João Ricardo Baptista dos Santos

BLOCKBATTLE
THE BEST BLOCKCHAIN WINS

1 2 9 0 UNIVERSIDADE D COIMBRA

UNIVERSIDADE D
COIMBRA

João Ricardo Baptista dos Santos

BLOCKBATTLE

THE BEST BLOCKCHAIN WINS

VOLUME 1

Dissertação no âmbito do Mestrado em Engenharia Informática, especialização em Engenharia de Software, orientada pelo Mestre Vítor Sousa e pelo Professor Doutor Paulo Alexandre Ferreira Simões e presente à Faculdade de Ciências e Tecnologia / Departamento de Engenharia Informática.

Junho de 2021

Faculdade de Ciências e Tecnologia da Universidade de Coimbra
Departamento de Engenharia Informática

Blockbattle

The best Blockchain wins

João Ricardo Baptista dos Santos

Dissertação no âmbito do Mestrado em Engenharia Informática, especialização em Engenharia de Software, orientada pelo Mestre Vítor Sousa e pelo Professor Doutor Paulo Alexandre Ferreira Simões e presente à Faculdade de Ciências e Tecnologia / Departamento de Engenharia Informática

Junho 2021

UNIVERSIDADE D
COIMBRA

This page is intentionally left blank.

Abstract

Blockchain technology operates on a decentralized peer-to-peer (P2P) network, whose nodes maintain a consistent replica of the ledger. The ledger stores groups of data in the form of block transactions. The hash generated from the content of each block allows you to reference it in the next one, creating an ordered sequence analogous to a linked list. The technology uses a consensus mechanism to ensure the consistency of the ledger among the respective nodes of its network. Given its architecture and the complexity of its mechanisms, it has scalability and performance limitations that make its application unfeasible in certain use cases.

The smart contracts mechanism allowed logic to be executed in the network nodes, expanding the application of the technology to several sectors. This dissertation is motivated by the use case of agri-food products certification. This is a traditionally bureaucratic process that presents a lack of autonomy in its stages and whose information presented to the stakeholders of its supply chain does not make transparent the legal norms that it enforces.

Therefore, the main objective of this dissertation is to study the application of blockchain technology in the certification of agri-food products. Its development aims to validate the implementation of the necessary logic through the smart contracts mechanism. Subsequently, and based on a performance analysis, it intends to conclude about the platform that best meets the requirements of the use case.

Keywords

Performance Analysis, Blockchain, Agri-Food Products Certification, Ethereum, Hyperledger

This page is intentionally left blank.

Resumo

A tecnologia *blockchain* opera numa rede descentralizada ponto-a-ponto (P2P), cujos nodos mantêm uma réplica consistente do *ledger*. O *ledger* armazena grupos de dados sob a forma de transações em blocos. A *hash* gerada a partir do conteúdo de cada bloco permite referenciá-lo no próximo, criando uma sequência ordenada análoga a uma lista ligada. A tecnologia recorre a um mecanismo de consenso para garantir a consistência do *ledger* entre os respetivos nodos da sua rede. Face à sua arquitetura e à complexidade dos seus mecanismos, apresenta limitações de escalabilidade e desempenho que inviabilizam a sua aplicação em determinados casos de uso.

O mecanismo de *smart contracts* permitiu que fosse executada lógica nos nodos da rede, expandindo a aplicação da tecnologia a diversos setores. Esta dissertação é motivada pelo caso de uso da certificação de produtos agroalimentares. Este é um processo tradicionalmente burocrático que apresenta falta de autonomia nas suas etapas e cuja informação apresentada aos *stakeholders* da sua cadeia de abastecimento não transparece as normas legais que faz cumprir.

Assim sendo, o principal objetivo desta dissertação é o estudo da aplicação da tecnologia *blockchain* na certificação de produtos agroalimentares. O seu desenvolvimento visa validar a implementação da lógica necessária através do mecanismo de *smart contracts*. Posteriormente, e com base numa análise de desempenho, pretende concluir acerca da plataforma que melhor cumpre os requisitos do caso de uso.

Palavras-Chave

Análise de Desempenho, Blockchain, Certificação de Produtos Agroalimentares, Ethereum, Hyperledger

This page is intentionally left blank.

Agradecimentos

À Mammasita e ao Velho,

queria agradecer todo o esforço e sacrifício que sempre fizeram pela nossa família. Este sempre foi o ingrediente secreto para tudo o que consegui e, se o sacrifício também foi vosso, o mérito sempre o será.

À minha mãe mais nova,

amiga, irmã, só quero agradecer por teres cuidado de mim como um filho e por teres dado tudo de ti para me dar o melhor. O universo devia dar-te muito mais, mas um irmão que te ama, extremamente orgulhoso de ti, e grato, terás sempre.

Ao Zé e ao Cadinho,

serei eternamente grato por tudo o que me ensinam, transmitem, e sei que o caminho que traçar vos levará comigo enquanto "segundos pais".

À minha avó,

agradecer-lhe-ei um dia por tudo o que nunca lhe agradeci, e por continuar a suportar a minha caminhada pelo muro.

Ao Luís Dias, João Marcos, João Ruivo e Andreia Borges,

agradeço por terem entrado na minha vida, por a terem enchido de histórias e de vida, e por se tornarem pessoas que amo e que sei que estão sempre comigo.

À minha loira,

"tiro o chapéu"para agradecer a mulher incrível que és, todo o apoio que me deste, por teres conhecido o meu pior lado profissional e nunca teres desistido de mim, e acima de tudo por me teres, tanta vez, presenteado com esse teu sorriso.

À D. Lúcia, ao Sr. João e ao Marcelo,

quero agradecer por terem feito parte deste caminho e por terem cuidado de mim com o mesmo carinho e respeito de quem cuida um filho/irmão.

Ao Vítor Sousa,

(e à sua gata) agradeço todos os minutos que me ouviu, aconselhou, puxou por mim, que me permitiu errar, prometer, falhar, cumprir, mas acima de tudo por me ter permitido ser transparente e me ter ajudado a crescer.

À Ubiwhere,

agradeço o excelente projeto que me disponibilizou, juntamente com todo o apoio necessário. Ficarão as sábias palavras do A. Duarte, as reuniões semanais com a Andreia, as brincadeiras da Liliana e toda a boa disposição dos meus colegas estagiários.

À Universidade de Coimbra e ao Prof. Dr. Paulo Simões,

agradeço terem disponibilizado os meios que necessitei para realizar este mestrado, nomeadamente a orientação nesta dissertação. A instituição permitiu-me crescer brutalmente em termos académicos, proporcionou-me oportunidades profissionais, e por tudo isso, e mais, sou bastante grato.

This page is intentionally left blank.

Conteúdo

Siglas

API Interface de Programação de Aplicações. 26, 28, 30, 44, 54, 56, 57, 58

BFT *Byzantine Fault Tolerance*. 17, 18, 26

CFT *Crash Fault Tolerance*. 20, 26, 92

DAG *Directed Acyclic Graph*. 13, 30, 66, 67

DOP Denominação de Origem Protegida. 8

DPoS *Delegated Proof of Stake*. 16, 19, 92

ETG Especialidade Tradicional Garantida. 8

GHOST *Greedy Heaviest Observed Subtree*. 18

IPAC Instituto Português de Acreditação. 8

NIF Número de Identificação Fiscal. 47, 48, 49

PBFT *Pratical Byzantine Fault Tolerance*. 16, 17, 18, 19, 26, 55, 64, 70, 72, 92

PoA *Proof-of-Authority*. 18, 19, 34, 70, 72

PoET *Proof of Elapsed Time*. 18, 19, 26, 27, 33, 55, 64, 70, 72, 92

PoS *Proof of Stake*. 13, 16, 18, 19, 22, 25, 67, 92

PoW *Proof of Work*. 10, 13, 15, 16, 17, 18, 19, 23, 24, 25, 26, 33, 44, 66, 67, 92

REST Transferência Representacional de Estado. 26, 28, 44, 54, 56, 57, 58

RPCA *Ripple Protocol Consensus Algorithm*. 92

SCP *Stellar Consensus Protocol*. 92

SUT Sistema em Teste. xiii, 29, 31, 32, 44, 57, 58, 59, 65, 66, 67, 68, 69, 70, 71, 72

YAC *Yet Another Consensus*. 92

This page is intentionally left blank.

Lista de Figuras

This page is intentionally left blank.

Lista de Tabelas

This page is intentionally left blank.

Capítulo 1

Introdução

Blockchain é uma sub-categoria de *Distributed Ledger Technology* (DLT) caracterizada pela descentralização, imutabilidade, transparência e rastreabilidade que garante aos dados que armazena no seu *ledger*. [1] A imutabilidade sustenta que a informação à qual o utilizador acede não foi alterada por terceiros, a descentralização sustenta a mitigação dos riscos de fraude e de segurança derivados da centralização de dados, e a transparência e rastreabilidade sustentam que qualquer utilizador com acesso de leitura é capaz de consultar os dados armazenados.

Os dados encontram-se armazenados no *ledger* sob a forma de transações. A capacidade de um nodo participar no mecanismo de consenso para validar novas transações faz divergir as plataformas *blockchain* em *permissionless* (e.g., Bitcoin e Ethereum) - qualquer nodo com acesso à rede pode participar no mecanismo de consenso - e *permissioned* (e.g., R3 Corda e Hyperledger Sawtooth) - podem ser restringidos os nodos que têm acesso à rede para participar no mecanismo de consenso. [1]

A tecnologia inicialmente direcionou-se ao setor financeiro, no entanto, apresenta-se atualmente integrada em diversas provas de conceito para múltiplos casos de uso. As suas limitações técnicas, como a privacidade [2], escalabilidade [1] e desempenho [1], limitam a sua aplicação em soluções empresariais de larga escala. A sua principal desvantagem é o desempenho quando comparado com o de tecnologias centralizadas. [1]

Estudos emergentes investigam soluções de escalabilidade [3] [4] e desempenho, e também soluções para desenvolver padrões na indústria que permitam realizar *benchmarking* destas plataformas. [5] [6]

O principal objetivo desta dissertação é o estudo da aplicação da tecnologia na certificação de produtos agroalimentares. Este objetivo encontra-se repartido por dois objetivos que são: implementação da lógica requerida pelo processo de certificação recorrendo ao mecanismo de *smart contracts* e a análise do desempenho de plataformas *blockchain*. Esta análise deve considerar o modelo de dados definido anteriormente, e deverá permitir concluir acerca da plataforma que melhor cumpre os requisitos deste caso de uso.

1.1 Contexto, Motivação e Oportunidades

Fatores como a globalização das cadeias de abastecimento [7], o desenvolvimento da preocupação por comércio ético e práticas agrícolas sustentáveis, e a crescente preocupação dos consumidores com a qualidade dos produtos alimentares, impulsionaram a certificação no setor agroalimentar. A certificação visa a qualificação das características ou dos processos de produção primários de um produto. De acordo com o serviço pretendido, o processo verifica se determinadas normas legais estão a ser cumpridas. [8]

No entanto, a escalabilidade que as cadeias de abastecimento sofreram através da globalização, juntamente com a escassez de digitalização na época, tornou a certificação num processo tradicional burocrático. Este processo, além de não digitalizado, apresenta-se como não sustentável e não automatizado. As limitações de dimensão da rotulagem, onde a certificação é apresentada, não permitem que seja apresentada informação do significado daquele símbolo, do conjunto de normas que faz cumprir ou das evidências que a entidade de produção apresentou à entidade certificadora. Assim, a certificação caracteriza-se também como um processo pouco transparente e rastreável. Apesar de emergirem soluções tecnológicas que procuram mitigar os problemas identificados através da digitalização dos dados, estes provêm de sistemas centralizados onde é possível que sejam alterados sem o conhecimento das restantes entidades da cadeia de abastecimento.

A Ubiwhere vem continuamente a integrar tecnologia *blockchain* nos seus projetos, nomeadamente no projeto SmartAgriChain. Da conceção deste projeto surgiu o caso de uso desta dissertação. Este consiste na digitalização do processo de certificação de produtos agroalimentares através dos mecanismos da tecnologia *blockchain*. O estudo desta aplicação visa mitigar os riscos/problemas identificados no respetivo processo.

1.2 Objetivos

O principal objetivo desta dissertação consiste no estudo da aplicação da tecnologia *blockchain* à certificação de produtos agroalimentares. Desta forma, o seu desenvolvimento engloba dois objetivos que permitirão contribuir para este estudo.

Inicialmente é pretendido que seja definido um modelo de dados que permita a implementação das estruturas de dados necessárias às aplicações *blockchain* (i.e., *smart contracts* ou terminologias semelhantes nas diferentes plataformas). O modelo de dados deve integrar a lógica necessária ao caso de uso considerando a possível integração das aplicações desenvolvidas numa rede em produção. Deste modo, deve validar o cumprimento dos requisitos necessários ao processo de certificação. As aplicações devem ser desenvolvidas para serem executadas nas plataformas *blockchain* que irão ser definidas para analisar.

Posteriormente, deve ser realizada uma análise de desempenho às plataformas *blockchain* a definir. Esta análise requer o seu design com a devida definição do plano de testes a executar. Este objetivo inclui a respetiva implementação das redes que suportaram a análise de desempenho e a implementação dos *workloads* que estas executaram. As aplicações desenvolvidas são integradas na execução desta análise, permitindo a posterior correlação entre os resultados desta e os requisitos do caso de uso.

Através do desenvolvimento desta dissertação, e do cumprimento destes objetivos, expecta-se que seja concluído acerca da plataforma que melhor se adequa aos requisitos do caso de uso.

1.3 Perguntas de Investigação

Quais os aspetos que obstruem o desempenho de uma plataformas *blockchain*?

O **Capítulo 5** analisa os resultados obtidos na análise de desempenho realizada em redes das plataformas *blockchain* definidas no **Capítulo 2**. Esta análise permite concluir acerca dos possíveis aspetos que obstruem o desempenho das plataformas, com base nas métricas recolhidas durante a execução do seu plano de testes e descritas no **Capítulo 4**. As respetivas conclusões encontram-se sintetizadas no **Capítulo 6**.

Qual a plataforma *blockchain* que melhor se adequa aos requisitos do caso de uso?

Com base na análise do desempenho, descrita no **Capítulo 6**, das plataformas *blockchain* e na resposta à primeira pergunta de investigação desta dissertação, será concluído acerca da plataforma que melhor cumpre os requisitos do caso de uso. As respetivas conclusões encontram-se sintetizadas no **Capítulo 6**.

1.4 Instituição Acolhedora

Ubiwhere é uma empresa de *software* fundada em 2007 em Aveiro, onde se encontra a sua sede. O *software* por si desenvolvido tem como especialização as áreas das *Smart Cities*, Telecomunicações e Internet do Futuro e Novas Tecnologias. A empresa mantém uma aposta contínua em assegurar as melhores práticas ao nível da gestão da qualidade tendo sido reconhecida através de certificações como a ISO9001 e CMMI Nível 3.

Zenithwings

Zenithwings é uma empresa *spin-off* da Ubiwhere cujo foco é a Investigação e Inovação (I&I) de soluções tecnológicas especializadas nas áreas da Agricultura de Precisão e Indústria 4.0. Pretende automatizar a produção agrícola e desenvolver soluções de rastreabilidade para sistemas integrantes na indústria logística.

WineChain foi um dos projetos da empresa nestas áreas que integrou blockchain para armazenar os detalhes de produção de vinhos. O seu objetivo era garantir rastreabilidade e segurança na informação dada aos consumidores.

O projeto em desenvolvimento SmartAgriChain visa ser um *marketplace* de serviços para o setor agroalimentar, tendo proporcionado a Proposta de Estágio desta dissertação. Uma das suas funcionalidades será a criação de certificados de qualidade dinâmicos, fazendo esta gestão *off-chain*, e procedendo posteriormente ao armazenamento dos dados na plataforma *blockchain* Hyperledger Sawtooth.

1.5 Estrutura do Documento

O presente documento visa relatar toda a pesquisa e trabalho desenvolvido durante o estágio académico, inserido no plano curricular do ano letivo 2020/2021 do Mestrado em Engenharia Informática, com especialização em Engenharia de Software. A sua organização é feita segundo capítulos, os quais estão estruturados da seguinte forma:

Capítulo 2 - Estado de Arte. Neste capítulo são apresentados os resultados da investigação dos conceitos e problemas inerentes aos temas desta dissertação. É ainda realizada a comparação das plataformas Blockchain existentes no mercado;

Capítulo 3 - Planeamento e Metodologia. Neste capítulo são apresentadas e descritas o conjunto de tarefas planeadas para o cumprimento dos objetivos desta dissertação, bem como a descrição da metodologia de desenvolvimento seguida. São enumerados os desvios que existiram face ao planeamento inicial e é descrita a análise dos riscos;

Capítulo 4 - Design da Análise de Desempenho. Neste capítulo é descrito o design da análise de desempenho que suporta o desenvolvimento desta dissertação. É assim descrita a metodologia a seguir, o conjunto de métricas e variáveis utilizadas, o modelo de dados implementado, os ambientes de teste e os *workloads* a executar;

Capítulo 5 - Análise de Resultados. Neste capítulo é feita a análise dos resultados obtidos pela execução da análise de desempenho. Esta análise segue o design descrito no capítulo anterior. Inicalmente são apresentados os problemas enfrentados e posteriormente a sua análise divide-se de acordo com os diferentes *workloads* e configurações utilizadas;

Capítulo 6 - Conclusão. Neste capítulo são apresentadas as conclusões retiradas do trabalho desenvolvido e é dada resposta às perguntas de investigação formuladas para esta dissertação;

Anexo A - Proposta de Estágio, proposta de estágio apresentada pela empresa orientadora deste estágio curricular;

Anexo B - Tabela Comparativa de Plataformas *Blockchain*, tabela comparativa das plataformas *blockchain* existentes;

Anexo C - Exemplo de Certificado de Qualidade Diferenciada, exemplo de um certificado de qualidade diferenciada.

Capítulo 2

Estado de Arte

O estado de arte desta dissertação consiste na descrição do conhecimento adquirido, face aos temas inerentes à mesma. Deste modo, pretende-se introduzir os conceitos relacionados com o caso de uso - certificação digital de produtos no setor agroalimentar - e de seguida introduzir a tecnologia *blockchain*, fazendo uma posterior comparação de plataformas *blockchain* existentes. Por fim, é feita a introdução dos conceitos de análise de desempenho de plataformas *blockchain*, e a análise de trabalhos relacionados.

2.1 Setor Agroalimentar

As entidades regulamentares do setor agroalimentar são responsáveis por garantir que os seus consumidores dispõem de recursos alimentares suficientes, saudáveis, seguros, nutricionalmente ricos, economicamente acessíveis e produzidos de forma sustentável, face às suas necessidades. [9] Na sua constituição encontram-se complexas cadeias de abastecimento que permitem o fluxo desde a produção até à venda do produto ao consumidor final.

A União Europeia, ao ano de 2019, aumentou o volume de negócios do setor face ao ano anterior, totalizando 270,5 mil milhões de euros - 151,2 mil milhões de euros em exportações e 119,3 mil milhões de euros em importações. [10] Portugal, em 2018, segundo dados estatísticos do INE [11], aumentou o volume de negócios comparativamente ao ano anterior, totalizando 17 mil milhões de euros.

Atualmente vêm a ser desenvolvidas soluções tecnológicas que apostam na garantia de qualidade, segurança e rastreabilidade dos alimentos na indústria agroalimentar. O termo Indústria 4.0 [12] caracteriza o conjunto de métodos tecnológicos que permitem armazenar e processar dados de IoT, robótica e sensores, com o objetivo de acelerar ou agilizar o processo industrial. Agricultura de Precisão [12] é o conjunto de técnicas com aplicação nas explorações agrícolas que permitem aumentar a segurança das decisões agronómicas na exploração agrícola, e consequentemente aumentar a produtividade das parcelas e reduzir os custos de produção e impactos ambientais.

O caso de uso desta dissertação surge do desenvolvimento de projetos pela Zenithwings nas áreas da Indústria 4.0 e da Agricultura de Precisão, sendo estes aliciados pela dimensão e contínua evolução da indústria agroalimentar na União Europeia, como referenciado.

2.1.1 Cadeias de Abastecimento

A globalização [7] permitiu às cadeias de abastecimento expandirem-se à escala global, quebrando fronteiras e descentralizando a produção. Também o desenvolvimento da indústria de retalho contribuiu para que as cadeias de abastecimento integrassem um grande número de intermediários, processos burocráticos e operações complexas.

Cadeia de abastecimento [13] é o conjunto de todas as entidades envolvidas desde o processo de produção de um produto até à sua venda ao seu consumidor final. Este processo segue uma lógica bidirecional e infinita, uma vez que a economia de uma empresa encontra-se em constante movimentação e dado que, embora a movimentação do produto siga do fornecedor até ao consumidor final, existe uma constante comunicação no sentido inverso (p.e. processos de devoluções, trocas, entre outros). O conjunto de entidades que a constitui pode variar consoante a dimensão da empresa, as necessidades de matérias-primas para a produção do produto final, ou processamento das mesmas, e/ou a estratégia de venda.

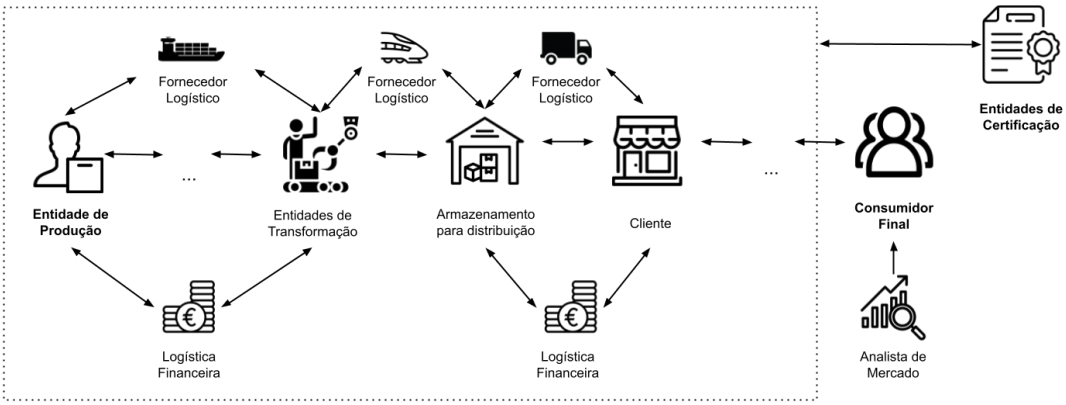


Figura 2.1: Representação de uma cadeia de abastecimento do setor agroalimentar

A **Figura 2.1** representa as entidades que integram, ou poderão integrar, uma cadeia de abastecimento do setor agroalimentar, bem como o fluxo bidirecional de matéria e financeiro entre estas. A continuidade de clientes que é apresentada até ao consumidor

final considera a indústria de retalho (i.e., intermediários grossistas e retalhistas) em cadeias complexas. Por outro lado, a continuidade que é apresentada entre as entidades de produção e as entidades de transformação considera o conjunto das várias entidades de produção e/ou de transformação no fluxo de matérias-primas de produtos complexos (i.e., um produto que requer diversas fontes de matéria-prima ou processos de transformação).

Esta dissertação define as entidades de produção, as entidades de certificação e o consumidor final enquanto *stakeholders* do seu caso de uso. Embora tenha sido uma decisão da instituição acolhedora, baseia-se nos *stakeholders* pretendidos para demonstrar, enquanto prova de conceito, a aplicabilidade da tecnologia ao caso de uso da certificação digital de produtos agroalimentares.

2.2 Certificação no Setor Agroalimentar

Nos Estados Unidos da América e na União Europeia, a certificação no setor agroalimentar foi impulsionada pela globalização [7], pela crescente preocupação por comércio ético e práticas agrícolas sustentáveis, e pela procura por qualidade diferenciada dos produtos agroalimentares. Nos países sub-desenvolvidos os fatores impulsionadores foram doenças transmitidas por alimentos e controvérsias com alimentos geneticamente modificados. [8]

Certificação [8], no setor agroalimentar, é o processo de atestar explicitamente a conformidade de um produto ou processo ou característica(s) do mesmo, segundo um determinado conjunto de normas legislativas e/ou especificações técnicas. Na **Figura 2.1** é possível observar a capacidade de serem certificados produtos ou processos nos vários constituintes da cadeia de abastecimento do setor agroalimentar.

Phil Hogan, no relatório da indústria agroalimentar de 2018 da União Europeia, [14] refere a ligação do sucesso da indústria agroalimentar com a reputação dos produtos produzidos de forma sustentável e seguindo elevados padrões de segurança e qualidade. Assim, algumas das vantagens resultantes da certificação de produtos no setor agroalimentar são: [8]

- Marketing implícito à rotulagem específica de certificação de qualidade diferenciada que, inerentemente, permite a diferenciação do produto pelos *stakeholders*;
- Possibilidade de integração de produtores em determinados nichos e/ou segmentos de mercado (p.e., mercado biológico onde a certificação assegura a legitimidade do produto e processo de produção);
- Aumento da eficiência das cadeias de abastecimento:
  - Diminuição dos custos para os retalhistas derivado, por exemplo, da minimização do risco de defeito de um produto e consequente desperdício;
  - Diminuição dos custos de produção derivado do desenvolvimento de práticas de produção sustentáveis.

Eventualmente, uma das maiores vantagens da certificação seria aumentar as oportunidades económicas no mercado para produtos agroalimentares certificados. No entanto, face ao elevado volume destes no mercado, a certificação deixou de ser um fator de diferenciação para se tornar num requisito para as entidades de produção manterem determinado produto no mercado. [8]

2.2.1 Entidades de Certificação

Entidades de certificação são entidades públicas ou privadas, devidamente acreditadas, que disponibilizam serviços nos quais são responsáveis por auditar e atestar a segurança e/ou qualidade diferenciada de um produto ou processo ou característica(s) do mesmo, segundo um determinado conjunto de normas legislativas e/ou especificações técnicas. [8] [15] Encarregam-se de estabelecer os requisitos necessários aos vários serviços que disponibilizam, bem como a consequente verificação da sua conformidade no decorrer desse.

Os serviços de certificação que disponibilizam encontram-se devidamente acreditados e com a evolução do são mercado estes inúmeros e categorizados de diversas formas. No entanto, na certificação de produtos agroalimentares destacam-se, por exemplo, a certificação Denominação de Origem Protegida (DOP), ou Especialidade Tradicional Garantida (ETG).

A acreditação é o processo onde uma única organização autorizada avalia e reconhece formalmente que uma entidade, neste caso de certificação, detém competências técnicas para efetuar atividades específicas de avaliação de conformidade. [8] Visa assegurar a existência de determinado nível de competências técnicas mínimas, reconhecidas internacionalmente, transmitindo confiança na execução das suas atividades. No caso da União Europeia cada Estado-Membro designou um único organismo de acreditação (p.e., em Portugal é o Instituto Português de Acreditação (IPAC) segundo o Decreto-Lei n.º 23/2011) que se rege por normas internacionais, permitindo a existência de Acordos de Reconhecimento Internacional e o cumprimento do Regulamento (CE) n.º 765/2008.

2.2.2 Processo de Certificação

Esta dissertação, por decisão da instituição acolhedora, foca-se apenas no processo de certificação de produtos no setor agroalimentar. Neste, a entidade de certificação verifica a conformidade do produto face aos requisitos necessários do serviço de certificação pretendido. Aquando da conformação, é emitido um certificado de qualidade, viabilizando a entidade de produção de divulgar o logótipo da certificação na rotulagem dos seus produtos. Tipicamente, o fluxo do processo de certificação é o seguinte: [8]

1. A entidade de produção faz um requerimento por determinado serviço à entidade de certificação, preenchendo um questionário dedicado;
2. A entidade de certificação revê e pré-avalia a documentação e, aceitando o respetivo pedido, inicia um processo de certificação;
3. De acordo com o serviço de certificação requerido, a entidade de produção necessita de apresentar determinada documentação (i.e., evidências) relativa ao seu produto, unidades e/ou processos de produção;
4. A entidade de certificação vai continuamente revendo e validando a documentação entregue. Caso seja rejeitada documentação, é também rejeitado o processo;
5. Aquando da validação de toda a documentação, é realizada uma auditoria por parte da entidade de certificação à entidade de produção;
6. Ao verificar a conformidade de todos os requisitos, é estabelecido um contrato de prestação de serviços de controlo e certificação entre as entidades, e no qual se comprometem a cumprir a legislação em vigor.

Esta certificação pode, a qualquer instante, ser cancelada por qualquer uma das partes, seguindo-se os procedimentos contratuais para o efeito. Por outro lado e numa fase inicial, caso exista um pedido de orçamento, a entidade de produção envia igualmente os dados necessários num questionário dedicado ao serviço de certificação pretendido. No entanto, é posteriormente enviado o orçamento à entidade de produção, que tem a possibilidade de validar o mesmo e prosseguir o processo de certificação.

A certificação atribuída detém um prazo de validade durante o qual podem ser realizadas auditorias periódicas com o objetivo de verificar a conformidade com as normas estabelecidas pelo serviço de certificação. [15] Caso sejam detetadas discrepâncias com as normas durante estas auditorias, e de acordo com o seu grau de gravidade ou importância, a certificação pode ser suspensa ou eventualmente revogada. Perante a expiração da validade da mesma, ou eventualmente antes, é necessário proceder à sua renovação.

As ações, mencionadas no anterior fluxo do processo, requere-se que sejam possíveis de executar pelos *stakeholders* nos resultados do desenvolvimento desta dissertação, tanto nas redes *blockchain* aplicadas ao caso de uso, como posteriormente na prova de conceito.

2.2.3 Certificação Digital

O processo de certificação apresenta-se tradicionalmente burocrático, dependente de ação humana e não automatizado. Consequentemente, não sendo digitalizado, é suscetível a uma maior probabilidade de erro (p.e., perda de papéis) e caracteriza-se como não sustentável face ao volume de papel necessário pelos documentos requeridos.

Esta dissertação visa a digitalização do processo de certificação de produtos agroalimentares com recurso à tecnologia *blockchain*. Esta digitalização é possível devido ao mecanismo de *smart contracts* (conceito detalhado na **Subsecção 2.3.5** deste capítulo) que permitirá não só o armazenamento dos dados do processo de certificação na *blockchain*, diminuindo a probabilidade de erro humano, como a sua automatização. As premissas do *smart contract* permitirão definir os dados necessários ao serviço de certificação e definir o fluxo para gerar o certificado digital. A sustentabilidade do processo de certificação é promovida através da submissão de ficheiros digitais como evidências. Estes serão armazenados num sistema de ficheiros descentralizado, cujo mapeamento é feito através do armazenamento do seu identificador nos dados do processo de certificação na *blockchain*.

A certificação de um produto agroalimentar é exposta a um *stakeholder* através de um logótipo impresso na rotulagem do mesmo, referente ao serviço de certificação que possui. Embora a dimensão da rotulagem seja limitativa, a representação da certificação através de um logótipo não permite ao *stakeholder*, por exemplo, obter informação acerca das evidências que a entidade de produção primária apresentou. Esta representação apresenta falta de transparência e rastreabilidade sob o processo perante um *stakeholder*.

Esta dissertação pretende mitigar este problema através da aplicabilidade da tecnologia *blockchain* ao respetivo caso de uso. O *modus operandi* desta tecnologia permite garantir o armazenamento de dados de forma imutável (i.e., garante ao *stakeholder* informação fidedigna), descentralizada (i.e., elimina riscos de fraude e de segurança derivados da centralização de dados) e transparente e rastreável (i.e., permite que um *stakeholder* com acesso ao *ledger* da *blockchain*, tenha acesso a toda a informação armazenada na rede). [16] Neste âmbito, é requerido que a prova de conceito desta dissertação permita ao *stakeholder* ter acesso à informação armazenada no *ledger* através da leitura de um código QR impresso na rotulagem do produto.

A evolução tecnológica permitiu o desenvolvimento de soluções que, através da digitalização de dados, permitem aos *stakeholders* obter informação, por exemplo, acerca do processo de produção de um produto. No entanto, os dados apresentados provêm de sistemas centralizados que podem alterar os mesmos sem conhecimento dos *stakeholders*, colocando em causa a confiabilidade da informação apresentada. Também a centralização torna os dados mais vulneráveis, colocando em causa a segurança destes.

Esta dissertação expecta mitigar o problema da centralização de dados através da digitalização da certificação com recurso a tecnologias descentralizadas. A tecnologia *blockchain* opera sob uma rede descentralizada de nodos, assim como o sistema de ficheiros descentralizado. Em suma, com base nas suas características, estas tecnologias permitem mitigar os problemas de falta de transparência, rastreabilidade, segurança e confiabilidade identificados.

