FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Blockchain: Privacidade e Partilha de Informação Clínica

Ricardo Alexandre Mariz Lopes

Mestrado Integrado em Engenharia Informática e Computação

Orientador: António Miguel Pontes Pimenta Monteiro

23 de Julho de 2018

Blockchain: Privacidade e Partilha de Informação Clínica

Ricardo Alexandre Mariz Lopes

Mestrado Integrado em Engenharia Informática e Computação

23 de Julho de 2018

Resumo

O *Blockchain* apareceu como solução à necessidade de maior transparência no mercado financeiro, através da eliminação dos intermediários, por estes não serem confiáveis mas, até então, necessários para a validação das transferências bancárias. Na prática, o *Blockchain* carateriza-se como uma base de dados distribuída, altamente segura e baseada em métodos de criptografia, onde toda a informação é acessível a todos. Tal como os mercados financeiros, a indústria da saúde movimenta grandes quantidades de dados, neste caso, Registos Médicos Eletrónicos de pacientes. No entanto, estes dados são guardados e usados exclusivamente pela respetiva instituição, isto é, não há partilha de dados entre diferentes instituições, nem garantia de persistência da informação.

Desta forma, não existe um controlo de dados livre e de fácil acesso por parte do paciente, quer para consulta própria, quer para uso futuro por parte de outras instituições de cuidados de saúde. Tal resulta na inevitável necessidade de confiança em relação à instituição responsável pela manutenção dos dados médicos e, ainda, na dispersão dos dados médicos pelas diferentes instituições, podendo ocorrer perda de dados.

Pretende-se solucionar este problema através do desenvolvimento de uma plataforma que implementa um *Blockchain* privado para possibilitar o controlo e acesso distribuído. Por outro lado, faz uso de contratos digitais inteligentes, de modo a garantir acesso privilegiado e restritivo aos dados. Desta forma, garante-se a autenticação e confidencialidade dos mesmos.

Devido à mudança do paradigma da era digital, a União Europeia levou a cabo uma reforma das regras de proteção de dados, permitindo um maior controlo e facilidade de acesso aos mesmos, independentemente da região geográfica. Assim, foi dada preferência a métodos de pseudonimização, como o *Blockchain*, que considera muitas das novas imposições e poderá servir de base para as instituições cumprirem o novo regulamento Europeu.

Palavras-chave: Blockchain, Blockchain com permissões, Partilha de registos clínicos, Proteção de dados da União Europeia, Base de dados hospitalar

Abstract

The Blockchain emerged as a solution to the necessity of higher transparency in the financial market, through the removal of the intermediates, since they are not reliable but, until now, they were required for the bank transfers validation. In practice, the Blockchain is featured as a distributed data base, highly secure and based on encryption methods, where all information is accessible to everyone. As all financial markets, the health industry deals with a high quantity of data, in this case with the patients' Electronic Health Records. However, these records are kept and used by the respective institution exclusively, meaning that the information is not shared between different institutions and there is no guarantee of information persistence.

Thus, there is no free or easy access data control for the patient to use, either for its own consult, or for other healthcare institutions future access. This results in the inevitable need for trust, regarding the institution responsible for the medical data maintenance and, also, in the medical data dispersion through different institutions, potentially affording data loss.

To solve this issue, a platform which implements a private Blockchain that enables a distributed control and access was developed. Also, it uses smart digital contracts in a way that ensures privileged and restricted access to data. Thus, data authentication and confidentiality are guaranteed.

Due to the change of the digital era paradigm, the European Union performed a renovation in the data protection rules, allowing a higher data control and an easier data access, regardless of the geographic region. Therefore, it has been given preference to pseudonymization methods, such as Blockchain, which regards many of the new impositions and could function as a basis for many institutions to comply with the new European regulation.

Keywords: Blockchain, Permissioned ledger, Healthcare data system, Healthcare data sharing, European Union data protection

CCS Concepts: • Security and privacy → Database and storage security → Data anonymization and sanitization; • Applied computing → Life and medical sciences → Health care information systems

Agradecimentos

Esta dissertação marca o fim da minha vida académica e o começo de uma nova etapa. Muitas foram as pessoas que me acompanharam neste percurso e a elas eternizo as minhas palavras.

