dados ao utilizador. 4b. Consultar documentos de um lote. 4b.1. O utilizador clica no botão "ver documentos" na linha da tabela de um determinado lote para consultar os documentos acerca do mesmo. 4c. Consultar documentos de um lote. 4c.1. O utilizador clica num botão "ver documentos" na linha da tabela de um determinado lote. 4c.2.O utilizador carrega o documento no espaço destinado a esse processo. 4c.3.O sistema guarda o documento inserido no lote escolhido. 4d. Consultar rastreabilidade do lote. 4d.1.O utilizador clica no botão "ver documentos" na linha da tabela de um determinado lote para consultar a rastreabilidade acerca do mesmo. 4d.2.O sistema retorna os dados de rastreabilidade do lote pretendido. 4d.3.O utilizador consulta os dados de rastreabilidade do lote.

Tabela 4.5: Descrição do caso de uso "Obter Listagem dos Lotes da Organização".

Caso de Uso	Procurar pela rastreabilidade de um lote
Atores	Consumidor de EEE
Casos de Uso Relacionados	1.Obter a rastreabilidade do lote à volta do mundo (mapa). 2.Obter a rastreabilidade do lote com todas as suas atividades em árvore.
Pré-condições	O autor ter o identificador de lote do seu produto.
Pós-condições	A rastreabilidade do seu produto é representada: 1.Num mapa onde foram executadas todas as atividades da cadeia de valor. 2.Num diagrama de árvore todas as atividades executadas.
Fluxo	1.O utilizador na página de *login* clica na opção "consultar rastreabilidade". 2.O sistema redireciona o utilizador para uma página, onde pode ser introduzido o identificador do seu produto. 3.O utilizador insere o identificador do produto e clica em procurar. 4.O sistema verifica se o identificador existe e devolve os dados de rastreio do produto. 5. O sistema apresenta esses dados ao utilizador através de uma interface gráfica.
Extensões	4a. Identificador do produto inválido. 4a.1 O sistema notifica que o identificador inserido é inválido.

Tabela 4.6: Descrição do caso de uso “Procurar pela rastreabilidade de um lote”.

4.2 Modelo de Domínio

O Modelo de Domínio é uma representação visual das principais entidades, ou classes conceptuais (ideias, coisas ou objetos), do domínio do problema em questão. Para desenvolver este diagrama, tal como do diagrama de casos de uso, na secção anterior, foi utilizada a notação Unified Modeling Language (UML), uma linguagem utilizada para modelar diferentes aspetos de um sistema de *software* de uma forma orientada por objetos [54]). Na figura 4.3 é apresentado o diagrama de domínio para a plataforma proposta. A verde estão representadas as classes que serão geridas pela *blockchain* e a branco estão representadas as classes geridas pela base de dados centralizada. Como se pode comprovar na figura 4.3 foram escolhidos como dados *on-chain* (dados guardados na *blockchain*) a informação do lote (classe "Lot") e todas as atividades associadas a um lote na cadeia de valor. As restantes classes são guardadas *off-chain*, já que são classes relativas a utilizadores e parâmetros da plataforma, como organizações, permissões, etc.

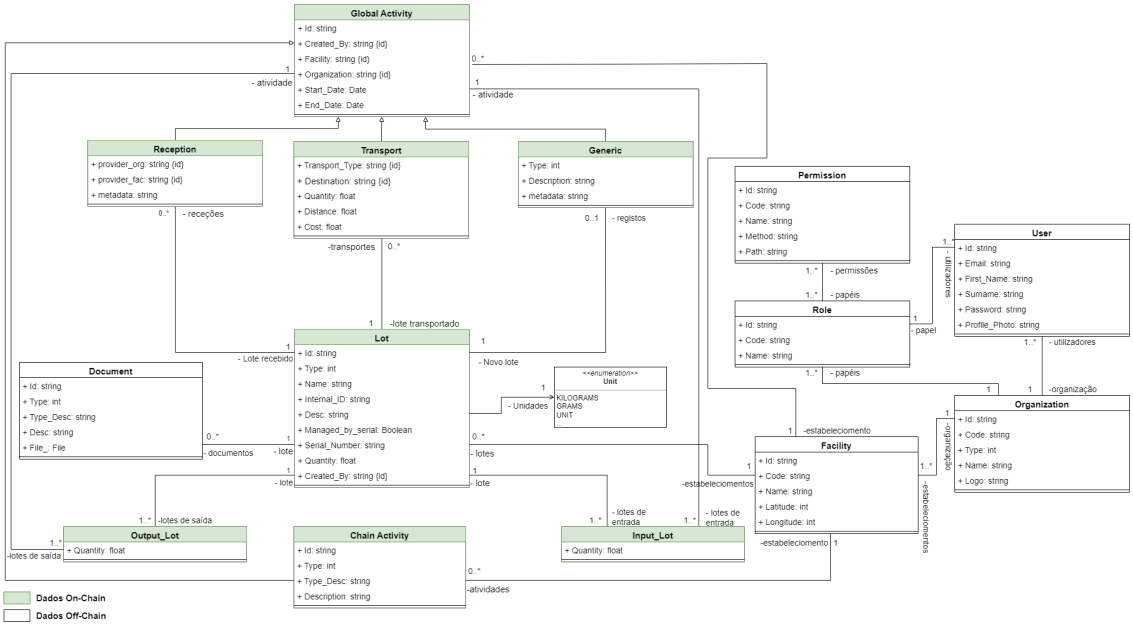

Figura 4.3: Modelo de domínio do sistema proposto.

Na tabela 4.7 estão representadas as várias classes presentes na fig. 4.3, assim como a descrição e propósito das mesmas, a sua herança e as ligações entre elas. Apesar de na tabela estarem representadas algumas das características das classes, existem alguns pormenores importantes a mencionar no caso de algumas delas. No caso da classe “Lot” é importante mencionar que a mesma se refere a um determinado lote de um produto, sendo que esse lote através da variável “Managed_by_serial” tanto pode ser número de série, como não (o lote com número de série apenas pode ter uma única unidade associada

à sua referência de lote, sendo que neste caso o ID de lote corresponde ao seu número de série). Outra classe particular é a classe "Chain Activity", esta classe representa, neste momento, as atividades de produção e as atividades de reciclagem. No entanto, através da variável "Type" é possível introduzir nesta classes outro tipo de atividade da cadeia de valor, desde que a mesma possua lotes de entrada ("Input_Lot") e lotes de saída ("Output_Lot"). Por fim, outra atividade importante de mencionar é a atividade "Generic", já que a mesma corresponde a atividades mais genéricas da cadeia de valor. Nesta classe também é possível distinguir os diferentes tipos de atividade através da variável "Type", tendo como um exemplo de atividade guardada nesta classe, a atividade de registo de um lote representada na tabela 4.1.

Classe	**Descrição**	**Off-Chain**	**On-Chain**	**Herança (Filhos)**	**Herança (Pai)**	**Relacionamentos**
Global Activity	Atividade Global do sistema que representa todas as outras atividades são filhas da mesma.	N\A	✓	Reception Transport Generic Chain Activity	N\A	List<Input_Lot> List<Output_Lot>
Transport	Classe que representa as atividades de transporte do sistema.	N\A	✓	N\A	Global Activity	Lot
Reception	Classe que representa as atividades de receção do sistema.	N\A	✓	N\A	Global Activity	Lot
Generic	Classe que representa as atividades mais genéricas do sistema.	N\A	✓	N\A	Global Activity	Lot
Chain Activity	Classe que representa as atividades de produção e reciclagem do sistema.	N\A	✓	N\A	Global Activity	List<Input_Lot> List<Output_Lot>
Lot	Lote de um produto do sistema	N\A	✓	N\A	N\A	List<Document> Facility Unit
Input_Lot	Lote de entrada de uma atividade	N\A	✓	N\A	N\A	Lot Global Activity
Output_Lot	Lote de saída de uma atividade	N\A	✓	N\A	N\A	Lot Global Activity
Document	Documentos de um lote	✓	N\A	N\A	N\A	Lot
Facility	Estabelecimentos de uma organização.	✓	N\A	N\A	N\A	Organization
Organization	Organização do sistema	✓	N\A	N\A	N\A	List<Facility> List<User> List<User>
Role	Papel (role) de uma organização	✓	N\A	N\A	N\A	List<Permission> Organization
Permission	Permissão para uma funcionalidade do sistema	✓	N\A	N\A	N\A	List<Role>
User	Utilizador do sistema	✓	N\A	N\A	N\A	List<Role> Organization

Tabela 4.7: Representação breve das classes do diagrama de domínio presente na fig. 4.3

4.3 Diagrama de Sequência de uma Atividade de Produção

Neste subcapítulo, é descrito o diagrama de sequência do caso de uso "Registar e Listar Atividades de Produção" (descrito na tabela 4.2), demonstrando as interações com os diversos componentes do sistema, componentes estes representados na figura 4.5 (secção 4.2). O diagrama de sequência em questão foi escolhido devido a ter o maior nível de complexidade dentro do sistema, já que para esta atividade ser registada tem que passar por diversas validações nos vários componentes.

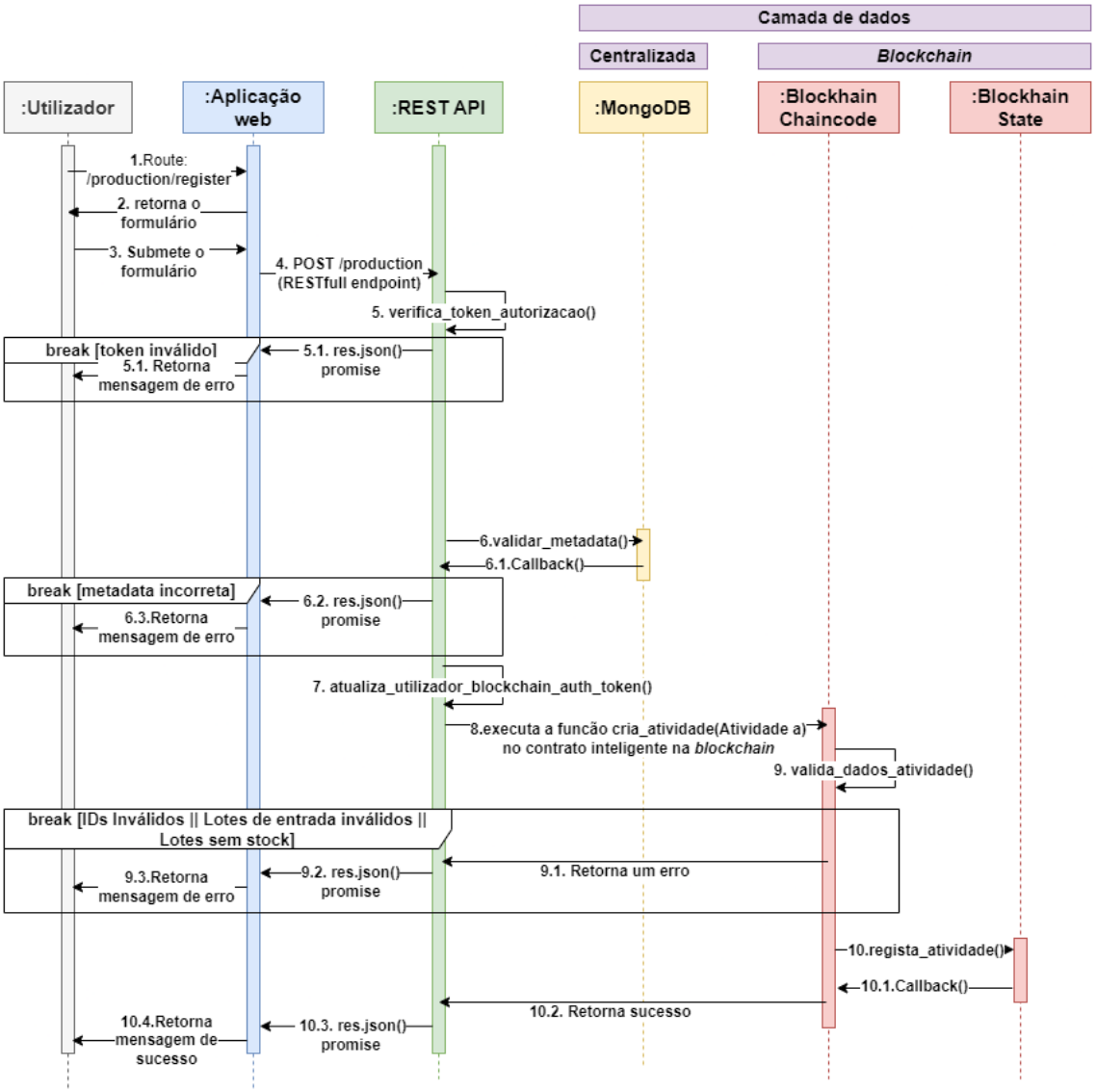

Figura 4.4: Diagrama de sequência do registo de uma atividade de produção

1. O utilizador acede à página de registo da atividade de produção, partindo do pressuposto que o mesmo tem autorização para a aceder, caso não tenha tal permissão vai ser mostrada uma mensagem de erro.

2. Caso o utilizador tenha permissão para registar uma atividade de produção vai ser mostrado o formulário para o registo da mesma.

3. O utilizador preenche o formulário devidamente (existem validações do lado da aplicação cliente para os campos obrigatórios) e submete-o.

4. Após a submissão do formulário, toda a meta-data inserida pelo utilizador é enviada para o serviço de registo da atividade de produção. É neste serviço, pertencente à REST API, que são verificadas todas as condições desta atividade no lado do servidor.

5. Já dentro do serviço de registo da atividade, o mesmo começa por validar se o *token* de autorização (JWT *token*) do utilizador é válido, e se tem permissão para aceder a este serviço.

    5.1. Caso o *token* do utilizador não esteja válido ou o utilizador não tenha permissão para o serviço, é retornado uma mensagem de erro para o utilizador.

6. Valida os meta-dados passados pelo utilizador, como, por exemplo, a validação dos IDs dos lotes de entrada e lotes de saída, se existe quantidade em *stock* dos lotes de entrada, etc.

    6.2. Caso falhe alguma verificação dos lotes passados é enviada uma mensagem de erro para o utilizador.

7. Obtém o *token* de acesso do utilizador à *blockchain*, de forma ao utilizador poder registar a atividade na mesma.

8. É chamado no serviço de criar uma atividade um outro serviço para registar uma atividade de produção na *blockchain*, através de um método de um *smart contract* presente na mesma.

9. Dentro do método do *smart contract* também é verificado se aquela atividade não se encontra já registada, se os lotes de saída não existem no sistema, e se as quantidades dos lotes de entrada são válidos, etc.

    9.1. Caso alguma destas verificações falhar, é mandada uma mensagem de erro para a camada o serviço da REST API, que por sua vez vai mandar a mensagem de erro para o utilizador.

10. Por fim, é registada a atividade de produção na *blockchain*, onde é posteriormente enviada uma mensagem de sucesso ao utilizador.

4.4 Arquitetura do Sistema

Tal como demonstrado na fig. 4.5, a arquitetura do sistema proposto é assente em três camadas. A primeira camada (camada base), denominada camada de dados, é responsável pelo armazenamento de todos os dados necessários para o funcionamento do sistema. Nesta camada, podemos encontrar duas vertentes. A vertente ”dados on-chain“, onde foi escolhido o *Hyperledger Fabric*, descrito na secção 3.4.5, para armazenar todos os dados de rastreabilidade de forma segura e descentralizada. Na outra vertente estão representados os dados “dados off-chain”, onde é utilizado o *MongoDB* (base de dados *NoSQL* orientada a documentos, e altamente escalável, que permite lidar com grandes quantidades de dados)[1] para armazenar todos os outros dados, como informação dos utilizadores, organizações, documentos dos lotes, etc. No que toca aos dados “dados on-chain”, foi escolhida a ferramenta *Fablo*, que permite levantar uma rede *blockchain Hyperledger Fabric* e executar diversos dos seus componentes em contentores *Docker*. Alguns destes componentes são bastante úteis para interagir com a *blockchain*, como o *Hyperledger Explorer* que fornece a possibilidade de analisar ao detalhe dados da *blockchain* (como transações) e o *Fablo REST* que possibilita chamar métodos de *smart contracts* através de normas REST.

A segunda camada (camada intermediária) representada na fig.4.5 é uma API responsável por fornecer todos os tipos de serviços à aplicação web e à aplicação móvel. Ou seja, sempre que um pedido é efetuado à API por uma das aplicações cliente, um determinado serviço é executado, o qual pode interagir com qualquer componente da camada de dados, dependendo do tipo de dados que esse serviço é responsável por gerir. Esta API está assente num servidor *NodeJs* (plataforma de desenvolvimento em *JavaScript* para executar código do lado do servidor)[2] com componentes necessários para o funcionamento desta camada:

- *Express.js*[3] é uma *framework* web para *NodeJs* que permite a criar aplicações web e serviços *RESTful* de forma rápida e eficiente. Esta *framework* fornece um conjunto de recursos que simplificam a criação de rotas, o processamento de requisições e respostas HTTP, a gestão de sessões e *cookies*, entre outros recursos.

[1] https://www.mongodb.com/
[2] https://nodejs.org/pt-br
[3] https://expressjs.com/

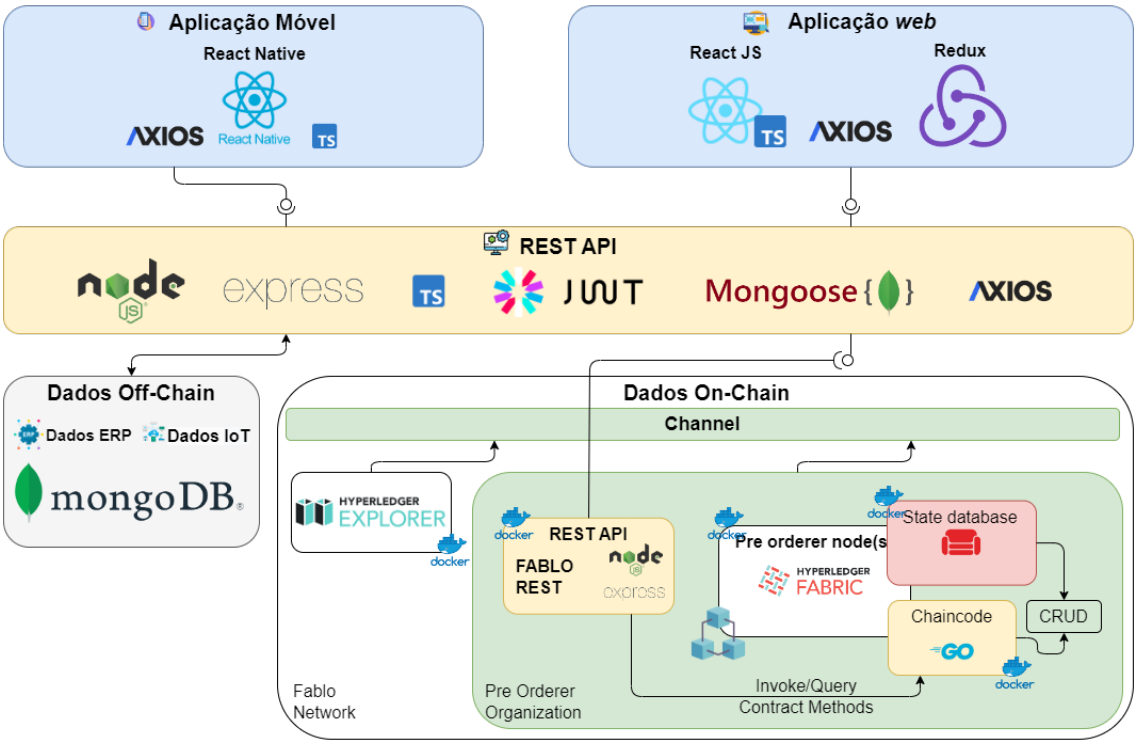

Figura 4.5: Arquitetura proposta do sistema

- JWT[4] é uma técnica de autenticação para API *RESTful*. Esta técnica consiste em criar um *token* encriptado que contem informações sobre o utilizador autenticado, como o seu nome, níveis de acesso e validade do *token*. Este *token* é enviado para a aplicação cliente após uma autenticação bem-sucedida, e é utilizado em todas as chamadas subsequentes do cliente para o servidor. O servidor, por sua vez, desencripta os dados do *token* para verificar a identidade do utilizador e autorizar o acesso às rotas protegidas. Isto torna a autenticação mais eficiente e segura relativamente aos métodos tradicionais, como *cookies* e sessões.