Todavia, a contrafação de produtos no setor agroalimentar, e consequente falsificação de certificação [17], é um problema que não é possível de mitigar pela impressão de um código QR no rótulo do produto. A desigualdade existente entre países desenvolvidos e em desenvolvimento, ou entre pequenos e grandes produtores, no acesso ao mercado da certificação é um problema que também a digitalização do processo de certificação não consegue mitigar. [8]

Face à digitalização do processo de certificação, é ainda necessário considerar que uma entidade de produção pode possuir diversos produtos, diversas áreas de produção, entre outros dados que seria necessário armazenar. Os dados necessários de armazenar na *blockchain*, ou de que forma esta pode ser impactada pelo volume de dados, é um ponto que requer análise face ao âmbito desta dissertação. A autenticação das entidades de certificação e de produção é também um ponto que deve ser analisado uma vez que apenas as entidades envolvidas no processo de certificação poderão colocar dados na *blockchain*, e as restantes apenas poderão consultar os dados armazenados.

2.3 Blockchain

A tecnologia *blockchain* foi introduzida através da plataforma Bitcoin [18] em 2008 por Satoshi Nakamoto. No entanto, esta tecnologia integra os desenvolvimentos de Stuart Haber e W. Scorr Stornetta [19] que desenvolveram uma técnica de numeração - *timestamp* - que permite identificar, de forma única, modificações em documentos digitais. Integra também os desenvolvimentos de Adam Back [20] no mecanismo de consenso *Proof of Work* (PoW), embora o seu trabalho fosse direcionado a limitar o *spam* em caixas de correio eletrónico e ataques distribuídos de negação de serviço (DDoS). Atualmente, a tecnologia *blockchain* é aplicada em inúmeras áreas (p.e., finanças, medicina, logística), uma vez que permitiu que aplicações que requeriam um intermediário de confiança passassem a operar de uma forma descentralizada (i.e., sem a necessidade de uma entidade central) atingindo o mesmo grau de confiança. [1]

Blockchain é uma sub-categoria de *Distributed Ledger Technology* (DLT) que opera sob um protocolo de rede descentralizado ponto-a-ponto (P2P), constituída por nodos que partilham um *ledger* distribuído e que executam *software* dedicado. Os nodos participantes da rede devem estabelecer consenso sob os dados legítimos a serem adicionados, sendo que o modo como esta decisão é alcançada é determinado por um algoritmo - o mecanismo de consenso. [1]

Os dados a inserir no *ledger* de uma plataforma *blockchain* são armazenados sob a forma

de transações, sendo estas agrupadas e armazenadas em estruturas de dados específicas - blocos. Cada bloco possui um identificador único - *hash* - gerado com base nas transações que armazena e no histórico de transações armazenadas no *ledger*. Os blocos armazenados no *ledger* encontram-se conectados através de uma parâmetro - *previous hash* - definido com o valor da *hash* do bloco anterior, como é possível de verificar na **Figura 2.2**, tornando o seu funcionamento análogo ao mecanismo de lista ligada. O primeiro bloco a ser inserido no ledger é intitulado de *genesis* e é o único caso particular deste funcionamento.

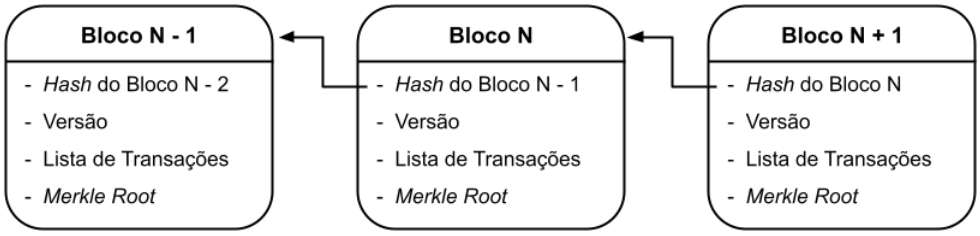


Figura 2.2: Modelo representativo de uma blockchain

2.3.1 Atributos

A tecnologia *blockchain*, face aos seus mecanismos e *modus operandi*, é caracterizada pela sua:

- **Descentralização**: é uma tecnologia, sub-categoria de DLT, que opera sob um protocolo de rede descentralizada P2P. O *ledger* que armazena os dados pertencentes ao seu estado é distribuído pelos múltiplos nodos pertencentes à rede. Deste modo, os nodos encontram-se conectados entre si, possuindo uma cópia igual do *ledger* e formando uma rede de dados descentralizada. Assim, a tecnologia garante tolerância a falhas uma vez que não é possível existir um ponto único de falha (SPOF);

- **Imutabilidade**: o valor da *hash* identificadora de um bloco é gerada com base no conjunto de transações por si armazenadas. Deste modo, caso alguma transação seja alterada, o valor da sua *hash* será diferente e consequentemente também a *hash* identificadora desse bloco. Esta alteração causaria incompatibilidades da rede uma vez que o parâmetro *previous hash* do bloco seguinte ao bloco alterado, não iria corresponder ao novo valor de *hash* do mesmo. Estas incongruências seriam detetadas pela rede que rejeitaria o bloco ou causaria o *fork* da mesma. No caso da plataforma *blockchain* Ethereum, por exemplo, o valor de *hash* para um determinado bloco é a *Merkle Root* resultante de uma *Merkle Tree* construída através do conjunto de transações que constituem o bloco.

- **Transparência e Rastreabilidade**: embora o tipo de plataforma *blockchain* (detalhado na **Subsecção 2.3.3** deste capítulo) possa fazer diferir esta característica, um nodo ao possuir uma cópia integral do *ledger* permite a uma entidade com acesso a si, auditar todos os dados armazenados neste. Deste modo, os dados armazenados no *ledger* de uma plataforma *blockchain* são transparentes a todas as entidades com acesso a estes, sendo possível rastrear os mesmos (p.e., verificar o processo logístico de transporte de um determinado produto). Esta é uma característica principalmente importante em plataformas *blockchain* públicas *permissionless* uma vez que, ao contrário dos atuais sistemas centralizados, permitem que qualquer entidade possua uma cópia integral do *ledger* ao integrar a rede.

- **Automatização**: o mecanismo de *smart contracts* (detalhado neste capítulo na **Subsecção 2.3.5**), desenvolvido e integrado em determinadas plataformas *blockchain*, permite aos nodos constituintes da rede executar lógica computacional. O objetivo deste mecanismo é permitir que digitalmente seja verificada e/ou executada lógica que permita que entidades medeiem, sem intermediários, de forma segura (i.e., com recurso a lógica imutável executada na rede) e automatizada (i.e., através de lógica computacional que permite despoletar determinado resultado através do cumprimento de determinadas premissas), um contrato.

2.3.2 Camadas de Abstração

Os desenvolvimentos de [1] focam-se na avaliação do desempenho de plataformas *blockchain*, indo de encontro ao foco desta dissertação. Assim, e tendo os autores de [1] definido de forma concisa as camadas de abstração de uma DLT, estas serão tidas em conta no desenvolvimento desta dissertação. As camadas de abstração são nomeadamente, a camada de rede, a camada de consenso, a camada de dados, a camada de execução e a camada de aplicação, como representado na **Figura 2.3**.

Fonte: [1]

Figura 2.3: Camadas de abstração de DLTs

- **Camada de Aplicação**: constituída pelo conjunto de aplicações que executam lógica no funcionamento da rede, e que são utilizadas pelos utilizadores finais. São estas, por exemplo, *wallets* para gerir as criptomoedas de uma entidade, *smart contracts*, e outras aplicações descentralizadas. Uma vez que esta camada apresenta os resultados finais da execução da lógica no atual estado do *ledger*, estes recorrem e são impactados pelas camadas inferiores, apresentadas na **Figura 2.3**. Deste modo, os resultados da avaliação do desempenho da camada de aplicação, refletem o desempenho geral da DLT a ser testada.

- **Camada de Execução**: encarregada de executar a lógica de um *smart contract* ou código de baixo-nível (i.e., *bytecode*), num ambiente de execução instalado nos nodos da rede. Este ambiente de execução deve ser eficiente, e os seus resultados devem ser determinísticos de forma a evitar incertezas e inconsistências de transações nos múltiplos nodos. O abortar de transações devido a inconsistências de execução resulta num desperdício de recursos computacionais e uma diminuição do desempenho.

Adicionalmente, as configurações de recursos (i.e., CPU e RAM) podem impactar o desempenho da execução.

- **Camada de Dados**: esta camada é constituída por um conjunto de tópicos que envolvem a manipulação dos dados de um *ledger*, como por exemplo, modelos transacionais (i.e., *Unspent Transaction Outputs* (UTXO) ou baseados em contas), estrutura de dados (i.e., *blockchain*, *Directed Acyclic Graph* (DAG), etc.), algoritmos de encriptação (p.e., SHA-256, SHA-512, etc.), entre outros. Todos estes fatores, e variações possíveis dos mesmos, podem influenciar o desempenho de uma plataforma *blockchain*.

- **Camada de Consenso**: o mecanismo de consenso estabelece o conjunto de regras, e força a execução destas, necessárias para que os nodos da rede estabeleçam um consenso sob os novos dados a inserir no *ledger*. Esta camada é definida pelo tipo de mecanismo de consenso (p.e., PoW, *Proof of Stake* (PoS), etc.), ou combinação destes, que a DLT a testar utilizar. O desempenho desta camada é definido pelo desempenho do mecanismo de consenso, e a eficiência deste impacta o desempenho da rede

- **Camada de Rede**: esta camada é constituída pela rede P2P, inerente a um sistema DLT, e pelos nodos que a constituem. A rede é responsável pelo processo de conexão entre nodos, pela gestão (i.e., validação e execução) de transações e pela propagação dos blocos. Os requisitos da rede de uma DLT é que esta providencie velocidade e seja estável. Por exemplo, o processo de conexão entre nodos e a sincronização do *ledger* entre estes, é diretamente afetado pela rede onde a velocidade define a eficiência da mesma. As métricas que permitem analisar o desempenho desta camada, e têm impacto no desempenho da DLT, são a taxa de transferência, a percentagem de pacotes perdidos, e a latência da rede.

2.3.3 Taxonomia

DLTs podem ser categorizadas segundo a arquitetura dos seus dados ou com base nas permissões do seu *ledger*. [1] Com base na arquitetura dos seus dados, podemos sub-categorizar uma DLT como *blockchain* (p.e., Bitcoin e Ethereum) ou DAG (p.e., IOTA e Nano). Com base nas permissões do seu *ledger*, as DLT podem ser sub-categorizadas enquanto *permissionless* ou *permissioned*, combinando com prefixo públicas ou privadas face à acessibilidade ao *ledger*.

Numa *blockchain* as transações são agrupadas e armazenadas em blocos que se encontram conectados através do valor da sua *hash*, como já mencionado. Num DAG as transações encontram-se conectadas através de uma relação de referência formando uma grafo orientado. Ainda assim, existem DLTs que possuem uma estrutura de dados única (p.e., Radix e Corda), não sendo possível integrá-las em nenhuma destas sub-categorias.

A categorização de uma DLT face às permissões do seu *ledger* não é um conceito padrão, deste modo foi tido em consideração o trabalho desenvolvido em [1], resumido na **Figura 2.4**. Face às permissões de um *ledger* ao nível da camada de rede, uma DLT *permissioned* difere de uma DLT *permissionless* no âmbito em que possui um mecanismo adicional de segurança que, análogo a uma camada de controlo de acesso, permite definir *roles* aos seus participantes. Os participantes de uma rede *permissioned* são identificáveis através de, por exemplo, uma conta. Face à acessibilidade, e ao contrário de *ledgers* privados, *ledgers* públicos permitem que qualquer entidade possa integrar um novo nodo na rede, participar

no mecanismo de consenso e ler os dados armazenados no *ledger*. Esta sub-categorização, resumida na **Tabela 2.1**, define DLTs:

- **Públicas *Permissionless*** (p.e., Ethereum): não existem restrições às entidades participantes, isto é, qualquer entidade anónima está capacitada a inserir um novo nodo na rede e participar no mecanismo de consenso;

- **Públicas *Permissioned*** (p.e., EOSIO): a rede pode definir entidades privilegiadas, capacitadas de executar diferentes ações dos restantes nodos da rede, no entanto, qualquer entidade identificável pode inserir um novo nodo na rede;

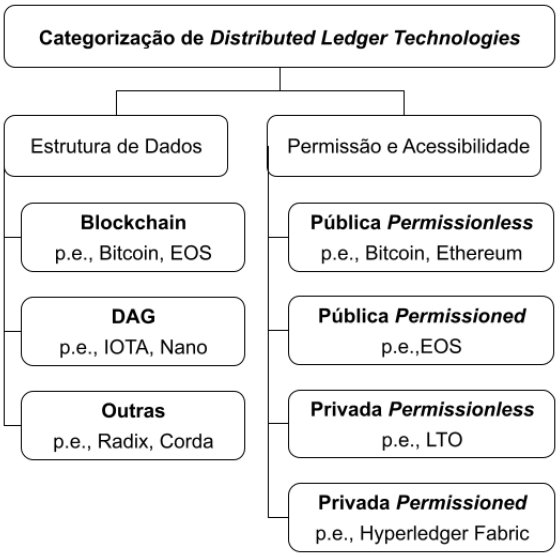
- **Privadas *Permissionless*** (p.e., LTO): qualquer entidade pode inserir um novo nodo na rede, no entanto, os restantes nodos vão reconhecer a sua existência mas não irão partilhar os seus dados. Um nodo armazena os dados que necessita para garantir determinado serviço a utilizadores previamente autorizados, em vez de armazenar toda a informação da rede. Os *smart contracts*, neste tipo de rede, não definem apenas quem pode realizar determinadas ações, como também quem pode ler o *smart contract* e os dados relacionados. Deste modo, uma DLT privada *permissionless* não possui uma única *chain* partilhada por todos os nodos da rede, em vez disso, cada instância de *smart contract* possui uma *chain* dedicada. As permissões de leitura de um *smart contract* não são garantidas a nodos específicos mas sim a entidades (i.e., pessoas ou organizações) identificadas por uma assinatura criptográfica. Para obter uma cópia do contrato e *chain* associada, além de previamente autorizado, é necessário saber o identificador único do contrato e o URL de um qualquer nodo que possua uma cópia da *chain* privada do contrato, de forma a participar no mesmo;

- **Privadas *Permissioned*** (p.e., Hyperledger Fabric): apenas entidades identificáveis previamente aprovadas podem integrar a rede e participar no mecanismo de consenso, podendo o gestor da rede definir entidades privilegiadas.

	Pública *Permissionless*	**Pública *Permissioned***	**Privada *Permissionless***	**Privada *Permissioned***
Autoridade	Descentralizada	Parcialmente Descentralizada	Descentralizada	Parcialmente Descentralizada
Velocidade Transacional	Lenta	Rápida	Lenta	Rápida
Custo	Alto	Baixo	Baixo	Baixo
Desafios/ Fraquezas	Escalabilidade Privacidade	Centralização Privacidade	Escalabilidade Consenso	Centralização Consenso
Exemplos	Ethereum	EOSIO	LTO	Hyperledger Fabric

Tabela 2.1: Comparação da taxonomia de DLTs face às permissões do seu *ledger*

É importante frisar que a anterior categorização tem por base as permissões de um *ledger* ao nível da camada de rede. Tendo como exemplo a plataforma *blockchain* Ethereum, uma DLT pública *permissionless*, é possível definir restrições de acesso aos métodos de *smart contracts* sendo que este controlo é feito ao nível da camada de aplicação.



Fonte: [1]

Figura 2.4: Categorização de DLTs

Desafios/Fraquezas

DLTs públicas apresentam a **privacidade** como um desafio/fraqueza, como mencionado na **Tabela 2.1**, uma vez que é possível a qualquer entidade integrar a rede e obter uma cópia integral do *ledger*. Embora seja uma característica que define este tipo de DLTs, pode tornar-se numa fraqueza caso haja uma falha de segurança que comprometa os respetivos dados. Além disso, torna-se numa fraqueza para casos de uso empresariais, onde seja gerida informação sensível de ser disponível publicamente, visto poder ser um *trade-off* crucial.

DLTs *permissioned* apresentam a **centralização** como um desafio/fraqueza, como mencionado na **Tabela 2.1**, uma vez que a autoridade destas é ditada por entidades previamente definidas e identificáveis. Estas DLTs tendem a recorrer a mecanismos de consenso, ou definição de permissões de rede, que atribuam a determinadas entidades privilégios (p.e., participar no mecanismo de consenso) que lhes permite impor autoridade na rede (p.e., definir os blocos que são adicionados ao *ledger*). Este pode ser um *trade-off* para determinados casos de uso, uma vez que tende a abdicar da descentralização que é a característica que permite mitigar o problema de existir uma unidade central em sistemas centralizados.

DLTs *permissionless* apresentam a **escalabilidade** como um desafio, como mencionado na **Tabela 2.1**, uma vez que não tendo a rede restrições à integração de novos nodos, o número destes pode escalar abruptamente. Esta escalabilidade pode afetar o desempenho da rede, se aumentar o volume de transações consequentemente aumentará o tempo para validação de uma transação, e/ou o custo, se o mecanismo de consenso adaptar os custos transacionais consoante o número de nodos de validação.

No caso por exemplo da plataforma *blockchain* Bitcoin, é utilizado o mecanismo de consenso PoW. Este mecanismo de consenso, tal como outros utilizados em outras plataformas *blockchain* públicas *permissionless*, requer elevados custos na validação de novas transações. O objetivo destes custos é garantir que apenas informação fidedigna é armazenada no *ledger*, tornando-se demasiado dispendioso para um atacante tentar dominar a rede. No entanto, os nodos de validação vêm recompensado este custo através das taxas transacionais que

o remetente é forçado a pagar e que são no final dadas a estes nodos, sob a forma de recompensa. As taxas transacionais por sua vez também flutuam consoante um conjunto de fatores como, por exemplo, o crescente número de transações da rede.

DLTs privadas apresentam o **consenso** como um desafio/fraqueza, como mencionado na **Tabela 2.1**, uma vez que este será gerido por entidades previamente definidas que impactam a descentralização do seu funcionamento. Deste modo, a validação de uma nova transação poderá estar suscetível a um consenso fraudulento ou de supremacia por parte de determinadas entidades, uma vez que este não é de carácter público e tende a ser centralizado.

Além do mencionado, a **regulamentação** é uma característica que caracteriza plataformas *blockchain permissioned*, uma vez que estas possuem capacidade de estabelecer permissões às várias entidades integrantes da rede. Deste modo, este tipo de plataformas *blockchain* tornam-se atrativas a soluções empresariais que dependam de regulamentação para definir regras.

Sendo a escolha do tipo de plataforma *blockchain* muito dependente dos requisitos do caso de uso, a **Subsecção 2.3.7** deste capítulo apresenta uma comparação de diferentes plataformas *blockchain*, não só face às suas características mas tendo em conta os requisitos do caso de uso da certificação digital de produtos agroalimentares.

2.3.4 Mecanismos de Consenso

Um mecanismo de consenso é um mecanismo tolerante a falhas utilizado em DLTs para atingir o consenso entre os múltiplos nodos da rede sob um novo estado do *ledger* (p.e., atualização do estado por inserção de uma nova transação). Operando sob uma rede descentralizada que se caracteriza por ser auto-reguladora (i.e., não possui uma unidade confiável de controlo central), os mecanismos de consenso demonstram-se eficientes, funcionais, de confiança e seguros, ao garantir que todas as transações na rede são fidedignas, e que os participantes desta consentem o estado atual do *ledger*. Existem DLTs híbridas que combinam diferentes tipos de mecanismos de consenso como, por exemplo, EOSIO que combina *Pratical Byzantine Fault Tolerance* (PBFT) e *Delegated Proof of Stake* (DPoS).

Aquando de uma nova e digitalmente assinada transação, esta é difundida pela rede e necessita de ser validada pelos nodos participantes no mecanismo de consenso. Após a sua validação, é adicionada a um bloco que será difundido pela rede e posteriormente armazenado no *ledger*, atualizando o seu estado. Os nodos participantes no mecanismo de consenso garantem a operabilidade da rede e, uma vez que permitem garantir que as transações e blocos são válidos e ordenados de forma não ambígua, asseguram a integridade e consistência do atual estado do *ledger* em todos os nodos da rede.

Sendo o mecanismo de consenso tolerante a falhas, como já mencionado, este deve saber lidar com nodos deliberadamente mal intencionados, ser resiliente perante falhas *byzantine* [21] (i.e., conjunto de falhas possíveis de ocorrer em sistemas distribuídos), particionamento da rede, atrasos de mensagens, mensagens fora de ordem e mensagens corrompidas. Deste modo, uma falha do mecanismo de consenso pode não garantir a capacidade da rede de atingir o consenso, impactando o desempenho da rede e possivelmente de todo o sistema, ou pode originar vulnerabilidades de segurança.

Mecanismos de consenso podem categorizar-se por serem baseados em provas (p.e., PoW, PoS, etc.) ou baseados em votação (p.e., Raft, DPoS, etc.). Os principais mecanismos de consenso são PoW, PoS e PBFT, embora existam múltiplos mecanismos de consenso que

derivam destes ou completamente distintos mas que não se demonstram tão presentes no mercado.

Mecanismos de consenso baseados em votação são utilizados em plataformas *blockchain permissionless*, uma vez que é requerida a identidade dos seus participantes no mecanismo de votação, e são representados pelos mecanismos Raft e mecanismos baseados em *Byzantine Fault Tolerance* (BFT) (p.e., PBFT). O seu funcionamento baseia-se em comunicação frequente entre os múltiplos nodos da rede, podendo estes ter diferentes funções, de forma a garantir o consenso.

Proof of Work

Proof of Work é um mecanismo de consenso de computação intensiva que requer que os nodos da rede participantes no mesmo - *miners* - solucionem um puzzle (i.e., um problema computacional) competindo pelo direito de inserir um novo bloco no *ledger*. O problema computacional consiste na adição de um parâmetro *nonce* ao corpo do bloco. Este parâmetro é um valor inteiro que visa que a *hash* resultante do bloco seja uma *string* que inicie com um determinado número de valores zero. A dificuldade de resolução deste problema será determinada pelo número de zeros consecutivos que a *hash* possuir inicialmente. O processo de solucionar este problema computacional, uma vez que o *miner* não sabe o valor do parâmetro *nonce*, é resolvido através de tentativa-erro e designa-se por *mining*. O primeiro *miner* a resolvê-lo fica capacitado a inserir o novo bloco no *ledger* e, devido aos elevados custos energéticos e longo tempo de processamento requeridos, é recompensado com um determinado montante de criptomoedas, correspondente ao montante que o remetente foi forçado a pagar de custos transacionais - *transaction fees*. No entanto, a complexidade do problema matemático aumenta consoante o número de problemas matemáticos resolvidos por um nodo, de forma a garantir um tempo de resolução médio equivalente a todos os *miners*.

O funcionamento do mecanismo de consenso PoW inicia-se pela seleção de transações da *memory pool* - local onde se encontram armazenadas as transações pendentes (i.e., para serem inseridas na blockchain) - por parte de um *miner*, e a adição destas a um novo bloco. Este bloco é de seguida difundido pela rede, de forma a que os *miners* iniciem o processo de *mining*. Ao ser descoberto o valor do parâmetro *nonce* que permite obter a *hash* correta do respetivo bloco, o *miner* difunde a *hash* pela rede de forma a que todos os nodos validem a mesma. Aquando da verificação pela rede, o novo bloco é adicionado ao *ledger*, o estado do *ledger* em todos os nodos é atualizado e o *miner* que solucionou o problema é recompensado. Por exemplo, na plataforma *blockchain* Bitcoin a dificuldade da resolução deste problema matemático é intencionalmente definida com o intuito de garantir que o tempo médio de verificação de um bloco seja 10 minutos.

Este mecanismo de consenso é principalmente utilizado em plataformas *blockchain* públicas *permissionless* onde, não existindo confiabilidade entre os nodos da rede, a computação intensiva requerida pelo mecanismo de consenso torna impraticável a inserção de transações falsas e o domínio da rede por um atacante, e mitiga os problemas de *double-spending* e *sybil attacks*. Assim, garante a segurança e integridade dos dados armazenados no *ledger*. Este mecanismo de consenso é utilizado, por exemplo, nas plataformas *blockchain* Bitcoin e Litecoin.

No entanto, a aplicabilidade de PoW torna a plataforma *blockchain* suscetível a problemas de escalabilidade devido a dois fatores limitativos: taxa de transferência transacional e tamanho de blocos. Caso aumentássemos a taxa de transferência aumentávamos o desem-

penho da rede mas aumentaria o risco desta sofrer bifurcações. No entanto, a baixa taxa de transferência limita a aplicação de PoW em casos de uso que necessitem de elevadas taxas transacionais. Ao aumentar o tamanho dos blocos, estes armazenariam um maior número de transações mas desacelaria a propagação destes pela rede, reduzindo a segurança. A alteração destes parâmetros pode colocar o sistema vulnerável a ataques como *selfish mining* e *double spending*. O elevado custo energético para a resolução do problema computacional implica que apenas seja rentável para um *miner* quando a recompensa é superior aos custos energéticos. Face ao mencionado, também as práticas de maximizar o poder computacional dos *miners* tendem a centralizar e profissionalizar o *mining* de novos blocos. Com o objetivo de aumentar a eficiência deste surgiram variantes como, por exemplo, *Greedy Heaviest Observed Subtree* (GHOST), PoS, *Proof-of-Authority* (PoA) e *Proof of Elapsed Time* (PoET).

Proof of Stake

Proof of Stake é um mecanismo de consenso para redes de plataformas *blockchain* que pretende mitigar os custos de computação intensiva em PoW. Baseia-se no princípio de que nodos que possuam maiores quantias de *tokens* na plataforma *blockchain* têm menos probabilidade de, intencionalmente, danificar o sistema.

O nodo participante do mecanismo de consenso PoS que irá gerar e validar o próximo bloco - *validator* -, assumindo a responsabilidade de manter a segurança e integridade do *ledger*, é escolhido pseudo-aleatoriamente com base no montante de *tokens* que aposta - *stake*. A probabilidade de um nodo ser *validator* aumenta quanto maior for o *stake* que aposta. Os blocos gerados e validados pelos *validators* são posteriormente difundidos pelos nodos da rede, que verificam e adicionam o novo bloco ao *ledger*. Os *validators* são recompensados com base no montante que apostam, incentivando os mesmos através de retorno sob o investimento.

Face ao seu funcionamento, PoS apresenta uma processamento de finalização rápida que permite reduzir o tempo de validação de novos blocos, aumentando o desempenho e taxa de transferência transacional da mesma. A nível de segurança, os *validators* são incentivados a atuar de forma fidedigna na produção de blocos e aprovação de transações. Por um lado, porque tipicamente estes *validators* possuem uma considerável porção dos *tokens* da rede e vulnerabilidades de segurança irão afetar negativamente o valor de mercado dos *tokens* (p.e., irá desvalorizar a criptomoeda da plataforma *blockchain*). Por outro lado, o seu *stake* fica bloqueado pelo mecanismo de consenso e, caso atue de forma maliciosa ou produza blocos falsos, o mesmo é-lhe removido.

No entanto, este mecanismo de consenso promove o armazenamento de *tokens*, em vez da sua circulação. Além disso, ao eliminar a validação de blocos por computação intensiva, a segurança em PoS pode ser um desafio para determinados casos de uso, tendo vindo a desenvolverem-se diversos protocolos.

Pratical Byzantine Fault Tolerance

Byzantine Fault Tolerance [21] é a característica de uma rede distribuída atingir o consenso, possuindo nodos que falham a resposta ou que respondem com informação incorreta. O mecanismo de consenso *Pratical Byzantine Fault Tolerance* visa replicar uma máquina de estados que permita operar uma rede descentralizada tolerante a *byzantine faults* (i.e., com nodos que falham a resposta ou que respondem com informação incorreta). Este foi

projetado para plataformas *blockchain permissionless* de forma que os nodos que não se encontram em correto funcionamento sejam facilmente identificáveis. Os nodos em PBFT são sequencialmente ordenados e um destes é alocado enquanto nodo primário, ou *leader*, e os restantes são alocados enquanto nodos secundários, ou *backup*. Qualquer nodo da rede é elegível a tornar-se *leader*, transitando de *backup* devido, por exemplo, à falha do *leader*.

O funcionamento de PBFT, em alto nível, divide-se em três fases por ronda. Inicialmente o processo desencadeia-se com o envio de um pedido por um cliente ao *leader*. Na fase de pré-preparação, o *leader* produz uma proposta com um conjunto de transações e reencaminha a mesma numa sequência ordenada a todos os nodos *backup*. Numa fase de preparação - primeira fase de votação -, e recebendo a mensagem do *leader*, os nodos *backup* vão verificar a mensagem. Se esta verificação for sucedida, os nodos *backup* difundem uma mensagem de preparação a todas os restantes nodos *backup*. Caso a verificação falhe, os nodos *backup* não fazem nada. Numa fase de *commit* - segunda fase de votação -, ao receber as mensagens de preparação de 3f + 1 nodos da rede, sendo f o número de nodos *faulty*, os nodos *backup* difundem mensagens de *commit* e enviam a resposta do pedido diretamente ao cliente. Se o cliente receber 3f + 1 respostas de diferentes nodos da rede com o mesmo resultado, o seu pedido foi concluído com sucesso.

A segurança e integridade do *ledger* depende da rede possuir 2/3, ou mais, de nodos fidedignos. No entanto, o seu funcionamento requer três fases de mensagens, incluindo duas fases de votação, onde cada mensagem é enviada a todos os nodos *backup* da rede. Deste modo, apresenta como desvantagem um *overhead* de comunicação de $O(n^2)$ mensagens, onde n é o número de nodos *backup*. Este *overhead* de comunicação dificulta a escalabilidade, nomeadamente numa rede de larga escala.

Outros Mecanismos de Consenso

Os múltiplos casos de uso, com diferentes requisitos, que foram integrando o mercado das DLTs, projetaram o desenvolvimento de diferentes mecanismos de consenso que se adaptam a diferentes necessidades. Deste modo, e além dos principais mecanismos de consenso - PoW, PoS e PBFT - já mencionados, podemos mencionar outros mecanismos de consenso, de entre muitos outros existentes (p.e., DPoS):

- *Proof-of-Authority* (PoA) [22]: deriva de PoS e é executado em plataformas *blockchain permissioned* onde os seus participantes são identificáveis (i.e., possuem identidade). O funcionamento deste baseia-se na aposta da identidade e reputação dos nodos participantes no mecanismo de consenso, de forma a poderem ser elegidos para gerar e validar o próximo bloco. Através de um algoritmo pseudo-aleatório e não determinístico, são selecionados um conjunto de *validators* para estabelecerem o consenso da rede. Sendo o novo bloco validado por um pequeno conjunto de nodos da rede autorizados, a velocidade do processo de validação aumenta. A nível de segurança, os *validators* são incentivados por colocarem a sua identidade e reputação em aposta e, por outro lado, sendo o algoritmo de seleção dos *validators* não-determinístico, este encontra-se menos sujeito a *sybil attacks*.
- *Proof of Elapsed Time* (PoET) [23]: executado em plataformas *blockchain permissioned* onde os seus participantes são identificáveis (i.e., possuem identidade). O funcionamento deste obriga a que todos os nodos da rede, participantes no mecanismo de consenso, gerem um período de tempo aleatório durante o qual vão ter que estar em espera. O primeiro nodo a terminar o respetivo tempo de espera (i.e., nodo que possuir um período de tempo de espera mais curto) fica capacitado a gerar e

validar o próximo bloco. Considera-se um algoritmo de lotaria justa, sendo que os nodos da rede devem possuir uma equitativa probabilidade de gerar o próximo bloco. Deste modo, o mecanismo de consenso deve garantir que os nodos participantes escolhem um período aleatório de tempo e não um período de curta duração por eles definido, e também que o nodo a ser recompensado de facto completou o tempo de espera gerado.

- *Raft* [24]: algoritmo *Crash Fault Tolerance* (CFT) com fluxo de dados unidirecional, desde o nodo elegido líder até aos restantes nodos da rede. Um nodo da rede pode assumir um dos seguintes três estados: líder - responsável por aceitar os pedidos dos clientes e gerir a replicação do *log*, ou estado, pelos restantes nodos da rede -, seguidor - em funcionamento normal todos os nodos da rede assumem este estado passivo, respondendo apenas aos pedidos do líder ou candidatos - ou candidato - estado utilizado quando nodos estão para ser eleitos novos líderes. Este mecanismo de consenso decompõem-se em três principais sub-problemas: eleição do líder - um novo líder necessita de ser eleito em caso de falha do atual -, replicação de *log* - o líder necessita de manter os *logs* dos restantes nodos da rede sincronizados com a sua réplica -, e integridade - se um dos nodos da rede fez *commit* de uma entrada de *log* num determinado índex, nenhum outro nodo pode fazer *commit* de outra entrada de *log* para o mesmo índex. O seu processamento divide-se em períodos de tempo - termos - arbitrários que se iniciam pela eleição do líder. Se um candidato vencer a eleição mantém-se líder pelo resto do termo, caso a votação seja dividida, o termo termina sem um líder.