Começo por agradecer aos meus pais por todo o apoio, ensinamentos e conselhos. Estarei eternamente agradecido por tudo o que fizeram por mim.

À minha família por estar sempre presente.

Um agradecimento especial à minha namorada por todas as palavras, momentos e incentivos ao longo do meu percurso académico.

A todos os meus amigos um enorme obrigado, por tudo.

Uma palavra de agradecimento aos meus colegas da *Glintt*, que durante cinco meses foi a minha segunda casa. Em especial aos Engenheiros Pedro Rocha, Francisco Correia, José Melo, Pedro Silva e Luís Costa pelo contributo na realização desta dissertação.

À Faculdade de Engenharia por cinco anos de magníficos ensinamentos e preparação.

À Tuna de Engenharia ficarei eternamente grato pelos momentos inesquecíveis.

Um agradecimento particular à Catarina Paiva pelo notável contributo nesta dissertação.

Por fim, ao Professor Pimenta Monteiro por ter orientado esta dissertação, que sem o seu contributo não teria sido possível.

Ricardo Alexandre Mariz Lopes

"Next time someone complains that you have made a mistake, tell him that may be a good thing. Because without imperfection, neither you nor I would exist."

Stephen William Hawking

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

ACL	*Access Control List*
API	*Application Programming Interface*
CA	*Certificate Authority*
CPU	*Central Processing Unit*
FHIR	*Fast Healthcare Interoperability Resources*
HF	*Hyperledger Fabric*
JSON	*JavaScript Object Notation*
MSP	*Membership Service Provider*
REST	*Representational State Transfer*
RME	*Registo Médico Eletrónico*
RPC	*Remote Procedure Call*
RPS	*Requests Per Second*
SDK	*Software Development Kit*
SQL	*Structured Query Language*
TLS	*Transport Layer Security*

Capítulo 1

Introdução

1.1 Enquadramento e domínio de intervenção

O modo tradicional do mercado financeiro pressupõe instituições centralizadas que tutelam as mais variadas operações financeiras. Estas instituições, que podem ser públicas ou privadas, garantem a estabilidade financeira através da monitorização e persistência de toda a atividade financeira, de forma privada e confidencial. Obviamente, desta organização advêm elevados custos que são pagos, por um lado, pelos seus utilizadores através de elevadas taxas de utilização e, por outro, por numerosos esquemas financeiros pouco transparentes por parte das próprias instituições, de modo a suportar quer o seu funcionamento, quer a apresentação de avultados lucros da sua operação. A longo prazo, estes esquemas tornam-se incomportáveis, resultando em fortes crises financeiras, como a de 2008. Foi por essa altura que o *Blockchain* começou a ser idealizado de forma a solucionar o problema de transparência do mercado financeiro, através da criação da criptomoeda *Bitcoin*, que tem no *Blockchain* a sua principal inovação. [Nak09]

O *Blockchain* traduz-se numa base de dados distribuída, criptograficamente segura e que garante persistência, imutabilidade e fácil acesso à informação. Estruturalmente, consiste numa cadeia de blocos, onde cada bloco adicionado referencia o anterior.

Com provas dadas no mercado financeiro, o *Blockchain* começou a ser alvo de estudo para aplicação em diferentes áreas e domínios para além da área financeira. Assim, foram surgindo vários casos de uso, bem como protótipos onde as principais características do *Blockchain* são utilizadas de modo a resolver, ou melhorar, problemas existentes.

A área da saúde pode beneficiar bastante das inovações advindas do *Blockchain* no que toca à gestão, manutenção e persistência de dados médicos. Isto deve-se à sua estrutura descentralizada, o que permite transparência e facilidade de acesso, mantendo, contudo, propriedades críticas como a segurança, privacidade e integridade da informação guardada. [Met16]

A *Glintt Healthcare Solutions* é uma empresa Portuguesa com mais de 20 anos de experiência em desenvolvimento de soluções para a saúde, orientadas ao setor público e privado, e preza pela inovação de soluções e tecnologias. Desta feita, propõe o desenvolvimento de um protótipo direcionado para a gestão de registos médicos, utilizando, para isso, o *Blockchain*.

1.2 O problema e sua caracterização

A indústria da saúde movimenta grandes quantidades de dados, tais como os Registos Médicos Eletrónicos (RME) de pacientes.