- *Mongoose*[5] é uma biblioteca para *NodeJs* que fornece um mapeamento objeto-documento para a base de dados *MongoDB*. A biblioteca permite definir esquemas para coleções *MongoDB*, fornece métodos de consulta simplificados e funções de validação de documentos. Além disso, o *Mongoose* disponibiliza uma API fácil de utilizador, que simplifica todas a interações com a base de dados, como criar e ler documentos, atualizar e eliminar dados, definir índices e realizar agregações.

- *Axios*[6] é uma biblioteca para *NodeJs* que permite realizar solicitações HTTP para

[4] https://jwt.io/
[5] https://mongoosejs.com/
[6] https://axios-http.com/docs/intro

servidores externos, podendo assim ser utilizada para integrar esta aplicação *NodeJs* com a *Fablo REST* API.

Por fim, a terceira camada são as aplicações do lado do cliente (*front-end*). Num lado temos a aplicação móvel desenvolvida com *React Native* (*framework* de desenvolvimento de aplicativos móveis)[7] que visa tirar partido dos sensores disponíveis num *smartphone* para registar atividade da cadeia de valor mais fácil e interativa. Noutro lado, está a aplicação web que permite fazer toda a configuração da plataforma, como gerir organizações, utilizadores, unidades de medida, etc. Ambas as aplicações consomem os serviços fornecidos pela API da segunda camada. A aplicação do lado do cliente permite que os utilizadores interajam com os dados rastreáveis armazenados na camada de dados, permitindo a visualização do histórico da rastreabilidade dos produtos e a realização de transações na *blockchain*.

[7] https://reactnative.dev/

Capítulo 5

Desenvolvimento do Sistema

Neste capítulo é apresentada a forma como os diferentes componentes foram desenvolvidos, de maneira a criar o primeiro protótipo do sistema, tendo como base toda a modelação do sistema apresentada no capítulo 4, como o diagrama de casos de uso, o diagrama de domínio, e por fim, a própria arquitetura do sistema. Com base nesta informação, todo o sistema e os seus componentes vão ser apresentados e descritos meticulosamente.

A primeira etapa do desenvolvimento do sistema foi a implementação do *backend*. Considerando a fig. 4.5, o *backend* corresponde a tudo que tem a ver com o lado do servidor do sistema, é onde toda a lógica do negócio é implementada, assim como a interação com a base de dados que vai ser utilizada para guardar os parâmetros do sistema, e a *blockchain*, onde se situa o *smart contract* que contém toda a lógica das operações relativas às operações na *blockchain*.

Após o desenvolvimento do *backend* e a execução dos devidos testes ao mesmo, o próximo passo é desenvolver o *frontend*, que corresponde às aplicações que o cliente vai interagir, no caso deste sistema vão existir duas aplicações do lado do cliente, uma aplicação móvel apenas dedicada ao registo de atividades da cadeia de valor, e uma aplicação *web* para a configuração de parâmetro da aplicação.

Tendo em conta esta infraestrutura, o sistema tem os diversos competentes separados, o que traduz na produção de diferentes aplicações (projetos), embora os mesmos tenham interação uns com os outros.

5.1 *Backend*

Ao analisar a arquitetura do sistema, apresentada na fig. 4.5, chega-se à conclusão de que o *backend* deste sistema será composto por três aplicações diferentes. A principal

aplicação será a REST API principal (camada intermediária representada na fig. 4.5), que vai ser responsável por interagir com a camada de dados, e por trocar requisições com o *frontend* e com a *blockchain*. A REST API do *smart contract* (aplicação representada no componente ”dados on-chain“ na fig. 4.5) é uma aplicação com o único propósito de trocar requisições entre o *smart contract* da *blockchain* e a REST API geral, sendo que esta aplicação é criada automaticamente num contentor *docker* sempre que a *blockchain* é levantada ou um determinado *smart contract* é publicado ou atualizado. O *smart contract* vai ser responsável por todas as interações com a *blockchain* (que neste sistema em específico, a maioria das operações serão Create, Retrieve, Update, Delete (CRUD)), e pela lógica dessas operações. No que toca a todos os componentes da *blockchain*, estes vão ser levantados em contentores *docker* para facilitar a execução e a portabilidade da *blockchain* em diferentes ambientes, já a REST API geral terá de ser executada num servidor *nodeJS*, de forma à mesma trabalhar sem qualquer tipo de problemas.

5.1.1 *Blockchain*

Um dos componentes mais importantes deste sistema é a *blockchain*, já que todo o propósito do desenvolvimento deste protótipo é assente nas características e benefícios que esta tecnologia traz (secção 3.4). Na secção 3.4.5 é justificado o porquê de neste projeto ser utilizada a *framework Hyperledger Fabric* para desenvolver uma solução *blockchain* adequada para este sistema. O *Fablo* é uma ferramenta que possibilita gerar *blockchain Hyperledger Fabric* e executar os seus diversos componentes em contentores *docker* (é possível consultar os contentores gerados pela configuração definida para este caso de uso na figura A.4). O Fabric é utilizado como protocolo por ser uma plataforma do tipo ”Consórcio“. O *Fablo* também fornece o poder de implementar ferramentas úteis, como o *Hyperledger Explorer*, que é uma ótima opção para examinar detalhadamente os dados da *ledger* on-chain, e o *Fablo* REST, um servidor que fornece uma simples REST API para facilitar a solicitação aos métodos do *smart contract* do Fabric (na figura A.5 é possível ver todos os contentores criados para fornecer a API a todas as organizações definidas). Para configurar esta ferramenta para levantar a *blockchain* utilizada para este sistema, é necessário a criação e configuração do ficheiro “fabloconfig.json” na raiz do projeto, possibilitando a ferramenta de executar os passos necessários para a criação da rede:

1. Após a criação do ficheiro de configuração, o *Fablo* lê o ficheiro e deteta a configuração pretendida para a *blockchain*.

2. Cria os vários componentes criptográficos como os certificados de Segurança de Camada de Transporte, certificados X.509, e chaves privadas.

3. Gera o primeiro bloco para o serviço de ordenação e inicia os contentores *docker* da *blockchain*.

4. Gera a configuração dos canais e cria os mesmos (neste caso apenas só vai ter um canal).

5. Cada organização junta-se ao canal definido.

6. Empacota o *smart contract* "traceability" e instala-o em todas as organizações definidas e em todos os seus *peers*.

Sempre que uma nova versão do *smart contract* é finalizada, o mesmo é empacotado de novo e instalada nos *peers* das organizações. A nova versão deve ser aprovada por organizações suficientes para passar pela política do ciclo de vida do protocolo. Caso a maioria aprove, o *smart contract* é publicado no canal através de uma Tx.

5.1.2 *Smart Contract*

Para operar com as classes "on-chain" do modelo de domínio definido na secção 4.2, vários métodos foram definidos no *smart contract* para dar suporte a todas as operações relacionadas com as funcionalidades de rastreabilidade da plataforma.

A tabela 5.1 apresenta esses métodos, que têm como principal função gerir as atividades que ocorrem na cadeia de valor de um determinado lote, bem como ler a informação acerca da rastreabilidade de um lote. Alguns dos argumentos de entrada passados para estes métodos são automáticos, como a organização e o estabelecimento onde essa atividade está a ser registada. No entanto, existem métodos que necessitam de ser inseridos por um utilizador, como, por exemplo, a quantidade do lote, leituras do mesmo (sendo que estas leituras embora sejam inseridas pelo utilizador, podem ser automatizadas recorrendo a etiquetas IoT), etc. Existem também métodos internos no *smart contract* para gerir os dados dos lotes ou das atividades, transferir a propriedade dos lotes, atualizar a quantidade dos mesmos, entre outros métodos que não foram representados na tabela 5.1. Os métodos internos não podem ser chamados pela REST API fornecida pelo *Fablo*, uma vez que esses métodos apenas devem servir como apoio para a execução de outros métodos.

Todos os argumentos dos métodos assinalados com asterisco (*) representam parâmetros que são preenchidos de forma automática e que representam uma entidade na base de

Método	**Descrição**	**Parâmetros entrada**	**Parâmetros saída**
CreateActivity	Cria um atividade de produção ou desmantelamento	ID;Organization*;Facility*;startDate;endDate Type;InputLots;OutputLots;Issuer*	Activity
ReadActivity	Lê uma atividade de produção ou desmantelamento	ID	Activity
ReadActivitiesByType	Obtém todas as atividades de produção ou desmantelamento de uma organização dado um determinado tipo	Organization*;Type	List<Activity>
CreateReception	Cria um atividade de receção	ID;TransportID;Organization*;Facility*;newLot ReceivedLot;Quantity;ProviderOrg*;ProviderFac*; Date;Issuer*	ReceptionActivity
ReadReception	Lê uma atividade de receção	ID	ReceptionActivity
ReadAllReceptions	Obtém todas as atividades de receção de uma organização	Organization*;Permission	List<ReceptionActivity>
ReadIncomingReceptions	Obtém todas as atividades de transporte que ainda não foram recebidas por uma organização	Organization*	List<TransportActivity>
CreateTransport	Cria um atividade de transporte	ID;Organization*;Facility*;startDate;endDate; Type;Destination;Lot;Quantity;Cost;Distance; Issuer*	TransportActivity
ReadTransport	Lê uma atividade de transporte	ID	TransportActivity
ReadAllTransportActivities	Obtém todas as atividades de transporte de uma organização	Organization*	List<TransportActivity>
CreateSale	Cria um atividade de venda	ID;Organization*;Facility*;Quantity;Lot;Date;Issuer*	SaleActivity
GetSaleActivityByLotID	Obtém as atividade de venda de um determinado lote	LotID	List<SaleActivity>
ReadAllSaleActivities	Obtém todas as atividades de venda de uma organização	Organization*	List<SaleActivity>
CreateGeneric	Cria uma atividade genérica da cadeia de valor, como a atividade de registo de um lote	ID;Organization*;Facility*;StartDate;EndDate; Type;Lot;Issuer*	GenericActivity
ReadGeneric	Lê uma atividade Genérica	ID	GenericActivity
ReadGenericActivitiesByType	Obtém todas as atividades genéricas de uma organização dado um determinado tipo	Organization*;Type	List<GenericActivity>
SearchLotByOrganization	Obtém a listagem de todos os lotes de uma determinada Organização	Organization*	List<Lot>
GetLotActivities	Obtém todas as atividades de um determinado lote, e todos as atividades dos lotes precedentes ao mesmo, é esta a função que retorna toda a rastreabilidade de um lote	LotID	List<GlobalActivity>

Tabela 5.1: Métodos do *smart contract*.

dados *MongoDB*, estes métodos antes de serem enviados para o método do *smart contract* são validados pela REST API principal, de forma a verificar se de facto esses dados existem e se o utilizador que pretende executar tal operação pertence de facto à organização que diz pertencer.

O algoritmo abaixo mostra a representação em pseudocódigo do método ”CreateActivity“. Este método contem uma série de verificações e validações necessárias para manter a integridade dos dados entre os lotes, e suas respetivas atividades. Estas podem ser restrições relativas a dados provenientes de outras fontes externas (como a base de dados *MongoDB*), relativas às quantidades utilizadas na atividade, ou simplesmente condições de integridade do ID, entre outras.

Algorithm 1 Pseudocódigo do método ”CreateActivity“

```
1:  function CreateActivity(Input: Parâmetros de entrada apresentados tabela 5.1)
2:      exists, err ← ActivityExists(ID)
3:      if err ≠ nil then
4:          throw “Could not read activity from world state.”
5:      else if exists then
6:          throw “The asset already exists”
7:      globalOutputLots ← make(map[string]float32)
8:      for k, v in InputLots do
9:          lot, err_code ← c.ReadLot(k)
10:         if err_code ≠ nil then
11:             throw “Could not read input lot from world state.”
12:         if v ≤ 0 then
13:             throw “Invalid lot amount.”
14:         if v > lot.Amount then
15:             throw “Invalid lot amount.”
16:         _, err ← c.UpdateLotAmount(k, lot.Amount − v)
17:         if err ≠ nil then
18:             throw “Couldn’t update the lot quantity”
19:     for key_, value in output_lots do
20:         exists, err ← c.LotExists(key_)
21:         if err ≠ nil then
22:             throw “Could not read output lot from world state.”
23:         else if exists then
24:             lot, err_code ← c.ReadLot(key_)
25:             if err_code ≠ nil then
26:                 throw “Could not read output lot from world state.”
27:             if value.Amount ≤ 0 then
28:                 throw “Output Lots’ amounts must be greater than 0.”
29:             _, err ← UpdateLotAmount(key_, lot.Amount + value.Amount)
30:             if err ≠ nil then
31:                 throw “Couldn’t update the lot quantity”
32:         else
33:             error_ ← c.CreateLot(value)
34:             if error_ ≠ 0 then
35:                 throw “Error creating the output lot.”
36:             globalOutputLots[key_] ← value.Amount
37:     Activity ← new Activity(ID, Organization, Facility, ..., InputLots, OutputLots)
38:     GlobalActivity ← new GlobalActivity(Activity)
39:     err ← c.GetStub().PutState(globalActivity.ID, bytes)
40:     if err ≠ nil then
41:         throw “Failed to save in the world state.”
42:     return GlobalActivity
```

5.1.3 REST API principal

A REST API principal fornecerá todos os serviços (*endpoints*) consumidos tanto pelas duas aplicações *frontend*, como também para processos de integração com outros sistemas. Esta camada é responsável por gerir todas as fontes de dados, já que a mesma vai interagir tanto com a base de dados ”off-chain“ (*MongoDB*), como com os métodos definidos no *smart contract* apresentado na subsecção 5.1.2.

Como foi explicado na secção 4.4, esta aplicação representa a camada intermediária na fig. 4.5. A aplicação foi desenvolvida em *NodeJs* em conjunto com uma *framework*, que fornece uma variedade de recursos para o desenvolvimento de uma REST API. Para interagir com uma base de dados *MongoDB* foi utilizada a biblioteca *Mongoose* para definir as coleções necessários para o sistema, assim como a gestão (CRUD) dos mesmos. Na tabela 5.2 é possível consultar as coleções criadas através da biblioteca *Mongoose*, o propósito e atributos de cada uma delas. Nesta tabela são apresentados os principais atributos de cada coleção, sendo que existem outros que estão presentes em todos os documentos do sistema, como o atributo ”CreatedAt“ que regista a data e hora em que um documento foi criado, o ”CreatedBy“ que guarda o ID do utilizador que registou esse documento na base de dados, e por fim, o atributo ”active“ que permite eliminar um documento, ou reverter o processo mais tarde (na figura A.1 é possível consultar as coleções criadas numa instância *MongoDB*). É também de mencionar que foi utilizada uma ferramenta incorporada no *MongoDB* chamada *GridFS*, que em vez de armazenar um ficheiro num único documento, a ferramenta divide o ficheiro em várias partes, ou blocos, e armazena cada uma dessas partes como um documento separado. Assim, quando é necessário consultar um ficheiro, a ferramenta vai voltar a montar os pedaços separados conforme necessário. Assim, é possível por vezes apenas consultar partes de um ficheiro (como apenas consultar parte um ficheiro de vídeo ou áudio). Esta ferramenta não é apenas útil para guardar ficheiros pesados, como também para armazenar qualquer ficheiro que se pretende aceder sem ter que o carregar por completa na memória [55]. Quando é pretendido gravar um ficheiro através do *GridFS*, a ferramenta cria duas coleções na base de dados, uma coleção ”<nome_da_coleção>.files“ (figura A.2) que grava o cabeçalho do ficheiro, ou seja, meta-dados acerca do mesmo, e outra coleção onde grava as várias partes que o ficheiro pode ser dividido. Na figura A.3 é representada a coleção onde são guardadas as partes de um ficheiro, nesta figura apenas é apresentada uma parte do ficheiro, já que o mesmo tem menos tamanho que o tamanho definido para fazer a divisão do mesmo.

Quando à vertente de integrar com o *smart contract* publicado na *blockchain* (passo desenvolvido na subsecção 5.1.2), foi utilizada a ferramenta *axios* (explicada na secção 4.4) que permite fazer pedidos HTTP. Esta ferramenta foi utilizada nesta aplicação inter-

Documento	Descrição	Principais Atributos	Relacionamentos
Organizationtypes	Permite guardar os tipos de uma organização	**_id**: Identificador único; **name:** Nome do tipo de organização;	
Organizations	Permite guardar as Organizações do sistema	**_id**: Identificador único; **internal_code:** Código interno da organização; **type:** Tipo da organização; **name:** Nome da organização; **blockchain**: Guarda um objeto com os dados do nó da organização da blockchain.	type = Organizationtypes['_id']
Facilities	Permite guardas os estabelecimentos das diversas organizações	**_id**: Identificador único; **internal_code:** Código interno do estabelecimento; **organization**: Organização de um estabelecimento; **latitude:** Latitude do estabelecimento; **longitude:** Longitude do estabelecimento; **name:** Nome da organização;	organization = Organizations['_id']
Roles	Permite guardas os papeis (roles) das diversas organizações	**_id**: Identificador único; **internal_code:** Código interno de um papel (role); **name:** Nome papel (role); **organization**: Organização a que pertence o papel (role); **latitude:** Latitude do estabelecimento; **longitude:** Longitude do estabelecimento; **permissões:** Guarda uma lista das permissões que este papel (role) tem no sistema;	organization = Organizations['_id']
Users	Permite guardar os utilizadores de uma organização	**_id:** Identificador único; **email:** Email do utilizador; **firstName:** Primeiro nome do utilizador; **lastName:** Último nome do utilizador; **password:** Palavra-passe do utilizador; **verified:** Se o utilizador verificou a conta pelo email; **organization:** Organização a que pertence o utilizador; **roles:** Lista dos diversos papéis (roles) que o utilizador pertence na sua organização; **isIntegration:** Permite definir se é um utilizador especifico para fazer integrações com outros sistemas já existentes, ou se é um utilizador normal do sistema. **blockchain**: Guarda um objeto com os dados de acesso à blockchain, como o id e a palavra-passe encriptada com o algoritmo SHA512. **apiKey:** Guarda a API key caso seja um utilizador para integração; **verificationCode:** Código de verificação que é gerado quando o utilizador pretende recuperar a palavra-passe;	organization = Organizations['_id'] roles = ListRoles['id']
Unitcosts	Permite guardar as unidades de custo da plataforma	**_id**: Identificador único; **internal_code:** Código interno da unidade de custo; **name:** Nome da unidade de custo;	
units	Permite guardar as unidades de medida da plataforma	**_id**: Identificador único; **internal_code:** Código interno da unidade de medida; **name:** Nome da unidade de medida; **isAbleToSerial:** Se é uma unidade de medida que pode ter produtos com número de série;	

Tabela 5.2: Documentos definidos na base de dados "off-chain"

média para interagir com a REST API que o *Fablo* (ferramenta que ajudou a levantar os vários componentes da *blockchain* na secção 5.1.1) forneceu, para facilitar o acesso aos métodos do *smart contract* definidos na tabela 5.1.