2.3.5 *Smart Contracts*

Em 1994, Nick Szabo [25] introduziu o conceito de *smart contracts* como sendo programas de computador que replicam, de forma automatizada, as ações descritas num contrato físico tradicional. Embora progredisse nos seus desenvolvimentos [26], à época não existia uma plataforma que viabilizasse a implementação do conceito de *smart contract*. [3]

A arquitetura da plataforma *blockchain* Bitcoin apenas utiliza linguagem de *scripts* para verificar e validar transações de criptomoedas. Embora não seja suficiente para o desenvolvimento de aplicações descentralizadas, foi considerada das primeira implementações de *smart contracts* na tecnologia *blockchain*. Permitiu demonstrar a aplicabilidade da tecnologia para a implementação de *smart contracts* através da sua linguagem de *script* e das permissões exclusivamente de leitura e escrita - propriedade *read-append*. [27] Em 2015, Vitalik Buterin [28] lançou a plataforma *blockchain* Ethereum que possui uma linguagem *Touring-complete*, permitindo o desenvolvimento de uma variedade de *smart contracts*. [27]

Smart contracts são código computacional cujo *deploy* é feito no *ledger* da plataforma *blockchain*, armazenando a sua instância e estado - *stateful*. [27] Permitem definir um conjunto de compromissos (i.e., "condições"que devem ser verificadas) e a forma como os seus participantes devem cumprir os mesmos (i.e., "ações"que podem ser executadas). [3]

A aplicabilidade deste mecanismo na tecnologia *blockchain* permite uma automatização descentralizada cujas alterações de estado são armazenadas no *ledger* desta. [2] De forma à sua execução obter os mesmos resultados nos vários nodos da rede descentralizada, o *smart contract* tem que ser determinístico. O seu funcionamento é análogo ao de um agente autónomo que se encontra em execução da rede da plataforma *blockchain*, eliminando a necessidade de terceiros para garantir confiabilidade aos seus participantes e consequentemente eliminando a existência de um SPOF. [3] [27]

Para fazer o *deploy* de um *smart contract* no *ledger* da plataforma *blockchain* é necessário: [27]

1. **Desenvolvimento do código fonte**: o código fonte, que representa a lógica computacional do *smart contract*, é desenvolvido numa linguagem de programação suportada pela plataforma *blockchain* em uso. De seguida, utilizando um compilador dedicado (habitualmente disponibilizado pela plataforma *blockchain*) é compilado o código fonte e é gerado o código máquina - *bytecode*.

2. ***Deploy***: é realizada uma transação que permitirá fazer o *deploy* do *bytecode* no *ledger* da plataforma *blockchain*. Dependendo da plataforma *blockchain*, e ao ser feito o seu *deploy*, o código fonte do *smart contract* poderá ser apenas de leitura ou modificável. Deste modo, determinadas plataformas *blockchain* (p.e., Ethereum) não permitem a atualização do código fonte do *smart contract* enquanto que outras plataformas *blockchain* (p.e., EOSIO) permitem reescrever o código fonte do mesmo. Plataformas *blockchain* que permitem apenas leitura de *smart contracts* obrigam a que os desenvolvedores façam um novo *deploy* do respetivo código fonte e redirecionem os utilizadores. [27]

O acesso a um *smart contract*, cujo *deploy* já foi realizado, depende da plataforma *blockchain*. Por exemplo, na plataforma *blockchain* Ethereum, o *deploy* do código fonte de um *smart contract* retorna um endereço que permite aos utilizadores enviarem transações destinadas ao mesmo. Estas transações requerem o nome do método a invocar e os parâmetros que este requer. Se for necessário o pagamento de algum valor para a execução do respetivo método, este deve também ser enviado na transação. Por outro lado, por exemplo, na plataforma *blockchain* EOSIO, o *deploy* do código fonte do *smart contract* é realizado numa conta armazenada na plataforma *blockchain*. O identificador da mesma irá ser utilizado para aceder e interagir com o mesmo. A próxima fase depende do mecanismo de consenso e do funcionamento da plataforma *blockchain*. No entanto, o objetivo é que as transações, quando retiradas da *pool*, sejam executadas no endereço do *smart contract* a que se dirigem. Os nodos da rede que executarem estas transações irão comparar os seus resultados e, de acordo com o resultado do mecanismo de consenso, será atualizado o estado do *smart contract* e consequentemente do *ledger*. [27]

O mecanismo de *smart contracts* possibilitou a estruturação da Proposta de Estágio desta dissertação, ao permitir a integração de lógica na rede de uma plataforma *blockchain* e o desenvolvimento de aplicações descentralizadas. O desenvolvimento desta dissertação iniciar-se-á pela implementação da lógica do caso de uso com recurso ao mecanismo de *smart contracts* para as plataformas *blockchain* a analisar. Pretende-se que os *smart contracts* a desenvolver, permitam a digitalização do processo de certificação de produtos agroalimentares. Enquanto prova de conceito, pretende-se o *deploy* do mesmo na plataforma *blockchain* seja possível de realizar pela entidade certificadora. O *smart contract* terá um funcionamento análogo ao de uma máquina de estados, permitindo restringir determinados métodos consoante a fase do processo de certificação. Estes métodos poderão também ser restringidos a determinada entidade, ou de certificação ou de produção.

Problemas

A utilização de *smart contracts* em plataformas *blockchain* públicas permite que *bugs*, inclusive falhas de segurança, sejam visíveis a todos os nodos da rede. No entanto, e dependendo da plataforma *blockchain*, a correção destes pode ser um processo lento. [3]

Um *smart contract* não consegue, de forma nativa, obter dados de fontes externas. Sendo as transações difundidas pela rede e processadas pelos vários nodos da mesma, os resultados ao dependerem de fontes externas poderiam variar de nodo para nodo. De forma a mitigar este problema foi desenvolvida tecnologia - oráculos - que permite que um agente externo à rede da plataforma *blockchain*, após obter os dados requeridos *off-chain*, invoque o método do *smart contract*, enviando os respetivos dados por parâmetro. [3]

O mecanismo de consenso utilizado pela plataforma *blockchain* pode tornar a aplicabilidade de *smart contracts* vulnerável. Por exemplo, uma plataforma *blockchain* que utilize PoS [29] enquanto mecanismo de consenso permite que os nodos que desvalorizem o *stake*, e possível queda do valor de mercado da criptomoeda, aplicando este no *deploy* de *smart contracts* que visem, por exemplo, a renovação ilegal ou viciação de *smart contracts* na rede da plataforma *blockchain*. No entanto, vêm surgindo plataformas *blockchain* híbridas (i.e., utilizam dois mecanismos de consenso distintos) que pretendem mitigar este problema. [29]

O facto de *smart contracts* permitirem a autonomia completa do processo, pode levar a que o sistema não consiga ser modificado e os utilizadores incorreram em perdas. É desta forma recomendado que sejam incluídos mecanismos de *fail-safe* nos *smart contracts* como, por exemplo, dados de entrada de um utilizador que mudem o seu comportamento, ou funções que permitam a utilizadores privilegiados destruir o contrato e removê-lo da Blockchain. [30] A controvérsia pode estar no facto de a Blockchain ser utilizada exatamente pela sua imutabilidade e desta forma estaríamos a corroborar o seu mecanismo.

2.3.6 Vulnerabilidades

As plataformas *blockchain* podem ser alvo de ataques associados com a arquitetura de rede distribuída P2P da tecnologia (p.e., bifurcações, ataques de 51% e ataques DDoS), e/ou associados à plataforma *blockchain* em causa, e seus mecanismos. [31]

Embora o foco desta dissertação não sejam as vulnerabilidades da tecnologia *blockchain*, serão apresentadas algumas com o intuito de alertar para a forma como estas podem comprometer dados, ou eventualmente toda a rede. Por exemplo, em Junho de 2016 um atacante desconhecido desviou 50 milhões de dólares do 'The DAO' - organização autónoma descentralizada. [31] Esta foi desenvolvida com base no mecanismo de *smart contracts*, funcionando como uma organização autónoma que é executada na plataforma *blockchain* Ethereum segundo um conjunto definido de regras.

Bifurcações

Uma bifurcação - *fork* - representa uma condição na qual os nodos da rede de uma plataforma *blockchain* apresentam uma visão distinta, que persiste durante longos períodos de tempo ou indefinidamente, acerca do estado da mesma. [31] Esta condição pode surgir involuntariamente aquando de, por exemplo, um erro do mecanismo de consenso, ou intencionalmente estabelecida aquando de, por exemplo, uma atualização de *software*. Uma bifurcação voluntária pode também derivar de nodos maliciosos que pretendam, por exemplo, integrar nodos *sybil* na rede, de modo a que estes sigam regras de validação fraudulentas ou realizem *selfish mining* [32] - grupo de nodos *miners* agrupados em número suficiente para conseguir obter recompensas superiores face à proporção do seu poder computacional de *mining*.

Com base na sua compatibilidade com a *chain* original e no prosseguimento da rede da plataforma *blockchain*, os *forks* categorizam-se segundo: [31]

- ***Soft forks***: representados na **Figura 2.5**, caracterizam-se por serem alterações compatíveis com versões anteriores. Aquando de um *soft fork* os nodos participantes do mecanismo de consenso continuam a reconhecer a validade de novas transações, no entanto, os blocos criados irão ser considerados inválidos pelos nodos atualizados. Assim, de forma a um *soft fork* ser bem sucedido, é necessário que os nodos atualizados representem a maioria dos participantes no mecanismo de consenso, caso contrário existe o risco de se tornar num *hard fork*.

- ***Hard forks***: representados na **Figura 2.6**, caracterizam-se por não serem compatíveis com versões anteriores. Aquando de um *hard fork* todos os participantes da rede devem ser atualizados, uma vez que os nodos da *chain* original não irão aceitar blocos criados por nodos atualizados. Neste caso, é como se a plataforma *blockchain* se dividi-se em duas redes distintas, uma vez que o consenso não é unânime. Este tipo de *fork* tende a ser utilizado para eliminar vulnerabilidades de segurança identificadas em versões anteriores, introduzir novas funcionalidades, ou reverter determinadas transações. Por exemplo, após ter sido identificado a vulnerabilidade do caso 'The DAO', a plataforma *blockchain* Ethereum realizou um *hard fork* de forma a reverter as transações associadas ao desvio de dinheiro, e os seus devidos donos foram reembolsados.

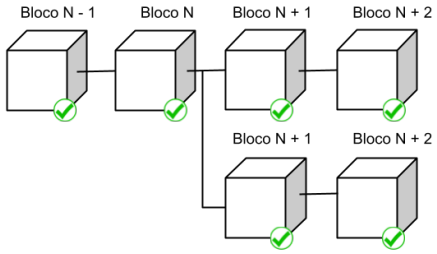


Figura 2.5: Representação de *hard fork*

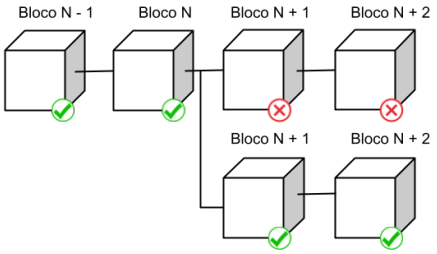


Figura 2.6: Representação de *soft fork*

Ataques de 51%

Um ataque de 51% [31] é definido por um atacante (singular ou coletivo) que visa deter a maioria do poder de participação no mecanismo de consenso (p.e., maioria da *mining hash rate* em PoW) com o intuito de manipular a rede da plataforma *blockchain*. Neste tipo de ataques, são utilizados, por exemplo, grupos de nodos *sybil* - nodos maliciosos que assumem diversas identidades, fingindo atuar de forma independente e honesta -, ou *mining pools* - conjunto de nodos que participam no mecanismo de consenso PoW e que partilham o seu poder computacional. Por exemplo, em Julho de 2014 a *mining pool* 'GHash.IO' deteve 51% da *hash rate* da rede durante um dia na plataforma *blockchain* Bitcoin.

Deste modo, o atacante conseguirá prevenir transações e blocos de serem validados, reverter transações para permitir *double spending* - técnica que consiste no armazenamento de transações no *ledger*, sob o mesmo ativo e com a chave privada do remetente embora com destinatários distintos - e prevenir outros participantes do mecanismo de consenso de encontrarem blocos num curto período de tempo. [31] Além disso, e comparativamente aos restantes, terá maior probabilidade de que os seus blocos sejam validados e armazenados no *ledger*, podendo estes conter transações fraudulentas ou que incorram em *double spending*.

As plataformas *blockchain* e respetivos mecanismos de consenso tentam mitigar este problema tentando torná-lo impraticável (i.e., dificultando a sua execução). Por exemplo, o

mecanismo de consenso PoW através da computação intensiva que requer para validação de um novo bloco, pretende tornar insustentáveis os custos de execução de um ataque de 51% ao seu atacante.

Ataques DDoS

Um ataque DDoS, aplicado à tecnologia *blockchain*, consiste num atacante (singular ou coletivo) sobrecarregar a rede da plataforma *blockchain*, tornando-a indisponível aos restantes utilizadores. [31] Este tipo de ataque, como referido em [31], poderá manifestar-se de diferentes formas de acordo com a arquitetura do sistema da respetiva plataforma *blockchain*, e do mecanismo de consenso que utiliza. O seu objetivo é inviabilizar os nodos de inserir e validar transações e gerar novos blocos, estagnando a rede ou aumentando o *delay* do processo.

Um ataque DDoS pode ser realizado por um atacante que crie uma infinidade de transações válidas num curto período de tempo - *overflood attack*. [33] Estas serão difundidas pelos nodos da rede que irão sobrecarregar as suas *memory pools*. Como resultado, pode aumentar o tempo de validação de uma transação e criação de novos blocos, e/ou os nodos vítimas deste ataque podem não ter capacidade de se manter na rede. Por exemplo, em 2015 a plataforma *blockchain* Bitcoin sofreu um ataque *overflood* que resultou na redução de mais de 10% do número total de nodos. [33] Por outro lado, também um ataque de 51% pode resultar num ataque DDoS caso este, por exemplo, previna os restantes participantes do mecanismo de consenso de validarem transações e criar novos blocos. [31]

2.3.7 Plataformas Blockchain

Existem diversas plataformas *blockchain* cujo design lhes permite mitigar determinados problemas identificados (p.e., escalabilidade) ou serem aplicadas a casos de uso com necessidades específicas. À vista disso foi feita uma análise das plataformas *blockchain* existentes, embora existam inúmeras mais no mercado, e com base nesta pesquisa resultou o quadro comparativo representado na **Tabela B.1** em anexo.

Considerando as plataformas *blockchain* mencionadas, foram definidas as que iriam integrar a amostra da análise de desempenho desta dissertação. Os critérios para a seleção da amostra consistiram nos seguintes parâmetros: foco industrial, compatibilidade com o mecanismo de *smart contracts*, modularidade da sua arquitetura, taxa de transferência transacional suportada e maturidade do seu sistema.

Face ao caso de uso desta dissertação, começaram por ser excluídas desta seleção as plataformas *blockchain* R3 Corda, XRP Ledger e Stellar. Estas são direcionadas ao setor financeiro e além disso, XRP Ledger não possibilita o desenvolvimento de *smart contracts*.

Hyperledger Fabric e Sawtooth apresentam elevados valores de taxa de transferência o que pode garantir alguma segurança face à escalabilidade da implementação do caso de uso numa rede em produção. No entanto, Hyperledger Sawtooth é selecionado uma vez que apresenta uma maior modularidade (i.e., permite a implementação de uma rede com diferentes possibilidades de taxonomia, e também permite múltiplos mecanismos de consenso) do seu sistema. Este fator permite um estudo mais abrangente da tecnologia *blockchain* e da respetiva plataforma *blockchain*.

Por fim, não é possível selecionar Ethereum e EOSIO e analisar três plataformas *blockchain*. Este poderia ser um risco às limitações do planeamento desta dissertação. Assim, foi

selecionada a plataforma *blockchain* Ethereum face à maturação do seu *software*.

Em suma, o desenvolvimento desta dissertação terá como amostra da sua análise as plataformas *blockchain* Ethereum e Hyperledger Sawtooth.

Ethereum

Ethereum é uma plataforma *blockchain* pública *permissionless turing-complete*, com foco em diversas indústrias e que utiliza PoW enquanto mecanismo de consenso. [2] Atualmente encontra-se na versão cujo nome de código é Berlin, no entanto, projeta-se a sua breve alteração para uma versão designada 2.0 que irá integrar o mecanismo de consenso PoS. Deste modo, a plataforma *blockchain* passará a classificar-se enquanto pública *permissioned*. Esta plataforma *blockchain* tem a sua própria criptomoeda - pesar de muitos outros, os seus clientes mais populares são Geth e Parity. asas Esta plataforma *blockchain* permite o desenvolvimento de *smart contracts* através das linguagens Solidity, Vyper, Yul e Yul+, e cujo *deploy* e execução destes é feita na *Ethereum Virtual Machine* (EVM). [3] Este mecanismo permite uma analogia à execução de um agente autónomo na EVM, que executa lógica computacional específica quando invocado por uma transação ou mensagem (i.e., quando invocado por outro *smart contract*). Para fazer o *deploy* de um *smart contract* é necessário que o código em Solidity seja compilado, pelo compilador dedicado da linguagem, e desta forma é gerado o código máquina - *bytecode* - que a EVM irá processar.

Bytecode é linguagem *assembly* constituída por múltiplos *opcodes*, onde cada um destes representa uma operação necessária de ser executada na EVM. Cada uma destas operações é simultaneamente executada por todos os nodos da rede da plataforma *blockchain*. Deste modo, Ethereum possui um sistema de preços interno para a execução de uma transação ou contrato na EVM que tem como unidade elementar o *gas*. Este custo visa limitar a utilização de recursos por um único *smart contract*. A EVM controla os recursos consumidos pela execução do *smart contract* e encarrega-se de cobrar, em *gas*, ao remetente pelos mesmos. [2] Cada operação invocada do respetivo *smart contract* tem um custo, em *gas*, e cada unidade de *gas* consumida pela transação é paga em *ether*, com base na sua quota de mercado à data.

Ethereum aplica uma estrutura de dados com duas camadas para organizar o conteúdo dos seus blocos. Os seus estados são armazenados numa base de dados chave-valor - LevelDB - em disco e indexados numa *hash tree* - *Patricia-Merkle Tree*. A raíz desta árvore é armazenada no cabeçalho dos respetivos blocos. A administração da rede é distribuída pelos múltiplos participantes da mesma, sendo que os respetivos nodos podem ser categorizados como: *full* - participam no mecanismo de consenso - ou *light* - não verificam transações e/ou blocos, e não possuem uma cópia do estado atual do *ledger*, recebendo informação de nodos *full*.

Sendo esta uma plataforma *blockchain* pública *permissionless*, não existem restrições à participação no mecanismo de consenso e qualquer nodo que participe no mecanismo de consenso recebe uma cópia das transações. Deste modo, Ethereum possui problemas de escalabilidade e privacidade que, no entanto, serão mitigados com a introdução do mecanismo de consenso PoS (i.e., serão limitados os participantes no mecanismo de consenso) na próxima versão.

Hyperledger Sawtooth

Hyperledger Sawtooth [34] é uma plataforma *blockchain permissioned*, desenvolvida pela Linux Foundation, de arquitetura modular. Esta, disponibiliza SDKs para as linguagens Python, JavaScript C++, Go, Rust, Swift e Java, que permitem o desenvolvimento de *transaction processors*, assinatura digital por clientes e subscrição de eventos. Uma rede em Hyperledger Sawtooth caracteriza-se pela conexão dos múltiplos *validators*, como representado na **Figura 2.7**, no entanto, um nodo da mesma é constituído por:

- **REST API** - interface projetada exclusivamente para o uso de clientes na interação com o *validator*, apenas reencaminhando os pedidos do mesmo. No entanto, é também possível aos clientes substituírem esta interface por uma interface ZMQ/Protobuf. A comunicação com a Interface de Programação de Aplicações (API) é feita através de HTTP/JSON sob TCP, permitindo uma interface agnóstica quanto à sua linguagem para a submissão de transações e leitura de blocos.
- ***Validator*** - componente fulcral ao funcionamento de Hyperledger Sawtooth enquanto DLT, sendo o seu principal objetivo a validação de todas as transições realizadas pelos clientes e o posterior redirecionamento destas ao *transaction processor* adequado.
- **Motor do Mecanismo de Consenso**
- ***Transaction Processor*(s)** - visa executar a lógica ao nível da aplicação e deste modo está capacitado para permitir, ou recusar, a alteração do estado do *ledger* com base na lógica que determinada transação pretende que seja executada.

Uma rede em Hyperledger Sawtooth pode ser definida enquanto *permissioned* ou *permissionless*, com base na configuração de permissões do *ledger*. O *ledger* armazena as configurações necessárias face a permissões, nodos e identidades. A arquitetura desta plataforma *blockchain* isola o funcionamento da rede do ambiente específico da aplicação, isto é, o processamento de lógica específica a uma aplicação, que é feito através do *transaction processor*, encontra-se isolado da restante operacionalidade da rede (i.e., o *transaction processor* apresenta-se enquanto um programa com execução isolada cujo término não coloca em causa a operacionalidade da rede Hyperledger Sawtooth). Este isolamento é garantida face aos módulos constituintes de um nodo, em Hyperledger Sawtooth, serem processos de execução independentes.

O mecanismo de consenso em Hyperledger Sawtooth é definido através de um parâmetro armazenado *on-chain*. Este parâmetro, embora seja requerido de especificar no bloco *genesis*, é possível de ser alterado durante a execução da rede. A alteração deste é possível uma vez que o seu valor se encontra armazenado no *ledger*, no entanto, requer a submissão de uma nova transação específica. Os mecanismos de consenso suportados pela plataforma *blockchain* são, PBFT, PoET CFT e Raft. São ainda suportados os mecanismos de consenso Devmode (não recomendável para produção), cujo objetivo é garantir um ambiente de teste simples, e PoET SGX, que requer *hardware* especializado e que permite garantir um funcionamento BFT semelhante a PoW.

O estado global do *ledger* é armazenado numa base de dados chave-valor - *Lightning Memory-Mapped Database* (LMDB) no entanto, representa-se para todas as *transaction families* através de uma única instância de *Radix-Merkle Tree* em cada *validator*. Os *transaction processors* armazenam *arrays* de *bytes* de dados em nós folha da árvore. Os nós,

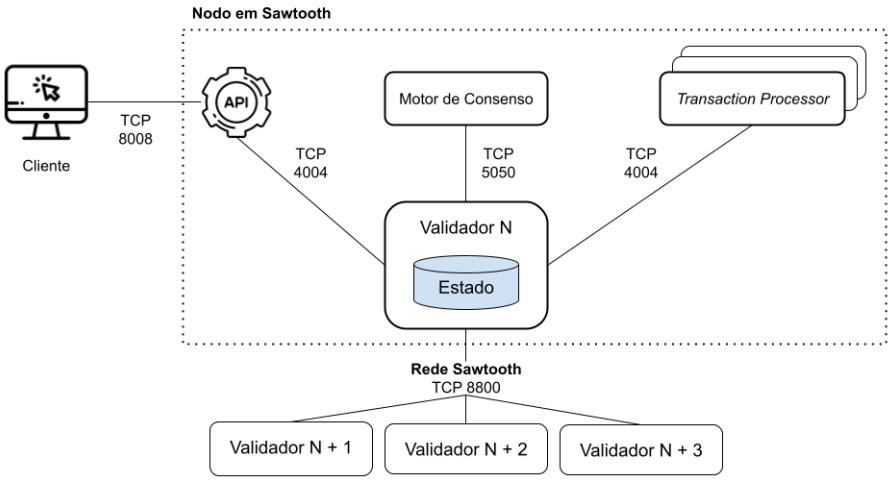


Figura 2.7: Arquitetura da plataforma *blockchain* Hyperledger Sawtooth

desde os nós folha até à raiz, desta estrutura de dados endereçável armazenam o seu valor de *hash*. A *hash* da raiz resulta de um processo cumulativo das *hashes* dos nós das suas folhas, o que lhe permite representar a versão do estado global. Se os dados de algum destes nós forem alterados, será consequentemente gerado um novo valor para a sua *hash*, o que provocará uma sucessiva alteração de todas as *hashes* na mesma ramificação e consequentemente da raiz. Um endereço é representado por uma *string* de 35 bytes (i.e., 70 caracteres) onde cada byte é um segmento de um caminho na *Radix-Merkle Tree* que identifica o próximo nó no caminho para o nó folha que contém os dados associados pelo endereço. Os primeiros 3 *bytes* de um endereço representam o *namespace* da *transaction family*. Os restantes 32 *bytes* são codificados com base nas especificações da *transaction family*, uma vez que esta pode incluir diversos esquemas para, por exemplo, distinguir o tipo de objetos a ser armazenado.

Transaction Families são o design do conjunto de transações possíveis para determinada aplicação. Uma *transaction family* define-se pelo seu nome, a sua versão - que permite atualizar uma *transaction family* em coordenação com os nodos da rede - e pelo seu *namespace*. Esta é constituída por um *transaction processor* que define a sua lógica (i.e., lógica da aplicação) e pelo conjunto de normas que esta lógica deve obedecer, nomeadamente na atualização do estado do *ledger* (i.e., deve ser utilizado o *namespace* da *transaction family* para identificar os endereços da mesma na indexação feita na *Radix-Merkle Tree*). A plataforma *blockchain* disponibiliza através do seu *software*, as *transaction families*: Settings - permite o armazenamento das configurações no *ledger* -, Identity, BlockInfo, IntegerKey, XO, PoET Validator Registry, Smallbank, Sawtooth Sabre e Sawtooth Seth - permite utilizar *smart contracts* desenvolvidos em Solidity para Ethereum.

Transações, em Hyperledger Sawtooth, são agrupadas dentro de um *batch* que se caracteriza por ser a unidade atómica de alteração de estado. Um *batch* agrupa uma ou mais transações relacionadas e cujo *commit* no estado global do *ledger* depende da validação de todas (i.e., se uma transação não é validada, todas as transações contidas no respetivo *batch* são consideradas como invalidadas). A execução de transações pode ser definida para operar em série ou em paralelo - permite que a execução de transações seja dividida em fluxos paralelos, cuja integridade é garantida através da subdivisão do processamento com base nos locais do estado que as transações modifiquem. *Batches* e transações são serializados com recurso a Protocol Buffers.

O fluxo transacional em Hyperledger Sawtooth - *Order-Execute-Commit* - inicia-se com

a submissão de uma nova transação, inserida num *batch*, pelo cliente à REST API de um nodo da rede. Este componente redireciona o *batch* ao *validator*, que irá verificar as condições de permissão (i.e., verificar a assinatura do respetivo *batch* de acordo com as permissões permitidas pelo *ledger* para a submissão de transações), de assinatura (i.e., verificar a integridade dos dados) e de estrutura (i.e., verificar a composição estrutural do *batch* como, por exemplo, transações duplicadas). Caso estas verificações seja bem sucedidas, as transações inseridas no *batch* são redirecionadas aos *transaction processors* a que se destinam. De seguida é executada a lógica do *transaction processor* que a transação invoca e, caso seja validada, é enviada uma mensagem ao *validator*. O *validator* insere a transação validada num bloco que será adicionado ao *ledger* e difundido pela rede para os restantes *validators*.

2.3.8 Soluções Existentes Aplicadas ao Setor Agroalimentar

O caso de uso desta dissertação, certificação digital no setor agroalimentar, pode enquadrar-se nos desenvolvimentos tecnológicos da indústria agrícola face à sua cadeia de abastecimento, agricultura inteligente ou *e-commerce*.

A aplicação da tecnologia *blockchain* na agricultura inteligente promove tecnologias *data-driven*, dadas as seguranças que oferece no armazenamento imutável de dados. A segurança que garante aos dados armazenados e mecanismos transacionais, elimina a necessidade de intermediários permitindo transações entre *stakeholders* sem mútua confiança. A sua integração em sistemas de cadeias de abastecimento complexas permite a descentralização dos dados pela rede e, tendo os *stakeholders* acesso à mesma, uma melhoria na rastreabilidade de produtos. O mecanismo de *smart contracts* permite a invocação de pagamentos entre *stakeholders*, aquando da inserção de determinados dados no *ledger*. [35] [36]

Os trabalhos de Patil *et al.* [37] e Lin *et al.* [38] [39] permitiram contínuos estudos e desenvolvimentos da aplicação da tecnologia *blockchain* no setor agroalimentar. As soluções propostas reúnem *blockchain* com IoT com o objetivo de recolha de dados para monitorização da produção (embora possa ser estendida à cadeia de abastecimento do setor), armazenando os mesmos de forma imutável num *ledger* acessível aos seus *stakeholders* com base em permissões. [36]

A Walmart [40] desenvolveu duas provas de conceito baseadas em *blockchain*, utilizando Hyperledger Fabric num sistema para rastreabilidade de carne de porco na China e mangas nos Estados Unidos da América. Foi-lhes possível, no caso de uso da carne de porco, submeter certificados de autenticidade na *blockchain* e, no caso de uso das mangas, rastrear a proveniência das mesmas em 2.2 segundos num processo que tradicionalmente demoraria 7 dias. Deste modo o sistema permitiu validar a aplicabilidade da tecnologia *blockchain* na rastreabilidade de produtos no setor agroalimentar.

A TE-FOOD [41][42] desenvolveu a sua própria plataforma *blockchain* - TrustChain - para integração num ecossistema para rastreabilidade de produtos agroalimentares. TrustChain é uma plataforma *blockchain* pública *permissioned* cujos nodos estão distribuídos geograficamente. Estes podem ser geridos por empresas integrantes das cadeias de abastecimento - *Supernodes* - ou por consumidores - *Masternodes*. Os dados armazenados no seu *ledger* são publicamente acessíveis para leitura, desde que disponibilizados pelas empresas para tal, mas a sua escrita requer permissões adicionais de segurança.

SmartAgriChain é um projeto em desenvolvimento pela Zenithwings que pretende agregar as entidades e fases associadas à cadeia de abastecimento do setor agroalimentar numa solução tecnológica *web*. Embora a sua visão seja desenvolver um *marketplace* de serviços

para a cadeia de abastecimento do setor agroalimentar, a sua investigação e desenvolvimento encontra-se atualmente a validar a aplicabilidade da automatização do processo de certificação integrando tecnologia *blockchain*.

2.4 Análise de Desempenho de Plataformas Blockchain

Benchmarking de *software* corresponde ao processo de avaliar o desempenho de um sistema, módulo, ou operação, comparativamente a outro(a), ou a diferentes versões do mesmo. [43] Tipicamente, os *benchmarks* subdividem-se em micro *benchmarks* [44], cujo objetivo é medir o desempenho de um módulo (p.e., *subroutines*) de *software*, e macro *benchmarks*, cujo objetivo é medir o desempenho de um conjunto de módulos ou do sistema na sua integra. Os *benchmarks* de desempenho caracterizam-se por avaliarem o desempenho de uma amostra com base em padrões formalizados pela indústria (p.e., organizações como a Standard Performance Evaluation Corporation (SPEC) ou a Transaction Processing Performance Council (TPC)). [43]

Por outro lado, a análise de desempenho é o processo de medir o desempenho de um Sistema em Teste (SUT) com o objetivo de observar e documentar o desempenho do mesmo, alterando determinadas variáveis. A veracidade destas medições está dependente da relevância - a sua interpretação deve ser apropriada ao sistema a ser testado -, compreensão - a análise deve ser compreensível - e aceitação - qualquer entidade deve reconhecer a imparcialidade dos resultados - dos *workloads* que serão definidos e executados. Este processo difere de *benchmarking* no âmbito em que não é tão rígido no seu controlo, sendo que os *benchmarks* são executados num ambiente padrão e recorrem a *workloads* devidamente documentados e padronizados. [43]

No contexto da tecnologia *blockchain*, o facto de existirem diversos tipos de plataformas *blockchain*, com diferentes níveis de complexidade e soluções arquiteturais, dificulta a padronização de *benchmarks* e de técnicas de análise. Por exemplo, no caso de plataformas *blockchain* públicas é impossível possuir controlo sob o *workload* real e os participantes no mecanismo de consenso. Deste modo, uma das alternativas é implementar uma rede privada da respetiva plataforma e avaliar esta rede com base em *workloads* artificiais. Porém, devem ser apresentados melhores valores de desempenho comparativamente à sua implementação na rede pública e devem ser considerados problemas de escalabilidade aquando da sua implementação pública. Outra alternativa é monitorizar e avaliar o desempenho em tempo-real da rede pública sob um *workload* realista. [45] [1]

Perante a falta de padrões para a execução de *benchmarking* em plataformas *blockchain*, e uma vez que esta dissertação se direciona a um caso de uso específico, o seu desenvolvimento visa realizar uma análise do desempenho das plataformas *blockchain*, Ethereum e Hyperledger Sawtooth, com recurso a um modelo de dados dedicado. Este modelo de dados deverá implementar a lógica requerida pelo caso de uso. Também os *workloads* a executar serão definidos de acordo com as análises a realizar aos SUT. Assim, torna-se necessário definir como se procederá à execução destes sobre o SUT. As seguintes subsecções pretendem validar o *software* a utilizar nesta análise através de uma análise das plataformas existentes para *benchmarking*.