Estes registos são guardados e utilizados exclusivamente pela instituição onde tiveram origem, ou seja, não há partilha de dados entre diferentes instituições médicas pois cada uma tem a sua base de dados, o que contribui para a dispersão de informação médica. [AEVL16]

Para o paciente, isto torna-se num problema sério visto que não tem controlo direto sobre os seus dados, nem um local único com acesso a toda a sua informação médica independentemente da instituição.

Uma vez que nem sempre os dados são guardados de forma segura pelas instituições, existe o risco de violação de privacidade e extravio de informação, não havendo certezas quanto à integridade da informação guardada. [AORBK17]

Deduz-se, portanto, que o problema é, por um lado, a falta de transparência e confiança nos intermediários e, por outro, a inexistência de uniformização de persistência, controlo e facilidade de acesso aos dados.

Devido à mudança do paradigma da era digital, a União Europeia levou a cabo a reforma da proteção de dados, garantindo um acesso e controlo mais fácil dos próprios dados, assim como o direito de portabilidade dos mesmos, independentemente da região geográfica Europeia.

Foi, como tal, dada preferência a métodos de pseudonimização, como o *Blockchain*, que considera muitas das novas imposições de segurança e poderá, assim, servir de base para as instituições cumprirem o novo regulamento Europeu. [EU1]

1.3 Objetivos a atingir

Os principais objetivos são o planeamento, desenvolvimento e implementação de um protótipo, funcional e realista, de um modelo de gestão de RMEs, no que toca às atuais necessidades empresariais, pretendendo-se cumprir as normas Europeias e ainda criar uma solução funcional, fácil e acessível aos utentes.

Este protótipo terá de assegurar a persistência, autenticidade e confidencialidade dos registos e ainda permitir o seu fácil acesso e partilha, sempre com consentimento do utente.

O ponto de diferenciação será a agregação de informação médica de diferentes instituições através da utilização do *Blockchain* como solução para o problema, servindo como prova de conceito em relação à utilização da tecnologia na área da saúde.

1.4 Benefícios esperados com a solução do problema

Os principais benefícios prendem-se com o cumprimento das normas Europeias e com a criação de uma solução fiável, segura e útil de modo a facilitar o dia a dia dos utentes.

Outro benefício é o baixo custo de implementação, que ocorre devido à utilização de tecnologias de código aberto, em estado avançado de desenvolvimento e modularmente orientadas para a reutilização em novas aplicações.

Os utentes serão beneficiados em termos práticos pela disponibilização e controlo dos seus dados, independentemente da instituição, num local único com fácil gestão, acesso e garantias de segurança e durabilidade, melhorando, significativamente, a qualidade de vida do utente. [EU2]

1.5 Estrutura da dissertação

Para além da introdução, esta dissertação contém mais 5 capítulos. No capítulo 2, é descrito o estado da arte com foco no domínio de intervenção, projetos e soluções *Blockchain* na saúde e tecnologias a serem utilizadas. O capítulo 3 expõe os requisitos, funcionalidades e a arquitetura da plataforma desenvolvida. O capítulo 4 diz respeito à implementação da plataforma. O capítulo 5 mostra os testes e resultados obtidos pela utilização da plataforma desenvolvida. Por fim, no capítulo 6 são apresentadas as conclusões e o trabalho futuro.

Capítulo 2

Estado da arte

2.1 Introdução

Na primeira secção é apresentado o domínio de intervenção que explica os fundamentos necessários para entender a tecnologia *Blockchain* e as suas possibilidades, como o funcionamento dos sistemas distribuídos, os contratos inteligentes e mecanismos de segurança. Seguidamente o *Blockchain* é escrutinado ao nível da sua base estrutural e categorização, passando-se pela sua criação com a *Bitcoin*, bem como pela sua evolução com o *Ethererum*. De forma mais concreta, na área do trabalho a desenvolver, é feita a caracterização dos RMEs e aborda-se a sua importância na evolução da indústria da saúde. É ainda apresentado o novo regulamento Europeu da proteção de dados, que se constitui como um fator determinante no desenvolvimento do protótipo.

Na segunda secção são apresentados alguns projetos e soluções *Blockchain* na área da saúde, com especial interesse nas suas características e inovações. Estes projetos são a prova da aplicabilidade do *Blockchain* às regras de negócio da indústria da saúde.