No excerto de código 2 é demonstrada a definição da rota "/api/activity" configurada na API principal. Esta rota é responsável por fornecer um serviço às aplicações *frontend*

para criar uma atividade de produção no sistema (atividade esta, guardada neste caso no estado da *blockchain* através da integração com método do *smart contract* ”CreateActivity“ presente na tabela 5.1).

```
import express from "express";
import { createActivityHandler, ... }
    from "../controller/activity.controller";
import requireUserPermission
    from "../middleware/requireUserPermission";

const router = express.Router();

router.post(
    "/api/activity",
    requireUserPermission,
    createActivityHandler
);
...
```

Listing 2: Código da rota (*endpoint*) para criar uma atividade de produção.

Como se pode consultar no código quando a rota é definida, segue-se a função ”requireUserPermission“ que funciona como um *middleware* nesta rota. Este *middleware* desenvolvido é bastante útil para utilizar em qualquer rota que necessite que o utilizador necessite de autorização (permissão) para aceder aos recursos da mesma. O pseudocódigo desta função está definido no algoritmo 2, onde tanto a lógica de toda a função está presente, como cada linha da mesma está comentada para melhor compreensão do algoritmo. Apesar da lógica do *middleware* estar devidamente comentado e explicado existem algumas funções externas utilizadas como função de apoio, sendo que algumas delas interagem com coleções ”off-chain“ representadas na tabela 5.2, estas funções são:

- *findRoleByIds* - Função de apoio que permite validar e verificar o papel (*role*) do utilizador. Ao desencriptar os dados do utilizador na linha 8 no algoritmo 2 é possível ir ao documento do utilizador e verificar o *role* dele na aplicação.
- *findAllPermissionsInRoles* - Permite validar e verificar as permissões de um determinado *role* através da coleção ”Role“, onde no documento de um determinado *role* estão definidas todas as suas permissões.

Algorithm 2 Pseudocódigo do *middleware* "requireUserPermission"

```
1:  function REQUIREUSERPERMISSION(request, response, next)
2:      //Obtêm o token de acesso no cabeçalho do pedido
3:      accessToken ← request.headers.authorization
4:      if !accessToken then
5:          return response.sendStatus(403)
6:      //A função "verifyJwt" permite desencriptar o token de acesso.
7:      //A variável "ACCESS_TOKEN_PUBLIC_KEY" está presente no ficheiro
8:      //de configuração da aplicação e guarda a chave para desencriptar o token.
9:      user_decoded ← verifyJwt(accessToken, ACCESS_TOKEN_PUBLIC_KEY)
10:     if !user_decoded then
11:         return response.sendStatus(403)
12:     //A função "findRoleByIds" é responsável por verificar se os roles que o utilizador
13:     //diz ter são de facto os registados no sistema.
14:     role_response ← await findRoleByIds(user_decoded.roles)
15:     if !role_response.success then
16:         return response.sendStatus(403)
17:     //A função "findAllPermissionsInRoles" é responsável por procurar
18:     //as permissões que aquele role do utilizador tem.
19:     permissions ← findAllPermissionsInRoles(role_response.payload)
20:     has_permission ← null
21:     //Este ciclo "For" é responsável por verificar se o
22:     //utilizador tem permissão para aceder a este route.
23:     for permission in permissions do
24:         if permission.method = request.method and permission.path =
    request.route.path then
25:             has_permission ← permission
26:     if has_permission ≠ null then
27:         return response.sendStatus(403)
28:     //Caso passe por todas as verificações anteriores é guardado o role na variável
29:     //response.locals.permission, de forma à próxima função ter acesso a estes valores.
30:     response.locals.permission ← role_response
31:     response.locals.user ← user_decoded
32:     //É chamada a função "next" para dar acesso à função de registo da atividade.
33:     return next()
```

Algorithm 3 Pseudocódigo do *controller* "createActivityHandler"

```
1: function CREATEACTIVITYHANDLER(request, response)
2:     //Obtêm os dados do utilizador que efetuou o pedido
3:     user ← response.locals.user
4:     //Obtem os dados do corpo do pedido
5:     {Facility, Organization, startDate, endDate, InputLots, OutputLots} ← request.body
6:     //Valida o estabelecimento enviado no corpo do pedido
7:     facility_response ← await getFacilityById(Facility)
8:     if !facility_response.success then
9:         return response.send({ success: false, payload: "erro ao validar o estabelecimento!"})
10:    //Valida a organização enviado no corpo do pedido
11:    organization_response ← await getOrganizationById(Organization)
12:    if !organization_response.success then
13:        return response.send({ success: false, payload: "erro ao validar a organização!"})
14:    //Obtem o token de acesso do utilizador ao canal da blockhcain
15:    refresh_response ← await RefreshUserBlockChainToken(user)
16:    if !refresh_response.success then
17:        return response.send({ success: false, payload: "erro ao obter token!"})
18:    Type ← 1
19:    //Formata os dados que vão ser enviados para o método do smart contract
20:    data ← JSON.stringify({'method': 'TraceabilityContract:CreateActivity',
21:                'args': [generateID(),Organization,Facility,startDate,endDate,
22:                Type,InputLots,OutputLots,user]})
23:    url ← getBlockchainUrl()
24:    //Efetua o pedido à API que permite a interação com o método do smart contract
25:    blockchain_response ← axios.post(url, data, {'Authorization': 'Bearer ' + refresh_response.token, 'Content-Type': 'application/json'})
26:    return blockchain_response
```

No caso de todas as verificações implementadas no *middleware* serem válidas, o *controller* ”createActivityHandler“ vai ser chamado e executado, para efetivamente registar a atividade de produção. O pseudocódigo desta função é representado no algoritmo 3, onde o mesmo está devidamente comentado. A função começa por obter os meta-dados enviados no corpo *body* do pedido e valida os mesmos através de funções externas (1. a função *getFacilityById* permite validar os dados do estabelecimento recebido no corpo do pedido; 2. a função *getOrganizationById* permite validar os dados da organização recebida no corpo do pedido;). Após verificar alguns dos dados recebidos, é pedido o *token* de acesso do utilizador ao canal da *blockchain*, de forma ao mesmo poder enviar os dados através da REST API fornecida pelo *Fablo* (consultar a secção 5.1.1). Por fim, é enviado um pedido ao método ”CreateActivity“ do *smart contract*, onde vai ser executado o código presente no algoritmo 1, cuja finalidade é validar alguns dos dados enviados, e no caso de tudo ser validado, registar a atividade de produção na *blockchain* (este processo foi modelado no diagrama de sequência definido na figura 4.3).

5.2 Frontend

O *frontend* é responsável por apresentar as informações e funcionalidades de uma aplicação de forma amigável e intuitiva. Na sua essência, o *frontend* gira em volta da criação de janelas visualmente atraentes e responsivas que se adaptam a diferentes dispositivos e tamanhos de ecrã, fornecendo uma aplicação interativa para lidar com as funcionalidades de todo o sistema. No caso deste projeto foram desenvolvidas duas aplicações *frontend*, uma aplicação móvel para facilitar o registo de atividades da cadeia de valor, e uma aplicação *web* que permite configurar e parametrizar toda a plataforma. No entanto, esta aplicação também permite a gestão de atividades, sendo que esta não é tão interativa como a desenvolvida para dispositivos móveis.

Apesar de terem sido desenvolvidas estas duas aplicações para demonstrar as funcionalidades deste sistema, estas podem ser melhoradas, tanto no requisito de usabilidade, como no de *design*, já que o principal foco desta dissertação foi a implementação de toda a infraestrutura *backend* descrita na secção 5.1 e a sua lógica.

Com o objetivo de demonstrar todas as funcionalidades implementadas através da aplicação *web* e da aplicação móvel, podemos assumir as seguintes condições:

- A plataforma é inicializada com uma organização do tipo administrador que permite gerir todas as outras (organização do sistema), esta organização contém um utilizador que permite parametrizar a aplicação como inserir novas organizações, utilizadores, unidades de medida e de custo, etc.

- Após a plataforma estar minimamente configurável, cada organização terá os seus utilizadores, e o administrador da mesma pode tanto gerir a sua organização e os seus estabelecimentos, como os seus utilizadores e as suas funções dentro da mesma.

5.2.1 Aplicação Web

Começando pela aplicação *web*, a mesma começa por apresentar a página de *login* demonstrada na figura 5.1, caso um utilizador digite corretamente os seus dados é redirecionado para o seu menu, caso os dados estejam errados vai ser mostrada uma mensagem de erro para o utilizador. No caso de um determinado utilizador se ter esquecido da sua palavra passe, poderá aceder à página "Esqueceu-se da palavra-passe" para enviar para o seu *email* um pedido de alteração da palavra-passe (processo demonstrado em sequências nas figuras A.6, A.7, A.8, A.9 respetivamente).

Após o administrador da plataforma entrar na sua conta, foi selecionada a opção "Organização" para gerir as organizações da plataforma. Na figura 5.2 são apresentadas duas interações da página, a primeira interação mostra a lista de todas as organizações registadas no sistema, sendo que podem ser adicionadas novas organizações através do botão "Adicionar Organização" (neste momento apenas são pedidos os dados mínimos para criar uma organização, como o código interno, o nome da mesma e o tipo da organização). No processo de criação de uma organização também é necessário inserir dados para facilitar a integração com os métodos do *smart contract* desenvolvido, como o nome/id da organização registada na *blockchain*, o canal a que esta organização pertence, o *smart contract* que é pretendido aceder recorrendo à API, e por fim, a porta em que a API dessa organização foi gerada pelo *Fablo* (na figura A.5 pode-se consultar as API criadas para cada organização e as portas para aceder às mesmas).

Embora neste caso de estudo tenha sido definido apenas um canal com um *smart contract* comum a todas as organizações, a necessidade desta configuração surge devido ao facto que diferentes organizações podem estar em diferentes canais com *smart contracts* específicos para os mesmo.

A segunda iteração surge após clicar na linha de uma organização, onde expande uma secção que mostra a listagem de estabelecimentos da mesma. Nesta secção, é também disponibilizado um botão para adicionar um novo estabelecimento com os campos pedidos no formulário presente na figura.

Após inserir uma organização é possível aceder à opção do menu "Roles" que permite criar uma função para um utilizador (*role*) de uma determinada organização. Neste formulário é possível adicionar dinamicamente as permissões que se vai adicionar a essa

função. No exemplo apresentado na figura 5.3, os dois primeiros ecrãs são referentes à criação de uma função de utilizador que apenas tem permissões para gerir as atividades de produção da sua organização. No que toca ao terceiro e último ecrã, este diz respeito ao formulário de criação de um utilizador, onde são apenas pedidos os dados mínimos para a criação do mesmo, como o primeiro e último nome, email, palavra-passe, seleção da organização do utilizador (como o utilizador autenticado é administrador pode selecionar qualquer organização), se o utilizador criado é um utilizador destinado a integrações com outros sistemas (1. Caso seja selecionada esta opção será criada uma *API KEY* para este utilizador, que apenas terá permissões para integrar os serviços presentes no *role* selecionado posteriormente, como se pode ver no apêndice B; 2. O indicado é criar anteriormente um role para apenas os serviços disponibilizados para integrações, e selecionar apenas os serviços que utilizador pode chamar.) ou não, e por fim, selecionar a função do utilizador.

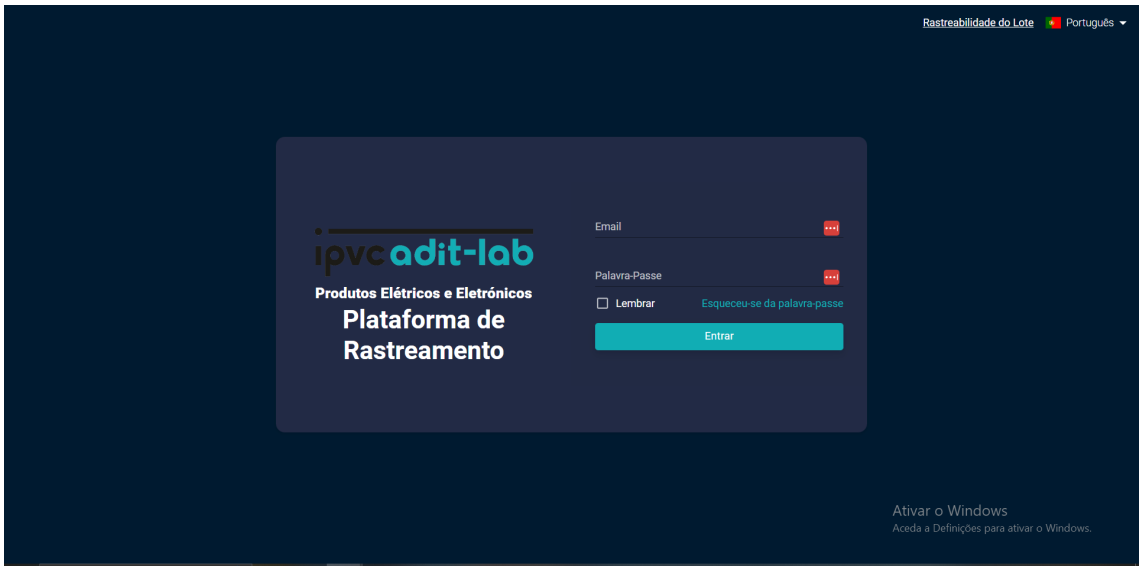

Figura 5.1: Página de *login* da aplicação *web*.

5.2.2 Aplicação Móvel

Partindo do princípio que a organização de mineração ”MINERACAO0001“ está minimamente configurada, como demonstrado na figura 5.2, é possível aceder à aplicação móvel para registar atividades da cadeia de valor por um utilizador que tenha permissões para registar as mesma. Na figura 5.4 é possível ver os ecrãs que permitem o registo dessa atividade, desde o ecrã de *login* até à criação da atividade (como se pode ver no terceiro ecrã desta figura, é possível dar ”scan“ de um código de barras ou de um código QR, para facilitar a leitura de lotes ou produtos). Nesta figura também é possível ver as últimas atividades de registo criadas, tanto no presente dia, como no presente mês.

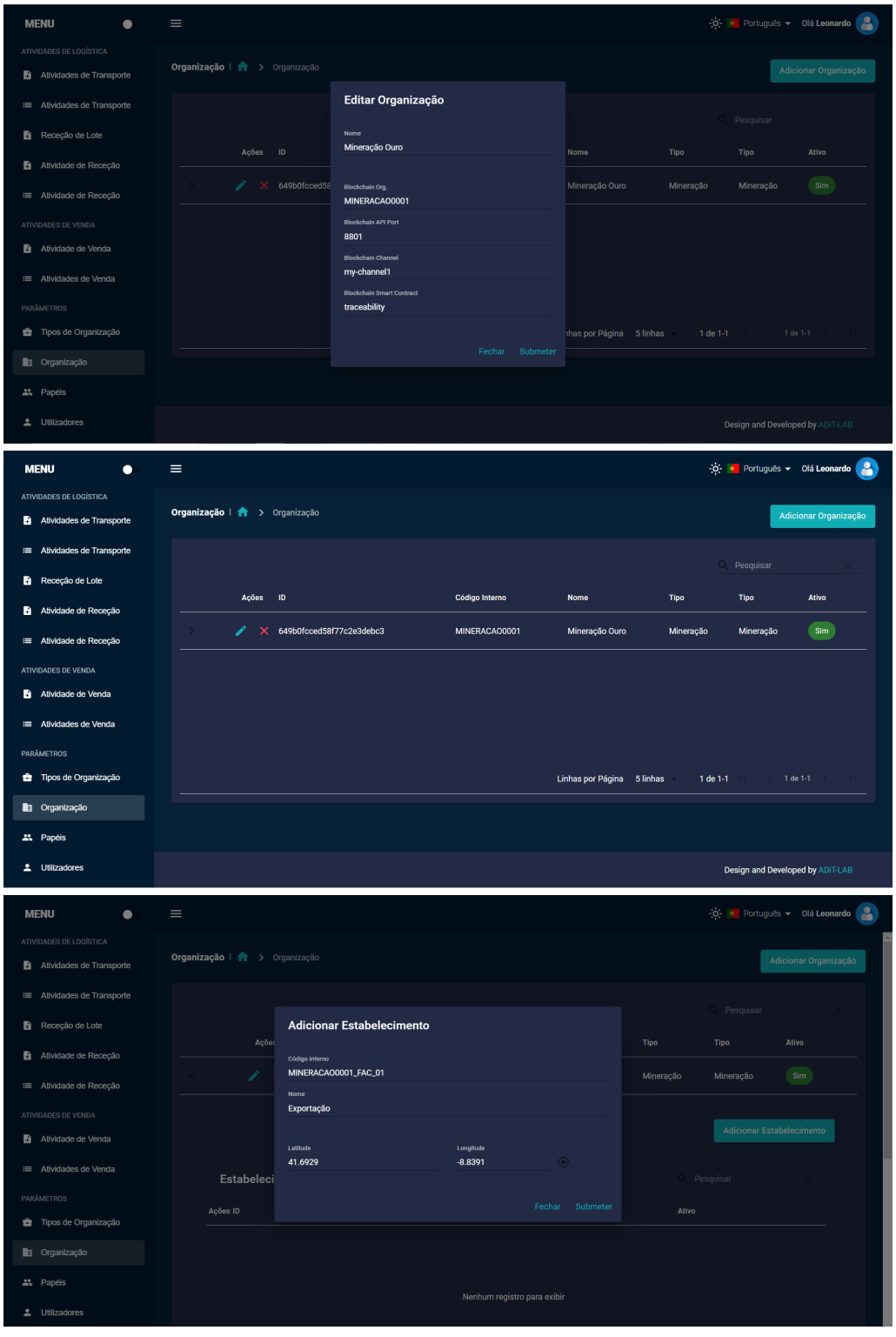

Figura 5.2: Página para gerir organizações e estabelecimentos das mesmas.