2.4.1 Plataformas Existentes

Com o desenvolvimento de *benchmarking* para sistemas em *cloud* (p.e., através de Hadoop, Mapreduce e Spark) e sistemas de bases de dados, foram desenvolvidos padrões industriais

como as *frameworks* TPC-C [46] - *benchmarks* para plataformas de bases de dados -, YCSB [47] - *benchmark* para bases de dados não relacionais - e Smallbank [48] - *benchmark* para *workloads* OLTP. [1]

A diversidade de mecanismos de consenso e APIs em DLTs que vêm continuamente a emergir, não possibilita a aplicação destes *benchmarks* através de um *benchmark* ou *framework* padrão. [1] No entanto, o contínuo desenvolvimento de DLTs que pretendem validar um aumento de desempenho, contribuíram para o desenvolvimento de três *frameworks* de *benchmarking* dedicadas à avaliação das mesmas - Blockbench, Hyperledger Caliper e DAGbench. [1] No entanto, a *framework* DAGbench é direcionada a DLTs DAG e portanto não se enquadra no âmbito desta dissertação, uma vez que se pretende analisar o desempenho de plataformas *blockchain*.

De notar que existem ainda *frameworks* de *benchmarking* que vêm a ser desenvolvidas específicas a determinadas métricas e plataformas *blockchain*. A *framework* OpBench [49] e a *framework* descrita em [50] pretendem avaliar as recompensas de um nodo *miner*, na plataforma *blockchain* Ethereum, face aos custos de execução de *smart contracts*.[1]

Sendo a análise de desempenho de plataformas *blockchain*, o principal objetivo desta dissertação, é necessário validar a *framework* a utilizar. Deste modo, a **Tabela 2.2** apresenta uma comparação das *frameworks* de *benchmarking* de plataformas *blockchain* mencionadas anteriormente.

Apesar dos dados dos repositórios não identificarem a maturidade do *software*, com base no *timestamp* número e de *commits*, é possível identificar que o repositório da *framework* Hyperledger Caliper apresenta constantes atualizações e que, face ao número de contribuidores, é amplamente revisto. [51] [52]

Por outro lado, e com base nas plataformas *blockchain* definidas para a análise desta dissertação, tanto Hyperledger Caliper como Blockbench suportam as plataformas *blockchain* Ethereum e Hyperledger Fabric, no entanto, o mesmo não ocorre para Hyperledger Sawtooth. Embora estas *frameworks* sejam multi-plataforma, seria necessário desenvolver conectores/adaptadores para que as *frameworks* Hyperledger Caliper v.0.4.2 e Blockbench integrassem redes Hyperledger Sawtooth. Este desenvolvimento seria um possível risco de falha no desenvolvimento desta dissertação.

Framework	**DLTs Suportadas**	**Métricas**	**Nº *Commits* / Contribuidores**	**Última Atualização**
Blockbench [53]	Ethereum; Parity; Quorum; Hyperledger Fabric (v0.6, v.1.4 e v.2.2) [53] [52]	Taxa de Transferência; Latência; Escalabilidade; Tolerância a Falhas [53]	70 / 6 [52]	Agosto 2020 [52]
Hyperledger Caliper v0.4.2 [54]	Hyperledger Besu; Ethereum; Hyperledger Fabric; FISCO BCOS [55]	Taxa de Transferência; Latência; Consumo de Recursos [55]	821 / 43 [51]	Março 2021 [51]
Hyperledger Caliper v0.3.2 [54]	Hyperledger Besu; Hyperledger Burrow; Ethereum; Hyperledger Fabric; FISCO BCOS; Hyperledger Iroha; Hyperledger Sawtooth [56]	Taxa de Transferência; Latência; Consumo de Recursos [56]	N/A	N/A

Tabela 2.2: Comparação das *frameworks* de *benchmarking*

Assim, o autor desta dissertação, validando com a instituição acolhedora e com base nos fatores mencionados, optou por selecionar a *framework* Hyperledger Caliper na versão 0.3.2 para o desenvolvimento desta dissertação.

Hyperledger Caliper

Hyperledger Caliper [54] é uma *framework* para *benchmarking* de plataformas *blockchain* privadas, cujo design promove a extensibilidade e escalabilidade para integração de soluções de infraestrutura e monitorização. A sua avaliação tem por base as métricas: taxa de transferência transacional, latência (mínima, máxima, média, por percentil) transacional e consumo de recursos (CPU, RAM, rede e E/S). O seu código fonte é desenvolvido em JavaScript.

A sua arquitetura modular é baseada em conectores para a integração de plataformas *blockchain*, permitindo-lhe a integração de plataformas além daquelas a que dá suporte. Na sua versão 0.3.2 suporta as plataformas *blockchain* Hyperledger Besu, Hyperledger Burrow, Ethereum, Hyperledger Fabric, FISCO BCOS, Hyperledger Iroha e Hyperledger Sawtooth. Na mais recente versão 0.4.2 suporta as plataformas *blockchain* Hyperledger Besu, Ethereum, Hyperledger Fabric e FISCO BCOS.

A execução de um *benchmark*, independentemente do SUT, requer, como argumentos:

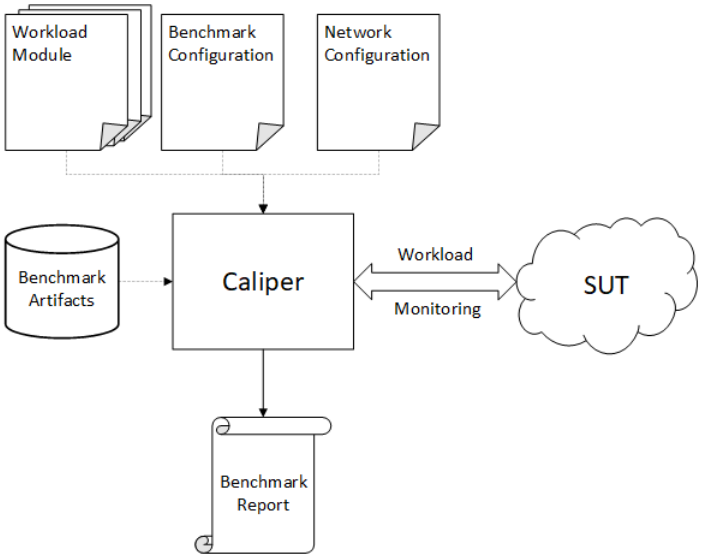
- Ficheiro de Configuração de *Benchmark*: descreve a execução do *benchmark* à *framework*, caracterizando-se como orquestrador. Indica o número de rondas que devem ser executadas, a que taxa as transações devem ser submetidas, o módulo que irá gerar o conteúdo da transação e as definições de monitorização e de observação do SUT. Sendo estas definições independentes do SUT, podem ser reutilizadas em *benchmarks* a diferentes SUT ou versões dos mesmos.

- Ficheiro de Configuração da Rede: descreve a topologia do SUT, isto é, os *endpoints* dos seus nodos, os endereços que irão assinar as transações a realizar pelo *workload*, e as aplicações *blockchain* que podem vir a ser *deployed* ou com as quais se poderá interagir. Estas configurações são específicas ao SUT.

- Definição dos Módulos de *Workload*: são as tarefas a executar nas rondas de *benchmark* cujo funcionamento consiste em gerar e submeter as transações pretendidas pelo teste. Estes módulos são módulos Node.js que devem exportar uma função de fábrica, e que podem conter qualquer lógica da linguagem.

- Artefactos: artefactos adicionais que podem ser necessários ao *benchmark*. Tipicamente incluem materiais criptográficos necessários para interagir com o SUT, código fonte de aplicações *blockchain* que a *framework* precisa para o seu *deploy*, invocação e consulta.

Como é possível verificar na **Figura 2.8**, os *benchmarks* geram *workloads* no SUT e, durante a sua execução, é monitorizada a informação do consumo de recursos e os seus clientes publicam a informação de taxa de transação num analisador de desempenho. No final da execução do *benchmark* é gerado um relatório baseado nas respostas observadas no SUT.



Fonte: `https://hyperledger.github.io/caliper/v0.3.2/architecture`

Figura 2.8: Arquitetura da *framework* Hyperledger Caliper

Além da sua extensibilidade/flexibilidade, Hyperledger Caliper visa escalabilidade no seu funcionamento, uma vez que o limite de recursos de uma máquina pode ser rapidamente atingido quando apenas esta gera o *workload* do *benchmark*. Deste modo, para corresponder às características de escalabilidade e desempenho do SUT, a *framework* recorre a uma abordagem distribuída através dos serviços/processos:

- *Manager*: inicializa o SUT e coordena a execução do *benchmark* e gere as estatísticas observadas nas transações realizadas, de forma a ser possível de gerar o relatório de *benchmark*;
- *Worker*: executa o *benchmark* de forma independente aos restantes, caso existam. Caso este processo atinja o limite de recursos, ao utilizar mais processos *worker* (em diferentes máquinas) é possível aumentar a taxa de *workload*.

2.4.2 Trabalho Relacionado

Apesar dos desenvolvimentos de *frameworks* de *benchmarking*, e outras ferramentas, continuam a não existir padrões para a comparação entre diferentes plataformas *blockchain*. [1] No entanto, recorrendo às *frameworks* de *benchmarking* anteriormente mencionadas ou outras ferramentas, tem sido desenvolvidos estudos para analisar o desempenho de plataformas *blockchain*. De seguida serão apresentadas análises empíricas cujos estudos pretendem concluir acerca do desempenho de plataformas *blockchain*.

Ethereum

Rouhani e Deters [57] analisaram o desempenho da plataforma *blockchain* Ethereum em redes privadas através da comparação entre o desempenho dos seus clientes Geth e Parity. Os seus resultados demonstram uma melhoria de desempenho de 89.92% em Parity, comparativamente a Geth, em termos do desempenho médio no processamento de transações sob diferentes *workloads*.

Bez *et al.* [58] desenvolveram uma análise acerca da escalabilidade de uma rede Ethereum num ambiente de teste extensível e tendo como métrica a taxa de transferência transacional. Os resultados indicaram que a plataforma *blockchain* Ethereum é sujeita ao trilema da escalabilidade mencionado por Vitalik Buterin: uma plataforma *blockchain* dificilmente consegue, simultaneamente, atingir a descentralização, escalabilidade e segurança. Este fator é referenciado na documentação de Ethereum [59] onde é mencionado que com o aumento dos pedidos ao sistema, a taxa de transferência transacional diminui e o preço do *gas* torna-se inviável.

No repositório da *framework* [52], e referente ao trabalho desenvolvido em [53], é identificado que Ethereum é mais resiliente a falhas nos seus nodos, mas apresenta maior vulnerabilidade a ataques de segurança que façam *fork* do *ledger*. Além disso, é mencionado que Ethereum apresenta maior *overhead* de consumo de memória e disco.

Hyperledger Sawtooth

Shi *et al.* [60] analisaram o desempenho de uma rede Hyperledger Sawtooth com PoET em ambiente *cloud*. A sua análise baseou-se na consistência (i.e., se o desempenho da plataforma se comportou de forma consistente a cada iteração para o mesmo *workload* e configuração de rede), estabilidade (i.e., se o desempenho da plataforma se manteve estável a cada iteração para o mesmo *workload*, mas para configurações de rede diferentes) e na escalabilidade (i.e., como é que a plataforma escalou com diferentes *workloads* e configurações de rede). Os seus resultados permitiram-lhes concluir que: a taxa de transferência transacional estabelecida no *benchmark* tem um impacto significante no desempenho da rede; quanto maior a taxa de transferência transacional, menor era o desempenho da rede, o que significa que o desempenho é afetado por maiores cargas em menores períodos de tempo; o desempenho da rede foi consistente face a uma escalabilidade do número de nodos participantes; o processamento paralelo aumentou o desempenho da rede em 30%.

Análises Comparativas

Caro *et al.* [61] analisaram o desempenho das plataformas *blockchain* Ethereum e Hyperledger Sawtooth, em integração com IoT para a rastreabilidade de produtos na cadeia de abastecimento do setor agroalimentar. Foi-lhes possível concluir que Hyperledger Sawtooth apresentou melhor desempenho, no entanto, apesar da latência em Ethereum, mencionam que poderia ser aplicada em redes com grande volume de participantes devido à segurança que garante e capacidade de suportar os mesmos. Referem ainda a computação intensiva requerida por PoW em Ethereum enquanto limitação à sua inclusão em IoT, no entanto, Hyperledger Sawtooth ainda é um *software* pouco maturado.

Pongnumkul *et al.* [62] concluíram que o desempenho das plataformas Hyperledger Fabric e Ethereum se mantém baixo, nomeadamente em *workloads* de carga elevada, comparativamente a sistemas de gestão de bases de dados (SGBD) centralizados (p.e., PostgreSQL). Esta análise foi testada e comprovada por Chen *et al.* [63], numa análise empírica que comparou o desempenho de uma rede Ethereum com o SGBD MySQL. A taxa de transferência transacional da tecnologia *blockchain* é afetada uma vez que o seu funcionamento consiste na integração de conjuntos de transações em blocos que são posteriormente inseridos e organizados num *ledger* único a todos os nodos da rede. Deste modo, o *ledger* não possibilita gerar blocos de forma concorrente. [1]

Benahmed *et al.* [64] analisaram o desempenho das plataformas *blockchain* Hyperledger

Sawtooth, EOSIO e Ethereum com recurso à *framework* de *benchmarking* Blockbench. Definiram os *workloads* CPUHeavy, KVStore e SmallBank para testar o consumo de CPU, memória, escalabilidade de carga e escalabilidade da rede. Concluíram que a plataforma *blockchain* EOSIO apresentou um desempenho superior às restantes no consumo de recursos e velocidade, no entanto, apresenta como lacuna a centralização. Por outro lado, direcionam Hyperledger Sawtooth para casos de uso com IoT e Ethereum, com o mecanismo de consenso PoA, para o desenvolvimento de aplicações *web*.

Dinh et al. [53] analisaram o desempenho das plataformas *blockchain* Ethereum (cliente Geth), Parity e Hyperledger Fabric em redes privadas, com recurso à *framework* de *benchmarking* Blockbench. Os seus resultados permitiram concluir que Hyperledger Fabric apresentou um desempenho superior às restantes plataformas nos macro-*benchmarks* e micro-*benchmarks*, no entanto, apresenta deficiências na escalabilidade da rede acima de 16 nodos. Concluiu também que o maior *bottleneck* das plataformas *blockchain* Hyperledger Fabric e Ethereum é o mecanismo de consenso. Os autores mencionam que as lacunas de desempenho destas plataformas provêm de decisões arquiteturais em diferentes camadas do *software*.

Visão Geral Final

Finalizado o estado da arte, pretende-se que o desenvolvimento desta dissertação cumpra os principais objetivos estabelecidos no **Capítulo 1**, **Secção 1.2**.

Inicialmente será definido o modelo de dados e lógica necessária para que as aplicações *blockchain* (i.e., *smart contracts*) executem os requisitos do caso de uso da certificação digital de produtos no setor agroalimentar. Posteriormente, e como já mencionado, serão implementadas as redes que permitem analisar o desempenho das plataformas *blockchain* Ethereum, e Hyperledger Sawtooth, integrando as aplicações previamente modeladas e com recurso à *framework* de *benchmarking* Hyperledger Caliper.

Face às análises de desempenho descritas neste capítulo na **Subsecção 2.4.2** foram identificados dois fatores com maior impacto no desempenho das respetivas plataformas *blockchain*: mecanismo de consenso e escalabilidade da rede. Deste modo, estes serão dois possíveis *bottlenecks* a ser analisados no desenvolvimento desta dissertação.

Além dos fatores mencionados, pretende-se que a análise desta dissertação valide as seguintes conclusões, sintetizadas com base nos resultados descritos neste capítulo na **Subsecção 2.4.2**:

Em suma, pretende-se que o desenvolvimento desta dissertação identifique possíveis *bottlenecks* e conclua acerca da plataforma *blockchain* que apresenta melhor desempenho face ao caso de uso desta dissertação.

Capítulo 3

Planeamento e Metodologia

O planeamento de um projeto tem como objetivo garantir que o resultado é apresentado no prazo previsto, dentro do orçamento e com a qualidade requerida. O desta dissertação baseou-se na definição de mecanismos que permitissem garantir o cumprimento do seu objetivo, tais como: enumeração de tarefas a realizar, cálculo do seu esforço estimado, definição da metodologia e processos a seguir, e definição dos riscos.

A metodologia consiste na descrição do ciclo de vida de desenvolvimento do projeto, e seus processos, que serão definidos de acordo com os objetivos a alcançar. Deste modo, será enunciada a metodologia e processos utilizados para suportar o desenvolvimento desta dissertação, e organização das suas tarefas, bem como as ferramentas de gestão de projeto utilizadas.

A existência de riscos é inerente a qualquer projeto e estes devem ser identificados numa fase precoce, permitindo a elaboração de planos de mitigação. Esta estratégia é um fator que pode permitir o aumento da probabilidade de sucesso do projeto, uma vez que os seus riscos são ponderados e a sua mitigação é estruturada. Assim, será apresentada a análise dos riscos identificados ao desenvolvimento desta dissertação, sob a forma de tabela, e serão posteriormente descritos os planos de mitigação elaborados.

3.1 Planeamento

Encontrando-se previamente definido o âmbito do projeto na Proposta de Estágio, foram definidas as tarefas necessárias à sua realização na **Tabela 3.1**. As tarefas foram distribuídas em duas fases de acordo com o semestre, e respetiva fase do projeto, na qual serão realizadas. As alocações do seu esforço estimado podem ser visualizadas graficamente para a 1º e 2º fase deste projeto, respetivamente, através dos diagramas de Gantt das **Figuras 3.1** e **3.2**.

Título	Tipo	Início	Fim	%
FASE 1 - ESTUDO DA SOLUÇÃO E PLANEAMENTO	Tarefa	14/09/2020	23/01/2021	100%
T1 – Elaboração do Estado da Arte	Tarefa	14/09/2020	31/10/2020	100%
T2 - Definição da Arquitetura	Tarefa	01/11/2020	31/12/2020	100%
T3 - Elaboração do Relatório Intermédio	Tarefa	21/09/2020	30/11/2020	100%
Entrega do Relatório Intermédio	Meta	-	18/01/2021	100%
Criação da Apresentação Intermédia	Tarefa	19/01/2021	24/01/2021	100%
FASE 2 - DESENVOLVIMENTO E CONCLUSÃO	Tarefa	04/01/2021	14/07/2021	100%
T4 – Aprendizagem das Tecnologias	Tarefa	04/01/2021	22/02/2021	100%
T5 – Definição do Modelo de Dados	Tarefa	22/02/2021	10/03/2021	100%
T6 – Desenvolvimento das Aplicações *Blockchain*	Tarefa	10/03/2021	05/04/2021	100%
T7 – *Deploy* das Redes para Teste	Tarefa	05/04/2021	05/05/2021	100%
T8 – *Benchmarking* e Análise de Resultados	Tarefa	05/05/2021	16/06/2021	100%
T9 – Elaboração do Relatório Final	Tarefa	01/03/2021	29/06/2021	100%
Entrega da Dissertação	Meta	-	30/06/2021	100%

Tabela 3.1: *Backlog* de Tarefas

3.1.1 1º Semestre

A tarefa T1 consistiu na investigação dos temas inerentes ao tema desta dissertação. O índice do Estado de Arte faz referência à linha temporal do processo de investigação que se iniciou com os temas Setor Agroalimentar e Certificação no Setor Agroalimentar, permitindo compreender o caso de uso. Posteriormente foi investigada a tecnologia Blockchain e o seu funcionamento, tendo sido analisadas as plataformas *blockchain* existentes e concluído acerca daquelas cujo desempenho seria testado na 2º fase deste projeto. Por fim, foi investigado o processo de Análise de Desempenho de Plataformas Blockchain e definida a *framework* de *benchmarking* a utilizar para a análise de desempenho das plataformas *blockchain*.

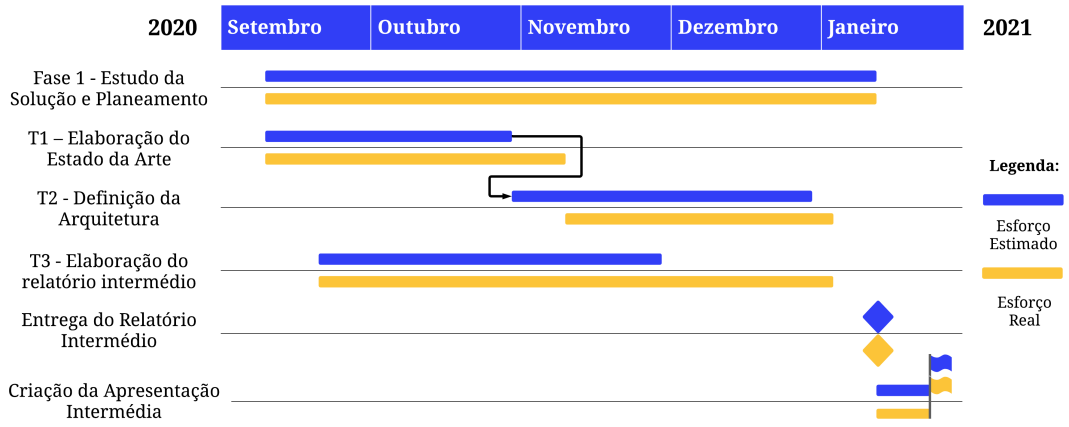


Figura 3.1: Diagrama de Gantt das tarefas do 1º semestre

A tarefa T2, face aos objetivos desta dissertação, consistiu na definição da arquitetura que

permitiu a realização dos *benchmarks* e a definição da arquitetura da prova de conceito. Face a *benchmarking*, a definição da arquitetura consistiu na planificação dos ambientes de teste que permitiram a análise de desempenho das plataformas *blockchain*.

Face ao trabalho concluído e descrito nas tarefas T1 e T2, e como resultado da tarefa T3 - meta desta 1º fase -, resultou a versão intermédia do relatório desta dissertação. Posteriormente foi necessário estruturar e preparar a apresentação intermédia.

3.1.2 2º Semestre

A tarefa T4 consistiu na aprendizagem das tecnologias necessárias ao desenvolvimento deste projeto. Foram desenvolvidos nodos locais para aprendizagem das plataformas *blockchain* Ethereum e Hyperledger Sawtooth, testadas ferramentas dedicadas ao desenvolvimento de aplicações *blockchain* (i.e., Truffle, Ganache e Web3.js) e seguidos tutoriais de desenvolvimento de *DApps* em Ethereum, de forma a perceber o que seria possível de desenvolver para a prova de conceito desta dissertação.

A tarefa T5 consistiu na definição do modelo de dados necessário ao desenvolvimento das aplicações *blockchain*.

A tarefa T6 consistiu nos desenvolvimentos do *smart contract* em Ethereum e *transaction family* em Hyperledger Sawtooth, que permitiram a execução da lógica necessária ao processo de certificação digital de produtos agroalimentares nas respetivas redes *blockchain*. Inicialmente foi desenvolvido o *smart contract* para Ethereum cujo contínuo desenvolvimento e validação foram feitos com recurso ao Remix IDE. Posteriormente foi desenvolvida a *transaction family* para Hyperledger Sawtooth, tendo como base a lógica do *smart contract* já desenvolvido, e cuja validação foi feita recorrendo a um nodo local.

A tarefa T7 consistiu no desenvolvimento e implementação das redes em ambiente de virtualização Docker para as plataformas *blockchain* Ethereum e Hyperledger Sawtooth. Esta tarefa requereu a devida aprendizagem dos conceitos de Docker e seu funcionamento.

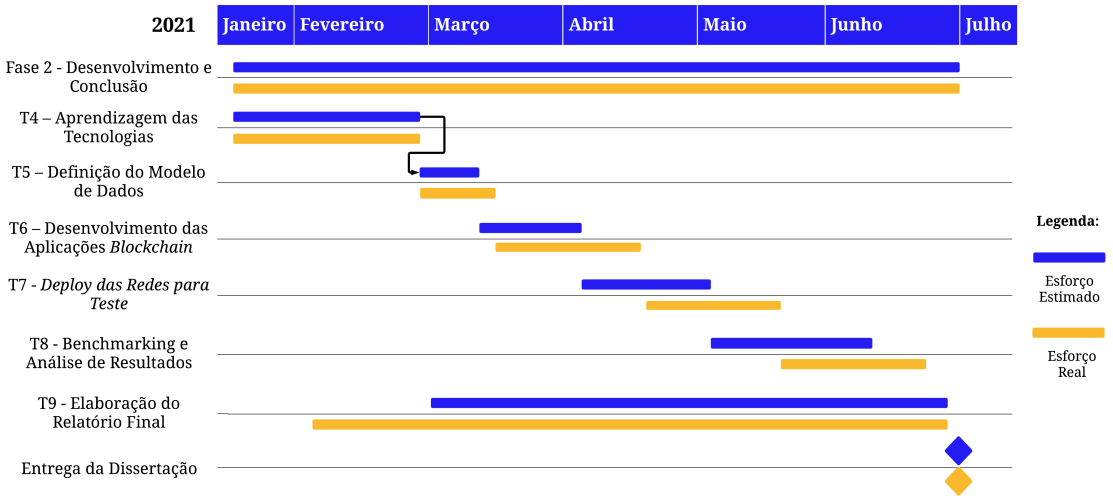


Figura 3.2: Diagrama de Gantt das tarefas do 2º semestre

A tarefa T8 consistiu na definição do plano de testes a executar e a validação deste perante a instituição acolhedora, com base nos requisitos que pretendem validar nesta dissertação. Posteriormente foram executados os *benchmarks* definidos no plano de testes e feita a sua

análise de acordo com o conhecimento adquirido e desenvolvido no Estado de Arte.

Face a todo o trabalho desenvolvimento neste projeto, e como resultado da tarefa T9 - meta desta 2º fase -, resultou a versão final do relatório desta dissertação. Posteriormente será estruturada e preparada a apresentação final da mesma.

3.2 Desvios ao Planeamento

Os diagramas de Gantt permitem-nos visualizar uma representação gráfica do esforço estimado e real face às tarefas alocadas, possibilitando o controlo do progresso das mesmas. Analisando os diagramas de Gantt das **Figuras 3.1** e **3.2** é possível identificar as seguintes anomalias:

- Atraso na realização da tarefa T1;
  - Causa: Carga letiva excessiva no decorrer desta tarefa (R05 da **Tabela 3.2**)
  - Impacto: Atraso na sua execução e início tardio da tarefa T2
  - Mitigação: Exercer cargas horárias extra na próxima tarefa de forma a garantir o cumprimento do planeamento para a 1º fase (PM01 da **Tabela 3.3**)
- Ligeiro atraso na finalização da tarefa T2;
  - Causa: Consequência do atraso sofrido na tarefa anterior (R05 da **Tabela 3.2**)
  - Impacto: Atraso na sua finalização face ao planeamento e consequente impacto no desenvolvimento da tarefa T3
  - Mitigação: Foram exercidas cargas horárias extra de forma a garantir o cumprimento do planeamento (PM01 da **Tabela 3.3**)
- Atraso da tarefa T3;
  - Causa: Subestimação da duração desta tarefa e paralela excessiva carga letiva (R07 da **Tabela 3.2**)
  - Impacto: Pouco tempo de revisão da versão intermédia relatório
  - Mitigação: Foram exercidas cargas horárias extra de forma a garantir o cumprimento do planeamento (PM01 da **Tabela 3.3**)
- Ligeiro atraso na tarefa T5;
  - Causa: Necessidade de realizar testes no âmbito do armazenamento de dados, uma vez que numa primeira fase de desenvolvimento não existiam conhecimentos técnicos suficientes das tecnologias (R05 da **Tabela 3.2**)
  - Impacto: A tarefa seguinte foi iniciada tardiamente face ao planeamento
  - Mitigação: Foram exercidas cargas horárias extra de forma a garantir o cumprimento do planeamento (PM01 da **Tabela 3.3**)
- Atraso na tarefa T6;
  - Causa: A complexidade das distintas arquiteturas das plataformas *blockchain* requereu uma investigação paralela ao desenvolvimento, tendo também este sido afetado pela inexperiência com as tecnologias e necessidade de executar diversos testes (R05 da **Tabela 3.2**)

  - Impacto: Atraso na sua execução e início tardio da tarefa T7
  - Mitigação: Foram exercidas cargas horárias extra de forma a garantir o cumprimento do planeamento (PM01 da **Tabela 3.3**)

- Atrasos na tarefa T7;

  - Causa: O seu atraso inicial deriva do atraso proveniente da tarefa T6 e na sua finalização existiram problemas no *deploy* das redes em Docker (R05 da **Tabela 3.2**)
  - Impacto: Atraso na sua execução e início tardio da tarefa T8
  - Mitigação: Foram exercidas cargas horárias extra de forma a garantir o cumprimento do planeamento (PM01 da **Tabela 3.3**)

- Atraso na tarefa T8;

  - Causa: Problemas na integração da *framework* de *benchmarking* com as redes anteriores (R05 da **Tabela 3.2**)
  - Impacto: Pouco tempo de revisão da versão final do relatório
  - Mitigação: Foram exercidas cargas horárias extra de forma a garantir o cumprimento do planeamento (PM01 da **Tabela 3.3**)

- Subestimação da tarefa T9;

  - Causa: Diversas notas de melhoria na defesa intermédia e escassez de muita informação no relatório (R05 da **Tabela 3.2**)
  - Impacto: Início precoce da sua realização e execução paralela com a tarefa T4
  - Mitigação: Foram exercidas cargas horárias extra de forma a garantir o cumprimento do planeamento (PM01 da **Tabela 3.3**)

3.3 Metodologia e Ferramentas de Gestão de Projetos

A instituição acolhedora previu o desenvolvimento desta dissertação num contexto isolado de investigação, onde o orientando desenvolveu o trabalho de forma individual, não sendo integrado numa equipa de desenvolvimento. Desta forma, a metodologia refere-se à logística que existiu, e representada na **Figura 3.3**, entre a instituição acolhedora, o seu orientando e respetivos orientadores. A logística aplicada assemelhou-se a uma metodologia ágil, dada a vantagem do desenvolvimento do projeto de forma iterativa com constante colaboração entre o orientando e, principalmente, o orientador Vítor Sousa. Sendo uma dissertação de investigação, a constante troca de *feedback*, tipicamente diária, permitiu aumentar a eficiência do desenvolvimento das tarefas, ao ser possível detetar, em fases precoces, defeitos/melhorias necessárias e rapidamente corrigir as mesmas.

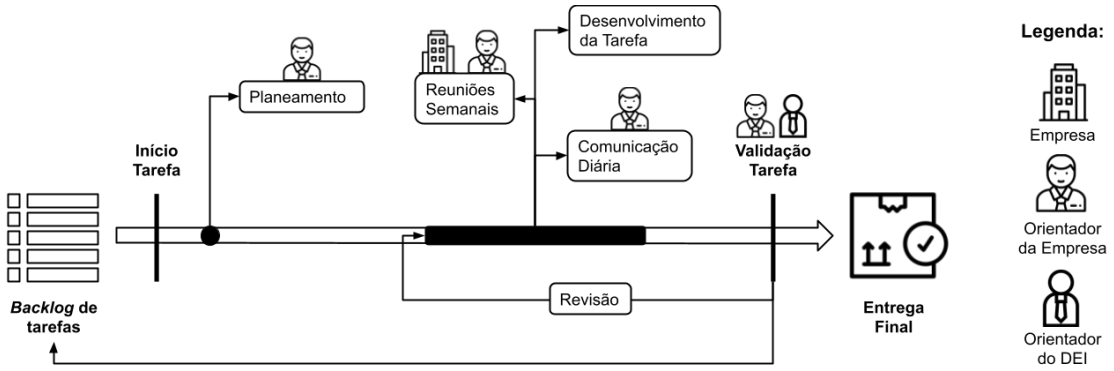


Figura 3.3: Metodologia ágil aplicada a esta dissertação

A política de desenvolvimento da instituição acolhedora segue abordagens ágeis, tendo a logística desta dissertação sido baseada no processo SCRUM. As entidades comunicaram diariamente entre si e existiram reuniões semanais entre a instituição acolhedora e o orientando, para controlo de progresso, e também entre o orientador da instituição acolhedora e o seu orientando, para controlo de progresso e revisão de *sprints* (i.e., validação de tarefas). Em conjunto com o orientador da universidade e com o orientador da instituição acolhedora foram definidas reuniões mensais que, análogas à revisão de *sprints*, visaram rever e validar o trabalho executado nas várias tarefas. Aquando da revisão de tarefas, estas seriam validadas ou, consoante os defeitos apontados e nível de impacto no projeto, planeava-se a extensão do seu desenvolvimento (como mencionado na Análise de Riscos). O *backlog* é uma característica do processo SCRUM que enumera o conjunto de tarefas necessárias à realização do projeto. Este encontra-se definido, juntamente com os prazos de execução das mesmas, na **Tabela 3.1**.