Por fim, na última secção, são caracterizadas e descritas duas plataformas possíveis para o desenvolvimento de soluções *Blockchain*, que serão cruciais na implementação do protótipo ao nível da funcionalidade e desempenho.

2.2 Domínio de intervenção

2.2.1 Sistemas distribuídos

De modo a explorar todas as capacidades do *Blockchain*, este é usado como um sistema distribuído e torna-se, como tal, importante entender a estrutura, comportamento e possíveis problemas inerentes a este sistema.

Ao contrário de um sistema centralizado, onde a informação é guardada apenas por uma entidade, num sistema distribuído a informação é partilhada entre todos os utilizadores, e esta propriedade é ainda o ponto de diferenciação entre um sistema apenas descentralizado e um sistema descentralizado e distribuído. Na Figura 2.1 é possível perceber a diferença de arquitetura e organização entre diferentes sistemas.

Um sistema distribuído pode ser interpretado como uma rede formada por vários nós, onde cada nó representa uma parte essencial na rede e corresponde a cada utilizador. Os nós comunicam entre si através da troca de mensagens, de modo a atingir um estado comum de consenso.

Num sistema distribuído é expectável a existência de consistência, disponibilidade e tolerância de partição. A primeira propriedade assegura que todos os nós tenham a mesma cópia da informação. A segunda propriedade está relacionada com a capacidade de resposta, ou seja, estar sempre pronto para responder a pedidos, sem falhas. E, por fim, a terceira está relacionada com a capacidade do sistema em lidar e recuperar de possíveis falhas de nós, ou seja, a capacidade do sistema em continuar a funcionar. [GL06]

O *Blockchain*, aliado a métodos de replicação e algoritmos de consenso, garante estas propriedades inerentes aos sistemas distribuídos.

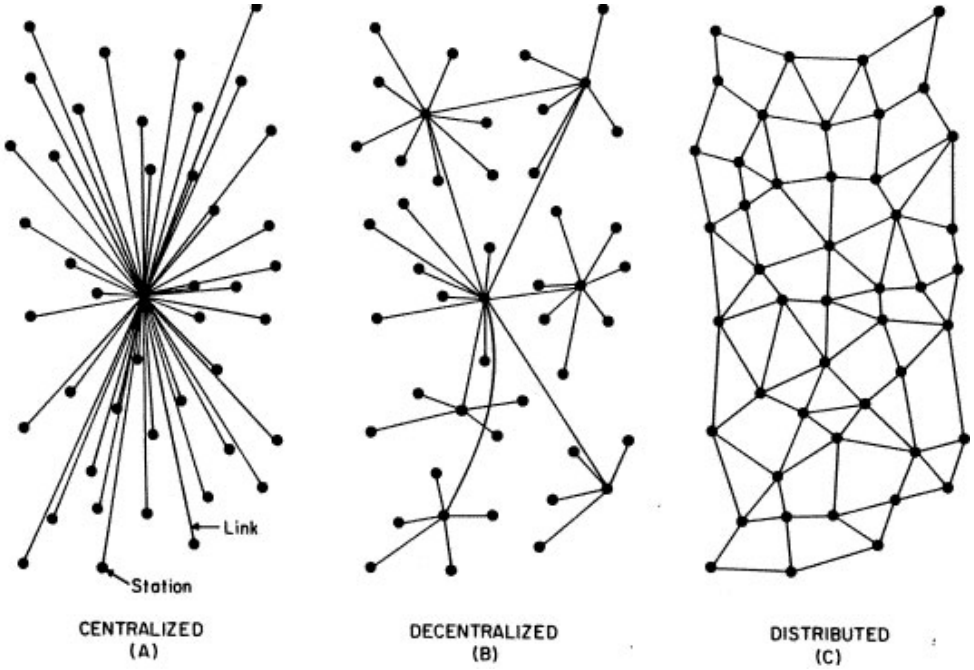

Figura 2.1: Esquema de sistemas de rede por Paul Baran.

2.2.1.1 Tolerância a falhas

Um determinado sistema ou componente falham quando não funcionam de acordo com as suas especificações. Por outro lado, um sistema é tolerante a falhas quando continua a funcionar corretamente, independentemente da existência de falhas de componentes constituintes.