Após demonstrar a criação de uma atividade de registo, é demonstrado o registo de uma atividade de produção na figura 5.5. Esta atividade tem alguma peculiaridade, visto

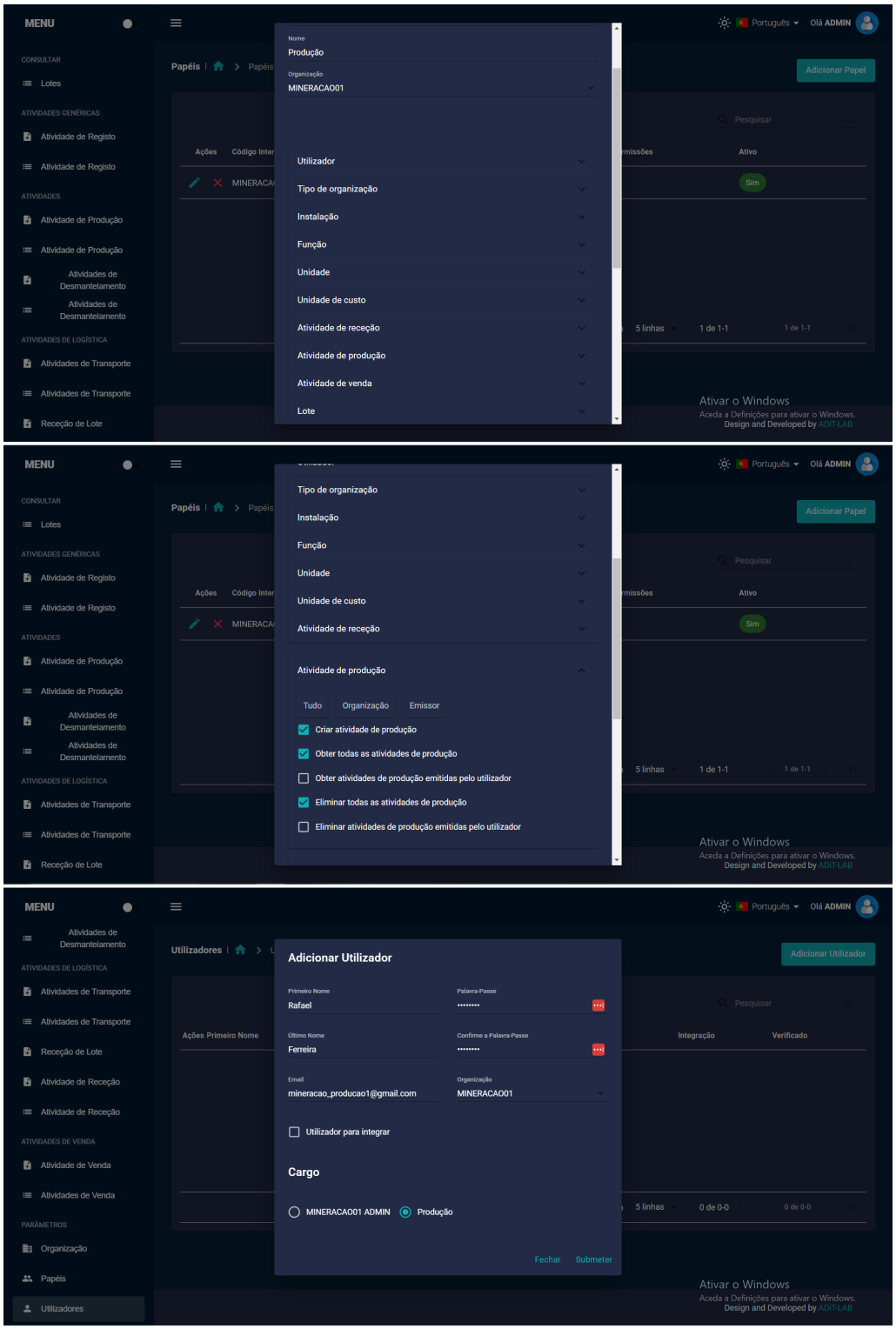

Figura 5.3: Páginas para gerir funções do utilizador e utilizadores.

que recebe vários lotes de entrada para produzir um ou mais lotes (lotes de saída). No primeiro ecrã já foram inseridos dois lotes de entrada, sendo possível adicionar mais atra-

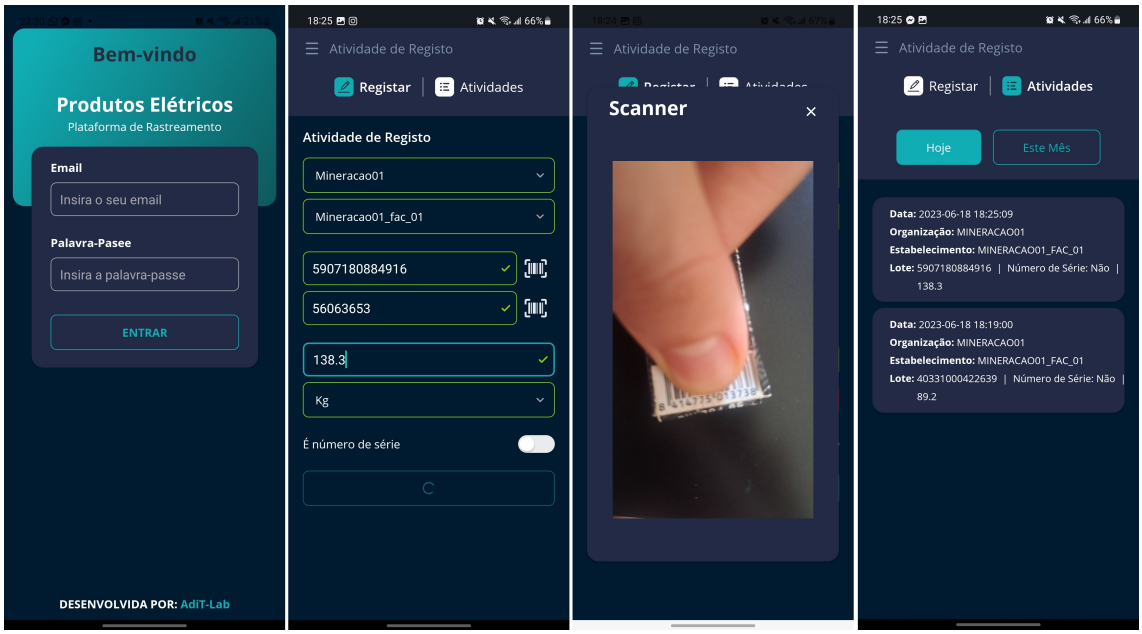

Figura 5.4: Ecrãs para registar uma atividade de registo de lote.

vés do ID do lote (pode ser lido através do *scanner*) e a quantidade gasta desse lote na atividade. Já mais em baixo encontra-se uma secção "Lotes de Saída", onde são inseridos os lotes de saída da atividade. Nesta secção, sempre que é inserido um novo lote é pedido o ID do lote, ID do produto, a quantidade produzida e a unidade de medida do lote produzido. Por fim, no último ecrã da figura 5.5 é possível ver a listagem dos lotes de produção registados e alguns dos seus dados.

Além da demonstração do registo destes dois tipos de atividades, na secção A.4 do apêndice A são apresentados os ecrãs de registo de outro tipo de atividades, como o registo de atividades de transporte, receção e venda (figuras A.12, A.13, A.14, respetivamente).

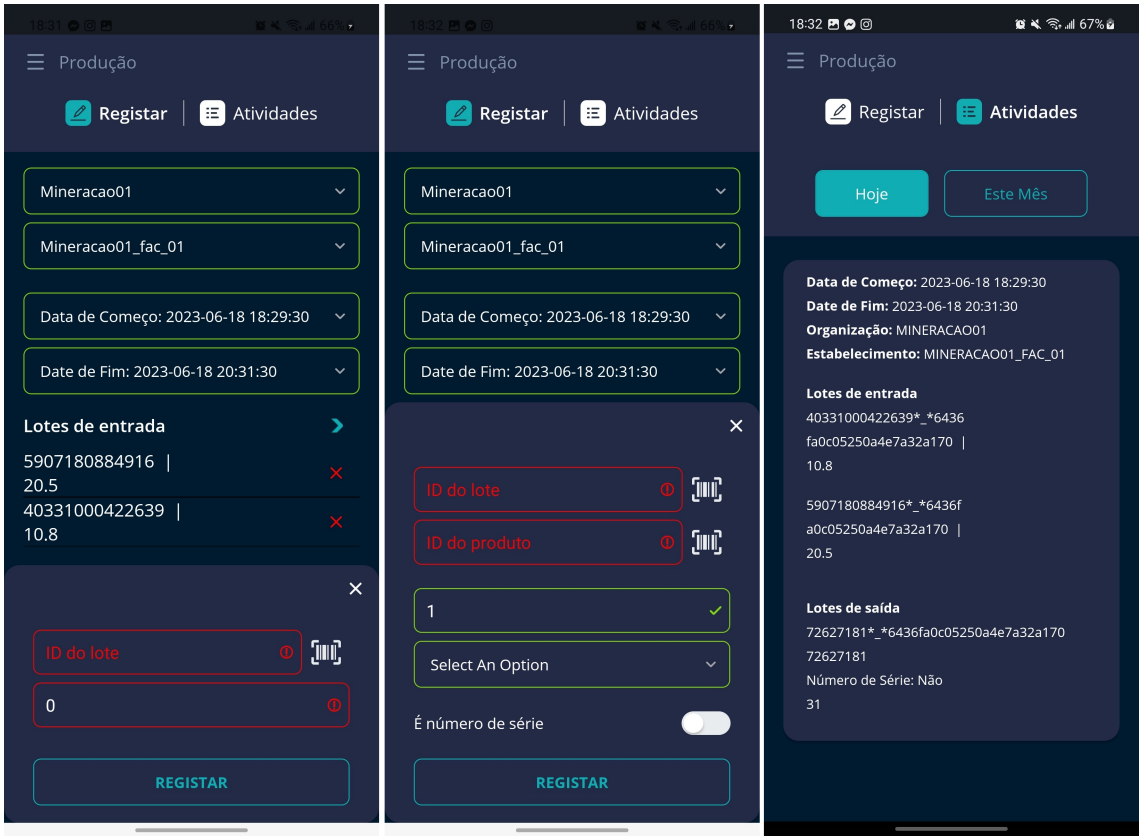

Figura 5.5: Ecrãs para registar uma atividade de produção.

Capítulo 6

Análise e Discussão

6.1 Demonstração

Para entender melhor como a plataforma fornece recursos para apoiar a rastreabilidade de lotes e atividades do modelo circular proposto na figura 3.2, vai ser colocado um cenário hipotético de rastreamento na cadeia de valor de PE. Este cenário hipotético foi também utilizado para explicar e modelar o sistema na figura 4.1 (secção 4). Embora este modelo seja apenas ilustrativo, e por consequente, não representa de facto todos os lotes ou componentes que um determinado produto necessita na realidade, o mesmo será utilizado como exemplo para demonstração.

No âmbito de preparar esta demonstração, primeiro foram definidas todas as organizações e os consequentes estabelecimentos necessários para replicar o fluxo de lotes representado na figura 4.1. Para além destas configurações, também foram criados diferentes *roles* e utilizadores para todas as organizações, de forma aos mesmos conseguirem registar os diversos tipos de atividades da cadeia de valor. Com o propósito de demonstrar a rastreabilidade de um produto neste documento, apenas vai ser demonstrada a rastreabilidade do lote "CPU_0001" devido ao facto que quanto mais nos aproximarmos de um produto final (por exemplo, o lote "PC_0001") maior vai ser o número de atividades e mais complicado será demonstrar a rastreabilidade em formas de figuras. Na figura 6.1 é demonstrado em forma de árvore todas as atividades antecedentes à produção do lote "CPU_0001".

Além de se poder consultar a rastreabilidade de um lote em forma de árvore também é possível fazê-lo através da representação da mesma num mapa. Na figura 6.2 pode-se consultar os pontos de todas as atividades da cadeia de valor que levaram à produção do lote "CPU_0001". Em ambas as formas de visualizar a rastreabilidade de atividades de um lote, é possível ver detalhes da mesma ao clicar em cima da atividade pretendida.

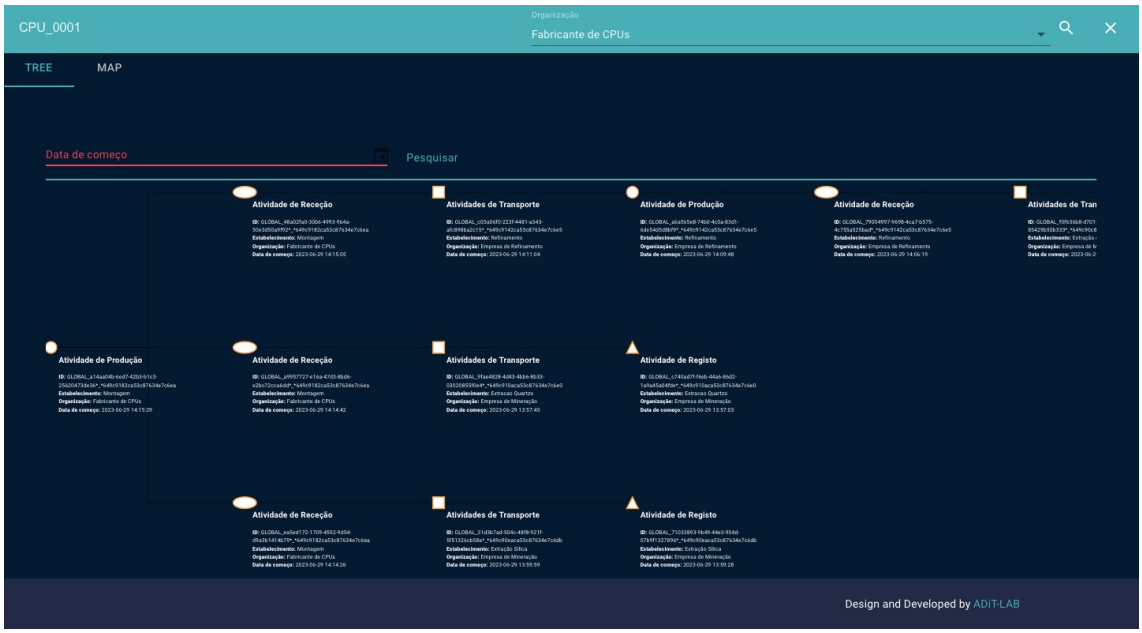

Figura 6.1: Dados de rastreabilidade do lote "CPU_0001" em forma de árvore.

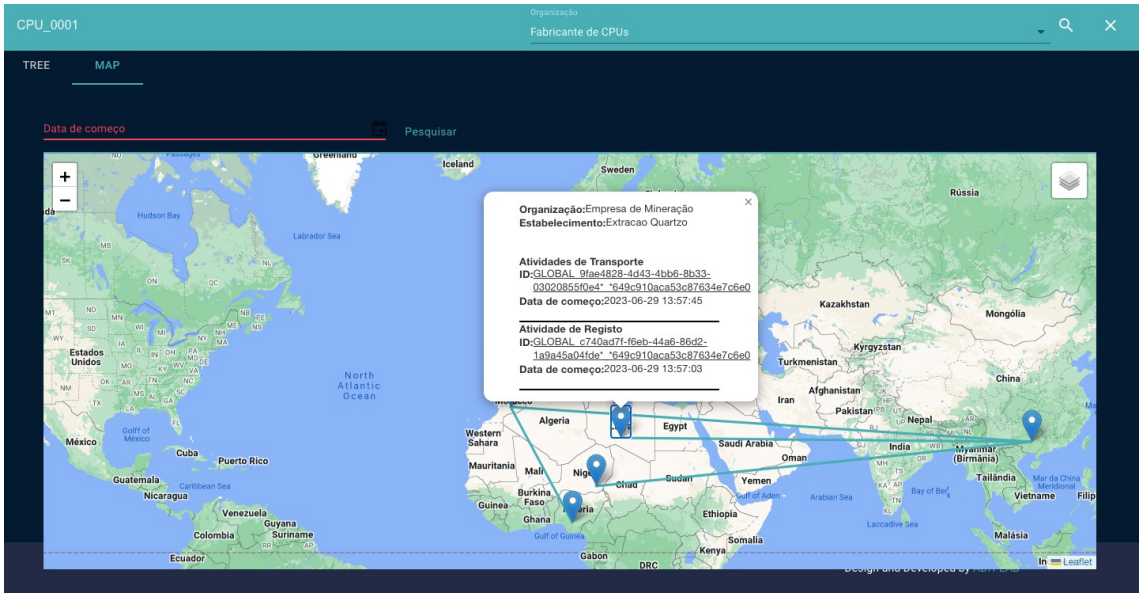

Figura 6.2: Dados de rastreabilidade do lote "CPU_0001" no mapa.

Até chegar a este ponto, e tendo como referência a figura 4.1, o processo de demonstração da rastreabilidade do lote começa por:

- A "Empresa de Mineração" cria três atividades de registo diferentes, para registar os lotes "SIL_0001", "QTZ_0001", "COP_0001" respetivamente. A empresa "Empresa de Ouro" cria uma atividade de registo para o lote "GLD_0001".
- Posteriormente a "Empresa de Mineração" cria uma atividade de transporte para

cada parte do lote que é pretendido transportar, ou seja, o lote "SIL_0001" tanto é transportado para a "Fabricante de CPUs" como para a "Fabricante de RAM", sendo assim criadas duas atividades de transporte diferentes. Por outro lado, tanto a "Empresa de Mineração" como a "Empresa de Ouro" criam atividades de transporte com destino a uma empresa de refinamento destes materiais ("COP_0001", "GLD_0001" respetivamente).

- A "Empresa de Mineração" cria três atividades de registo diferentes, para registar os lotes "SIL_0001", "QTZ_0001", "COP_0001" respetivamente. A empresa "Empresa de Ouro" cria uma atividade de registo para o lote "GLD_0001".
- A organização "Refinamento de Materiais" regista duas atividades de receção, concluindo as atividades de transporte relativas aos lotes "COP_0001" e "GLD_0001".
- A organização "Refinamento de Materiais" regista duas atividades de produção. A primeira atividade recebe uma determinada quantidade do lote "COP_0001" e refina o mesmo, produzindo um novo lote "COP_0001_REF". Quanto à segunda atividade, a lógica é a mesma só que refina o lote "GLD_0001".
- Após as duas atividades de produção efetuadas pela organização "Refinamento de Materiais", parte das quantidades dos lotes produzidos vão ser transportados para a organização (criação de duas atividades de transporte) "Fabricante de CPUs".
- A organização "Fabricante de CPUs" cria uma atividade de receção para todos os lotes, cujas outras organizações criaram uma atividade de transporte com destino à mesma.
- A organização "Fabricante de CPUs" regista uma atividade de produção com os lotes necessários para o fabrico do lote "'CPU_0001".