Para a gestão do projeto, respetivas tarefas e alocação de tempo das mesmas, a empresa propôs as ferramentas que já utilizava:

- ProofHub - plataforma tecnológica utilizada para a definição das tarefas necessárias e alocação do seu esforço (real e estimado), e para implementação dos diagramas de Gantt;
- Clockify - plataforma tecnológica utilizada para inserção das horas de trabalho realizadas, permitindo a gestão do esforço aplicado por parte da instituição acolhedora.
- BambooHR - plataforma tecnológica utilizada para inserção das horas de trabalho realizadas, permitindo a gestão financeira dos recursos humanos da instituição acolhedora.

3.4 Análise de Riscos

A análise dos possíveis riscos associados ao projeto é apresentada na **Tabela 3.2**, sendo estes caracterizados pela sua probabilidade de ocorrência e impacto estimados. A probabilidade de ocorrência é definida numa escala de 1 a 5, sendo que elementos com probabilidade 5 estimam-se mais prováveis de ocorrerem. O impacto é definido numa escala de 1 a 5, sendo que elementos com impacto 5 estimam-se que tenham maior impacto no projeto.

ID	Risco	Probabilidade	Impacto
R01	Indefinição, pela instituição acolhedora, do trabalho expectável, causando alterações tardias	2	4
R02	Adição de requisitos ou complexidade, pela Ubiwhere, causando anomalias no planeamento	2	4
R03	Remoção de requisitos ou complexidade, pela Ubiwhere, causando anomalias no planeamento	2	3
R04	Descontinuar de tecnologias em utilização, estagnando o desenvolvimento	1	5
R05	Atrasos na realização de tarefas	3	3
R06	Incapacidade da realização de tarefas por problemas derivados das tecnologias a utilizar	1	4
R07	Subestimação do esforço das tarefas	2	3

Tabela 3.2: Análise de riscos do projeto

Na **Tabela 3.3** são apresentados, quando possível, planos de mitigação para os riscos identificados na **Tabela 3.2**. Planos de mitigação consistem numa ação, ou conjunto de ações, que deve ser tomada de forma a minimizar os possíveis impactos inerentes à ocorrência de algum destes riscos.

ID	Plano de Mitigação	Riscos
PM01	Exercer cargas horárias extra de forma a garantir a execução das tarefas planeadas	R02; R05; R06; R07
PM02	Reunir com os orientadores desta dissertação de forma a validar reajustes no planeamento	R01; R03; R06
PM03	Atualização do projeto desenvolvido até ao momento, de forma a suportar as novas atualizações	R04

Tabela 3.3: Planos de mitigação face aos riscos do projeto

This page is intentionally left blank.

Capítulo 4

Design da Análise de Desempenho

Face ao principal objetivo desta dissertação, foi realizada uma análise do desempenho das plataformas *blockchain* Ethereum e Hyperledger Sawtooth, com recurso à *framework* Hyperledger Caliper. A respetiva análise requereu a definição/implementação dos seguintes componentes: métricas analisadas e o seu processo de medição, variáveis independentes da análise, modelo de dados e aplicações *blockchain* (i.e., *smart contracts* e *transaction processor*, redes das plataformas *blockchain*, *workloads* e metodologia.

Deste modo, este capítulo engloba as primeiras duas fases - Design e Implementação - do desenvolvimento desta dissertação. A primeira fase - Design - consiste em explicitar o design da análise e a segunda fase - Implementação - consiste em referir o trabalho desenvolvido para permitir a execução da mesma. Estas serão apresentadas homogeneamente para cada aspeto, visando referir o seu processo de design e de seguida a forma como foi implementada e as considerações que foram sendo realizadas.

4.1 Métricas

A análise de desempenho a desenvolver nesta dissertação, terá como base as métricas possíveis de obter pela *framework* Hyperledger Caliper. Deste modo, a *framework* permite-nos obter dois tipos de métricas: métricas de desempenho e de consumo de recursos.

Face ao consumo de recursos serão medidos, durante a execução dos testes, os parâmetros: [56] Processador - percentagem de processador consumido -, Memória RAM - quantidade máxima, mínima e média, em MB -, Tráfego - tráfico *ingress* (i.e., proveniente da rede exterior e com destino a um nodo da rede onde a máquina se encontra) e *egress* (i.e., o inverso de *ingress*), em KB - e Disco - quantidade de dados, em bytes, lidos e escritos em disco.

A documentação da *framework* [56] menciona que são suportadas as seguintes métricas de desempenho: taxa de transferência de leitura, taxa de transferência transacional, latência

de leitura e latência transacional. No entanto, após realizados testes com a *framework* conclui-se que as métricas disponíveis para análise do desempenho de plataformas *blockchain* são as seguintes: [43]

- **Taxa de Sucesso**: percentagem de transações bem sucedidas. Embora não seja uma métrica por defeito da *framework*, a mesma retorna o número total de transações bem sucedidas e falhadas, permitindo o seu cálculo.
- **Latência Transacional**: diferença, em segundos, entre o instante de submissão de uma transação e o instante do seu *commit* na rede, consequentemente afetando ou tornando-se acessível aos seus participantes. Esta métrica é calculada por transação e inclui o tempo de propagação da mesma pela rede e a atuação do mecanismo de consenso. No entanto, como mencionado em [43], para obter uma visão geral da rede acerca destes fatores é necessário fazer a medição da latência transacional em todos os nós do SUT. A *framework* apresenta os seus valores máximos, médios e mínimos, em segundos, obtidos durante a realização dos testes.

  Latência Transacional = (*Timestamp do commit da transação* @ limite da rede) - *Timestamp* da submissão da transação

  De notar que o parâmetro *limite da rede* considera a percentagem da rede sob a qual necessita de obter os tempos de confirmação para considerar a transação como avaliada. No entanto, como referido por [43], apenas é significativa para mecanismos de consenso probabilísticos (p.e., PoW) onde um limite como, por exemplo, 90%, pode ser desejado.
- **Taxa de Transferência Transacional**: quantidade de transações inseridas no *ledger* (apenas são consideradas as transações bem sucedidas) do SUT num determinado período de tempo, expresso em transações por segundo - TPS. Esta taxa é limitada pela combinação do intervalo de blocos - tempo decorrido entre a publicação de blocos consecutivos - e o tamanho dos blocos - número máximo de bytes por bloco -, que determina o número máximo de transações que podem ser incluídas num bloco.

  Taxa de Transferência Transacional = Número total de transações inseridas / Tempo total decorrido nas respetivas transações em segundos @ Total de nodos participantes do mecanismo de consenso

De realçar que estas métricas permitir-nos-ão analisar comparativamente o desempenho das plataformas *blockchain* Ethereum e Hyperledger Sawtooth. Estas são as métricas suportadas pela *framework* para uma análise do desempenho de plataformas *blockchain*. No entanto, poderiam ser utilizadas outras métricas específicas às plataformas *blockchain* como, por exemplo, em Ethereum o custo de *gas* ou em Hyperledger Sawtooth a latência média de resposta de um pedido da REST API ao *validator*.

4.2 Variáveis Independentes

Como mencionado no **Capítulo 1.2**, a instituição acolhedora visa o estudo da tecnologia *blockchain* na certificação de produtos agroalimentares. Deste modo, a análise de desem-

penho das plataformas *blockchain* Ethereum e Hyperledger Sawtooth, foca-se na análise da escalabilidade da rede, sendo uma limitação identificada nas mesmas, e na análise do seu desempenho perante a lógica implementada através das estruturas de dados descritas no **Capítulo 4.3**.

Relativamente à análise da escalabilidade da rede, este é um parâmetro que é definido face à camada de rede da mesma e que é caracterizado pela capacidade da rede da plataforma *blockchain* escalar o seu número de nodos (i.e., aumentar o número de nodos que a integram). Assim, para esta análise, a variável independente será o número de nodos integrantes da rede. Este parâmetro será configurado nos ficheiros de configuração de rede, como detalhado no **Capítulo 4.4**.

Relativamente à análise do seu desempenho face à lógica implementada, este parâmetro é definido face à camada de aplicação das plataformas *blockchain* Ethereum e Hyperledger Sawtooth, e está relacionado com os métodos que são invocados nos seus *smart contract* e *transaction processor*, respetivamente. Assim, para esta análise, a variável independente será o método a ser invocado nas aplicações *blockchain* mencionadas e será um parâmetro configurado nos ficheiros de configuração dos *workloads* (detalhados no **Capítulo 4.5**) a executar nas respetivas redes.

4.3 Modelo de Dados

Como mencionado no **Capítulo 2.4**, a falta de padrões para *benchmarking* de plataformas *blockchain*, permite que qualquer análise de desempenho da tecnologia recorra a diferentes dados nos seus *workloads*. Face ao mencionado, e uma vez que o desenvolvimento desta dissertação é motivado pelo estudo da tecnologia *blockchain* na certificação de produtos agroalimentares, os *workloads* a executar recorreram a um modelo de dados que modela a lógica requerida pelo caso de uso.

A implementação deste modelo de dados, como mencionado no **Capítulo 2.3.5**, é possível através do mecanismo de *smart contracts* da tecnologia *blockchain*. A sua implementação não só permitiu definir os *workloads* para a análise de desempenho das plataformas *blockchain* Ethereum e Hyperledger Sawtooth, como validar a capacidade destas executarem lógica que cumprisse os requisitos do caso de uso.

Análise de Requisitos

De forma a ser possível definir o modelo de dados a utilizar no desenvolvimento desta dissertação, a instituição acolhedora (através do orientador Vítor Sousa) explicitou a lógica requerida para o processo de certificação de produtos agroalimentares, tendo este sido definido no diagrama de sequência presente na **Figura 4.1** (embora o mesmo não contemple todos os casos possíveis do respetivo processo). Com base no processo definido, deve ser possível às seguintes entidades:

- Produtor:
  - Visualizar os serviços de certificação disponibilizados por uma entidade de certificação;
  - Requisitar um serviço de certificação, disponibilizado por uma entidade de certificação, sob um dos seus produtos;

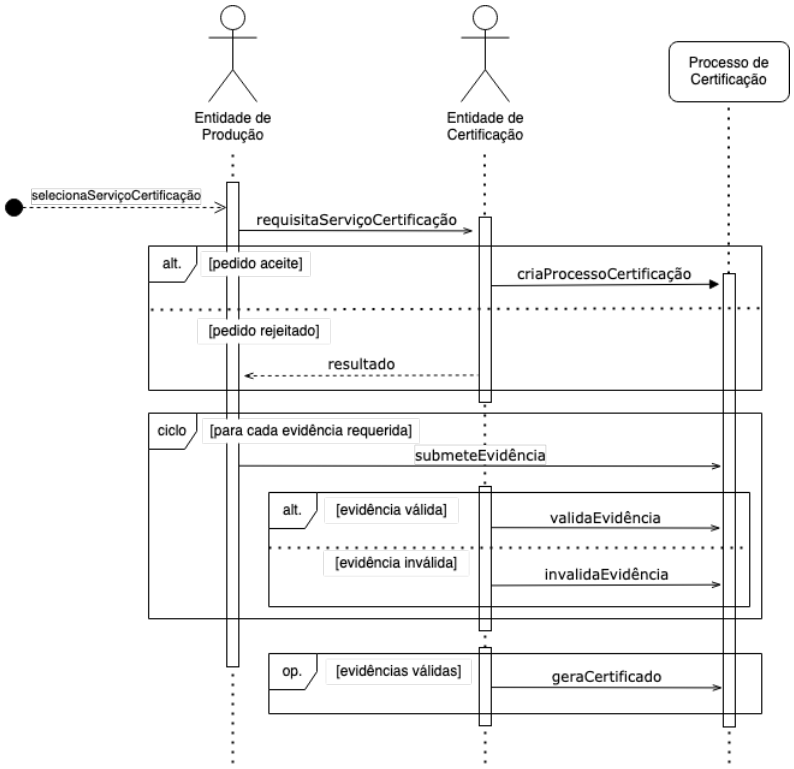


Figura 4.1: Lógica do processo de certificação de produtos agroalimentares

  - Submeter evidências a um processo de certificação em curso, sob um dos seus produtos;
  - Cancelar um processo de certificação em curso;
  - Visualizar os dados de um processo de certificação.

- Entidade de Certificação:

  - Disponibilizar serviços de certificação;
  - Aprovar/Rejeitar um pedido de certificação;
  - Aprovar/Rejeitar evidências submetidas por entidades de produção, face a um processo de certificação em curso;
  - Cancelar um processo de certificação em curso.

- Consumidor (i.e., qualquer entidade):

  - Visualizar a informação do processo de certificação de um produto;

Estruturas de Dados

Face às limitações de taxa de transferência identificadas nas plataformas *blockchain*, comparativamente a sistemas de bases de dados tradicionais (embora tenha surgido recentemente um conjunto de soluções que pretendem mitigar este problema como, por exemplo, Polkadot [65]), foi necessário definir os dados fulcrais (i.e., tendo excluído o armazenamento de, por exemplo, contactos das entidades) ao processo de certificação que deveriam ser armazenados no *ledger* das mesmas. No entanto, foi feita uma pequena pesquisa acerca

de possíveis soluções para mitigar este problema, tendo sido encontradas duas possíveis soluções:

1. Utilização de uma plataforma *blockchain* com maior taxa de transferência.

    - Contras: face ao dilema da escalabilidade, estas plataformas tendem a abdicar de outros aspetos para garantir maiores taxas de transferência, implicando um trade-off na gestão dos dados armazenados em plataformas *blockchain*. Por outro lado, embora se pudesse ter duas plataformas *blockchain*, para o armazenamento de dados com diferentes níveis de impacto, o custo do projeto escalaria podendo não ser rentável à instituição acolhedora.

2. Armazenamento dos dados menos fulcrais num SGBD centralizado, armazenando uma *hash* dos mesmos no *ledger* da plataforma *blockhcain* permitindo o mapeamento e garantia da imutabilidade dos dados entre estes sistemas.

    - Contras: o maior *trade-off* desta solução consiste em manter um sistema centralizado apesar da implementação de uma rede descentralizada para a plataforma *blockchain*. Ainda assim, a implementação destes dois sistemas iria novamente causar um aumento do custo do projeto, que poderia não ser rentável à instituição acolhedora.

Assim, no contexto da anterior análise de requisitos, foram identificadas as diferentes estruturas de dados referentes às entidades fulcrais ao caso de uso e que permitem a implementação da sua lógica. Deste modo, é possível promover uma melhor separação do código e assegurar a separação das respetivas entidades no modelo de dados, representado na **Figura 4.6**. Assim, foram definidas as seguintes estruturas de dados para o processo de certificação de produtos agroalimentares:

- **Produto**: armazena os dados acerca do produto de uma entidade de produção que pretende certificar a qualidade diferenciada do mesmo;

    – Uma vez que os requisitos da instituição acolhedora nada referem acerca dos dados necessários de armazenar de um produto, e de forma a simplificar a implementação, foi apenas definido armazenar o nome do produto.

- **Entidade de Produção**: armazena os dados necessários à identificação de uma entidade de produção (singular ou coletiva), cujo objetivo na sua interação com a rede é obter serviços de certificação sob os seus produtos. Os dados armazenados foram analisados com base numa pequena pesquisa e cujo exemplo de resultado se encontra no **Anexo C**;

    – Foi considerado como parâmetro de identificação único de uma entidade de produção o seu Número de Identificação Fiscal (NIF). Com o intuito de simplificar a implementação, foi limitado o âmbito do projeto a entidades com NIF português.

    – No desenvolvimento do *smart contract*, em Solidity, para a plataforma *blockchain* Ethereum, o NIF foi considerado *uint256*. Sendo o NIF um número de 9 dígitos, consideremos o seu valor(sem verificação) mais elevado - 999999999 -. Este pode ser representado com 30 bits, no entanto e devidamente testado, como a *Ethereum Virtual Machine* (EVM) funciona com 256 bits, ao definirmos

uma variável com uma potência diferente são realizadas conversões que fazem aumentar o custo de *gas* da transacção. Este foi um fator testado e tido em consideração neste desenvolvimento, uma vez que os custos de *gas* são um fator de bastante peso numa rede em produção.

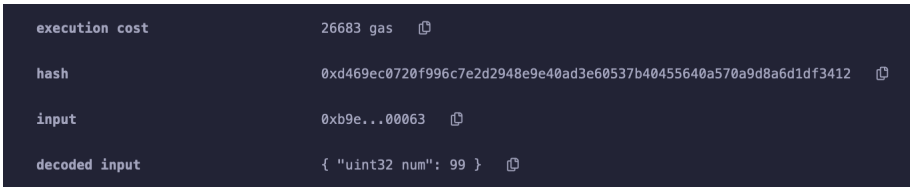


Figura 4.2: Custo de *gas* na escrita de uma variável *uint32* num *smart contract*

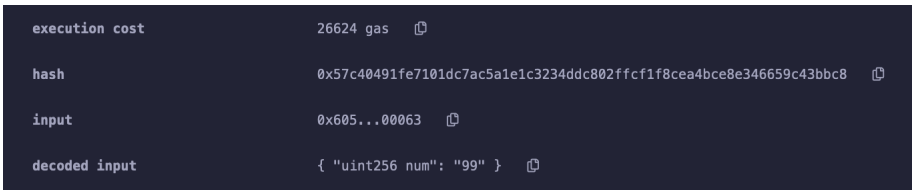


Figura 4.3: Custo de *gas* na escrita de uma variável *uint256* num *smart contract*

- **Entidade de Certificação**: armazena os dados necessários à identificação de uma entidade de certificação, cujo objetivo na sua interação com a rede é gerir os seus processos de certificação. Os dados armazenados foram analisados com base numa pequena pesquisa e cujo exemplo de resultado se encontra no **Anexo C**;
  - Face ao âmbito do projeto, e embora não exista nenhuma validação na lógica implementada, considera-se a entidade de certificação responsável pela criação do *smart contract*. Esta decisão arquitetural tem por base aproximar a lógica deste processo ao de um sistema em produção onde, com as devidas validações, seria a entidade de certificação responsável pela criação e gestão de estado do processo de certificação. Em termos de implementação, esta decisão arquitetural é visível, em ambas as plataformas, ao ser armazenada a chave pública do remetente das respetivas transações, na variável que armazena a chave pública da entidade de certificação no *ledger*.
  - Foram considerados os mesmos aspetos face ao NIF da entidade de certificação.
- **Lei**: armazena os dados necessários à identificação de uma lei que determinado serviço de certificação faz cumprir (p.e., Regulamento (UE) nº1151/2012 como referenciado no **Anexo C**), e cujo produto valida aquando de ser atestada a sua qualidade diferenciada. De forma à informação apresentada tentar seguir convenções existentes, foi realizada uma pequena pesquisa e foi considerada a convenção mencionada em [66];
  - Em relação às variáveis numéricas, foram considerados os mesmos aspetos face ao NIF da entidade de certificação.
- **Evidência**: armazena os dados necessários para a definição de uma evidência (p.e., submissão de uma carta de motivação) no processo de certificação, cuja representação é a de um passo que é necessário à entidade de produção de ver validado pela entidade de certificação de forma a progredir no processo de certificação (constituído por uma lista deste evidências);

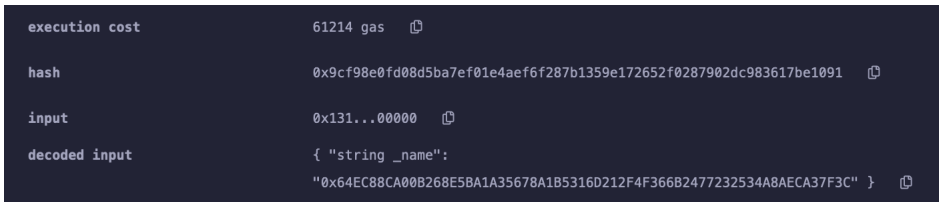


Figura 4.4: Custo de *gas* na escrita de uma variável *bytes32* num *smart contract*

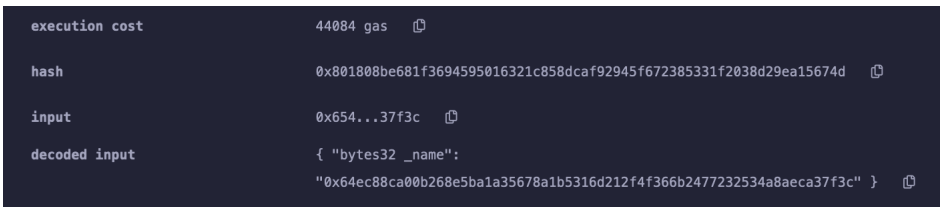


Figura 4.5: Custo de *gas* na escrita de uma variável *string* num *smart contract*

  - Em relação às variáveis numéricas, foram considerados os mesmos aspetos face ao NIF da entidade de certificação.
  - A evidência fornecida pela entidade de produção, com base na informação transmitida pela instituição acolhedora, consiste no conjunto de dados relativos a processos de produção, fertilização, entre outros processos. No entanto, estes devem ser registados num Caderno de Campo que visa já ser um requisito de documentação requerido pelas entidades de certificação. Deste modo, para simplificar o processo de implementação, considerou-se que apenas seriam consideradas evidências a submissão de ficheiros.
  - Face ao contexto de desenvolvimento descentralizado, considerou-se que os ficheiros mencionados seriam armazenados num sistema de armazenamento distribuído, e apenas a sua *hash* é armazenada no *ledger* da plataforma *blockchain*. Deste modo, é possível garantir a descentralização do sistema, bem como é utilizada a tecnologia *blockchain* para garantir a imutabilidade da *hash* que permite mapeamento com o ficheiro.
  - Como já mencionado, sendo o custo de *gas* uma preocupação no desenvolvimento de *smart contracts* em Ethereum, foi considerado o custo de criação de *strings* para o armazenamento da *hash* do ficheiro de evidências. O tamanho da *hash* gerada no armazenamento do ficheiro depende depende do algoritmo de *hashing* utilizado pelo sistema. Ao considerarmos a utilização de SHA-256, é possível utilizar uma variável *bytes32* e deste modo ocupar todos os bytes desta (i.e., otimizando o armazenamento). Caso sejam utilizados algoritmos de *hashing* que gerem *hashes* de tamanho superior, é necessário considerar o seu armazenamento num *array* de *bytes32* ou numa variável do tipo *string*. Existem diversas considerações para mitigar este problema, dependentes do âmbito do projeto, no entanto, foi considerado a utilização do algoritmo de *hashing* SHA-256 para facilitar a implementação.

- **Certificado**: estrutura de dados que representa o conjunto de dados de um processo de certificação, incorporando, além das suas variáveis de estado específicas, instâncias das estruturas de dados definidas anteriormente. Representam um processo de certificação face a um pedido de certificação de uma entidade de produção sob um serviço disponibilizado por uma entidade de certificação. Este processo, além de poder ser

cancelado, é considerado concluído aquando da validação de todas as evidências e gerado o certificado pela entidade de certificação, ou rejeitado caso contrário.

- Na *hash* que representa o ficheiro do certificado gerado, foram considerados os mesmos aspetos que para a *hash* do ficheiro de evidência.

No desenvolvimento de *smart contracts* para Ethereum, a estrutura de dados "Certificado"representa o conjunto de variáveis de estado do *smart contract* definido para o processo de certificação, sendo apenas adicionada posteriormente a lógica para o processamento do mesmo. No desenvolvimento do *transaction processor* para Hyperledger Sawtooth, a estrutura de dados "Certificado"é considerado o Protocol Buffer que define a entrada de estado referente a um processo de certificação.

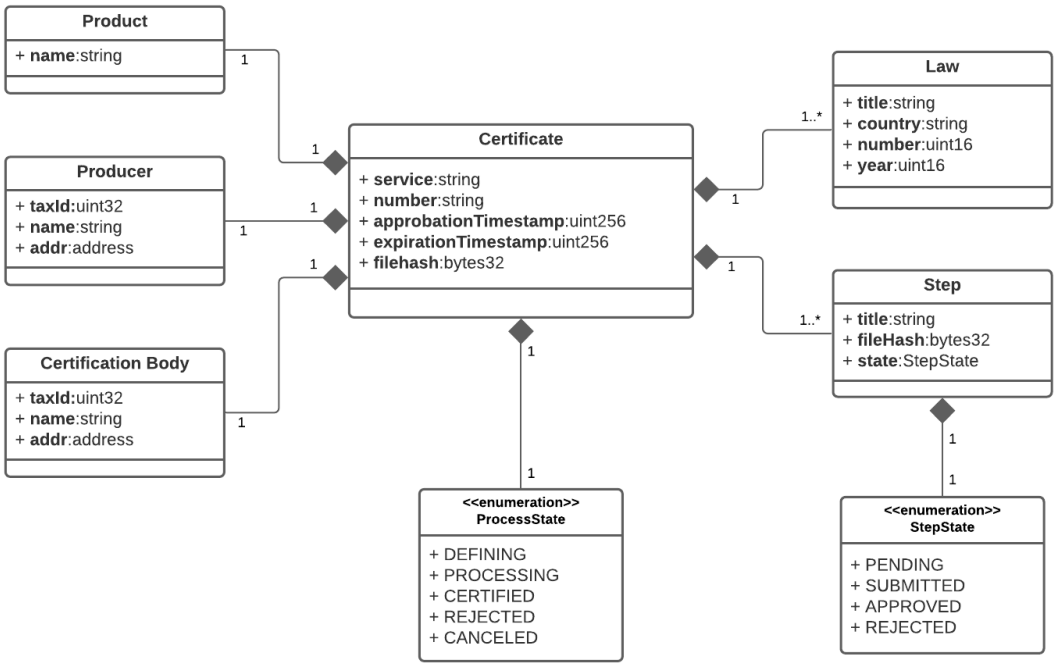


Figura 4.6: Modelo de Dados das Aplicações *Blockchain*

Por fim, os métodos que permitem a transição entre estados e consequente implementação da lógica do processo de certificação, são simples implementações de código que podem ser verificadas no *smart contract*. A única particularidade destes são as restrições que têm face a validações de parâmetros de entrada e restrições de acesso.

Máquina de Estados

Máquina de Estados é um padrão de design de *software* que permite a um objeto alterar o seu comportamento com base no seu estado interno. Existe um número finito de estados pré-determinados que o objeto pode assumir, comportando-se de forma distinta consoante o mesmo, e as suas transições de estado são finitas, pré-determinadas e podem ocorrer instantaneamente. Este conceito está relacionado com o conceito de Máquina de Estados Finita [67].

Relativamente ao processo de certificação de produtos agroalimentares, e com base nos seus requisitos, apenas é pretendido que determinadas ações estejam disponíveis com base

no atual progresso do mesmo. De igual forma, o processamento de evidências apenas deve permitir determinadas ações com base no seu progresso.

Com base neste fator, e uma vez que é possível definir o progresso como um momento exato (i.e., é possível definir exatamente quando uma evidência é submetida), foi implementada uma máquina de estados para representar o progresso do processo de certificação, e das evidências que o constituem, como é possível de verificar nas **Figuras 4.7** e **4.8**.

A máquina de estados finita, referente ao processo de certificação, possui o seguinte conjunto finito de estados, de acordo com a sua representação na **Figura 4.7**:

- **EM DEFINIÇÃO**: estado inicial que assinala a aprovação do pedido de uma entidade de produção face a um serviço de certificação. Este estado permite à entidade de certificação definir os parâmetros do respetivo contrato (p.e., evidências necessárias), até à mesma marcar esta fase como terminada, bloqueando qualquer interação de escrita do produtor (excepto o cancelamento do processo);

- **EM PROCESSAMENTO**: estado que permite à entidade de produção submeter as evidências requeridas pelo serviço de certificação requerido. De igual forma, permite à entidade de certificação a aprovação/rejeição das evidências já submetidas. Aquando da validação das evidências necessárias ao serviço de certificação, é possível do mesmo ser gerado;

- **CERTIFICADO**: estado final que marca a aprovação de um produto de uma entidade de produção, face ao serviço de certificação requerido;

- **CANCELADO**: estado final que marca o cancelamento do serviço de certificação requerido;

- **REJEITADO**: estado final que marca a rejeição do serviço de certificação para um produto de uma entidade de produção;

A máquina de estados finita, referente às evidências, possui o seguinte conjunto finito de estados, de acordo com a sua representação na **Figura 4.8**:

- **PENDENTE**: estado inicial que assinala que uma evidência se encontra pendente da sua submissão;

- **SUBMETIDA**: estado definido pela submissão de uma evidência por parte da entidade de produção, bloqueando qualquer interação desta entidade com a mesma e possibilitando a entidade de certificação de a validar/rejeitar;

- **APROVADA**: estado final que marca a aprovação de uma evidência submetida pela entidade de produção;

- **REJEITADA**: estado final que marca a rejeição de uma evidência submetida pela entidade de produção;

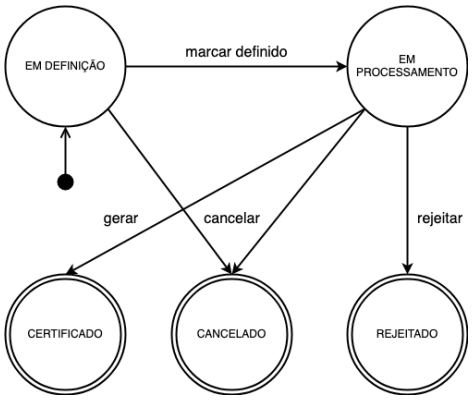


Figura 4.7: Máquina de estados - Processo de Certificação

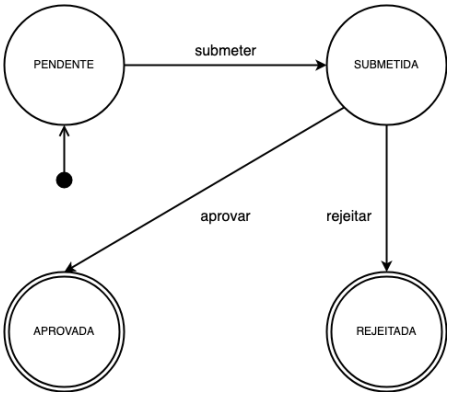


Figura 4.8: Máquina de estados - Evidências

Esta implementação foi considerada viável [68] para o desenvolvimento de *smart contracts* em Solidity, para a plataforma *blockchain* Ethereum. Face ao desenvolvimento do *transaction processor* em Hyperledger Sawtooth, uma vez que é possível implementar qualquer lógica utilizando a linguagem do respetivo SDK, e dependendo esta implementação de condições, verifica-se possível a mesma.

Ambas as máquinas de estados foram definidas nos respetivos códigos fonte (i.e., em Solidity para o desenvolvimento do *smart contract* para Ethereum, e em Protocol Buffers para a representação do estado em Hyperledger Sawtooth) enquanto enumerações. Desta forma é possível pré-definir o conjunto finito de estados, e posteriormente é possível representar o estado atual através de uma variável.

As transições de estado são realizadas através dos métodos dedicados a implementar a respetiva lógica, e cuja função final é a alteração do valor da variável que define o estado atual.

Restrições de Acesso

Os requisitos do caso de uso indicam explicitamente que determinados métodos devem ser explicitamente bloqueados a determinadas entidades, além do seu bloqueio face ao atual estado do processo de certificação ou das evidências do mesmo. Estas são as validações que constam dos requisitos de implementação do caso de uso, além das possíveis validações lógicas.

Por consequência é necessário identificar as respetivas entidades no contexto do *smart contract*, ou *transaction processor*. De acordo com os mecanismos da tecnologia *blockchain*, esta identificação é realizada com recurso à chave pública das respetivas entidades, que por sua vez é utilizada para assinar as transações enviadas para a rede.

No entanto, no contexto do caso de uso, as entidades de produção e de certificação poderiam ser entidades coletivas, obrigando à definição de uma metodologia para identificação dos seus intervenientes com os métodos do *smart contract*. Face ao esforço que poderia requerer tal implementação (p.e., adicionar uma camada superior que permitisse a autenticação de utilizadores e posterior mapeamento com as chaves públicas das entidades de certificação) para a identificação das mesmas ou para a sua restrição de acesso aos métodos do *smart contract*, foi simplificado o desenvolvimento desta metodologia assumindo que existe apenas uma chave por entidade e que, caso esta seja coletiva, os seus intervenientes utilizam a

mesma. Assim, no construtor do *smart contract*, e *transaction processor*, é armazenada a chave pública do remetente da transação numa variável que identifica a entidade de certificação, e a chave pública da entidade de certificação é enviada por parâmetro sendo armazenada na variável destinada a esta entidade.

A plataforma *blockchain* Ethereum é estruturada para executar uma rede pública *permissionless* e deste modo, a sua implementação privada não possui mecanismos de gestão de permissões. No entanto, é definido um padrão de design de *software* na documentação de Solidity [68] - Restrição de Acesso - que demonstra como pode ser aplicada a restrição de acessos num *smart contract* em Etheruem.

Esta abordagem foi utilizada no desenvolvimento do *smart contract* para o processo de certificação em Solidity e consiste na comparação da chave pública, que assina a transação que invoca o *smart contract*, com a chave pública da entidade desejada, e armazenada enquanto variável de estado do mesmo. Caso este processo seja validado, o respetivo método é invocado e a sua lógica processada, caso contrário a transação é revertida.