Uma métrica de avaliação da fiabilidade de um sistema distribuído corresponde à tolerância a falhas bizantinas [TS06]. Uma falha bizantina consiste numa falha indetetável do sistema, ou seja, não é identificada como uma falha apesar do seu funcionamento incorreto. Um exemplo deste tipo de falhas corresponde a um nó que sofreu uma violação das regras de segurança e está a ser ilegalmente controlado por uma entidade que não tinha direito de acesso.

2.2.2 Contratos inteligentes

Os contratos inteligentes foram, pela primeira vez, usados no *Blockchain* através da sua implementação na plataforma *Ethereum*. Desde então, tornaram-se numa funcionalidade fulcral nestes sistemas.

O contrato inteligente é um programa, executado automaticamente por cima do *Blockchain*, quando determinadas condições se verificam. Pode englobar acordos entre diferentes partes na forma de lógicas de negócio, guardar valores e até estados do próprio contrato. Uma vez que não pode haver interrupções na execução de um contrato inteligente, este tem de ser tolerante a falhas.

Uma das principais funções dos contratos inteligentes, é o controlo de acesso à informação nos registos distribuídos. [Bas17]

2.2.3 Segurança e métodos criptográficos

2.2.3.1 Access Control List

A *Access Control List* (ACL) consiste numa lista com a informação necessária para controlar os acessos, por parte de uma entidade, em relação a um determinado recurso do sistema. Esta lista pode ser mantida pelo próprio recurso ou pelo módulo que o gere. [Men92]

2.2.3.2 Chave criptográfica

A chave criptográfica é uma cadeia de *bits* fixa necessária para usar em mecanismos criptográficos. Pode ser pessoal e estar ligada a uma entidade ou de sessão e estar ligada a canais de comunicação.

Existem três fases estritamente ligadas a uma chave criptográfica: a sua criação, a sua persistência e a sua distribuição.

A primeira fase consiste em utilizar métodos criptográficos seguros e pseudo-aleatórios.

A segunda fase tem de ser adaptada a cada situação, considerando a possibilidade de persistência em diferentes locais de forma segura. Por fim, a terceira fase pode passar pela utilização de canais de comunicação seguros, ou, no caso de ser inseguro, pela utilização de protocolos que considerem as necessidades de comunicação.

As chaves podem ser simétricas ou assimétricas. A chave simétrica corresponde a uma única chave secreta, apenas conhecida na origem e destino. As chaves assimétricas são um par relacionado, em que uma delas é privada, de um único utilizador, e a outra é pública, podendo ser divulgada.

A chave pode ser pública ou privada. A chave pública serve para ser partilhada enquanto que a privada tem de ser secreta ao utilizador.

2.2.3.3 Certificado digital

Um certificado digital consiste num ficheiro eletrónico que mapeia uma entidade a uma chave pública criptográfica, e tem como objetivo assegurar a identidade de uma determinada entidade.

O certificado tem de ser emitido por uma autoridade certificadora de confiança que assina o certificado, de modo a assegurar a validade do seu conteúdo. O certificado contém, entre outras, a informação da entidade a certificar, a informação do certificador que inclui a sua assinatura digital e o tempo de expiração do certificado. O X.509 é o standard para a criação de certificados digitais.

2.2.3.4 Função de *hashing*

Uma função de *hashing* transforma um qualquer bloco de informação, com tamanho variável, numa cadeia de carateres de tamanho fixo. Uma característica importante das técnicas de *hashing* é o determinismo, ou seja, os mesmos dados de entrada vão sempre dar os mesmos dados de saída. Outra característica é a irreversibilidade, ou seja, não é possível obter os dados de entrada a partir dos dados de saída. As funções de *hashing* criptográficas devem ainda garantir a impraticabilidade de construir dados de entrada com resultado igual a um valor pré-determinado, ou obtido noutro conjunto de dados de entrada.

2.2.3.5 Algoritmos de encriptação

Tradicionalmente, um algoritmo de encriptação tem como objetivo ocultar informação de forma a torná-la ininteligível e, para isso, utiliza técnicas de encriptação para esconder e técnicas de desencriptação para recuperar a informação. Para ambas é necessário usar chaves criptográficas. Em criptografia simétrica é utilizada a mesma chave nas operações de encriptação e desencriptação. Na criptografia assimétrica encripta-se com uma chave e desencripta-se com a outra do mesmo par: pública ou privada. Pode também servir para identificar a informação de forma inequívoca através de uma *hash*, que funciona como uma impressão digital da informação original. Isto serve para verificar a adulteração da informação.