Por fim, este lote pode ser enviado para outras organizações, tal como demonstrado na figura 4.1

Concluindo esta demonstração, pode-se constatar na figura 6.3 as organizações definidas na configuração da *blockchain*, assim como outros dados estatísticos, como o número de nós da rede, o número de blocos, transações por organização, etc.

Utilizando ainda esta ferramenta, para além de se poder consultar o número de transações e os seus ID num bloco, também se pode consultar que interação é que uma determinada Tx teve com a *blockchain*. A Tx apresentada na figura 6.4 representa a criação de uma atividade de registo pela organização "Empresa de Mineração" ("MINERACAO0001"), podendo se consultar vários dados acerca desta Tx, como o *smart contract*

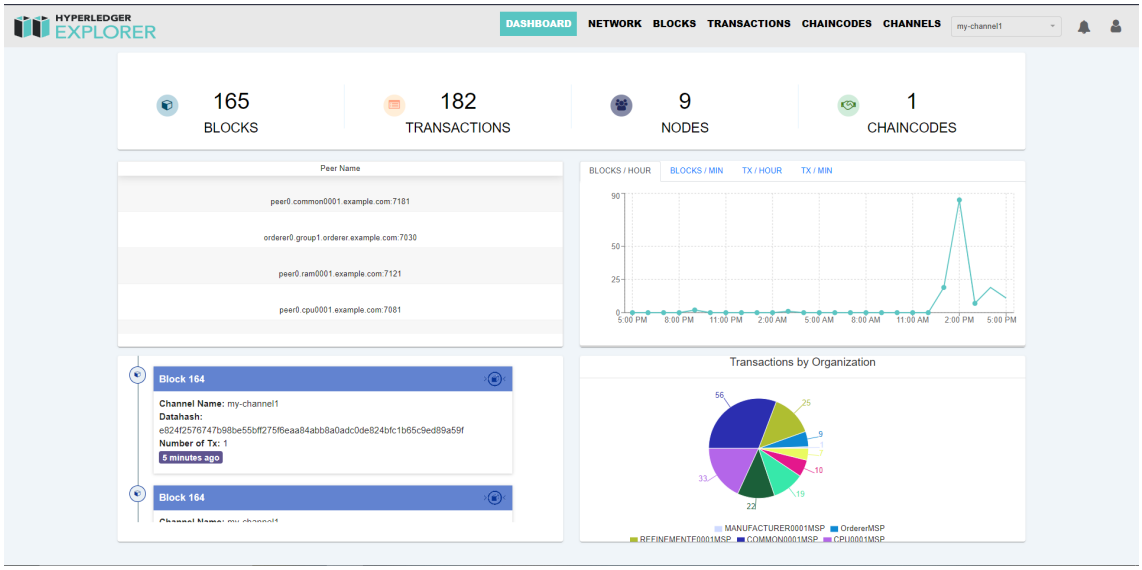

Figura 6.3: *Hyperledger explorer dashboard*.

que interagiu, as organizações que a validaram, e os parâmetros passados para o método de criar uma atividade de registo presente no *smart contract*.

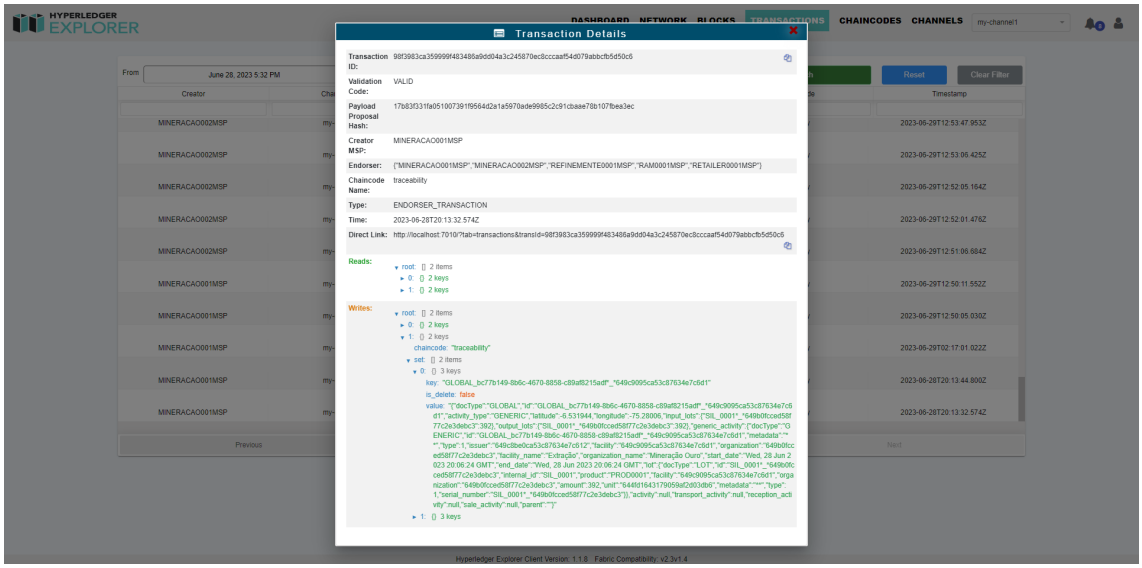

Figura 6.4: Detalhes de uma Tx no *Hyperledger explorer*.

6.2 Performance da REST API

Uma das medidas que se pode avaliar um sistema é a desempenho do mesmo no que toca a lidar com os dados. No sistema proposto neste documento a camada intermediária representada na arquitetura 4.5 é a responsável por lidar com as várias fontes de dados,

Rota	Método HTTP	Descrição	Fonte de dados	Contém recursividade	Tempo de execução
/api/organization	GET	Obtém as organizações do sistema.	*MongoDB*	N\A	87ms
/api/organization	POST	Cria uma nova organização.	*MongoDB*	N\A	132ms
/api/signin	POST	Permite a um utilizador entrar no sistema.	*MongoDB*	N\A	245ms
/api/user/me	GET	Obtém informações do utilizador autenticado.	*MongoDB*	N\A	124ms
/api/registerLot	POST	Cria uma atividade de registo	*MongoDB & Blockchain*	N\A	3420ms
/api/registerLot	GET	Obtém todas as atividades de registo de uma org.	*MongoDB & Blockchain*	N\A	2790ms
/api/activity	POST	Cria uma atividade de produção.	*MongoDB & Blockchain*	N\A * $n = 4$	3330ms
/api/activity	GET	Obtém todas as atividades de produção de uma org.	*MongoDB & Blockchain*	N\A	2790ms
api/traceability/:internal_id	POST	Obtém a rastreabilidade de um lote.	*MongoDB & Blockchain*	✓**$n = 2$	2830ms
api/traceability/:internal_id	POST	Obtém a rastreabilidade de um lote.	*MongoDB & Blockchain*	✓**$n = 5$	3770ms

Tabela 6.1: Performance de algumas rotas da REST API

logo é a aplicação que terá que ser avaliada neste requisito. Na tabela 6.1 é possível ver os tempos médios de execução de várias rotas disponibilizadas pela API. Antes analisar os dados presentes na tabela é importante realçar que estas aplicações não estão em produção, e que a máquina de desenvolvimento das mesmas foi um computador pessoal. Assim, tem que se ter em conta que estes testes foram efetuados com 94% de utilização da memória, e em volta dos 81% do uso do CPU. Este uso intenso de recursos deve-se ao facto de todos os nós e componentes da *blockchain* estarem em execução na máquina local.

A primeira ideia que se pode tirar relativamente à tabela 6.1 é que todas as rotas que apenas interagem com *MongoDB* são executadas em questões de milissegundos, isto deve-se à alta desempenho que esta base de dados fornece (estas rotas ainda passam por várias verificações como a verificação do *token* de autorização, etc.). Esta grande diferença em tempos de execução devem-se a vários fatores, já que a *blockchain*:

1. É um sistema distribuído, onde cada Tx tem que ser validada por diferentes nós (como explicado na 3.4), o que logo à partida faz com que demore mais tempo tanto a registar dados, como a ler os mesmos.

2. Quando é requisitada uma rota que interaja com *blockchain*, por exemplo, a rota POST "api/activity", alguns dados para o registo da atividade têm que ser validados

através do *MongoDB*, e apenas posteriormente enviados para o método pretendido no *smart contract* (este processo é representado do diagrama de sequência 4.3).

3. Enquanto a base de dados *MongoDB* se encontra na nuvem, toda a infraestrutura da *blockchain* está a correr localmente.

Na tabela 6.1 também se pode encontra uma linha com um asterisco (*) representando que os tempos de execução deste serviço dependem do número de lotes de entrada, já que para cada lote de entrada é necessário fazer múltiplas verificações. No caso apresentado na tabela apenas foram anotados os tempos com quatro lotes de entrada. As duas linhas assinaladas com dois asteriscos (**) referem-se ao método que retorna todas as atividades de rastreabilidade de um lote por uma função recursiva. O n refere-se ao número de vezes que a função se chama a si própria (ou número de atividades de rastreamento de um lote), tendo este método sido testado com $n = 2$ e $n = 5$. Comparando as duas linhas podes ver que a diferença de tempos entre as duas é considerável, podendo assumir que à medida que o n a complexidade de efetuar a operação também.

6.3 Problemas Identificados

Considerando que esta é a primeira iteração do protótipo desenvolvido, pode-se considerar que existem vários aspetos do sistema a melhorar, no entanto, esses aspetos vão ser mencionados na secção 7.1. Nesta secção apenas vão ser mencionados os problemas identificados com os métodos utilizados para desenvolver esta primeira iteração.

O grande problema identificado neste sistema é a utilização de recursividade para obter todas as atividades de rastreabilidade de um lote. No caso deste projeto, a recursividade pode introduzir problemas significativos, como *stack overflow* que acontece quando o número de iterações da recursividade é muito longa, já que a pilha (*stack*) vai ser chamada muitas vezes para se manter a par das chamadas da função, causando normalmente um erro. Para além deste problema, a recursividade utiliza muita memória, podendo até mesmo a levar a fugas da memória. Tendo isto em conta, e o facto de este tipo de soluções demorarem mais tempo a serem executadas, torna o uso deste método instável e incerto.

Para além do problema da recursividade, também foi identificado na tabela 6.1 problemas com a interação com os métodos do *smart contract*, no que toca aos tempos de execução. Este problema também é importante, visto que nesta indústria o registo de atividades é uma constante, sendo indesejável para qualquer utilizador que as operações de registo e de leitura demorem à volta de três segundos.

Existem diversos tipos de soluções para contornar estes problemas, no entanto, o uso da base de dados *MongoDB* em conjunto com a *blockchain* para dados de rastreabilidade

talvez seja a melhor e mais rápida opção a implementar. No caso de ser utilizada esta solução, os dados dos lotes e das atividades vão ser registados em ambas as fontes de dados, sendo que o registo na *blockchain* ficará a ser feito numa tarefa secundária, dando ao utilizador a perceção que este foi quase instantâneo. Para efeitos de consistência, também seria necessário registar o histórico de cada uma das tarefas secundárias. Enquanto que a base de dados *MongoDB* permite gerir e consultar informação de forma rápida e eficaz, a *blockchain* apenas funcionará como a fonte verdadeira dos dados de rastreio através das suas inerentes características. Quanto à sincronização destas duas tecnologias também podem haver várias soluções, como, por exemplo, o desenvolvimento de uma tarefa que seja responsável por verificar se os dados de ambas as fontes são consistentes. Esta solução para além de melhorar significativamente os tempos de execução dos serviços de registo de atividades, também pode ajudar no requisito de obter todas as atividades de rastreabilidade de um lote. Ao contrário da recursividade no *smart contract*, pode ser utilizado o sistema de pesquisa do *MongoDB* para obter estes dados.

Capítulo 7

Conclusões

Em suma, considerando que a indústria de EEE é um dos maiores setores do mundo, é urgente direcionar esta indústria para um modelo mais sustentável. Nesta dissertação é mencionado o conceito de CE, já que é um dos modelos mais estudados e promissores para o desenvolvimento de sustentabilidade numa cadeia de valor, proporcionando a reutilização contínua de materiais e recursos, e consequentemente a redução de lixo eletrónico. A adoção de um modelo de sustentabilidade requer aplicações que consigam rastrear os produtos da cadeia de valor e ler possíveis métricas de sustentabilidade.

O uso de *blockchain* para sistema de rastreabilidade traz várias vantagens devido à sua arquitetura, como a imutabilidade das Tx e a transparência que esta tecnologia oferece. Estas características são bastante importantes no requisito de rastreabilidade, já que uma não permite a adulteração de dados, exceto se a rede seja comprometida, e a outra oferece o histórico transparente de um determinado produto.

Apesar da *blockchain* trazer vários benefícios na sua utilização, a mesma também traz consigo alguns problemas técnicos inerentes à sua arquitetura. Visto que é uma tecnologia distribuída (como explicado na secção 3.4) a seu desempenho (um dos principais problemas encontrados na arquitetura proposta nesta dissertação) e os custos de implementação são os dois problemas mais apontados a esta tecnologia. Para além dos problemas técnicos da sua arquitetura, a BCT também traz consigo desafios na sua implementação, como ter o conhecimento de como trabalhar com sistemas distribuídos e desenvolver *smart contracts* para os mesmos.

Assim, a implementação da BCT neste caso de estudo deu para conseguir avaliar e comparar as diferenças do uso desta tecnologia com sistema tradicionais. O uso BCT não é indicada para todos os casos, no entanto, nos vários setores de cadeias de valor, finanças, entre outras, pode ser de facto uma mais-valia devido aos benéficos que fornece. Quanto às suas desvantagens as mesmas podem ser contornadas utilizando outras tecnologias em

conjunto, como foi discutido na secção 6.3.

7.1 Trabalho Futuro

Na secção 6.3 foram encontrados alguns problemas relativos a este sistema, maioritariamente relativos com o desempenho dos métodos do *smart contracts* publicado na *blockchain*. Como trabalho futuro, a prioridade será implementar uma solução para melhorar os tempos de execução que o utilizador visualiza nas aplicações *frontend*. Uma solução já foi proposta no final dessa mesma secção, que implementada irá, sem dúvida, baixar consideravelmente os tempos de execução.

Ainda no que toca a melhorar a desempenho em serviços que necessitem da *blockchain*, outra ideia passa por adicionar um sistema de filas, onde todos os pedidos aos métodos do *smart contract* estarão em fila, para não sobrecarregar a rede com demasiado pedidos em simultâneo. Outra iteração desta ideia, é criar métodos no *smart contract* que registam um determinado número (n) de atividades na rede com apenas uma chamada ao método (por outras palavras é mandar uma lista com n atividades para o método). Considerando $n = 5$, em vez de estarem cinco pedidos na fila para adicionar atividades na *blockchain*, apenas vai estar um pedido na fila, e o método apenas vai ser executado uma vez (no entanto, vai demorar mais tempo a ser executado, uma vez que são mais atividades a serem registadas).

Relativamente ao processo de integração de outros sistemas, é necessário adicionar mais serviços para a integração, visto que neste momento apenas foram disponibilizados os serviços mínimos para o registo de atividades na *blockchain* e a leitura do rastreio de um lote.

Para além de adicionar novas atividade da cadeia de valor (como a de armazenamento), também há bastante trabalho a ser feito nas aplicações *frontend*, tanto seja a melhorar a experiência do utilizador a fazer algumas tarefas, principalmente na sequência "Criar organização" - "Criar estabelecimento" - "Criar *roles*" - "Criar utilizador", como adicionar novas funcionalidades na aplicação móvel, já que neste momento apenas permite registar atividades da cadeia de valor.

No que toca a mais funcionalidades, seria uma mais-valia a implementação de mecanismos para integrar diversos sensores IoT nas atividades de rastreio, para ter mais indicadores de sustentabilidade automaticamente.

Por fim, para incentivar o consumidor final a participar do processo de reciclagem, uma aplicação direcionada a *gamification* poderia ser estudada e/ou desenvolvida.

Bibliografia

[1] L. Fernandes, A. M. Rosado da Cruz, E. F. Cruz, and S. I. Lopes, "A review on adopting blockchain and iot technologies for fostering the circular economy in the electrical and electronic equipment value chain," *Sustainability*, vol. 15, no. 5, 2023. [Online]. Available: https://www.mdpi.com/2071-1050/15/5/4574

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008.

[3] J. J. Bullón Pérez, A. Queiruga-Dios, V. Gayoso Martínez, and Á. Martín del Rey, "Traceability of ready-to-wear clothing through blockchain technology," *Sustainability*, vol. 12, no. 18, 2020. [Online]. Available: https://www.mdpi.com/2071-1050/12/18/7491

[4] C. P. Baldé, V. Forti, V. Gray, R. Kuehr, and P. Stegmann, *The global e-waste monitor 2017: Quantities, flows and resources*. United Nations University, International Telecommunication Union, and International Solid Waste Association, 2017.

[5] K. Wiens, "Smartphone repairs you can do on earth day," 04 2014. [Online]. Available: https://pt.ifixit.com/News/6448/smartphone-repairs

[6] E. Jardim, "From smart to senseless: The global impact of 10 years of smartphones," *Washington: Greenpeace Inc*, 2017.

[7] A. Manhart, M. Blepp, C. Fischer, K. Graulich, S. Prakash, R. Priess, T. Schleicher, and M. Tür, *Resource efficiency in the ICT sector*. Greenpeace eV, 2016.

[8] P. Limata, "Speculating on the application of blockchains in the circular economy," CERBE Center for Relationship Banking and Economics, CERBE Working Papers wpC32, Nov. 2019. [Online]. Available: https://ideas.repec.org/p/lsa/wpaper/wpc32.html

[9] A. M. R. da Cruz and E. F. Cruz, "Blockchain-based traceability platforms as a tool for sustainability," in *22st International Conference on Enterprise Information Systems (ICEIS 2020)*, vol. 2. SciTePress, 2020, pp. 330–337.

[10] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007. [Online]. Available: https://doi.org/10.2753/MIS0742-1222240302

[11] V. Surwase, "Rest api modeling languages-a developer's perspective," *Int. J. Sci. Technol. Eng*, vol. 2, no. 10, pp. 634–637, 2016.

[12] Y. Gu, Y. Wu, M. Xu, X. Mu, and T. Zuo, "Waste electrical and electronic equipment (weee) recycling for a sustainable resource supply in the electronics industry in china," *Journal of Cleaner Production*, vol. 127, pp. 331–338, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0959652616303092

[13] E. M. Foundation, "Towards the circular economy," 2015. [Online]. Available: https://www.werktrends.nl/app/uploads/2015/06/Rapport_McKinsey-Towards_A_Circular_Economy.pdf

[14] T. Wautelet, "Exploring the role of independent retailers in the circular economy: a case study approach," Master's thesis, eufom European University for Economics & Management, 02 2018.