Por outro lado, a arquitetura da plataforma *blockchain* Hyperledger Sawtooth contempla a sua aplicação na implementação de redes públicas e privadas, *permissioned* e *permissionless*. A definição de permissões é implementada ao nível dos seus componentes *validator* e *transaction processor*. [69] No entanto, esta seria uma implementação interessante de abordar na definição de uma rede em produção privada e *permissioned*, mas que não foi implementada no nissertação de forma a manter as redes com o mesmo nível de permissões.

Embora o fator anterior seja um *trade-off* do desenvolvimento da dissertação, permitirá que o processo de *benchmarking* valide o desempenho de duas redes de iguais permissões. Deste modo, a restrição de acesso em Hyperledger Sawtooth aos métodos do *transaction processor* é feita com validação lógica no mesmo, e as chaves públicas das entidades são armazenadas no estado do *ledger* através de Protocol Buffers.

De notar que Solidity possui um tipo de variável, com mecanismos específicos, destinado ao armazenamento de endereços. Em Hyperledger Sawtooth as chaves são armazenadas em variáveis do tipo *string*.

4.4 Redes de Teste

Numa primeira fase de desenvolvimento desta dissertação, utilizaram-se máquinas virtuais, em ambiente local, para implementação de nodos das plataformas **blockchain**. No entanto, foi adotada a utilização de virtualização através da plataforma Docker (explicitamente o motor Docker e Docker Compose). Este implementação permite um menor consumo de recursos da máquina em causa, uma vez que os *containers* em Docker fazem uso de um único núcleo partilhado, e também um igual ou melhor desempenho das aplicações executadas nestes, comparativamente à sua execução em máquinas virtuais. [70]

No desenvolvimento da arquitetura da rede Ethereum, com recurso ao cliente Geth, foram considerados os tipos de nodos [71] que iriam constituir a mesma. Uma vez que se pretende realizar a análise de desempenho da rede, e que apenas os nodos que participam no mecanismo de consenso executam determinada lógica de validação de transações, apenas foram considerados nodos *full*.

Para conectar os diferentes nodos integrantes da rede, é possível utilizar mecanismos estáticos ou mecanismos dinâmicos. Os mecanismos estáticos consistem ou na utilização da *flag –nodiscover* na configuração do nodo e posterior adição de nodos manualmente

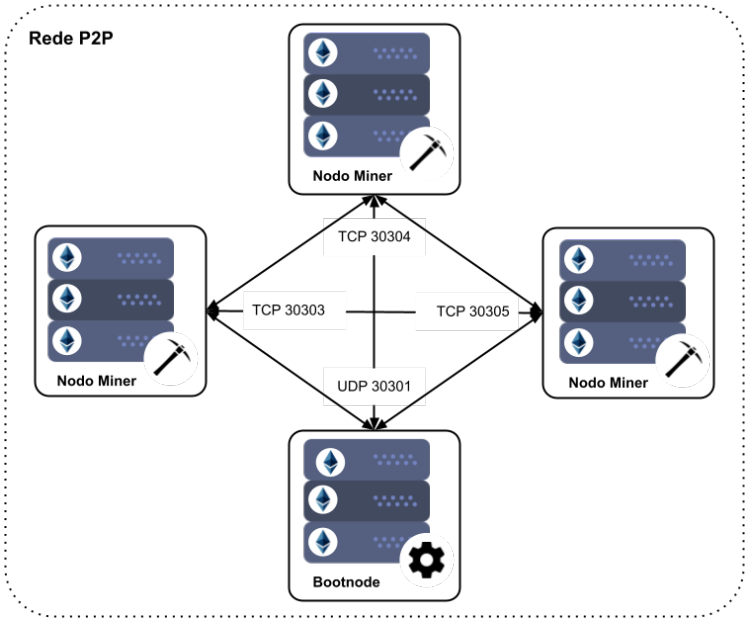


Figura 4.9: Esquema da rede implementada para a plataforma *blockchain* Ethereum

(i.e., através da consola do cliente Geth), ou na definição de um ficheiro 'static-nodes.json' que é definido no nodo e carregado na sua inicialização (adicionando os nodos que estiverem declarados no mesmo), ou, análogo ao processo anterior, na definição de um ficheiro 'trusted-nodes.json' que é definido no nodo e carregado na sua inicialização. Os mecanismos dinâmicos constistem na definição de *bootnodes*. [72]

Com vista na escalabilidade da rede, foi adicionado um *bootnode* - nodo que permite que a primeira vez que um nodo se conecta à rede, ao definir este *bootnode* na sua configuração, se conecte aos restantes nodos da rede - de forma a que os nodos da rede Ethereum desta dissertação se comuniquem a este nodo e, através do *discovery protocol*, estabelecem comunicação com os restantes nodos da rede. De realçar que, numa rede de produção este nodo é replicado com o objetivo de eliminar um possível SPOF.

Face ao mencionado, encontra-se na **Figura 4.9** a arquitetura da rede Ethereum cujo desempenho será analisado. Na implementação da mesma, devido a diversos problemas na conexão entre nodos, foram testadas as diversas formas de *peering* até que fosse possível a definição de um *bootnode* funcional.

O ficheiro de Docker Compose define cada nodo como um serviço independente que instala o *software* Geth. Posteriormente é feito o gerado o bloco *genesis* em cada nodo *full* copiando o mesmo ficheiro de configuração para o sistema de ficheiros do seu *container*. Posteriormente é definido um nodo que irá abrir o *web socket* (único mecanismo permitido para a integração de Hyperledger Caliper, visto que foi testado HTTP e não foi permitido) para a posterior integração da *framework* Hyperledger Caliper, e que integra a comunicação com InfluxDB (explicitado no seguinte sub-capítulo). À parte destas definições, os nodos iniciam abrindo um porto para o protocolo Ethereum (diferente em cada nodo, como recomendado em [72]) e iniciam o seu processo de *mining* de forma a validarem novas transações.

Como já mencionado no Estado de Arte, um nodo em Hyperledger Sawtooth é constituído pelo conjunto de componentes: *validator*, REST API, *transaction processors* (obrigatório, pelo menos, o *settings transaction processor*) e motor de consenso. Assim sendo, o ficheiro de Docker Compose define estes serviços de forma independente para o número de nodos a incluir na rede.

Pelo que é possível analisar, cada nodo terá que iniciar, no mínimo, 4 processos na respetiva máquina. Embora este fator não indique nada, uma vez que é perfeitamente aceitável e nada especifica acerca da carga destes, é possível questionar o possível impacto no consumo de recursos (nomeadamente de CPU) que poderá ter, comparativamente a um cliente Geth Ethereum que apenas inicia um processo.

Numa fase preliminar, onde ainda eram executados ambientes de desenvolvimento em máquinas virtuais, foi testada a capacidade de alteração dinâmica do mecanismo de consenso numa rede Hyperledger Sawtooth. Embora este fator não seja considerado no *benchmarking* da plataforma *blockchain* foi importante para perceber as capacidades e particularidades deste funcionamento. Assim, foi possível anotar as seguintes particularidades:

- Aquando de uma rede em funcionamento, que pretenda posteriormente alterar para o mecanismo de consenso PBFT, apenas é possível caso o bloco *genesis* inclua as chaves públicas de todos os nodos *validator* na rede inicial. [73] Embora esta particularidade esteja devidamente documentada na documentação da plataforma *blockchain*, foram testadas outras alternativas de forma a validar todas as hipótese de alteração dinâmica, tendo estas sido rejeitadas;

- A alteração dinâmica de qualquer mecanismo de consenso para o mecanismo de consenso PoET, e vice-versa, tendia a que existissem diversos *forks* na rede, demonstrando a vulnerabilidade da alteração dinâmica de mecanismo de consenso através de um mecanismo de consenso que permite a existência de *forks*;

- A alteração do mecanismo de consenso requeria a consequente inicialização de um novo motor de consenso, adequado ao respetivo mecanismo, e a opção de término do motor em execução. No entanto, esta troca demonstrou-se um pouco confusa no sentido em que, poderia ser um mecanismo que terminava internamente (uma vez que existe comunicação entre o *validator* e o motor de consenso) e em vez disso cabe ao utilizador administrar este processo, sendo o mesmo não muito claro;

- Foi também notado que a troca de alteração do mecanismo de consenso depende de uma transação para alteração das configurações, armazenadas em estado através do *settings transaction processor*. [73] Este mecanismo centraliza a administração da rede, embora se possam configurar.

De realçar que o processo de inicialização do *validator* responsável pela criação do bloco *genesis*, na rede definida com Docker Compose para os mecanismos de consenso PBFT e Raft, requer a partilha de um volume (ou qualquer outra metodologia que permita a mesma finalidade) de forma a ser possível ao mesmo aceder às chaves públicas dos restantes nodos. Como mencionado em [73], os mecanismos de consenso PBFT e Raft requerem que o bloco *genesis* contenha esta informação definida.

Por fim, a arquitetura de uma rede Hyperledger Sawtooth encontra-se definida na **Figura 4.10**.

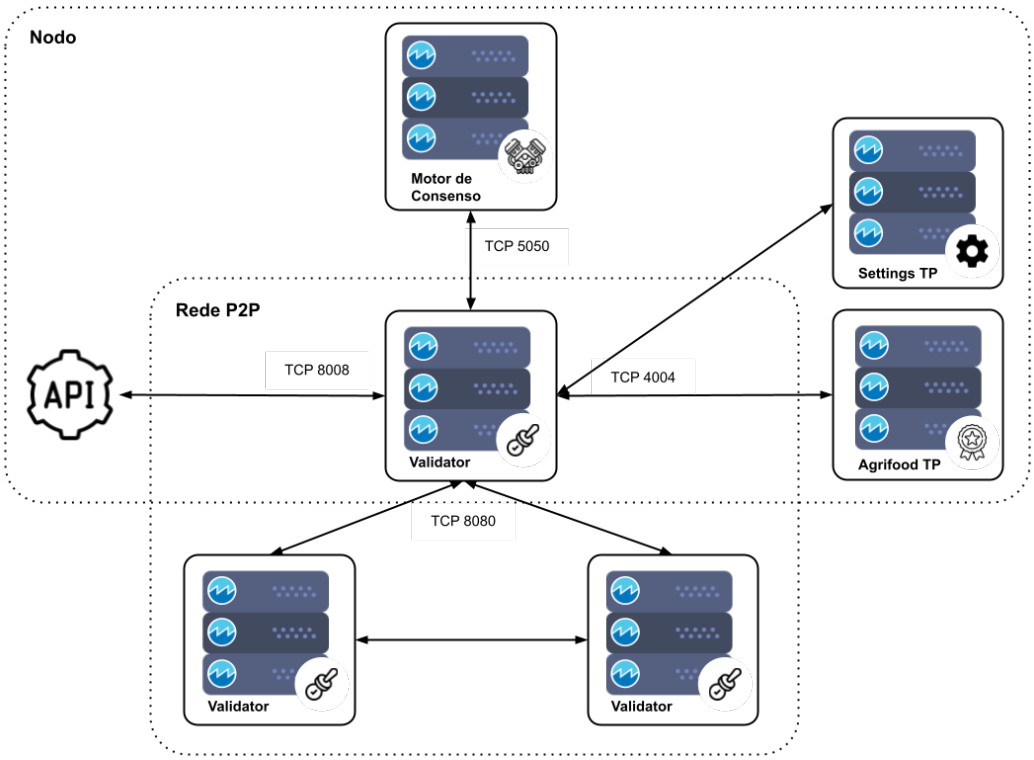


Figura 4.10: Esquema da rede implementada para a plataforma *blockchain* Hyperledger Sawtooth

Monitorização

InfluxDB é uma base de dados de séries temporais que permite armazenar dados sob a forma chave-valor, sendo a principal característica deste tipo de bases de dados o armazenamento do *timestamp* enquanto chave dos dados armazenados (i.e., tempo-valor). Embora as bases de dados relacionais permitam o armazenamento de dados temporais, o seu funcionamento não está otimizado para *workloads* de séries temporais. Deste modo, InfluxBD foi modelado para armazenamento de grandes volumes de dados de séries temporais, e a forma como armazena os seus dados permite uma otimização do seu processamento, permitindo-lhe realizar análise em tempo real dos mesmos. A sua implementação segue o modelo *push*, obrigando a que as aplicações que enviem os dados que pretendem armazenar.

No contexto do presente desenvolvimento, os nodos na plataforma *blockchain* Ethereum e os componentes *validator* e REST API na plataforma *blockchain* Hyperledger Sawtooth, enviam métricas para a base de dados InfluxDB através da comunicação com o serviço Docker responsável pela sua instância. Tais configurações são definidas através de *flags* no respetivo comando que inicia o processo do componente em causa.

Outra alternativa, para a plataforma *blockchain* Hyperledger Sawtooth, é a definição destas configurações nos ficheiros 'validator.toml' e 'rest_api.toml', que podem posteriormente ser copiados para o sistemas de ficheiros dos respetivos *containers* e iniciados aquando do processo. No entanto, esta abordagem foi testada e, ao serem carregados estes ficheiros, parâmetros referentes a outraas configuração que estavam a ser enviados por *flags*, não estavam a fazer *override* (i.e., não estavam a ser escritos) dos que se encontravam no ficheiro por defeito.

Outro problema de implementação residiu no facto do componente REST API estar a

apresentar erro na leitura das *flags* utilizadas para definir a comunicação com InfluxDB. Embora estivessem a ser seguidas as configurações para a execução do respetivo comando, definidas em [74], este não aceitou nenhuma das diversas tentativas de valor. Uma vez dado o problema mencionado anteriormente com a definição de um ficheiro de configuração para o componente, foi impossível a implementação da comunicação entre a REST API e a base de dados InfluxDB, estando apenas esta a ser executada com as métricas do *validator*.

Inicialmente foi testada a implementação da versão 2.0 de InfluxDB, no entanto, esta versão apresentou incompatibilidades com ambas as plataformas *blockchain* devido ao facto de estas não possuírem a capacidade de enviar o *token* de autenticação requerido por defeito nesta nova versão. Assim, é utilizada a anterior versão estável 1.8 de InfluxDB.

Grafana é uma aplicação *web open-source* para visualização e análise de dados. Permite realizar consultas, visualizar dados, estabelecer alertas, e explorar métricas através da sua conexão a uma fonte de dados. Esta tecnologia é integrada no desenvolvimento desta dissertação para a visualização dos dados monitorizados nas redes das plataformas *blockchain* e armazenadas na base de dados InfluxDB.

Assim, é necessário adicionar a fonte de dados InfluxDB ao Grafana através da sua interface gráfica *web*. Embora este processo possa ser automatizado, o mesmo não foi desenvolvido nesta dissertação face à sua relevância e planeamento. Posteriormente a esta configuração, é carregada uma *dashboard*, específica à plataforma *blockchain* em uso, que irá definir um conjunto de consultas que serão feitas à fonte de dados configurada anteriormente. Os resultados destas consultas serão apresentados nos devidos gráficos.

Integração da *Framework* de *Benchmarking*

Tal como descrito na Metodologia, a *framework* de *benchmarking* Hyperleger Caliper foi instalada localmente na versão 0.3.2. Após a instalação da *framework* foi necessário fazer *binding* desta para as plataformas *blockchain* Ethereum e Hyperledger Sawthooth. Como mencionado no Estado de Arte, a execução de *benchmarks* em Hyperledger Caliper requer, como argumentos, os ficheiros de configuração de *benchmark*, os ficheiros de configuração da rede, os módulos que definem os *workloads* e outros artefactos necessários à sua execução.

Os artefactos necessários à realização dos *benchmarks* desta dissertação são o *bytecode* do *smart contract* e o *transaction processor* desenvolvidos para definir o processo de certificação nas plataformas *blockchain* Ethereum, e Hyperledger Sawtooth respetivamente.

No caso particular da definição destes módulos para a execução de *benchmarks* com o SUT Ethereum, o facto de ser necessário gerar o bytecode forçou a que fosse desenvolvido um projeto externo que recebe como parâmtero de entrada um *smart contract* em Solidity e, com recurso ao compilador da linguagem, gera o código máquina de acordo com a estrutura pretendida por Hyperledger Caliper. Outra alternativa, embora não automatizada, seria a geração deste ficheiro manualmente com base na compilação do *smart contract* através de uma ferramenta como, por exemplo, Remix IDE.

As configurações de rede, como definem um conjunto pré-definido de serviços Docker que irão representar os nodos das redes *blockchain*, têm por base o número de nodos das mesmas ser uma variável independente desta análise. Seguindo a mesma lógica para os mecanismos de consenso, sendo estes uma variável independente desta análise, foram definidas o conjunto de configurações de rede que combinam estes dois aspetos com base nos valores que as variáveis irão assumir.

Além da definição da rede, Hyperledger Caliper requer um ficheiro de configuração cuja

estrutura se divide em dois principais objetos: definição do ambiente de teste (i.e., definido o nome da plataforma *blockchain* utilizada e é possível também utilizar comandos para serem executados aquando da iniciação e término do processo Caliper), semelhante às duas plataformas *blockchain*, definição do ponto do ponto de entrada da *framework* no SUT, estrutura especifica a cada plataforma *blockchain*.

A definição do ponto de entrada no SUT Ethereum requer que sejam definidos: o *endpoint* de entrada, cujo valor é referente ao *web socket* iniciado no nodo já mencionado; o endereço e chave privada (ou palavra-chave) da conta que irá assinar as transações a serem geradas pela *framework*; e os *smart contracts* que serão utilizados pela *framework*, sendo necessário mapear com o artefacto no diretório e estimar os custos de *gas* dos métodos que serão utilizados.

Os custos de *gas* foram gerados automaticamente com o desenvolvimento do *bytecode* do *smart contract*, no entanto, os valores não estavam a ser corretamente gerados e foi necessário definir um valor arbitrariamente grande para permitir a sua implementação e invocação dos seus métodos.

Um dos problemas de design identificados na *framework* foi o facto de não permitir a definição de, por exemplo, diferentes assinaturas para diferentes métodos a invocar no *smart contract*. Além de poder ser uma funcionalidade requerida por plataformas *blockchain* que permitem definir a invocação de métodos com a necessidade de múltiplas assinaturas, no caso de uso desta dissertação não permitiu a definição do envio de diferentes chaves para a inovcação de diferentes métodos no *smart contract*. Deste modo não foi possível aplicar as restrições de acesso no nível das aplicações *blockchain*, sendo que estavam a surgir sempre transações falhadas na execução de testes.

A definição do ponto de entrada no SUT Hyperledger Sawtooth requer que sejam definidos: o *batchBuilder*, que é um programa dedicado à execução de *benchmarks* desta plataforma *blockchain* e que visa a geração dos *batches* que serão enviados da *framework* de *benchmarking* ao SUT; os *endpoints* dos componentes *validator* e REST API da rede. De realçar que à definição do *batchBuilder* deve fazer corresponder-se o nome e versão definidas à *transaction family* do *transaction processor* a utilizar.

Como mencionado, o único problema identificado nesta configuração para a plataforma *blockchain* Hyperlegder Sawtooth reside no facto de ser necessário desenvolver um artefacto adicional, o gerador de *batches* a enviar para o SUT que funciona como um cliente de que irá enviar os *workloads* para o SUT.

Ainda enquanto argumento necessário à execução de *benchmarks* em Hyperledger Caliper, foram definidos os módulos de *workload* que permitiram a execução de *benchmarks* com recurso aos *smart contract* e *transaction processor* desenvolvidos para definir o processo de certificação nas plataformas *blockchain* Ethereum, e Hyperledger Sawtooth respetivamente. Os módulos definidos, são definidos como *callbacks* das rondas onde se inserem nos *benchmarks* a realizar, e sub-categorizam-se em operações de leitura e escrita. A lógica destes é adaptada à plataforma *blockchain* em causa, de forma aos seus argumentos e identificadores serem devidamente configurados.

Um dos problemas enfrentados na geração de testes face aos artefactos desenvolvidos, nomeadamente o *smart contract* para *benchmark* do SUT Ethereum, foi o facto da *framework* Hyperledger Caliper não suportar o construtor de Solidity. Isto é, o *smart contract* desenvolvido fazia uso de um construtor que recebia parâmetros do remetente da transação para inicialização de determinadas variáveis de estado. No entanto, a *framework* não suporta a configuração destes parâmetros na sua fase de inicialização no processo *manager* e tornou-

se impossível a criação do respetivo *smart contract* para efeitos de *benchmarking*. Como solução optou-se por abdicar do construtor, reformular o *smart contract* desenvolvido, e atribuir valores por defeito às variáveis para efeitos da análise desta dissertação.

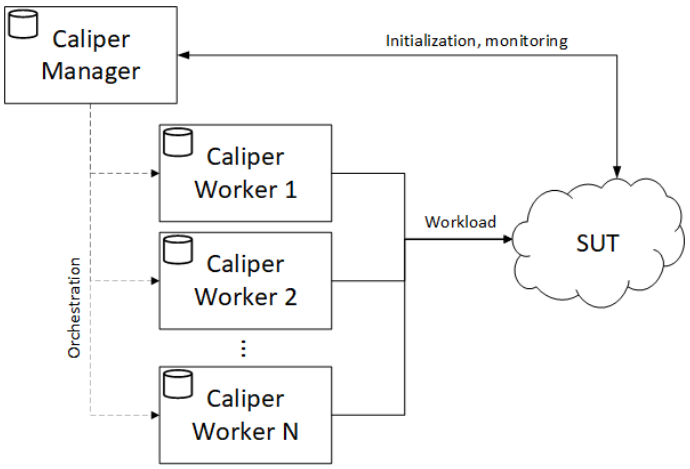
No caso particular da definição destes módulos para a execução de *benchmarks* com o SUT Hyperledger Sawtooth, foi necessário adaptar os argumentos que eram enviados através do método 'invokeSmartcontract()'. Este método invoca o *batchBuilder* para a criação dos respetivos *batches*, e foi necessário desenvolver o processamento dos argumentos enviados para a correta geração do *payload* das transações com base no método pretendido de invocar no *transaction processor*.

Por fim, o último argumento necessário à execução de *benchmarks* na *framework* Hyperledger Caliper são os ficheiros de *benchmark*. Estes ficheiros definem como serão executados os *benchmarks* e, deste modo, permitem implementar os *workloads* da análise desta dissertação.

Um dos aspetos possíveis de configurar no ficheiro de *benchmarks* são os monitores que permitem recolher estatísticas sobre o consumo de recursos durante o processo de *benchmarking*. Estes monitores podem ser definidos para recolher métricas de um processo a correr na máquina onde é executado o *benchmarking*, de um *container* (ou múltiplos) em Docker e de Prometheus (não foi explorado por limitação de tempo). No caso de uso desta dissertação foram testadas as alternativas de processo e de *containers* Docker, no entanto, a recolha de métricas face ao processo não se demonstravam capacidade para uma análise objetiva, ao contrário do que ocorre em Docker uma vez que as métricas são específicas a um serviço que possui uma determinada função no sistema. No entanto, mantiveram-se os dois tipos de recolha de métricas (possíveis de visualizar nos relatórios gerados) de forma a obter o máximo possível de dados para a respetiva análise.

Outro dos aspetos possíveis de configurar no ficheiro de *benchmarks* são os observadores, que permitem recolher, em tempo real, o estado das transações submetidas pela *framework* ao SUT. Este estado é obtido pelo processo *manager* sob os processos *workers* gerados para o respetivo *benchmark*. Observadores podem ser do tipo nulo, local ou com recurso novamente a Prometheus (não foi explorado por limitação de tempo). Deste modo, está a ser utilizado um observador local que recolhe as estatísticas das transações submetidas em intervalos de 1 segundo.

A injeção de carga distribuída pelos diversos nodos da rede permitiria aproximar o mais possível a distribuição da carga de forma heterogénea e promovendo, mais realisticamente, a interação entre os diversos nodos. Além disso, seria um aspeto para observar se o desempenho da plataforma melhora ou piora com a injeção de carga por diferentes nodos, em vez de apenas em um único nodo. [43] Visando este desenvolvimento seria interessante explorar a configuração dos processos *worker* que é possível de configurar no ficheiro de *benchmarks*. Esta configuração permite gerar um determinado número de processos *worker* que injetarão carga no SUT, como é possível de verificar pelo seu funcionamento na **Figura 4.11**. No entanto, além de não ter sido encontrada muita documentação na definição destes *workers* com os diferentes nodos do SUT, foi dada conta da seguinte informação na documentação da *framework* [75] "test.workers.type - Currently unused". Assim, concluise que será uma funcionalidade ainda em desenvolvimento e apenas foi explorada a injeção de carga através de um único processo *worker* local a único nodo na rede.



Fonte: `https://hyperledger.github.io/caliper/assets/img/arch_processes.png`

Figura 4.11: Funcionamento dos processos *manager* e *worker* em Hyperledger Caliper

O parâmetro de configuração de rondas no ficheiro de *benchmarks*, será mencionado para cada *workload* da rede uma vez que a sua configuração é o que permite definir o *workload* a executar.

Por fim, encontram-se modeladas nas **Figuras 4.12** e **4.13** as arquiteturas completas dos sistemas utilizados na análise desta dissertação para o *benchmarking* das plataformas *blockchain* Ethereum e Hyperledger Sawtooth, respetivamente.

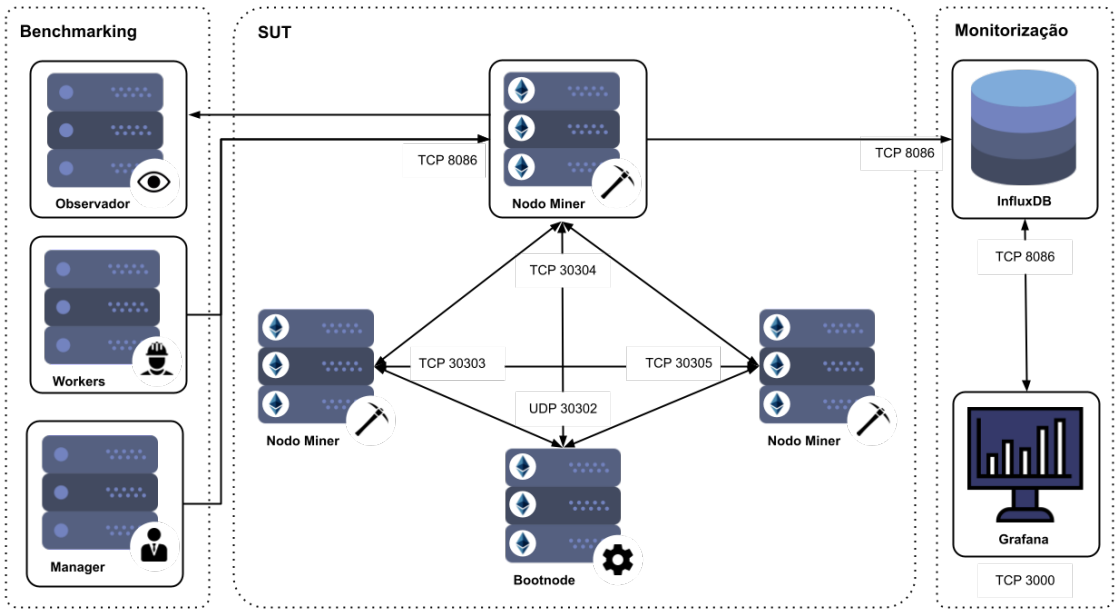


Figura 4.12: Esquema do ambiente de testes implementado para a plataforma *blockchain* Ethereum

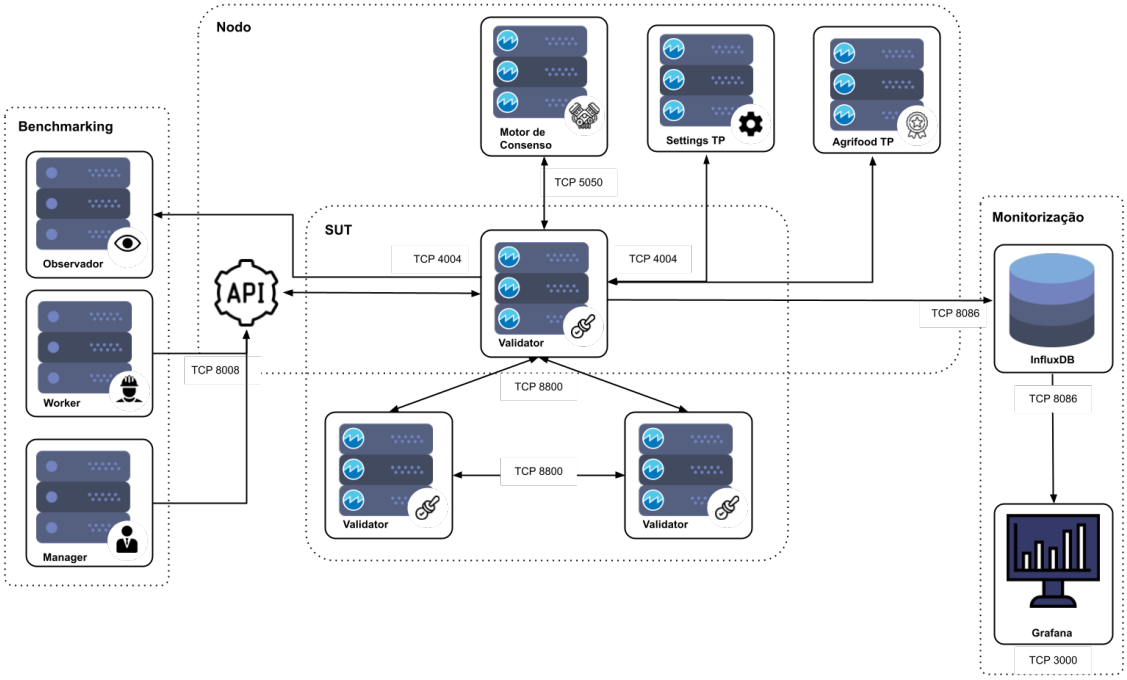


Figura 4.13: Esquema do ambiente de testes implementado para a plataforma *blockchain* Hyperledger Sawtooth

4.5 Workloads

Como mencionado no **Capítulo 4.3**, a lógica definida no modelo de dados foi utilizada na definição dos *workloads* executados nesta análise. Os métodos das aplicações *blockchain*, desenvolvidas e em execução na rede, são invocados consoante a configuração do *workload* e o objetivo da análise.

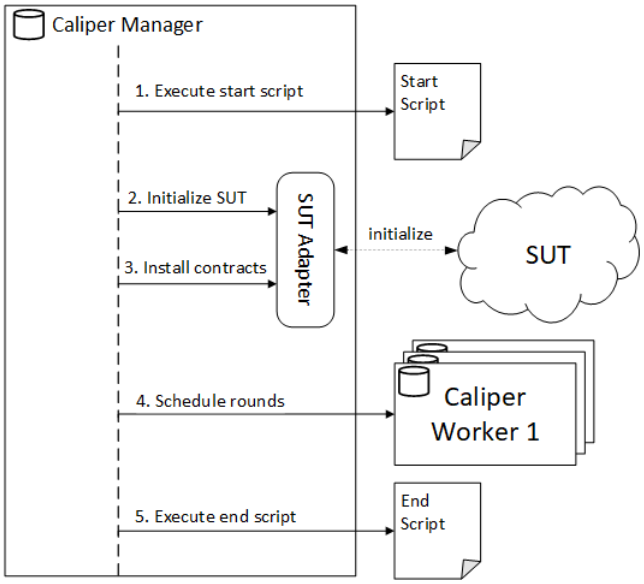
De notar que a lógica desenvolvida possui dependências transacionais (p.e., não é expectável que se tente validar uma evidência que ainda não foi adicionada) que foram ponderadas de forma a que uma transação não falhasse por validações lógicas das respetivas aplicações. As considerações consistiram em colocar as transações de adicionar leis e evidências antes de qualquer processamento das suas estruturas de dados. Embora a análise desta dissertação não escale a esse âmbito, caso seja utilizado processamento paralelo na rede ou na execução de testes, a dependência transacional deve ser ponderada.

Uma das particularidades identificadas na aplicação da *framework* Hyperledger Caliper à análise desta dissertação, é o seu processo de inicialização das aplicações *blockchain* (i.e., o processo de implementação dos *smart contracts* na rede Ethereum e da criação do estado na rede Hyperledger Sawtooth). A **Figura 4.14** representa este processo que, no entanto, apresentou as seguintes limitações:

- Não dispõe de suporte para a criação dos *smart contracts* na rede Ethereum com envio de argumentos ao método construtor, nem possibilita fazer *override* do respetivo método de criação.
  - **Alternativa**: foi alterado o código fonte do *smart contract* pre-definndo as variáveis que eram inicializadas no construtor.
- Não possui implementação do método de criação de *smart contracts* para a plata-

forma *blockchain* Hyperledger Sawtooth. Este método deveria estar implementado de forma a criar um novo estado no estado global da plataforma *blockchain*, permitindo a alteração das suas variáveis de estado e execução da sua lógica. Assim, foram feitas diversas execuções de teste, até detetar que o problema provinha da *framework* não criar o estado no seu processo de inicialização.