[15] X. M. González, M. Rodríguez, and Y. Pena-Boquete, "The social benefits of weee re-use schemes. a cost benefit analysis for pcs in spain," in *Waste Management*, vol. 64, 2017, pp. 202–213.

[16] R. Atlason, D. Giacalone, and K. Parajuly, "Product design in the circular economy: Users' perception of end-of-life scenarios for electrical and electronic appliances," *Journal of Cleaner Production*, vol. 168, pp. 1059–1069, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0959652617320711

[17] M. Geissdoerfer, P. Savaget, N. M. Bocken, and E. J. Hultink, "The circular economy – a new sustainability paradigm?" *Journal of Cleaner Production*, vol. 143, pp. 757–768, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0959652616321023

[18] T. Zhang and K. Kraisintu, "The role of traceability in sustainable supply chain management," Master's thesis, CHALMERS UNIVERSITY OF TECHNOLOGY, Göteborg, Sweden, 2011.

[19] H. Richardson, "Traceability for electronics manufacturing," November 2020. [Online]. Available: https://www.electronicsonline.net.au/content/assembly/sponsored/traceability-for-electronics-manufacturing-1107037678

[20] W. Li, K.-M. Chao, G. Jin, K. Xia, and L. Gao, "Sustainable information management for waste electrical and eletronic equipment," in *Proceedings of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, 2012, pp. 875–881.

[21] S. Capecci, E. Cassisi, G. Granatiero, C. Scavongelli, S. Orcioni, and M. Conti, "Cloud-based system for waste electrical and electronic equipment," in *2017 13th Workshop on Intelligent Solutions in Embedded Systems (WISES)*, 2017, pp. 41–46.

[22] T. K. Dasaklis, F. Casino, and C. Patsakis, "A traceability and auditing framework for electronic equipment reverse logistics based on blockchain: the case of mobile phones," in *2020 11th International Conference on Information, Intelligence, Systems and Applications (IISA*, 2020, pp. 1–7.

[23] J. Sunny, N. Undralla, and V. Madhusudanan Pillai, "Supply chain transparency through blockchain-based traceability: An overview with demonstration," *Computers & Industrial Engineering*, vol. 150, p. 106895, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835220305829

[24] C. Magrini, J. Nicolas, H. Berg, A. Bellini, E. Paolini, N. Vincenti, L. Campadello, and A. Bonoli, "Using internet of things and distributed ledger technology for digital circular economy enablement: The case of electronic equipment," *Sustainability*, vol. 13, no. 9, 2021. [Online]. Available: https://www.mdpi.com/2071-1050/13/9/4982

[25] M. Kuhn, F. Funk, and J. Franke, "Blockchain architecture for automotive traceability," *Procedia CIRP*, vol. 97, pp. 390–395, 2021, 8th CIRP Conference of Assembly Technology and Systems. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212827120314785

[26] D. DiMase, Z. A. Collier, J. Carlson, R. B. Gray, Jr, and I. Linkov, "Traceability and risk analysis strategies for addressing counterfeit electronics in supply chains for complex systems," *Risk Anal*, vol. 36, no. 10, pp. 1834–1843, Jan. 2016.

[27] T. Ko, J. Lee, and D. Ryu, "Blockchain technology and manufacturing industry: Real-time transparency and cost savings," *Sustainability*, vol. 10, p. 4274, 11 2018.

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

[29] D. Bayer, S. Haber, and W. S. Stornetta, "Improving the efficiency and reliability of digital time-stamping," in *Sequences Ii*. Springer, 1993, pp. 329–334.

[30] W. Ethereum, "Ethereum whitepaper," 2014. [Online]. Available: https://ethereum.org/en/whitepaper/

[31] R. Colomo-Palacios, M. Sánchez-Gordón, and D. Arias-Aranda, "A critical review on blockchain assessment initiatives: A technology evolution viewpoint," *Journal of Software: evolution and process*, vol. 32, no. 11, p. e2272, 2020.

[32] P. Verhoeven, F. Sinn, and T. T. Herden, "Examples from blockchain implementations in logistics and supply chain management: exploring the mindful use of a new technology," *Logistics*, vol. 2, no. 3, p. 20, 2018.

[33] S. Huang, G. Wang, Y. Yan, and X. Fang, "Blockchain-based data management for digital twin of product," *Journal of Manufacturing Systems*, vol. 54, pp. 361–371, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612520300091

[34] B. Bailey and S. Sankagiri, "Merkle trees optimized for stateless clients in bitcoin," in *International Conference on Financial Cryptography and Data Security*. Springer, 2021, pp. 451–466.

[35] L. Ribeiro and O. Mendizabal, "Introdução à blockchain e contratos inteligentes," Tech. Rep., 2021.

[36] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *International journal of web and grid services*, vol. 14, no. 4, pp. 352–375, 2018.

[37] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Advances in Cryptology – CRYPTO 2017*, J. Katz and H. Shacham, Eds. Cham: Springer International Publishing, 2017, pp. 357–388.

[38] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi, "On security analysis of proof-of-elapsed-time (poet)," in *Stabilization, Safety, and Security of Distributed Systems*,

P. Spirakis and P. Tsigas, Eds. Cham: Springer International Publishing, 2017, pp. 282–297.

[39] S. D. Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, "Pbft vs proof-of-authority: applying the cap theorem to permissioned blockchain," in *Italian Conference on Cyber Security (06/02/18)*, January 2018. [Online]. Available: https://eprints.soton.ac.uk/415083/

[40] S. Aggarwal and N. Kumar, "Cryptographic consensus mechanisms," in *The Blockchain Technology for Secure and Smart Applications across Industry Verticals*, ser. Advances in Computers, S. Aggarwal, N. Kumar, and P. Raj, Eds. Elsevier, 2021, vol. 121, pp. 211–226. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0065245820300668

[41] I.-C. Lin and T.-C. Liao, "A survey of blockchain security issues and challenges," *International Journal of Network Security*, vol. 19, pp. 653–659, 09 2017.

[42] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[43] C. Cachin, S. Schubert, and M. Vukolić, "Non-determinism in byzantine fault-tolerant replication," *arXiv preprint arXiv:1603.07351*, 2016.

[44] D. Tapscott and A. Tapscott, "How blockchain will change organizations," *MIT Sloan Management Review*, vol. 58, no. 2, p. 10, 2017.

[45] S. Nandi, J. Sarkis, A. A. Hervani, and M. M. Helms, "Redesigning supply chains using blockchain-enabled circular economy and covid-19 experiences," *Sustainable Production and Consumption*, vol. 27, pp. 10–22, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352550920313592

[46] A. Prato, F. Mazzoleni, and A. Schiavi, "Metrological traceability for digital sensors in smart manufacturing: calibration of mems accelerometers and microphones at inrim," in *2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0 IoT)*, 2019, pp. 371–375.

[47] J. Grecuccio, E. Giusto, F. Fiori, and M. Rebaudengo, "Combining blockchain and iot: Food-chain traceability and beyond," *Energies*, vol. 13, no. 15, 2020. [Online]. Available: https://www.mdpi.com/1996-1073/13/15/3820

[48] K. M. Botcha, V. V. Chakravarthy, and Anurag, "Enhancing traceability in pharmaceutical supply chain using internet of things (iot) and blockchain," in *2019 IEEE International Conference on Intelligent Systems and Green Technology (ICISGT)*, 2019, pp. 45–453.

[49] A. Budak, A. Ustundag, M. S. Kilinc, and E. Cevikcan, *Digital Traceability Through Production Value Chain*. Springer International Publishing, 2018, pp. 251–265. [Online]. Available: https://doi.org/10.1007/978-3-319-57870-5_15

[50] L. Alves, E. F. Cruz, S. I. Lopes, P. M. Faria, and A. M. R. da Cruz, "Towards circular economy in the textiles and clothing value chain through blockchain technology and iot: A review," *Waste Management & Research*, vol. 40, no. 1, pp. 3–23, 2022, pMID: 34708680. [Online]. Available: https://doi.org/10.1177/0734242X211052858

[51] C. C. Agbo and Q. H. Mahmoud, "Comparison of blockchain frameworks for healthcare applications," *Internet Technology Letters*, vol. 2, no. 5, p. e122, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/itl2.122

[52] S. Zhang and J.-H. Lee, "Analysis of the main consensus protocols of blockchain," *ICT Express*, vol. 6, no. 2, pp. 93–97, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S240595951930164X

[53] S. Jabbar, H. Lloyd, M. Hammoudeh, B. Adebisi, and U. Raza, "Blockchain-enabled supply chain: analysis, challenges, and future directions," *Multimedia Systems*, vol. 27, no. 4, pp. 787–806, Aug 2021. [Online]. Available: https://doi.org/10.1007/s00530-020-00687-0

[54] O. UML and I. MOF, "The unified modeling language uml," 2011.

[55] S. Wang, G. Li, X. Yao, Y. Zeng, L. Pang, and L. Zhang, "A distributed storage and access approach for massive remote sensing data in mongodb," *ISPRS International Journal of Geo-Information*, vol. 8, no. 12, 2019. [Online]. Available: https://www.mdpi.com/2220-9964/8/12/533

Anexo A

Screenshots de apoio ao documento

Neste apêndice, alguns *screenshots* de apoio ao documento podem ser utilizadas como exemplos.

A.1 *MongoDB*

A figura A.1 mostra todos as coleções geradas pela REST API na base de dados *MongoDB*, enquanto que as figuras A.2 e A.3 mostram as coleções criadas pela ferramenta *gridFS* para guardar ficheiros.

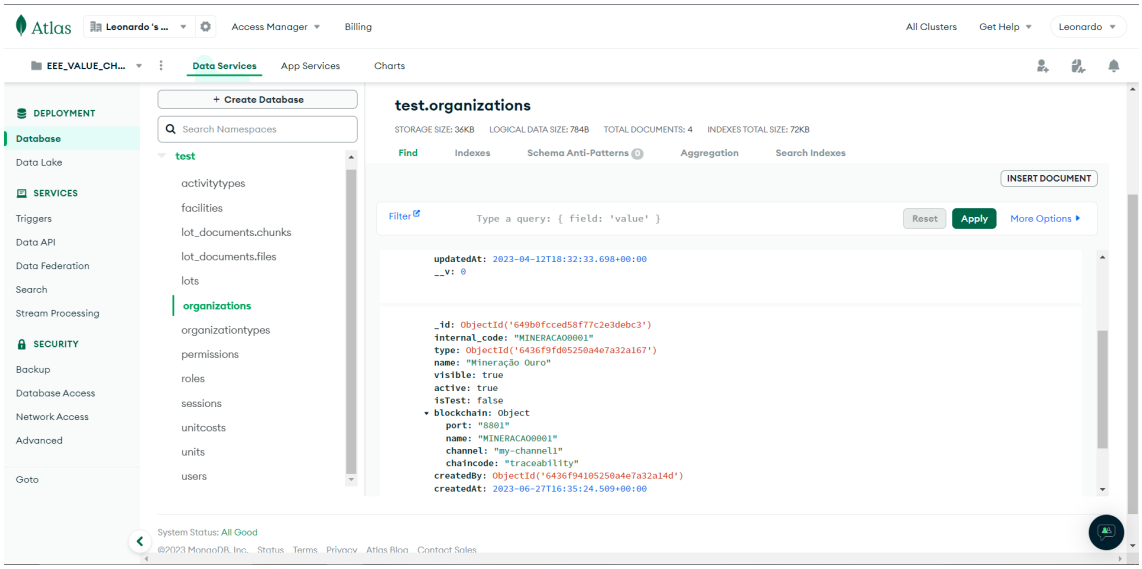

Figura A.1: Coleções do sistema na base de dados *MongoDB*.

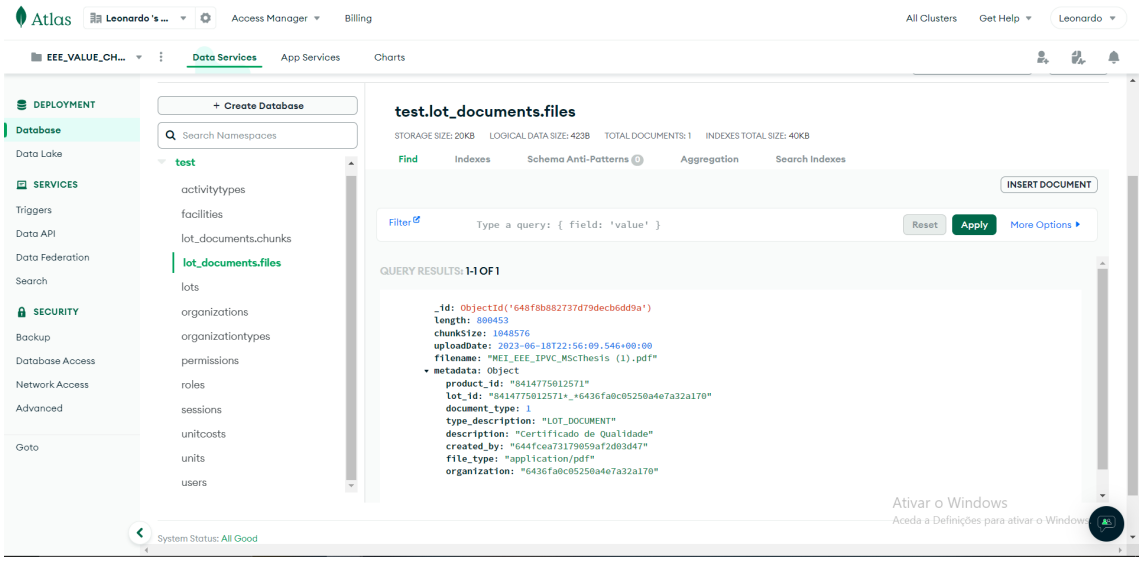

Figura A.2: Coleção dos cabeçalhos dos ficheiros criada pela ferramenta *GridFS*

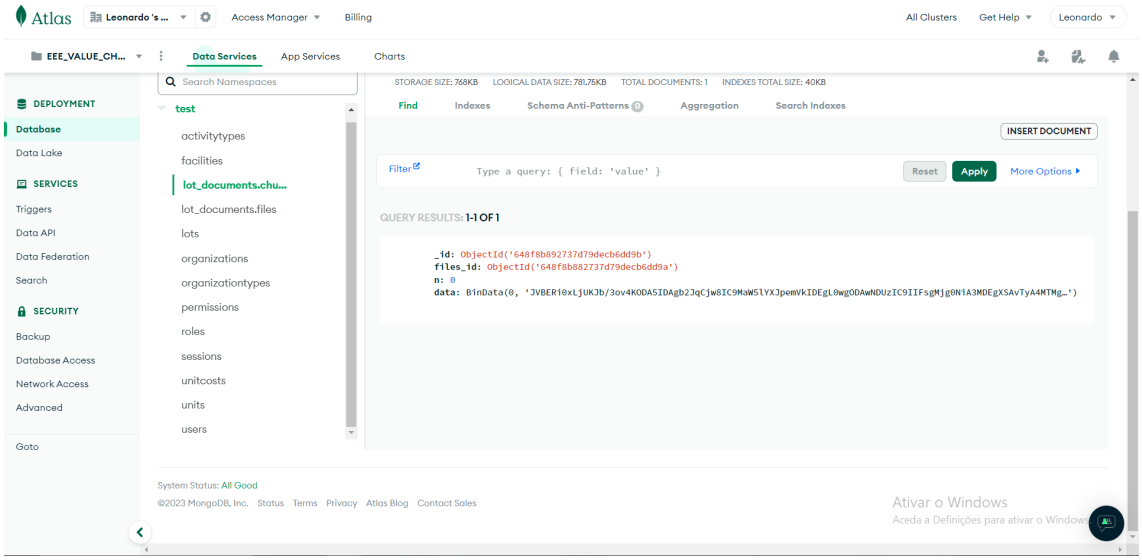

Figura A.3: Coleção de várias partes dos ficheiros criada pela ferramenta *GridFS*

A.2 Contentores *Docker*

Nesta secção estão presentes as figuras de apoio à dissertação relativamente à ferramenta *docker*. Esta ferramenta foi utilizada para levantar os vários componentes da *blockchain* configurada para este estudo de caso. A figura A.4 mostra todos os contentores necessários para a *blockchain* funcionar, enquanto a figura A.5 apenas mostra os contentores utilizados para gerar a API para as diferentes organizações acederem ao *smart contract*.

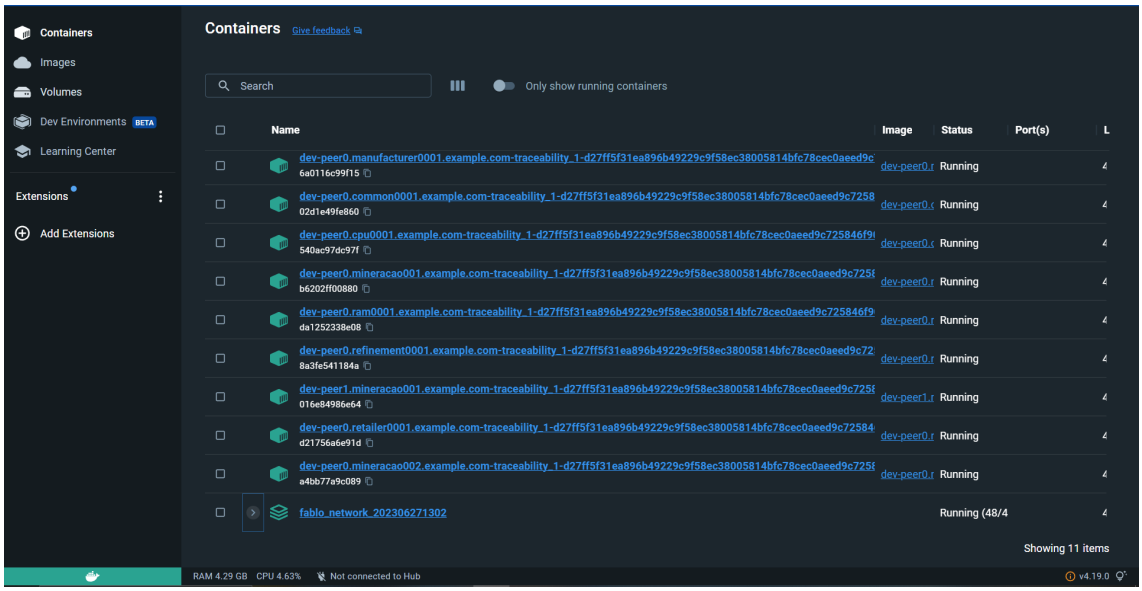

Figura A.4: Todos os contentores *Docker* para correr a *blockchain*

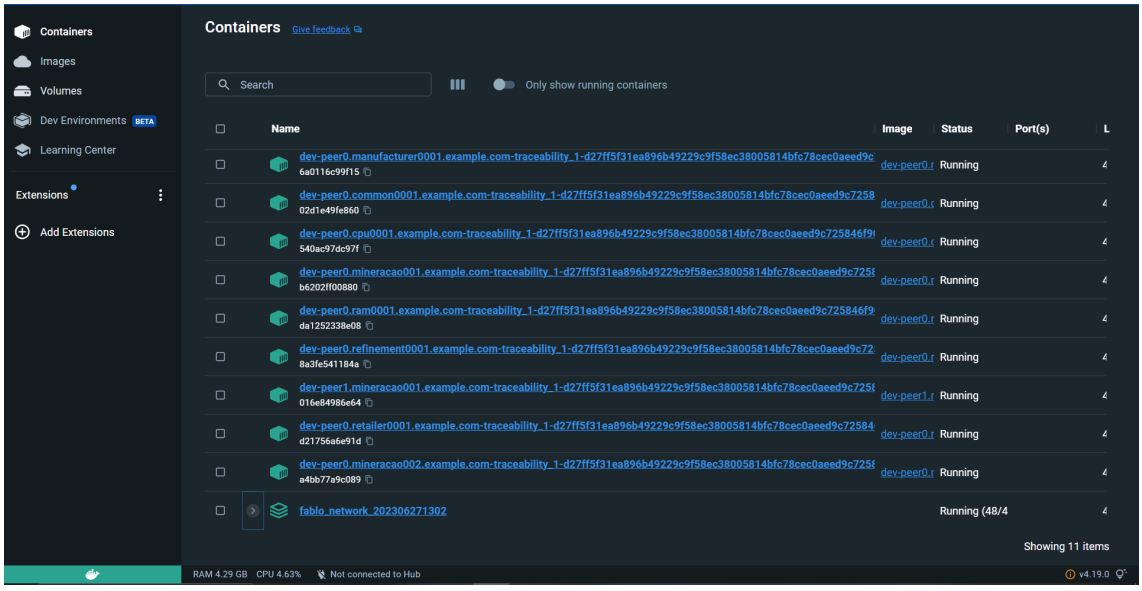

Figura A.5: Contentores *Docker* que correm as API das diferentes organizações

A.3 Aplicação Web

Para além de algumas funcionalidades demonstradas na secção 5.2, outras foram aqui demonstradas, nas figuras A.6, A.7, A.8, A.9, respetivamente é possível ver os vários passos o utilizador recuperar a palavra-passe. No que toca às figuras A.10 e A.11 são representados ecrãs que ambos servem para parametrizar a plataforma.

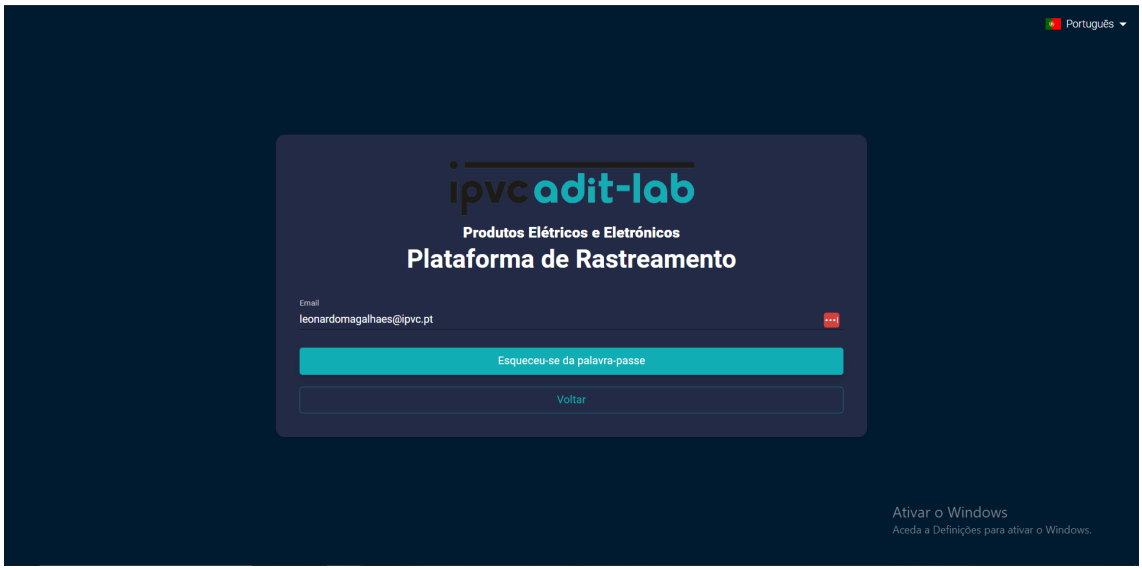

Figura A.6: Recuperar palavra-passe - Passo 1

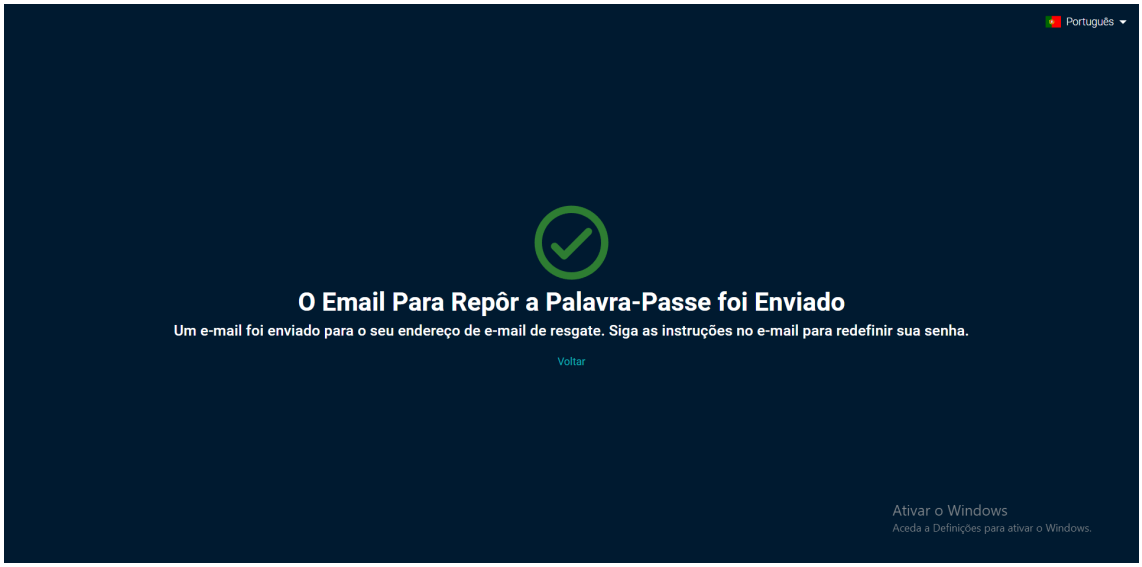

Figura A.7: Recuperar palavra-passe - Passo 2

A.4 Aplicação móvel

Nesta secção são demonstrados os ecrãs da aplicação móvel que não foram possíveis demonstrar na 5.2, na figura A.12 é possível ver os ecrãs referentes às atividades de transporte, na figura A.13 é possível ver os ecrãs referentes às atividades de receção, e por fim, na figura A.14 pode-se ver os ecrãs referentes às atividades de venda.

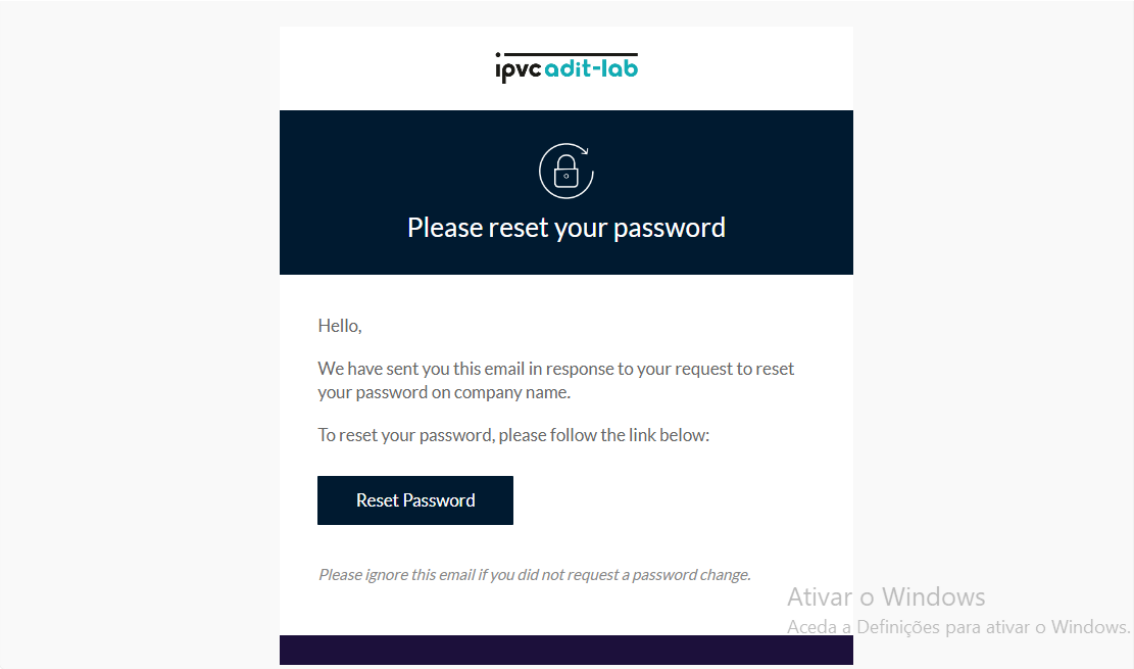

Figura A.8: Recuperar palavra-passe - Passo 3

Figura A.9: Recuperar palavra-passe - Passo 4

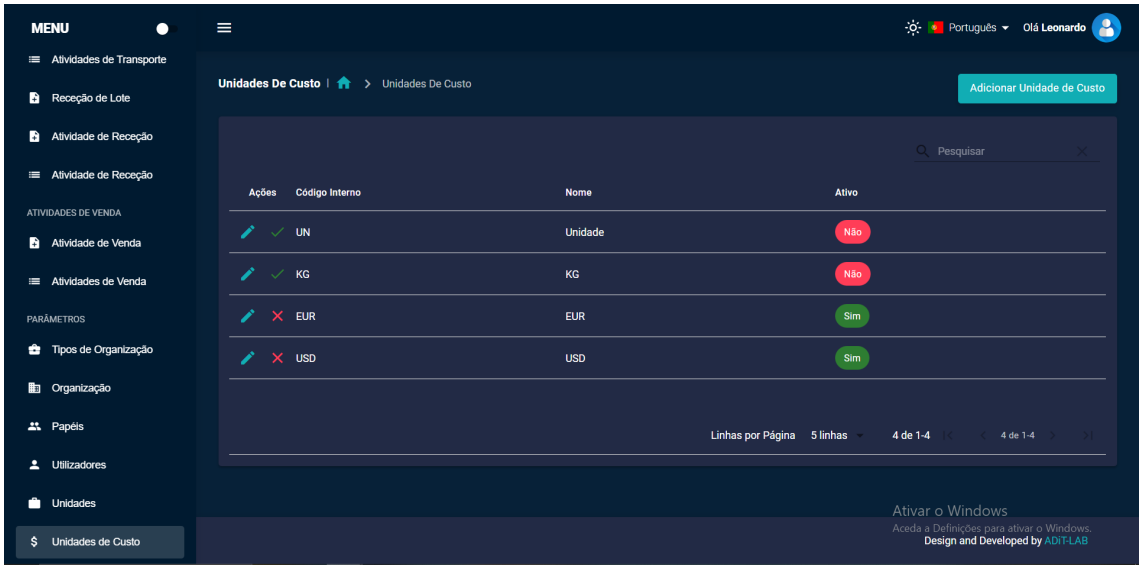

Figura A.10: Gerir unidades de custo da plataforma

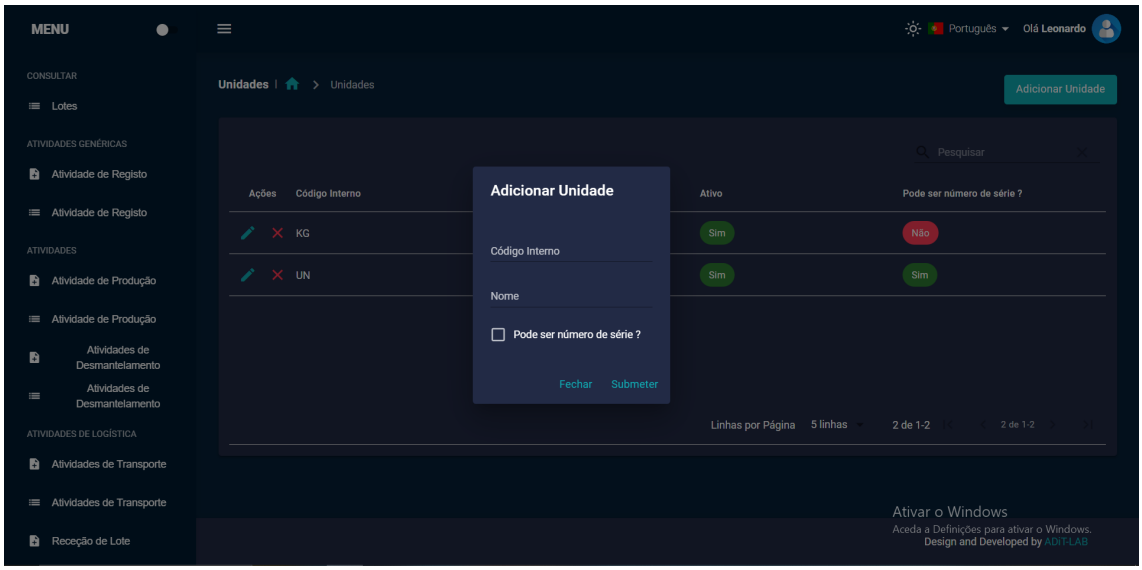

Figura A.11: Gerir unidades de medida da plataforma

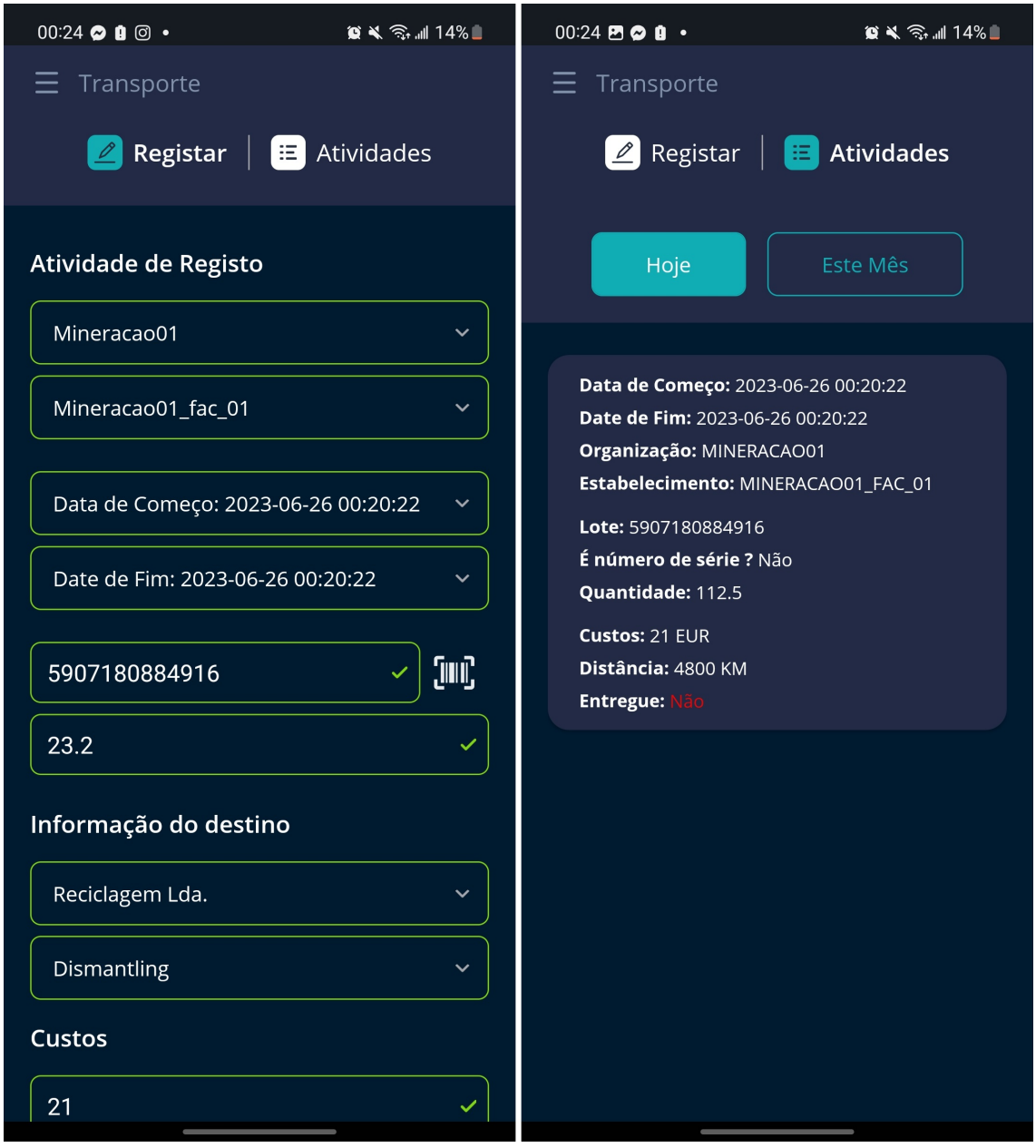

Figura A.12: Ecrã de criação e listagem de atividades de transporte.