- **Alternativa**: uma das alternativas seria o desenvolvimento de um conector para a plataforma *blockchain* Hyperledger Sawtooth. No entanto, esta alteração poderia colocar em causa o planeamento da dissertação. Deste modo, a alternativa identificada foi a configuração de uma ronda de testes que apenas realiza uma transação para a criação de um novo processo de certificação. Esta é apenas incluída nos *workloads* à plataforma *blockchain* Hyperledger Sawtooth.



Fonte: `https://hyperledger.github.io/caliper/v0.3.2/architecture`

Figura 4.14: Execução do processo *manager* na *framework* Hyperledger Caliper

Os *workloads* definidos para a análise desta dissertação, independentemente do seu âmbito, iniciaram-se com uma ronda designada de "Inicialização". O objetivo desta foi o carregamento do *ledger* para verificação do seu bom funcionamento, e dos restantes componentes da rede. Esta verificação só é feita com base no relatório gerado no final da execução dos testes, e apenas indica se a rede iniciou o seu funcionamento corretamente para uma melhor análise dos restantes testes. Deste modo, qualquer *workload* foi configurado para, numa ronda inicial, submeter transações numa taxa fixa de 5 TPS até um limite de 50 transações. Estas transações invocam o método "setAsDefined", definido por apenas alterar o valor de uma das variáveis de estado da aplicação *blockchain*.

Para análise de escalabilidade da rede, foi configurado um *workload* com uma ronda para submeter transações numa taxa fixa de 40 TPS. As métricas, referidas neste capítulo na **Secção 4.1**, foram recolhidas até um limite de 400 transações. Estas transações invocam o método que permite gerar um novo certificado, e cujo objetivo foi selecionar o método que manipulava um maior número de variáveis de estado, tendendo a ser um método que demore mais tempo a ser processado (embora este tempo seja ínfimo).

Para análise dos métodos das aplicações *blockchain*, foi configurado um *workload* com uma

ronda para submeter transações a cada método - total de 8 rondas. Cada ronda recolhe métricas até um limite de 400 transações, a uma taxa fixa de 40 TPS.

Na execução dos testes anteriores foram sendo identificados determinados problemas (a mencionar no seguinte capítulo). Deste modo, foi definido um *workload* de carga cujo objetivo é validar as taxas de transferência transacional que as redes das plataformas *blockchain* suportam. O *workload* foi configurado para executar 5 rondas cujos valores das taxas de transferência transacional vão incrementando de 20 TPS até 200 TPS. As métricas foram recolhidas até um limite de transações 5x superior ao valor da taxa de transferência transacional, permitindo que sejam submetidas transações durante aproximadamente 5 segundos. As suas transações invocam o método que permite gerar um novo certificado, seguindo a mesma lógica da análise de escalabilidade.

4.6 Metodologia

As redes experimentais desta análise, Ethereum e Hyperledger Sawtooth, foram implementadas numa única máquina que possui as especificações apresentadas na **Tabela 4.1**. O principal problema desta abordagem, além da centralização da análise (i.e., os nodos das redes não serão distribuídos), é que o processo gerado para a execução da *framework*, e os processos que executam as redes, serem executados na mesma máquina. Além do elevado consumo de recursos, a preemptividade do sistema operativo pode afetar as métricas analisadas (p.e., aumentar a latência derivado de retirar a prioridade do processo de um nodo para garantir processamento na execução da *framework* e processamento de métricas) dado o elevado número de serviços que irão ser executados numa única máquina.

MacBook Pro (Retina, 13-inch, Early 2015)

OS	masOS Big Sur v11.4
CPU	Intel Core i5 2,7 GHz Dual-Core com 3 MB de cache L3 partilhada
RAM	8 GB 1867 MHz DDR3
GPU	Intel Iris Graphics 6100 1536 MB
SSD	121 GB de armazenamento flash com base em PCIe

Tabela 4.1: Especificações da máquina para execução do plano de testes

A rede da plataforma *blockchain* Ethereum foi implementada recorrendo ao cliente Geth v1.10.1, e a rede da plataforma *blockchain* Hyperledger Sawtooth foi implementada na v1.2 (Chime). A *framework* Hyperledger Caliper foi instalada no sistema operativo da máquina na v0.3.2, com recurso a Node.JS v10.24.1 e NPM v6.14.12. De realçar que existiram conflitos com versões do Node.JS na execução desta versão específica, tendo-se verificado a sua compatibilidade apenas com Node.JS 8 LTS, 9 LTS e 10 LTS.

Para a implementação destas redes foi utilizada a ferramenta Docker Compose v1.29.2, com recurso ao motor de Docker v20.10.7. Os seus ficheiros de configuração definem diferentes configurações de rede com base na plataforma *blockchain* configurada, no mecanismo de consenso que esta utiliza, e no número de nodos que o teste definir para estas.

Como mencionado na **Secção 4.4** deste capítulo, para a monitorização da rede foi integrada InfluxDB v1.8.6 e Grafana v7.4.1, para a rede Ethereum, e v4.4.3 para a rede Hyperledger Sawtooth. Esta diferença de versões da aplicação Grafana surge face aos requisitos das *dashboards* utilizadas. A *dashboard* para monitorizar a rede Hyperledger Sawtooth proveio do seu repositório Git, e para monitorizar a rede Ethereum foi utilizada a *dashboard* com identificador 13877 no Grafana Labs.

O plano de testes, que visa o desenvolvimento dos objetivos desta dissertação, incide em três principais análises com base em diferentes parâmetros a validar. Cada conjunto de testes será realizado sob redes das plataformas *blockchain* Ethereum, com mecanismo de consenso Clique e Ethash, e Hyperledger Sawtooth, com mecanismo de consenso PBFT, PoET e Raft.

A análise de escalabilidade da rede implementa redes configuradas com 4, 8, 12 e 20 nodos. A análise de métodos das aplicações *blockchain*, e a análise de carga, utilizam as configurações de rede, anteriormente implementadas, com 8 nodos. No entanto, o objetivo de ambas as análises é garantido pela execução dos devidos *workloads*, definidos na **Secção 4.5** deste capítulo.

Para a execução desta análise foi definido um *Shell script* que cria os diretórios necessários e executa ciclos para a execução dos diversos testes, inicializando as redes definidas na **Secção 4.4** deste capítulo e executando os *workloads* definidos na **Secção 4.5** deste capítulo. Os *logs* desta execução foram periodicamente analisados manualmente, para verificar o bom funcionamento da rede e da execução dos respetivos testes.

Fan *et al.* [1] mencionam que, antes de se proceder a uma análise de desempenho, devem ser consideradas as condições: bom funcionamento do *software* utilizado, correto funcionamento dos *drivers*, remoção de ficheiros temporários, término de processos desnecessários e verificação de atualizações. Deste modo, os serviços definidos nos ficheiros de configuração Docker Compose iniciarão uma imagem com configurações específicas e necessárias ao funcionamento do respetivo *container*, validando as condições mencionadas.

Por fim, foram gerados gráficos através dos relatórios HTML gerados automaticamente pela *framework* Hyperledger Caliper. Esta última fase recorre aos gráficos desenvolvidos para uma análise objetiva das métricas recolhidas, e encontra-se desenvolvida no **Capítulo 5**.

Capítulo 5

Análise de Resultados

Descrito o design da análise de desempenho desta dissertação, foi seguida a metodologia da mesma para a execução do plano de testes. No entanto, antes da análise dos resultados finais foram ocorrendo problemas preliminares à análise final mas cujas explicações permitem perceber determinados pontos dos resultados obtidos.

O presente capítulo encontra-se sub-dividido com base nas diferentes análises realizadas às redes das plataformas *blockchain* Ethereum e Hyperledger Sawtooth. Os testes que antecederam as mesmas permitiram gerar os artefactos necessários à sua objetividade na validação dos aspetos pretendidos.

5.1 Problemas Preliminares

Como já referido, e antes da execução dos testes finais cujos resultados são apresentados nas seguintes seções, foram sendo realizados testes para validação do bom funcionamento das redes e dos *workloads* implementados. Consequentemente, foram sendo detetados determinados problemas cuja importância em mencioná-los nesta dissertação se deve ao tempo consumido para a sua deteção e consequente mitigação, ou pela sua permanência nos resultados finais.

5.1.1 Fila de Espera Esgotada

Inicialmente as análises previam a definição de *workloads* com maior taxa transacional, de forma a submeter um maior volume de transações (no mesmo espaço de tempo) ao SUT. No entanto, foram identificados problemas na execução de *workloads* cujas taxas de transferência transacionais eram definidas, por exemplo, a 500 TPS ou 1000 TPS.

O primeiro volume de transações - ronda de Inicialização - era submetido e estas eram validadas e inseridas no *ledger*. A partir de determinado momento de sobrecarga, começou a ser repetidamente retornada a mensagem de erro: "*The validator cannot currently accept more batches, due to a full queue*".

Um dos componentes da arquitetura de um *validator*, em Hyperledger Sawtooth, é o *journal*. Este é constituído por um grupo de sub-componentes cujo objetivo é gerirem *batches* e propostas de novos blocos. [76] Um dos seus sub-componentes é o BlockPublisher cuja responsabilidade é receber *batches* enviados pelos clientes, ou outros nodos, e adicioná-los à sua fila de espera. *Batches* inseridos na fila de espera são posteriormente retirados desta e processados para a criação de novos blocos.

Deste modo, a mensagem de erro retornada deriva da fila de espera no BlockPublisher ficar obstruída face ao elevado volume de transações submetidas num período de tempo menor do que aquele em que esta é capaz de processar as transações que detém em espera. Este problema foi também identificada por Corso [77] que menciona que idealmente a taxa de transferência transacional deveria estabilizar na capacidade máxima da rede, não sendo afetada por transações que não serão possíveis de validar. No entanto, conclui que o que realmente ocorre é uma quebra no desempenho após a obstrução da referida fila de espera.

5.1.2 Tempo Limite Esgotado

Este problema emergiu, tal como o problema mencionado anteriormente, da intenção de definir *workloads* que executassem maiores taxas de transferência transacional, resultando num maior volume de transações submetidas ao SUT. Ao verificar os *logs* gerados pela execução do mesmo, iam sendo apresentadas mensagens de informação do observador que referia que teriam sido submetidas x transações e ainda se encontravam x transações inacabadas. Ao ser esgotado o tempo limite, e após esta mensagem surgir de acordo com o tempo definido de intervalo para o observador, era apresentada uma mensagem com a mensagem "*getBatchEventResponse err: Error: Timeout, batchID* (...)".

Este problema derivava do tempo de espera para validação de um bloco ser esgotado. O módulo do núcleo da *framework* Hyperledger Caliper possui um método - *getBatchEventResponse* - no código fonte do conector para Hyperledger Sawtooth que obtém a mensagem de confirmação da inserção do bloco no *ledger*. O que sucede aquando desta mensagem de erro é que o *timeout* definido no módulo de *callback* do método invocado na aplicação *blockchain* é esgotado. Ao ser esgotado, a transação é marcada como falhada no relatório gerado automaticamente pela *framework*, no entanto, o respetivo estado é posteriormente acessível aquando da validação e inserção do bloco no *ledger*.

Assim, de forma a mitigar este problema, embora com outras consequências, pode ser aumentado o valor do *timeout* definido no módulo implementado para *callback* do método da aplicação *blockchain* a invocar.

5.1.3 Geração de DAG

A necessidade de gerar um DAG [78] expressou-se na implementação das redes para a plataforma *blockchain* Ethereum com mecanismo de consenso Ethash. Este, recorre a um DAG na execução do seu mecanismo de consenso PoW que é gerado a cada 30000 blocos - época. Ao ser necessário gerar este DAG, aquando da implementação das redes a analisar com o mecanismo de consenso Ethash, era imposto um longo período de tempo para gerar o primeiro bloco da nova época.

Este é um processo utilizado pelo cliente Geth e que é automaticamente iniciado ao ser utilizada a *flag* --mine"no comando inicial de um nodo. O cliente Geth mantém dois DAGs consecutivos de forma a suavizar as transições entre épocas, apesar desta abordagem implicar um custo computacional elevado inicialmente.

Os nodos das redes configuradas para esta análise utilizam o cliente Geth com recurso à referida *flag* para iniciarem o processo de *mining* aquando da sua implementação. No entanto, este processo requer um elevado custo temporal e de recursos computacionais que emergiu como problema na implementação de redes superiores a 8 nodos. Foi possível implementar uma rede Ethereum com mecanismo de consenso Ethash configurada com 4 nodos. No entanto, a partir de 8 nodos a mesma iniciava o processo de inicialização do DAG e utilizava a maioria dos recursos da máquina utilizada para o desenvolvimento desta análise.

Considerando a versão 2.0 de Ethereum que irá substituir o mecanismo de consenso PoW pelo mecanismo de consenso PoS, e face às dificuldades presenciadas na máquina a executar esta análise, retirou-se o mecanismo de consenso Ethash da amostra de redes a analisar.

5.1.4 Validações Lógicas

Na execução do conjunto de testes realizados inicialmente, foram detetadas transações falhadas e que não eram posteriormente inseridas no estado do *ledger*. Estes resultados eram anormais no sentido em que não derivavam de problemas no tempo de validação de blocos.

Posteriormente, analisados os *logs* dos respetivos processos, foi identificado que as transações não eram validadas na lógica dos seus métodos e eram descartadas pela camada de aplicação e posteriormente não inseridas no *ledger*. Neste sentido, foi necessário retirar maioria das validações lógicas que eram definidas nas aplicações *blockchain*, uma vez que estas geriam o atual estado do processo de certificação, ou de alguma evidência. Sendo este processo baseado na transição de estados, ao ser validada uma transação que alterasse o mesmo, inviabilizava a validação das restantes.

Deste modo, foi necessário adaptar o código fonte das aplicações *blockchain* para a execução dos testes. Por consequência, esta alteração retira lógica aos métodos das aplicações que pode causar diferentes valores de desempenho numa rede em produção. Este é um *trade-off* necessário uma vez que a *framework* Hyperledger Caliper também não contempla a criação de diversos *smart contracts* nem a adaptação das rondas a esta distribuição de carga por aplicações *blockchain*.

5.2 Escalabilidade da Rede

A presente analise foca-se na camada de rede de forma a verificar na latência e taxa de transferência transacional o impacto do aumento do número de nodos na rede do SUT. Desta forma, e como já mencionado, variou-se o número de nodos na configuração de cada rede e executou-se o *workload* descrito no **Capítulo 4**, **Secção 4.5**.

Inicialmente é possível observar resultados incomuns do SUT com mecanismo de consenso Raft. Este fenómeno ocorreu nas suas configurações de rede para qualquer número de nodos e deve-se ao tempo de espera para validação de um bloco ser esgotado (problema descrito neste capítulo na **Secção 5.1**. Foi verificado que o SUT executava e validava a

ronda de criação de estado mas a partir da primeira transação da ronda de carga inicial da rede gerava a mensagem de erro do problema associado. O teste terminava sem que fossem validadas transações da ronda de análise que define o *workload*. Deste modo, é possível concluir que a latência da rede nestes casos foi superior ao tempo limite definido, não tendo sido validado blocos no período em que a *framework* aguardava a inserção destes no *ledger*. Estes ficaram em *backlog* até serem posteriormente validados.

Face aos gráficos das **Figuras 5.1** e **5.2**, é possível identificar que as redes da plataforma Hyperledger Sawtooth apresentam um contínuo aumento dos valores de latência com o aumento do número de nodos na rede. Este aumento de latência deriva do facto da rede integrar um maior número de nodos, necessitando de um maior tempo disponível para a sua propagação e validação. Por outro lado, a taxa de transferência transacional diminui com o aumento do número de nodos uma vez que reflete os valores de latência discutidos anteriormente. Esta métrica contempla o tempo total decorrido nas respetivas transações e o número total de transações inseridas no *ledger*. Não sendo inseridas transações no *ledger*, esta taxa atinge o valor 0.

Atingindo 0% de taxa de sucesso, é possível verificar que o SUT Hyperledger Sawtooth não escalou além de 12 nodos, derivado dos resultados das métricas referidas. Apesar de não ser possível identificar o número de nodos a partir dos quais a rede deixa de conseguir validar novos blocos, é possível verificar que este fenómeno se verificou na configuração de rede com 20 nodos. Assim conclui-se que a escalabilidade causa obstrução no desempenho de uma rede Hyperledger Sawtooth.

Face aos fatores mencionados anteriormente, é necessário mencionar que os recursos disponíveis para o desenvolvimento desta análise foram limitados à maquina referida no **Capítulo 4**, **Secção 4.6**. Deste modo, para solidificar as conclusões desta análise de escalabilidade, seria necessário realizar novamente a mesma aumentando os recursos de *hardware* dos nodos da rede.

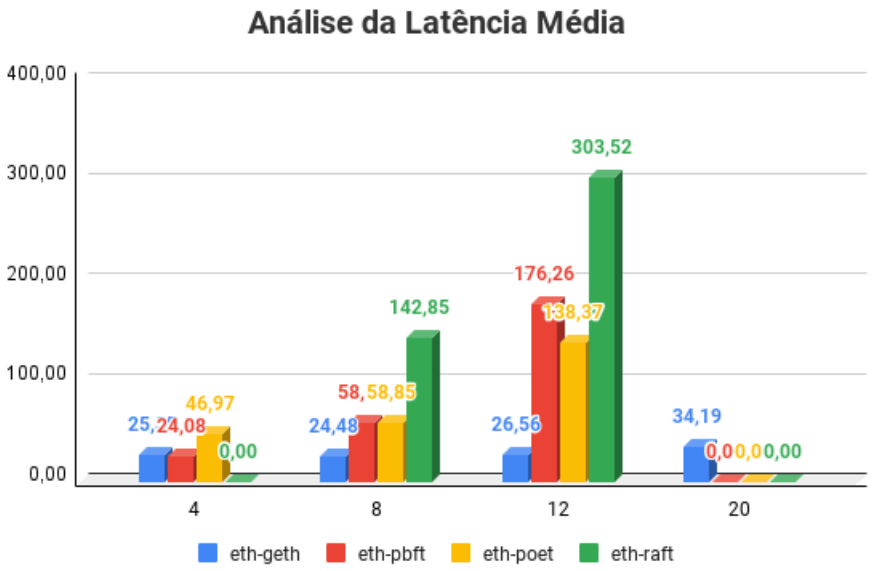


Figura 5.1: Análise da latência média em testes à escalabilidade da rede

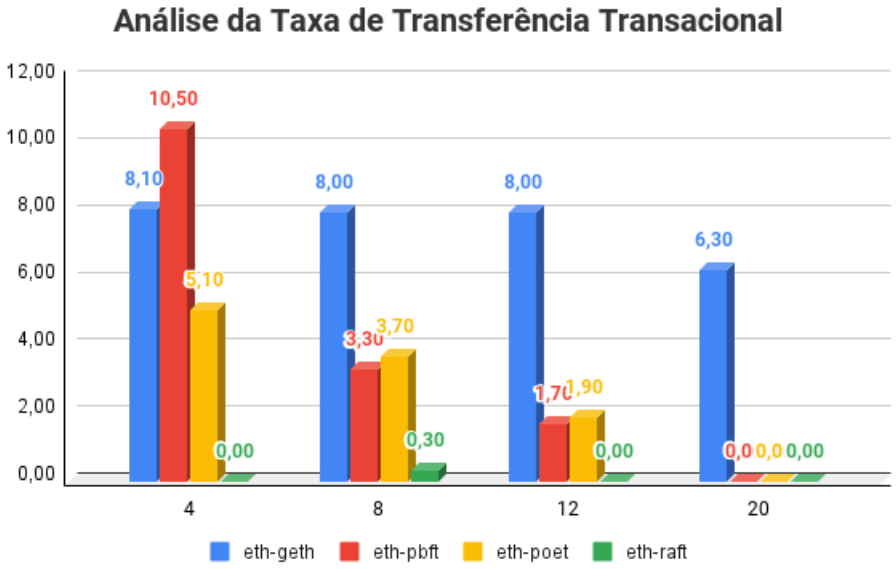


Figura 5.2: Análise da taxa de transferência transacional em testes à escalabilidade da rede

5.3 Carga

Derivado do problema de fila de espera identificado neste capítulo na **Secção 5.1**, esta análise surge com o intuito de identificar o intervalo de taxas de transferência transacional que origina o mesmo nas redes Hyperledger. Desta forma, torna-se possível perceber a capacidade de carga que a rede suporta.

Relativamente aos valores obtidos ao SUT com mecanismo de consenso Raft, este incorreu novamente no processo mencionado na análise de escalabilidade. No entanto, desta vez foi analisado o consumo médio de memória do *validator* que comunica com a *framework* Hyperledger Caliper. O seu consumo encontra-se representado na **Figura 5.3** para todos os SUT Hyperledger Sawtooth.

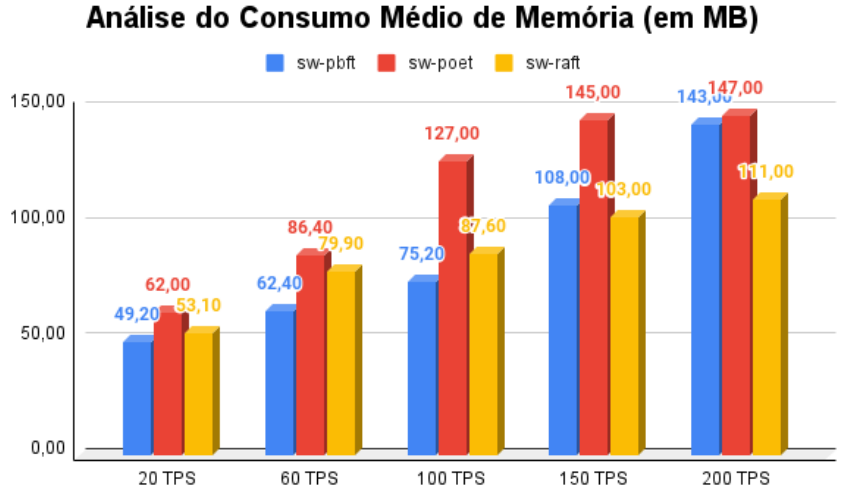


Figura 5.3: Análise do consumo médio de memória do *validator* em testes de carga à rede

Como é possível verificar, apesar do *validator* no SUT com mecanismo de consenso Raft não apresentar transações bem sucedidas, com o aumento da taxa de transferência transacional em cada ronda de teste, este continua a consumir maior quantidade de memória. Este fenómeno ocorre uma vez que, apesar da sua fila de espera entupir e não permitir a inserção de novas transações, o *validator* continua a receber novas transações e a necessitar de processamento para retornar as devidas mensagens de erro.

O gráfico da **Figura 5.4** permite identificar o contínuo aumento do valor da latência média face ao aumento da taxa de transferência transacional de entrada no teste. Por outro lado, o gráfico da **Figura 5.5** demonstra que a taxa de transferência transacional da rede, nomeadamente para o SUT Ethereum e Hyperledger Sawtooth com PBFT, se mantém praticamente constante a partir da segunda ronda. Deste modo, é possível concluir que, embora o tempo de validação aumente, o *validator* tem capacidade de validar novos blocos às taxas referidas, permitindo-lhe não ser necessário descartar transações.

Esta afirmação já não ocorre para o SUT com mecanismo de consenso PoET, uma vez que é possível afirmar, por observação do gráfico, que o seu desempenho ficou obstruído no decorrer da ronda com taxa de transferência transacional de 150 TPS. A taxa de sucesso desta ronde foi de 23,94% uma vez que a partir de um determinado ponto o *validator* começou a retornar erro de fila de espera cheia. Apesar dos seus valores de 7,50 TPS e 7,70 TPS em rondas anteriores, obteve um valor de 0,9 TPS de taxa na ronda de 60 TPS que poderá ter contribuído para que a fila de espera fosse acumulando transações durante as seguintes rondas embora só impactasse posteriormente na ronda de 150 TPS.

O SUT com mecanismo de consenso PBFT apresenta uma maior latência média comparativamente à mesma rede com o mecanismo de consenso PoET, e consequentemente, considerando as rondas de 20 TPS e 100 TPS, apresenta uma maior taxa de transferência transacional. A origem deste fenómeno poderá residir no funcionamento dos respetivos mecanismos de consenso. Em PoET a validação de um novo bloco é atribuída ao *validator* que esgotar um período de tempo aleatório mais rapidamente. Em PBFT o processo de validação recorre a um líder e nodos *backup* que trocam informação entre si nas fases de preparação e *commit*. Assim, o mecanismo de consenso PBFT apresenta uma maior sobrecarga de comunicação entre nodos que poderá ser justificação para os valores observados.

O SUT Ethereum, clinte Geth, com mecanismo de consenso PoA apresenta um desempenho superior aos restantes nas métricas analisadas. Este desempenho, comparativamente ao SUT com mecanismo de consenso PoET, poderá ser justificado pelo facto de este ser um mecanismo de consenso determinístico enquanto que PoET é um probabilístico e coloca os seus nodos numa espera imprevisível. Por outro lado, e comparativamente ao SUT com mecanismo de consenso PBFT, o seu desempenho poderá ser justificado pelo facto de não recorrer a uma sobrecarga na comunicação entre nodos. Enquanto que PoA determina um grupo de nodos com os mesmos privilégios e processa de igual forma a sua resposta, em PBFT existe um líder que processa as informações devolvidas em troca de informação com nodos *backup*. No entanto, esta análise tende a ser superficial considerando a maturidade e diferente arquitetura das plataformas.

5.4 Métodos das Aplicações Blockchain

A presente análise, face à camada de aplicação, pretende analisar na latência e taxa de transferência transacional dos SUT o impacto dos diferentes métodos da estrutura de dados descrita no **Capítulo 4**, **Secção 4.3**. Em análise à camada de aplicação, são os tempos relativos à validação de transações que nos permitem identificar o desempenho dos respeti-

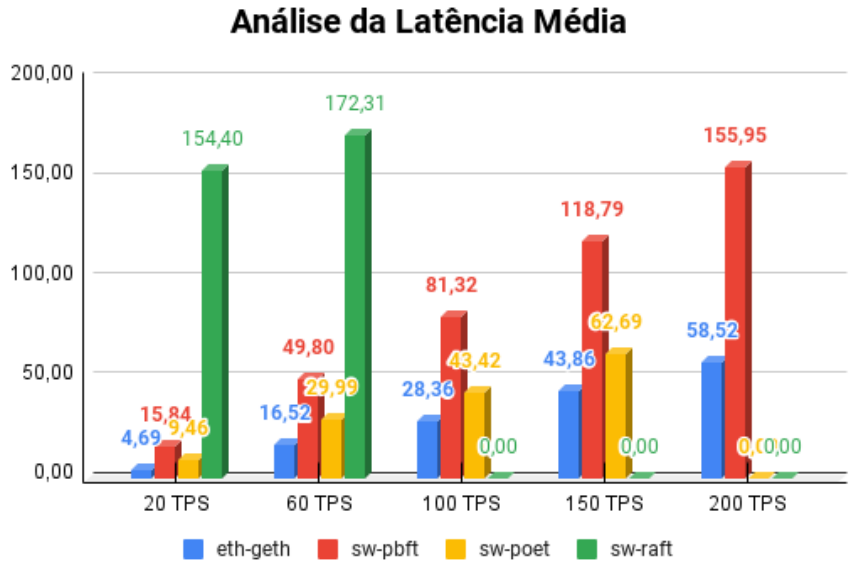


Figura 5.4: Análise da latência média em testes de carga ao SUT

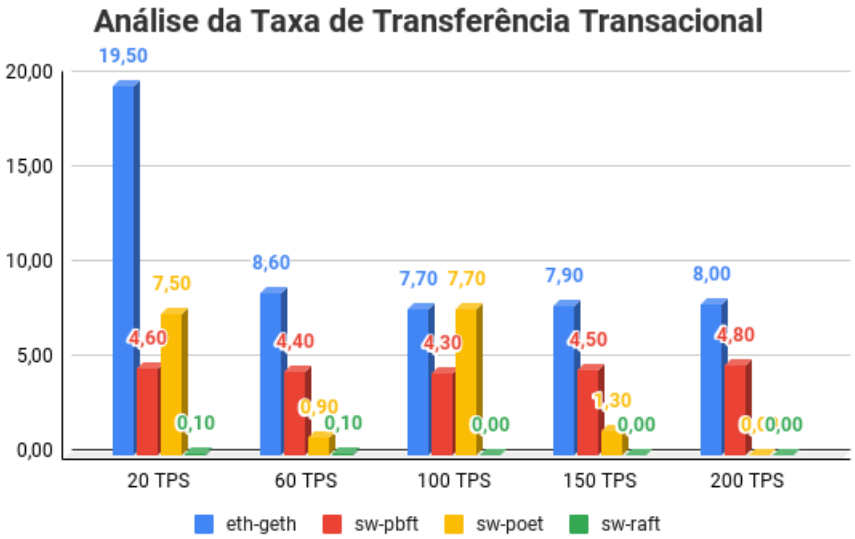


Figura 5.5: Análise da taxa de transferência transacional em testes de carga ao SUT

vos métodos. Assim, será possível concluir o quão intensiva é computação da lógica destes e o respetivo impacto no desempenho da rede.

De realçar que a presente análise não irá considerar os SUT com mecanismo de consenso PoET e Raft, sendo que estes foram novamente sujeitos às consequências do problema de fila de espera (descrito no presente capítulo na **Secção 5.1**) nos seus resultados.

Em análise dos gráficos das **Figuras 5.6** e **5.7**, é possível observar que os resultados dos SUT tendem a não sofrer variações notórias além das observadas nos resultados dos métodos "addLaw"e "addStep"para o SUT Ethereum. Os referidos métodos são os únicos cuja execução requer a escrita de um novo elemento num *array*. Este processo tende a exigir um maior poder computacional comparativamente à lógica dos restantes. Deste modo, os resultados obtidos através do SUT Ethereum apresentam custos bastante superiores face aos restantes métodos. Este custo apresenta um impacto notável no desempenho da rede da plataforma *blockchain* Ethereum com mecanismo de consenso PoA. De notar que os resultados obtidos para o SUT com mecanismo de consenso PBFT apesar de apresentarem ligeiras variações, não permitem concluir a sua origem. Todavia as linguagens de programação utilizadas para o desenvolvimento das respetivas aplicações *blockchain* são diferentes, como já referido nesta dissertação.

Foram igualmente analisados os métodos para leitura de dados do estado da *blockchain*, no entanto, estes expectavelmente obtiveram taxas de sucesso de 100% com latências médias entre 0,2 e 0,8 nos SUT Hyperledger e entre 0,01 e 0,06 no SUT Ethereum. Estes valores derivam do facto do nodo ao qual nos conectamos devolver o estado atual que possui do *ledger*, não sendo necessário propagar transações na rede nem a execução do mecanismo de consenso para validação.

Exceptuando a análise já realizada acerca dos resultados obtidos nos dois primeiros métodos, os resultados demonstram que o desempenho dos vários métodos é pelo menos constante (variações ínfimas) entre estes no seu desempenho. Deste modo é possível concluir que, caso a rede execute apenas esta lógica, a execução dos diferentes métodos não terão impacto entre si no desempenho da mesma.

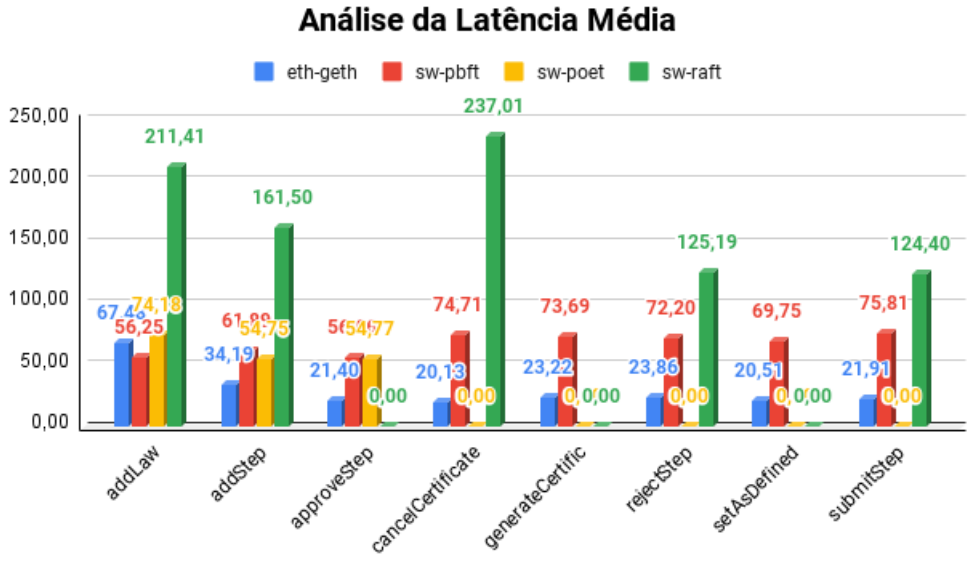


Figura 5.6: Análise da latência média em testes aos métodos da estrutura de dados

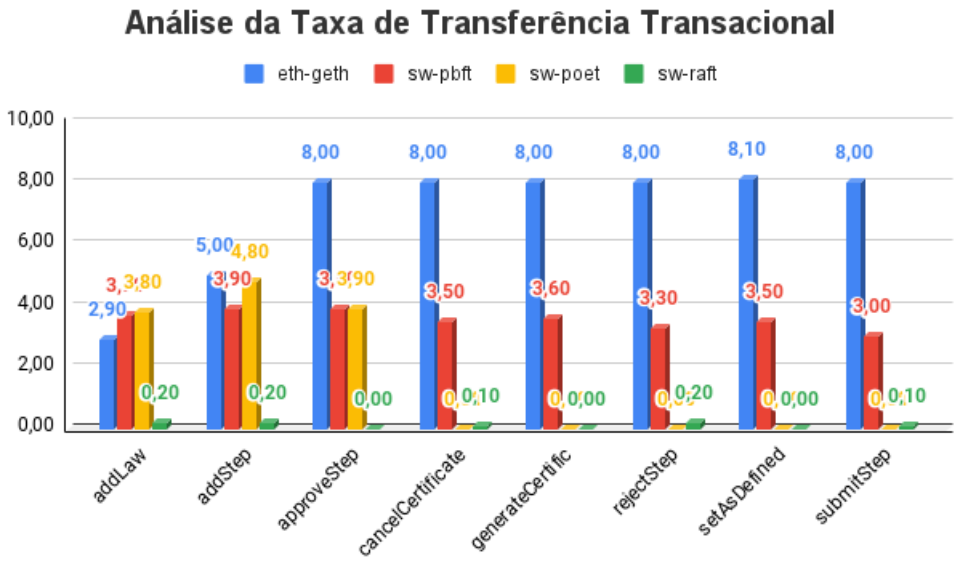


Figura 5.7: Análise da taxa de transferência transacional em testes aos métodos da estrutura de dados

This page is intentionally left blank.

Capítulo 6

Conclusão

Dos diversos desenvolvimentos emergentes, e analisados nesta dissertação, esta dissertação marcou os primeiros desenvolvimentos na análise do desempenho de uma plataforma *blockchain* aplicada a um caso de uso real. Apesar das análises referidas no Estado de Arte recorrerem a *benchmarks* pré-definidos de forma a seguir um determinado padrão na indústria, é necessário validar a aplicabilidade desta análise em casos de uso para produção. Consequentemente, foi possível identificar lacunas nas ferramentas que não possibilitam determinada execução (p.e., o problema com validações lógicas mencionado no **Capítulo 5** na **Secção 5.1**) que se torna fundamental na análise de desempenho destes sistemas.

De acordo com os resultados obtidos é necessário mencionar as dificuldades em desenvolver resultados concisos face às limitações de recursos de *hardware* neste desenvolvimento. é também necessário mencionar as dificuldade em desenvolver análises significativas nos sistemas distribuídos e complexos da tecnologia *blockchain*. Ainda assim esta dissertação pretende ser uma base para considerações em futuras análises de desempenho aplicadas a casos de uso que requeiram o desenvolvimento de aplicações *blockchain* dedicadas.

Em suma, o desenvolvimento desta dissertação permitiu a implementação de um ecossistema que aplica o processo de certificação de produtos agroalimentares na execução da rede de uma plataforma *blockchain*. Este é constituído pelos diversos sistemas isolados que foram configurados e validados: sistema representativo da rede da plataforma *blockchain*, sistema de monitorização e sistema de análise de desempenho da rede. Assim, esta dissertação abrange o desenvolvimento dos diversos sistemas, monitorização, execução de análise de desempenho e a análise de plataformas *blockchain*.

6.1 Descobertas

Este sub-capítulo apresenta uma visão geral final acerca dos resultados da análise de desempenho das plataformas *blockchain*, em resposta às perguntas de investigação definidas no **Capítulo 1.3**.

Quais os aspetos que obstruem o desempenho de uma plataformas *blockchain*?

A análise de resultados, realizada no **Capítulo 5**, permitiu observar que as redes da

plataforma *blockchain* Hyperlegder Sawtooth configuradas com 20 nodos não executaram a validação de transações no tempo limite definido. Na mesma análise já viria a diminuir a taxa de transferência transacional e a aumentar os valores de latência na rede, ao longo dos sucessivos testes realizados. Sendo que as rede configurada com 12 nodos ainda permitiu observar, à exceção da rede com mecanismo de consenso Raft, uma taxa de sucesso próxima aos 100% é possível afirmar que a rede não escala, pelo menos, a partir de 20 nodos. Assim, a escalabilidade é um fator de obstrução do desempenho de redes da plataforma *blockchain* Hyperledger Sawtooth.

Outro aspeto, embora não de obstrução, é a complexidade da lógica de um método executado numa rede da plataforma *blockchain* Ethereum, recorrendo ao seu cliente Geth. Como já haveria sido mencionado no Estado de Arte, esta plataforma previne determinados vetores de ataque (p.e., *spam* da rede), ou execução de ciclos lógicos infinitos, através de taxas em *gas* cobradas de acordo com a complexidade lógica que uma transação requer que seja executada na EVM. Na análise realizada foi possível verificar que a taxa de transferência transacional e a latência foram impactadas pela execução de métodos com lógica mais complexa. Assim, embora não seja um aspeto de obstrução do desempenho da rede, é um fator que deve ser considerado na escolha da plataforma *blockchain*.

Qual a plataforma *blockchain* que melhor se adequa aos requisitos do caso de uso?

Os requisitos do caso de uso foram descritos no **Capítulo 4** na **Secção 4.3**. No entanto, a instituição acolhedora, através do respetivo orientador, apenas definiu a lógica que pretendia validar na sua aplicação na tecnologia *blockchain*. Não foram definidos valores que deveriam ser validados na análise de desempenho, uma vez que este se trataria de uma dissertação que visava o estudo da tecnologia e não a definição de um projeto empresarial real. Assim sendo, as seguintes conclusões baseiam-se nos resultados obtidos e no conhecimento adquirido face à tecnologia e às plataformas *blockchain* investigadas.

Face aos resultados da análise de desempenho, foram possíveis identificar os aspetos críticos das plataformas *blockchain* na resposta à primeira pergunta de investigação desta dissertação. Não existindo qualquer definição da empresa para a projeção da escala da rede, a escalabilidade ser uma obstrução no desempenho de uma rede Hyperledger Sawtooth pode não ser considerado um problema.

Por outro lado, o facto da complexidade lógica numa rede Ethereum com cliente Geth afetar o desempenho da rede em produção já é um fator considerável. Uma vez que a complexidade lógica foi eventualmente reduzida, e que a estrutura de dados foi definida face aos requisitos do caso de uso, o facto de esta impactar o desempenho de uma rede é um ponto a ter em atenção. No entanto, é necessário considerar que o desempenho da rede, embora afetado, apresentou valores não muito discrepantes das redes Hyperledger Sawtooth.

É também necessário referir que apenas foram consideradas duas plataformas *blockchain* de um amplo mercado de DLTs. Portanto, em suma, e considerando que não existem dados mais concretos do desempenho expectável para a rede em produção e as limitações de *hardware* que existiram e não permitiram uma análise com valores mais próximos do real, não existem condições que permitam concluir sobre a plataforma *blockchain* que cumpre melhor os requisitos do caso de uso.

6.2 Trabalho Futuro

Estrutura de Dados

Como mencionado no Estado de Arte, o funcionamento de um *smart contract* é análogo ao de um agente autónomo que se encontra em execução na rede da plataforma *blockchain*. No desenvolvimento desta dissertação apenas foi considerado o processo de certificação enquanto agente autónomo, de forma a simplificar a implementação. No entanto, com a possibilidade de escalar o trabalho desenvolvido num sistema mais complexo, seria interessante abordar as entidades de produção e de certificação enquanto agentes autónomos, uma vez que estas possuem as suas próprias características (i.e., a entidade de produção gere os seus próprios produtos e entidades de produção e a entidade de certificação gere os seus serviços de certificação) que poderiam ser automatizadas e geridas na rede.

Outro ponto a considerar é a escalabilidade dos *stakeholders* no âmbito do processo de certificação de produtos agroalimentares. Isto é, apesar do caso de uso desta dissertação ter sido definido e restringido principalmente às entidades de produção e de certificação, seria interessante escalar os *stakeholders* a toda a cadeia de abastecimento e tornar o projeto numa aplicação descentralizada disponível ao setor.

Design da Análise

Apesar de não ter sido realizado nesta dissertação por limitações de planeamento,seria fulcral a análise de desempenho com base na distribuição geográfica dos nodos, de forma a obter valores e conclusões mais próximas de uma rede em produção. Deste modo, poderia ter sido implementada uma rede em diversas máquinas dispersas por diferentes pontos geográficos e utilizar um ou múltiplos nodos, como ponto de entrada para os *benchmarks*. A configuração ideal passaria por aplicar a distribuição de carga dos *workers*, como mencionado na dissertação, por diversas máquinas distribuídas geograficamente, no entanto não é conhecida a viabilidade desta implementação.

Relativamente ao desenvolvimento de artefactos para análise, seria interessante o desenvolvimento de código fonte com recurso, por exemplo, à ferramenta Jupyter Notebook, para automatizar o processamento dos dados dos relatórios gerados em HTML pela *framework* Hyperledger Caliper. Deste modo seria também possível desenvolver outro tipo de gráficos para permitir diferentes análises daquelas realizadas.

Por fim, um último aspeto que foi notado no desenvolvimento desta dissertação mas que não foi possível de desenvolver por limitações de planeamento, consistiria na implementação de uma rede com recurso às ferramentas da *cloud*. Desta forma, era possível validar a integração do *software* existente na *cloud* para implementação de rede *blockchain* e simplificaria a implementação de uma rede cujos nodos se encontrassem distribuídos geograficamente.

This page is intentionally left blank.

Referências

[1] C. Fan, S. Ghaemi, H. Khazaei, and P. Musilek, "Performance evaluation of blockchain systems: A systematic survey," *IEEE Access*, vol. 8, 2020.

[2] N. B. Truong, K. Sun, G. M. Lee, and Y. Guo, "Gdpr-compliant personal data management: A blockchain-based solution," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1746–1761, 2020.

[3] X. Liu, R. Chen, Y. Chen, and S. Yuan, "Off-chain data fetching architecture for ethereum smart contract," in *2018 International Conference on Cloud Computing, Big Data and Blockchain (ICCBB)*, pp. 1–4, Novembro 2018.

[4] Q. Zhou, H. Huang, Z. Zheng, and J. Bian, "Solutions to scalability of blockchain: A survey," *IEEE Access*, vol. 8, pp. 16440–16455, 2020.

[5] H. Leppelsack, "Experimental performance evaluation of private distributed ledger implementations," 2018.

[6] Q. Nasir, I. Qasse, M. Talib, and A. Nassif, "Performance analysis of hyperledger fabric platforms," *Security and Communication Networks*, vol. 8, pp. 1–14, Setembro 2018.

[7] A. Bonanno, "A globalização da economia e da sociedade: fordismo e pós-fordismo no setor agroalimentar," *Globalização, Trabalho, Meio Ambiente: mudanças socioeconómicas em regiões frutícolas para exportação. Recife: Ed. Universitária da UFPE*, pp. 47–94, 1999.

[8] M. Hatanaka, C. Bain, and L. Busch, "Third-party certification in the global agrifood system," *Food Policy*, vol. 30, no. 3, pp. 354 – 369, 2005. Private Agri-food Standards: Implications for Food Policy and Agri-food Systems.

[9] F. das Indústrias Portuguesas Agro-Alimentares, "Políticas de competitividade para o setor agro-alimentar," Maio 2011.

[10] E. Comission, "Monitoring eu agri-food trade: Developments in 2019," 2020.

[11] Instituto Nacional de Estatística, "Estatísticas agrícolas: 2018," 2019.

[12] Y. Liu, X. Ma, L. Shu, G. P. Hancke, and A. M. Abu-Mahfouz, "From industry 4.0 to agriculture 4.0: Current status, enabling technologies, and research challenges," *IEEE Transactions on Industrial Informatics*, vol. 17, pp. 4322–4334, Junho 2021.

[13] J. T. Mentzer, W. DeWitt, J. S. Keebler, S. Min, N. W. Nix, C. D. Smith, and Z. G. Zacharia, "Defining supply chain management," *Journal of Business Logistics*, vol. 22, no. 2, pp. 1–25, 2001.

[14] E. Comission, "Agri-food trade in 2018: another successful year for agri-food trade," Setembro 2019.

[15] M. Hatanaka and L. Busch, "Third-party certification in the global agrifood system: An objective or socially mediated governance mechanism?," *Sociologia Ruralis*, vol. 48, no. 1, pp. 73–91, 2008.

[16] L. Ge, C. Brewster, J. Spek, A. Smeenk, J. Top, F. van Diepen, B. Klaase, C. Graumans, and M. de Ruyter de Wildt, *Blockchain for agriculture and food: Findings from the pilot study*. Wageningen Economic Research, 2017.

[17] S. I. A. Meerza and C. Gustafson, "Consumer response to fraudulent producer behavior in the agri-food marketing system," *Cornhusker Economics*, Maio 2018.

[18] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system [white paper]," Março 2009.

[19] S. Haber and W. Stornetta, "How to time-stamp a digital document," *J. Cryptology*, vol. 3, p. 99–111, Janeiro 1991.

[20] A. Back, "Hashcash - a denial of service counter-measure," Setembro 2002.

[21] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," in *Concurrency: the Works of Leslie Lamport*, pp. 203–226, 2019.

[22] V. Foundation, "Vechain whitepaper 2.0 [white paper]," tech. rep., VeChain, Dezembro 2019.

[23] S. Hyperledger, "Intel corporation, poet 1.0 specification." `https://sawtooth.hyperledger.org/docs/core/releases/1.0/architecture/poet.html`. Acedido em: 29 de Maio 2021.

[24] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14)*, pp. 173–186, 1999.

[25] N. Szabo, "Smart contracts." `https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html`, 1994. Acedido em: 29 de Maio 2021.

[26] N. Szabo, "Smart contracts: Building blocks for digital markets." `https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html`, 1996. Acedido em: 29 de Maio 2021.

[27] V. Y. Kemmoe, W. Stone, J. Kim, D. Kim, and J. Son, "Recent advances in smart contracts: A technical overview and state of the art," *IEEE Access*, vol. 8, pp. 117782–117801, 2020.

[28] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *White Paper*, vol. 3, no. 37, 2014.

[29] H. Watanabe, S. Fujimura, A. Nakadaira, Y. Miyazaki, A. Akutsu, and J. Kishigami, "Blockchain contract: Securing a blockchain applied to smart contracts," in *2016 IEEE International Conference on Consumer Electronics (ICCE)*, pp. 467–468, Janeiro 2016.

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

[31] S. S. Shetty, C. A. Kamhoua, and L. L. Njilla, *Blockchain for Distributed Systems Security*, pp. 51–61. 2019.

[32] I. Eyal and E. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," vol. 8437, Novembro 2013.

[33] "Preventing ddos attack in blockchain system using dynamic transaction limit volume," *International Journal of Control and Automation*, vol. 10, pp. 131–138, Dezembro 2017.

[34] Hyperledger, "Hyperledger sawtooth documentation." `https://sawtooth.hyperledger.org/docs/core/releases/latest/`. Acedido em: 2 de Junho 2021.

[35] G. Sylvester *et al.*, *E-agriculture in action: blockchain for agriculture, opportunities and challenges.* FAO, 2019.

[36] H. Xiong, T. Dalhaus, P. Wang, and J. Huang, "Blockchain technology for agriculture: Applications and rationale," *Frontiers in Blockchain*, vol. 3, p. 7, 2020.

[37] A. Patil, B. Adhi Tama, Y. Park, and K. H. Rhee, *A Framework for Blockchain Based Secure Smart Green House Farming.* Janeiro 2018.

[38] Y.-P. Lin, J. R. Petway, J. Anthony, H. Mukhtar, S.-W. Liao, C.-F. Chou, and Y. Ho, "Blockchain: The evolutionary next step for ict e-agriculture," 2017.

[39] J. Lin, Z. Shen, A. Zhang, and Y. Chai, "Blockchain and iot based food traceability for smart agriculture," in *ICCSE'18*, 2018.

[40] H. Foundation, "How walmart brought unprecedented transparency to the food supply chain with hyperledger fabric." `https://core.ac.uk/download/pdf/345088356.pdf`. Acedido em: 20 de Março 2021.

[41] "Te-food." Acedido em: 30 de Novembro 2020, em: `https://te-food.com`.

[42] TE-FOOD, "Introduction of te-food's technology." Acedido em: 30 de Novembro 2020, em: `https://medium.com/te-food/introduction-of-te-foods-technology-732cdd90bb16`, Setembro 2018.

[43] H. Performance and S. W. Group, "Hyperledger blockchain performance metrics [white paper]." `https://www.hyperledger.org/learn/publications/blockchain-performance-metrics`, Outubro 2018. Acedido em: 18 de Junho 2021.

[44] S. Sakr and A. Y. Zomaya, eds., *Microbenchmark*, pp. 1143–1152. Cham: Springer International Publishing, 2019.

[45] M. Tuler de Oliveira, G. Carrara, N. Fernandes, C. Albuquerque, R. Carrano, D. Medeiros, and D. Menezes, "Towards a performance evaluation of private blockchain frameworks using a realistic workload," pp. 180–187, Fevereiro 2019.

[46] S. T. Leutenegger and D. Dias, "A modeling study of the tpc-c benchmark," *SIGMOD Rec.*, vol. 22, pp. 22–31, Junho 1993.

[47] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with ycsb," pp. 143–154, 2010.

[48] D. E. Difallah, A. Pavlo, C. Curino, and P. Cudre-Mauroux, "Oltp-bench: An extensible testbed for benchmarking relational databases," *Proc. VLDB Endow.*, vol. 7, pp. 277–288, Dezembro 2013.

[49] A. Aldweesh, M. Alharby, M. Mehrnezhad, and A. Van Moorsel, "Opbench: A cpu performance benchmark for ethereum smart contract operation code," in *2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 274–281, Julho 2019.

[50] A. Aldweesh, M. Alharby, E. Solaiman, and A. van Moorsel, "Performance benchmarking of smart contracts to assess miner incentives in ethereum," in *2018 14th European Dependable Computing Conference (EDCC)*, pp. 144–149, Setembro 2018.

[51] H. Foundation, "Hyperledger caliper." `https://github.com/hyperledger/caliper`. Acedido em: 20 de Março 2021.

[52] B. C. Ooi, "Blockbench." `https://github.com/ooibc88/blockbench`. Acedido em: 20 de Março 2021.

[53] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "Blockbench: A framework for analyzing private blockchains," in *Proceedings of the 2017 ACM International Conference on Management of Data*, (New York, NY, USA), pp. 1085–1100, Association for Computing Machinery, 2017.

[54] H. Performance and S. W. Group, "Hyperledger blockchain performance metrics v1.01 [white paper]." `https://www.hyperledger.org/wp-content/uploads/2018/10/HL_Whitepaper_Metrics_PDF_V1.01.pdf`, Outubro 2018. Acedido em: 20 de Março 2021.

[55] C. Project, "Getting started [v0.4.2]." `https://hyperledger.github.io/caliper/v0.4.2/getting-started`. Acedido em: 20 de Março 2021.

[56] C. Project, "Getting started [v0.3.2]." `https://hyperledger.github.io/caliper/v0.3.2/getting-started`. Acedido em: 20 de Março 2021.

[57] S. Rouhani and R. Deters, "Performance analysis of ethereum transactions in private blockchain," in *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pp. 70–74, Novembro 2017.

[58] M. Bez, G. Fornari, and T. Vardanega, "The scalability challenge of ethereum: An initial quantitative analysis," in *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, pp. 167–176, Abril 2019.

[59] P. Wackerow, "Scaling." `https://ethereum.org/en/developers/docs/scaling/`. Acedido em: 20 de Março 2021.

[60] Z. Shi, H. Zhou, Y. Hu, S. Jayachander, C. de Laat, and Z. Zhao, "Operating permissioned blockchain in clouds: A performance study of hyperledger sawtooth," in *2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*, pp. 50–57, Junho 2019.

[61] M. P. Caro, M. Ali, M. Vecchio, and R. Giaffreda, "Blockchain-based traceability in agri-food supply chain management: A practical implementation," *2018 IoT Vertical and Topical Summit on Agriculture - Tuscany (IOT Tuscany)*, pp. 1–4, 2018.

[62] S. Pongnumkul, C. Siripanpornchana, and S. Thajchayapong, "Performance analysis of private blockchain platforms in varying workloads," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–6, Julho 2017.

[63] S. Chen, J. Zhang, R. Shi, J. Yan, and Q. ke, *A Comparative Testing on Performance of Blockchain and Relational Database: Foundation for Applying Smart Technology into Current Business Systems*, pp. 21–34. Janeiro 2018.

[64] S. Benahmed, I. Pidikseev, R. Hussain, J. Lee, S. A. Kazmi, A. Oracevic, and F. Hussain, "A comparative analysis of distributed ledger technologies for smart contract development," in *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1–6, Setembro 2019.

[65] D. G. WOOD, "Polkadot: Vision for a heterogeneous multi-chain frameworkdraft." `https://polkadot.network/PolkaDotPaper.pdf)`. Acedido em: 18 de Junho 2021.

[66] Wikipedia, "Wikipedia talk:naming conventions (legislation)." `https://en.wikipedia.org/wiki/Wikipedia_talk:Naming_conventions_(legislation)`. Acedido em: 18 de Junho 2021.

[67] C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, p. 800. 01 2010.

[68] Ethereum, "Common patterns." `https://docs.soliditylang.org/en/v0.8.6/common-patterns.html`. Acedido em: 18 de Junho 2021.

[69] H. Foundation, "Permissioning design." `https://sawtooth.hyperledger.org/docs/core/nightly/1-1/architecture/permissioning_requirement.html`. Acedido em: 18 de Junho 2021.

[70] S. Seshachala, "Docker vs vms." `https://devops.com/docker-vs-vms`. Acedido em: 15 de Junho 2021.

[71] E. Organization, "Nodes and clients." `https://ethereum.org/en/developers/docs/nodes-and-clients/`. Acedido em: 19 de Junho 2021.

[72] G. Ethereum, "Private network tutorial." `https://geth.ethereum.org/docs/getting-started/private-net`. Acedido em: 19 de Junho 2021.

[73] H. Foundation, "About dynamic consensus." `https://sawtooth.hyperledger.org/docs/core/nightly/1-2/sysadmin_guide/about_dynamic_consensus.html`. Acedido em: 19 de Junho 2021.

[74] H. Foundation, "sawtooth-rest-api." `https://sawtooth.hyperledger.org/docs/core/nightly/1-2/cli/sawtooth-rest-api.html`. Acedido em: 19 de Junho 2021.

[75] C. Project, "Benchmark configuration." `https://hyperledger.github.io/caliper/v0.3.2/bench-config`. Acedido em: 18 de Junho 2021.

[76] H. Foundation, "Journal." `https://sawtooth.hyperledger.org/docs/core/releases/latest/architecture/journal.html`. Acedido em: 20 de Junho 2021.

[77] A. Corso, "Performance analysis of proof-of-elapsed-time (poet) consensus in the sawtooth blockchain framework," 2019.

[78] E. Foundation, "Mining." `https://eth.wiki/en/fundamentals/mining`. Acedido em: 20 de Junho 2021.

[79] Block.one, "Consensus protocol." https://developers.eos.io/welcome/latest/protocol/consensus_protocol. Acedido em: 4 de Junho 2021.

[80] T. Maas, "Everything they don't want you to know about eos.." https://medium.com/hackernoon/everything-they-dont-want-you-to-know-about-eos-the-ethereum-killer-9939c43aa2df. Acedido em: 4 de Junho 2021.

[81] Block.one, "Technical features." https://developers.eos.io/welcome/v2.0/overview/technical_features. Acedido em: 4 de Junho 2021.

[82] K. Ziechmann, "Networks." https://ethereum.org/en/developers/docs/networks/. Acedido em: 4 de Junho 2021.

[83] S. Richards, "Consensus mechanisms." https://ethereum.org/en/developers/docs/consensus-mechanisms/. Acedido em: 4 de Junho 2021.

[84] Ripple, "Xrp: The best digital asset for global payments." https://ripple.com/xrp/. Acedido em: 4 de Junho 2021.

[85] S. Richards, "Smart contract languages." https://ethereum.org/en/developers/docs/smart-contracts/languages. Acedido em: 4 de Junho 2021.

[86] Hyperledger, "Introduction." https://hyperledger-fabric.readthedocs.io/en/latest/blockchain.html. Acedido em: 4 de Junho 2021.

[87] Hyperledger, "The ordering service." https://hyperledger-fabric.readthedocs.io/en/latest/orderer/ordering_service.html. Acedido em: 4 de Junho 2021.

[88] H. Anwar, "Hyperledger sawtooth vs. fabric: How are they different?." https://101blockchains.com/hyperledger-sawtooth-vs-fabric/. Acedido em: 4 de Junho 2021.

[89] Hyperledger, "Smart contract processing." https://hyperledger-fabric.readthedocs.io/en/latest/developapps/smartcontract.html. Acedido em: 4 de Junho 2021.

[90] I. Corporation, "Introduction." https://sawtooth.hyperledger.org/docs/core/releases/latest/introduction.html. Acedido em: 4 de Junho 2021.

[91] Soramitsu, "Overview of iroha." https://iroha.readthedocs.io/en/main/overview.html. Acedido em: 4 de Junho 2021.

[92] Soramitsu, "What's inside iroha?." https://iroha.readthedocs.io/en/main/concepts_architecture/architecture.html. Acedido em: 4 de Junho 2021.

[93] Soramitsu, "What's inside iroha?." https://iroha.readthedocs.io/en/main/develop/libraries.html. Acedido em: 4 de Junho 2021.

[94] R3, "Network." https://docs.corda.net/docs/corda-os/4.8/key-concepts-ecosystem.html. Acedido em: 4 de Junho 2021.

[95] R3, "Contracts." https://docs.corda.net/docs/corda-os/4.8/key-concepts-contracts.html. Acedido em: 4 de Junho 2021.

[96] D. Schwartz, N. Youngs, A. Britto, *et al.*, "The ripple protocol consensus algorithm," *Ripple Labs Inc White Paper*, vol. 5, no. 8, p. 151, 2014.

[97] D. Mazieres, "The stellar consensus protocol: A federated model for internet-level consensus," *Stellar Development Foundation*, vol. 32, 2015.

[98] S. D. Foundation, "Software and sdks." `https://developers.stellar.org/docs/software-and-sdks/`. Acedido em: 4 de Junho 2021.

This page is intentionally left blank.

Appendices

Apêndice A

Proposta de Estágio

Titulo Estágio

Blockbattle - The best Blockchain wins

Áreas de especialidade

Engenharia de Software

Sistemas de Informação

Local do Estágio

Rua Pedro Nunes - IPN , Escritório 2.18 3030–199 Coimbra

Enquadramento

A empresa Ubiwhere, Lda. foi constituída em Setembro de 2007 em Aveiro por três investigadores de telecomunicações provenientes do Instituto de Telecomunicações e da PT Inovação. Com sede em Aveiro, o objetivo primário da empresa é o desenvolvimento e investigação de tecnologias de ponta, para conceber a tecnologia mais avançada e criar propriedade intelectual de grande valor. A Ubiwhere tem vindo a investir, desde a sua criação, no aumento da sua capacidade tecnológica e na diferenciação dos seus produtos e serviços, apostando em nichos que apresentam vantagens competitivas, mas que exigem um forte investimento no contínuo desenvolvimento tecnológico. A empresa trabalha em diversos sectores, como a Internet das Coisas, com foco nas Cidades Inteligentes e Agricultura, nas tecnologias de Telecomunicação (e.g. 5G) e em tecnologias do futuro.

Uma das principais spin-offs da Ubiwhere, a Zenithwings, está neste momento a explorar uma plataforma para agricultura de precisão e Industria 4.0. Esta plataforma tem como principal objetivo a digitalização das cadeias de abastecimento e os processos que há muito estão relacionados com a genese de uma empresa e de um processo/produto.

A ideia para o presente estágio, passa por investigar uma nova funcionalidade para este produto que se baseia na aplicabilidade da tecnologia blockchain para a certificação de produtos.

Objetivo

O objectivo principal desta dissertação centra-se no estudo da aplicação da tecnologia para a certificação de produtos. Os objetivos secundários passam por estudar as tecnologias que existem no mercado e testar a sua aplicabilidade prática neste cenário. Pretende-se assim, que sejam realizados prototipos de implementação de um caso de uso de exemplo e que seja feita uma comparação entre as tecnologias levantadas em estado da arte.

Plano de Trabalhos - Semestre 1

Fase 1 - Estudo da Solução e Planeamento

* T1 – Estudo do problema e Elaboração do estudo do Estado da Arte sobre tipos de DLTs que resolvam o descrito

* T2 - Definição das diferentes arquiteturas da solução a serem testadas e sua composição:

- arquitetura de software

- interface de ligação diferentes componentes

- mapeamento com a tecnologia blockchain e integração com o restante ambiente de software (terceiros, frontend, etc)

* T3 - Escrita do relatório intermédio

Plano de Trabalhos - Semestre 2

Fase 2 - Desenvolvimento e conclusão da dissertação

* T4 – Implementação das diferentes soluções

* T5 – Implementação e testes à solução (compatibilidade dos componentes, benchmarks à performance e escalabilidade)

* T6 – Elaboração do Relatório Final do Projeto

Condições

Bolsa de estágio - Valor subsídio de alimentação diário

Orientador

Vítor Sousa - vsousa@ubiwhere.com 📩

Apêndice B

Tabela Comparativa de Plataformas *Blockchain*

	Foco Industrial	**Taxonomia**	**Mecanismo de Consenso**	***Throughput***	**Smart Contract**	**Linguagens *Scripting***
EOSIO	Diversos	Desconhecido	aBFt + DPoS [79]	1000 TPS [80]	Sim	C++ [81]
Ethereum	Diversos	Pública/Privada *Permissionless* [82]	PoW; PoS (Casper) [83]	15 TPS [84]	Sim	Solidity; Vyper; Yul; Yul+ [85]
Hyperledger Fabric	Diversos	Privada *Permissioned* [86]	Raft [87]	> 2000 TPS [88]	Sim	Go; Node.js; Java [89]
Hyperledger Sawtooth	Diversos	Privada *Permissioned*/ *Permissionless* [88]	PBFT; Raft; PoET CFT ou SGX [90]	> 1000 TPS [88]	Sim	Python; JavaScript; Go; Rust; Java; Swift; C++; Solidity (Seth) [90]
Hyperledger Iroha	Diversos	Privada *Permissioned* [91]	YAC [92]	Desconhecido	Sim	Java; JavaScript; Python; Swift [93]
R3 Corda	Finanças	Privada *Permissioned* [94]	Desconhecido	Desconhecido	Sim	Java; Kotlin [95]
XRP Ledger	Finanças	Desconhecido	RPCA [96]	1500 TPS [84]	Não	-
Stellar	Finanças	Desconhecido	SCP [97]	Desconhecido	Sim	JavaScript; Java; Go [98]

Tabela B.1: Comparação das plataformas blockchain existentes

Apêndice C

Exemplo de Certificado de Qualidade Diferenciada

Certificado nº AZT0013UT

AZEITE DE TRÁS-OS-MONTES

DENOMINAÇÃO DE ORIGEM PROTEGIDA

Pelo presente, a SATIVA, Desenvolvimento Rural, Lda., organismo de controlo e certificação, reconhecido pelo Ministério da Agricultura e indicado pela AOTAD – Associação dos Olivicultorees de Tras-os-Montes e Alto Douro, para o controlo e certificação da **DENOMINAÇÃO DE ORIGEM PROTEGIDA** do **AZEITE DE TRÁS-OS-MONTES**, certifica que o operador:

MARIA DO CARMO RABAÇAL & ARAGÃO. LDA

contribuinte nº 507098196

ESTRADA NACIONAL 215

5350-051 ALFÂNDEGA DA FÉ

submeteu as suas actividades a controlo e satisfaz os requisitos do Regulamento (UE) nº1151/2012 e do caderno de especificações relativo ao AZEITE DE TRÁS-OS-MONTES (esquema de certificação do tipo 5), aprovado pelo Regulamento (CE) nº 1107/96 da Comissão de 12 de Junho de 1996, para o seguinte produto:

Azeite deTrás-os-Montes

Válido até: 30 – 09 – 2016 .

Lisboa, 1 de Fevereiro de 2016

O Departamento de Certificação,

SATIVA

Antonio Mantas

(Responsável da Certificação)

IPAC acreditação C0005 Certificação Produtos

Este documento é propriedade da SATIVA e deverá ser devolvido se solicitado.

Fonte: `http://www.casaaragao.eu/wp-content/uploads/2016/03/CERTIFICADO-2016.jpg`

Figura C.1: Exemplo de certificado de qualidade diferenciada