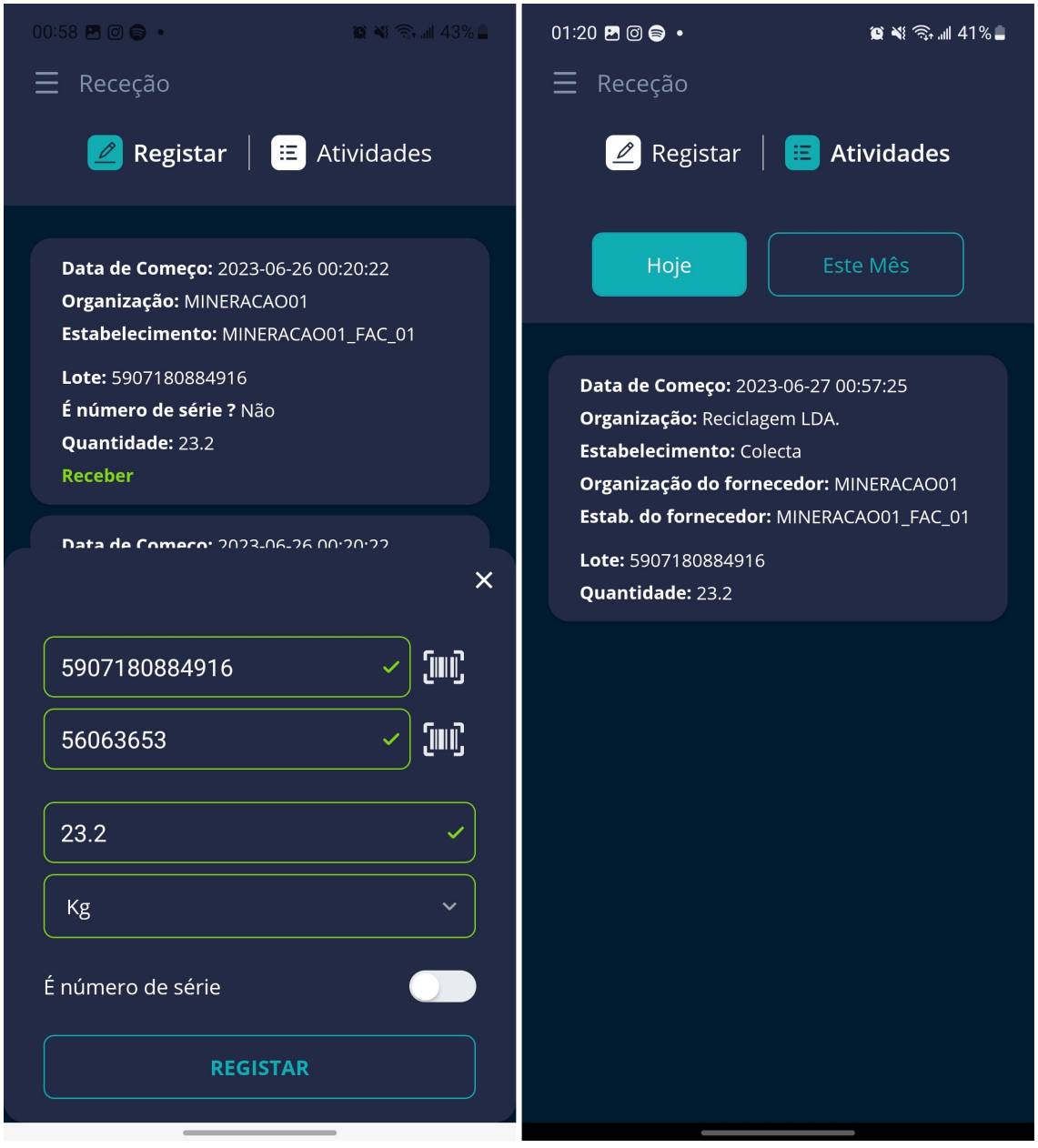

Figura A.13: Ecrã de criação e listagem de atividades de receção.

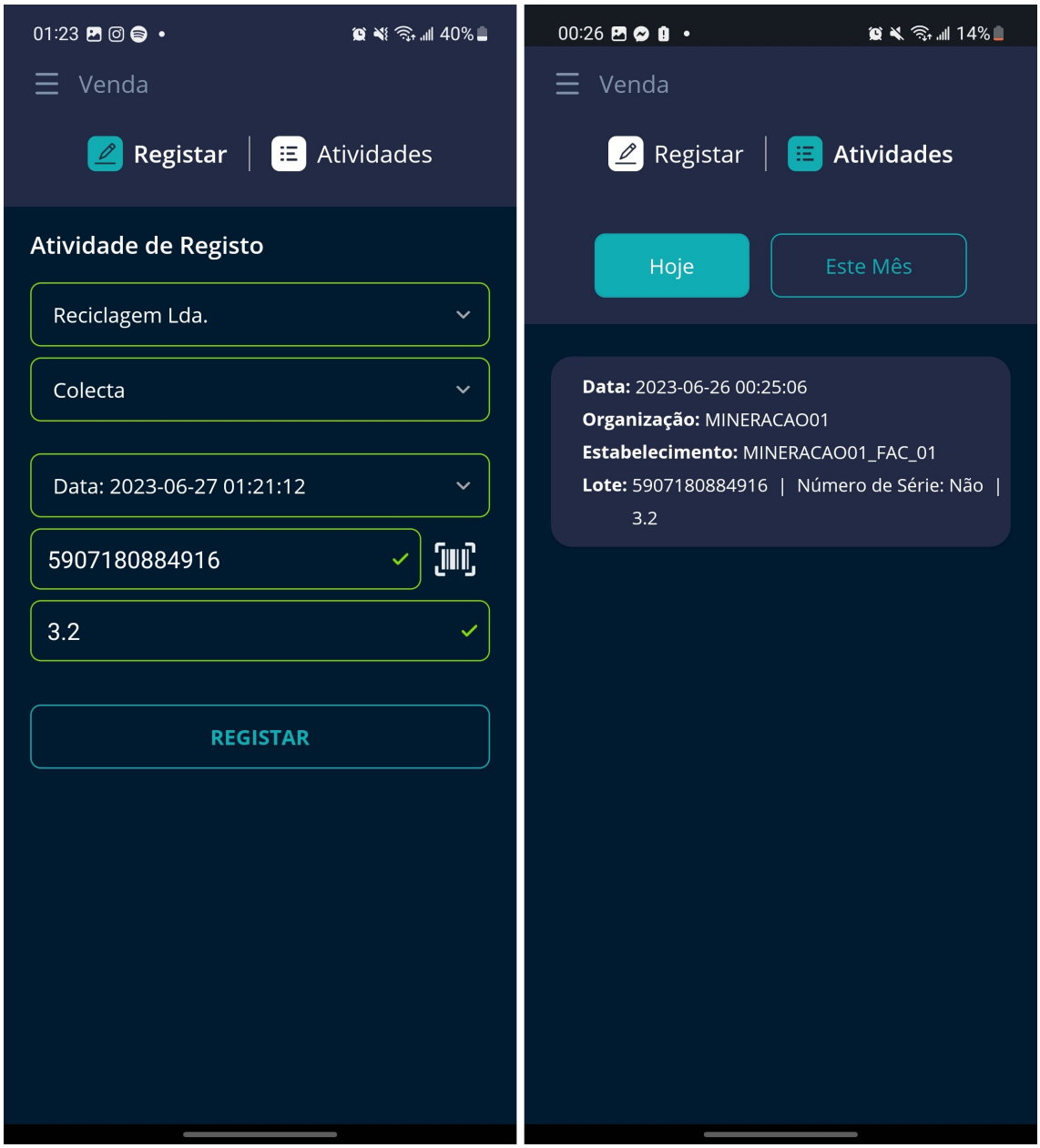

Figura A.14: Ecrã de criação e listagem de atividades de venda.

Anexo B

Criação de utilizador para integração

Neste apêndice vai ser demonstrado o processo de criação de um utilizador para permitir integrar determinados *endpoints* em sistema já existentes. O processo começa por uma determinada organização criar uma função (*role*) com as permissões para rotas definidas para integração. Estas rotas são verificadas por um *middleware* específico, ao contrário do *middleware* que valida todos os rotas utilizadas pelas aplicações *frontend* (representado no algoritmo 2). A diferença do *middleware* de integração para o outro, passa por ter que desencriptar a *API KEY* (esta é encriptada utilizando o algoritmo (*SHA512*) do utilizador (caso o utilizador possua uma) e validar posteriormente os dados, como por exemplo, verificar se o utilizador é para integrar, etc.

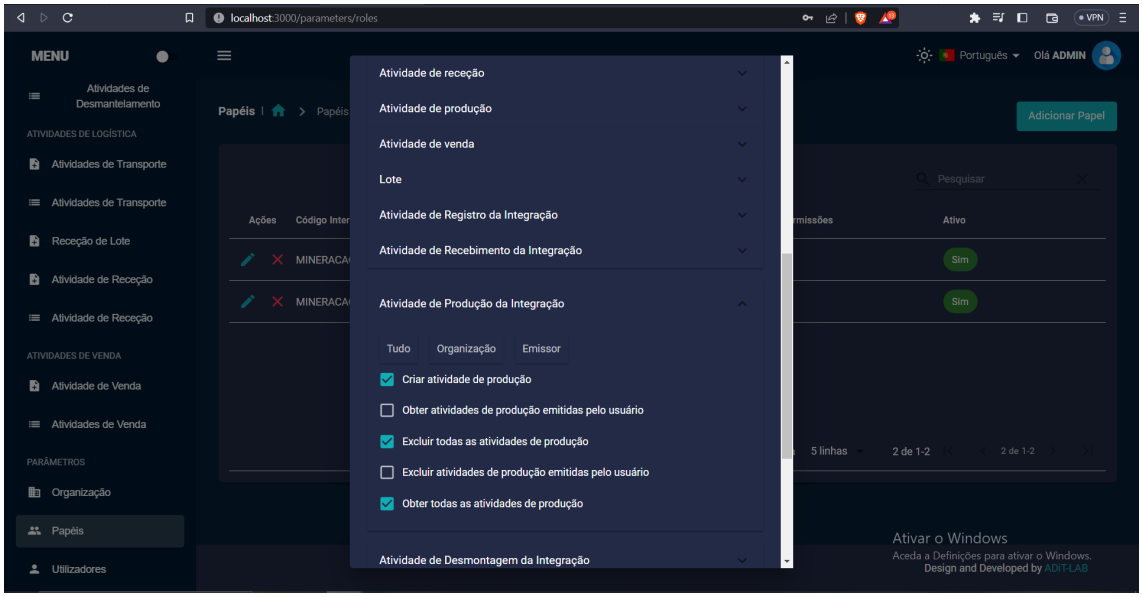

Figura B.1: Criar função (*role*) de integração

O processo de criar uma *API KEY* para integração, começa por criar um *role* com permissões para rotas definidas para integrar. No caso da figura B.1 é criada uma função

de utilizador com as rotas de integração para atividades de produção. Posteriormente, é criado um utilizador com esse *role* associado e com a opção ”Utilizador para integrar“ selecionada (consultar figura B.2), para ser enviado um *email* para o novo utilizador com a sua *API KEY* (figura B.3).

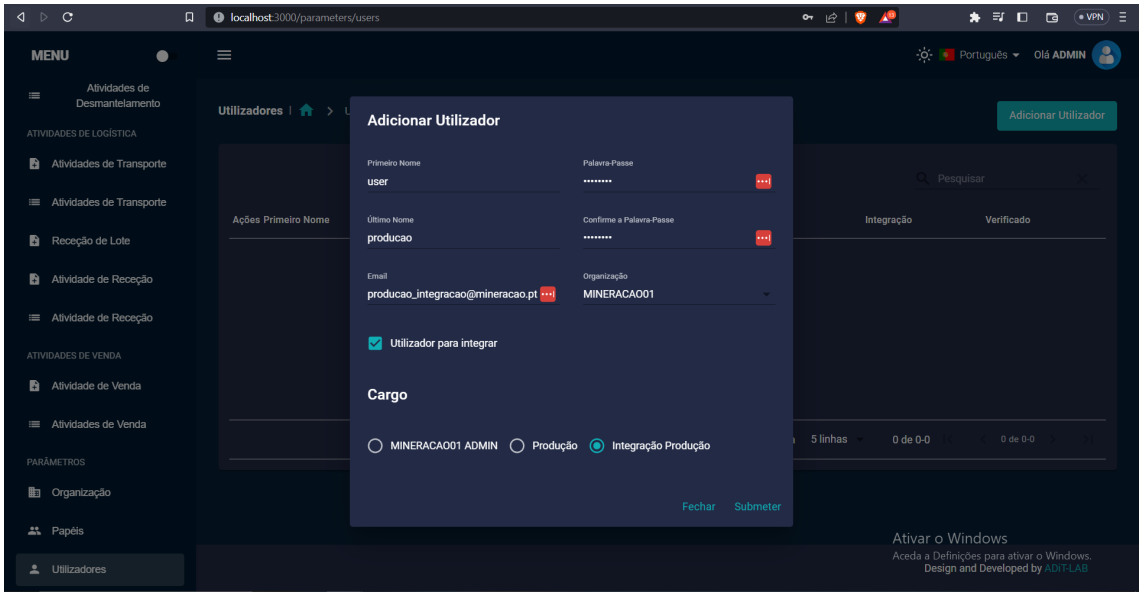

Figura B.2: Criar utilizador de integração

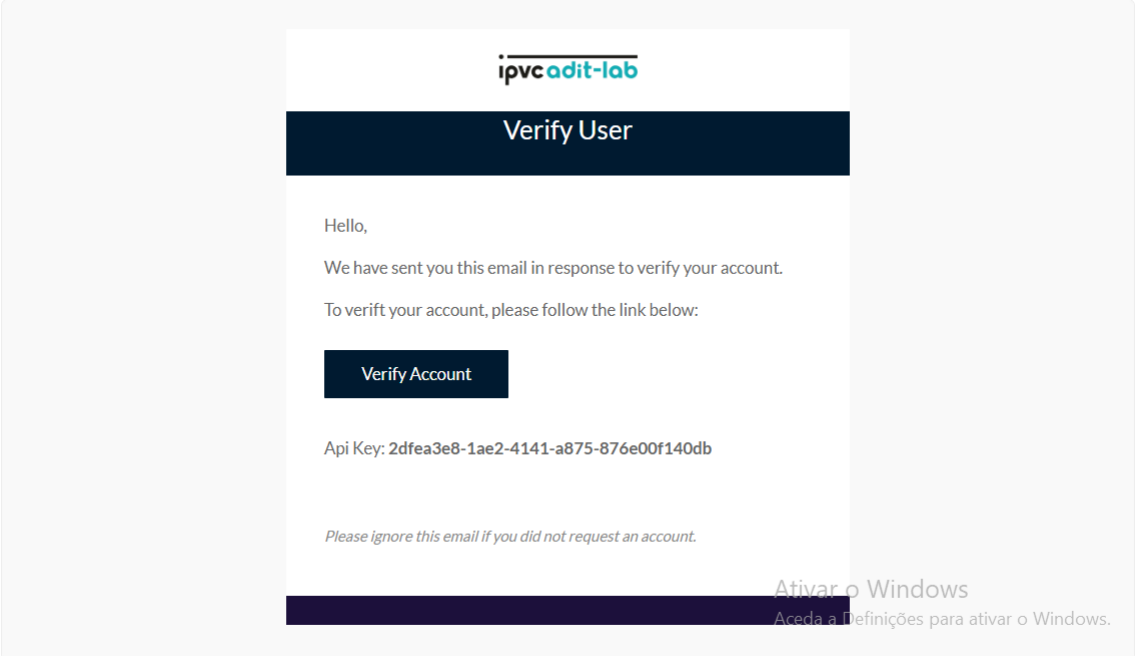

Figura B.3: Email enviado ao utilizador para integração

Após todo o processo de configuração da *API KEY* já será possível fazer pedidos às rotas definidas para integração. Neste caso foi configurado um *role* para apenas permitir

o acesso a rotas relacionadas com atividades de produção. Na figura B.4 é efetuado o pedido à rota de criação de uma atividade de produção, dada a *API KEY* enviada por *email* na figura B.3.

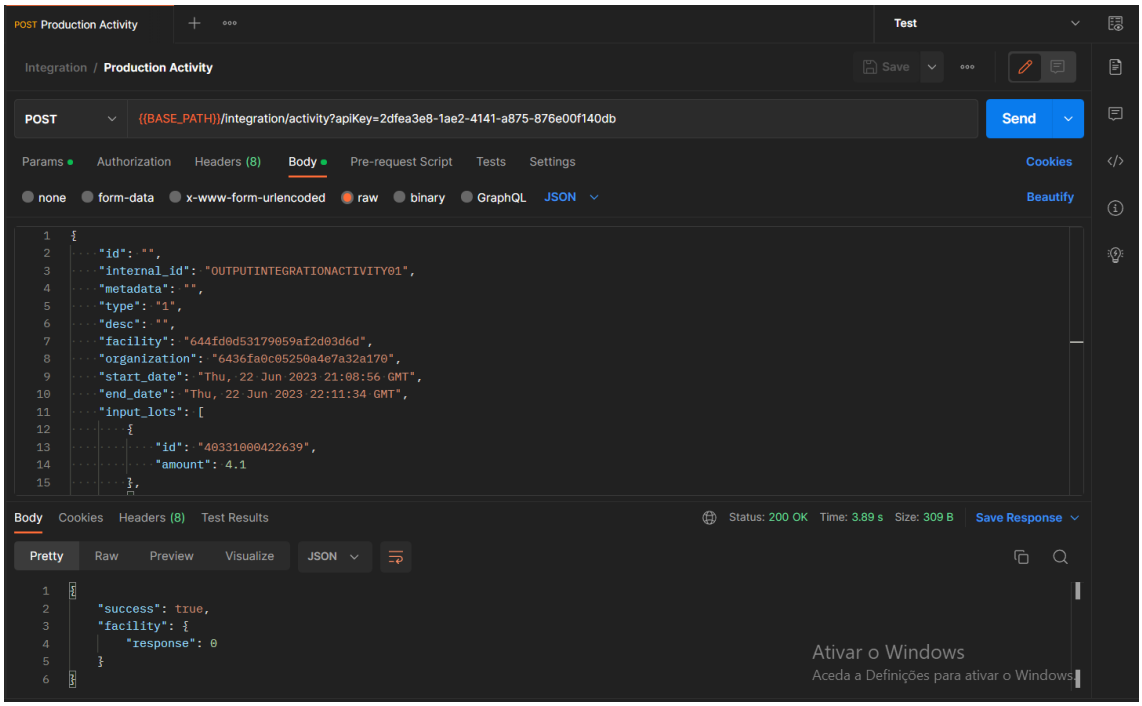

Figura B.4: Pedido efetuado pelo *Postman* à rota de integração

Como se pode ver na resposta do pedido efetuado na figura B.4, o mesmo devolveu resposta com código "0" que significa que a atividade foi adicionada com sucesso através do *smart contract*. Por fim, no código 3 é representado o *json* enviado no corpo do pedido da figura B.4.

```
{
    "id": "",
    "internal_id": "OUTPUTINTEGRATIONACTIVITY01",
    "metadata": "",
    "type": "1",
    "desc": "",
    "facility": "644fd0d53179059af2d03d6d",
    "organization": "6436fa0c05250a4e7a32a170",
    "start_date": "Thu, 22 Jun 2023 21:08:56 GMT",
    "end_date": "Thu, 22 Jun 2023 22:11:34 GMT",
    "input_lots": [
        {
```

```
            "id": "40331000422639",
            "amount": 4.1
        },
        {
            "id": "5907180884916",
            "amount": 5.3
        }
    ],
    "output_lots": [
        {
            "id": "OUTPUTINTEGRATIONOUTPUT01",
            "lot": {
                "internal_id": "OUTPUTINTEGRATIONOUTPUT01",
                "product": "PROD_INTEGRATION",
                "metadata": "",
                "amount": 3,
                "facility": "644fd0d53179059af2d03d6d",
                "organization": "6436fa0c05250a4e7a32a170",
                "unit": "644fd1713179059af2d03dbb",
                "type": 1,
                "serial_number": ""
            }
        }
    ]
}
```


Listing 3: Corpo (*body*) do pedido efetuado