A encriptação pode ser simétrica ou assimétrica. No caso de ser simétrica, em que existe a partilha de uma chave, é computacionalmente eficiente mas é preciso ter em conta o método de partilha da mesma, uma vez que tem de ser secreta aos intervenientes. Por outro lado, quando a encriptação é assimétrica, existindo pares de chaves públicas e privadas, é um caso computacionalmente exigente. No entanto, apenas é necessário partilhar as chaves públicas, o que não representa um risco, visto que é sempre necessária a chave privada, que tem de ser mantida num local seguro.

2.2.3.6 Políticas baseadas na função do sujeito

O acesso a um determinado recurso por parte de um sujeito está dependente do papel que desempenha no sistema, ou seja, o papel prevalece em relação ao sujeito.

2.2.4 *Blockchain*

O interesse no *Blockchain*, devido às suas principais caraterísticas diferenciadoras em várias áreas, conduziu a uma crescente evolução que, por razões de organização e conveniência, pode ser dividida em três categorias: *Blockchain 1.0*, *Blockchain 2.0* e *Blockchain 3.0*. [Swa15]

Blockchain 1.0 é a designação da primeira utilização do *Blockchain* na área financeira, sob a forma de criptomoeda, possibilitando transações financeiras descentralizadas.

Blockchain 2.0 é uma evolução da primeira categoria, pois introduziu a utilização de contratos sobre o *Blockchain*, o que permitiu a descentralização de mercados, ou seja, outras operações mais complexas em comparação com simples transações monetárias como derivativos ou obrigações.

Blockchain 3.0 carateriza-se pela sua implementação em diferentes áreas, para além da financeira, como é o caso da utilização em serviços governamentais, na indústria da saúde, ciências, literatura, cultura ou arte. [Adv16]

O *Blockchain* foi utilizado pela primeira vez com a criação da *Bitcoin*, sendo a principal tecnologia por detrás da criptomoeda [Nak09]. Na sua base, está uma estrutura de dados *peer-to-peer* em forma de cadeia de blocos de informação, onde apenas é possível acrescentar informação à cadeia, não sendo possível modificar sem consenso da rede. Para o *Blockchain 1.0* cada bloco contém um grupo de transações financeiras; no *Blockchain 2.0*, os blocos contêm transações financeiras com contratos inteligentes; e, no *Blockchain 3.0* contêm qualquer tipo de informação, como, por exemplo, RMEs.

Dependendo da implementação do *Blockchain*, o bloco poderá ter vários campos, dos quais se destacam a marca temporal e a referência para o bloco anterior da cadeia. A marca temporal serve como garantia que um determinado evento, como a adição de um bloco à cadeia, aconteceu a uma determinada altura. Para referenciar o bloco anterior, é utilizada uma técnica de *hashing*, de forma a criar uma *hash* identificadora.

Uma vez que cada bloco referencia o anterior, o primeiro bloco da cadeia, chamado de *genesis*, é gerado diretamente no momento da criação do *Blockchain*.

O *Blockchain* pode ser visto como uma máquina de estados finita, com estados definidos, transitando entre eles até um estado final. De modo a tornar o *Blockchain* numa máquina completa de *Turing* é necessário usar contratos digitais.

De forma a garantir que todos os nós na rede possuam a mesma cadeia e, portanto, a mesma informação, é necessário um mecanismo de consenso, de modo a todos os nós concordarem com um estado da rede.

O *Blockchain* pode ser qualificado relativamente à participação e restrição de operações na rede: *Blockchain* público, *Blockchain* privado, *Blockchain* com permissões, ou *permissioned Blockchain*, e *Blockchain* sem permissões, ou *permissionless Blockchain*.

O *Blockchain* público está disponível publicamente para todos e não pertence a ninguém. Cada utilizador do *Blockchain* público possui uma cópia da cadeia e utiliza mecanismos de consenso, de modo a concordar com o estado da cadeia [Bas17]. Pelo contrário, o *Blockchain* privado é restrito a um determinado grupo de utilizadores, ou organizações, que decidiram partilhar o registo entre eles.

No *permissionless ledger* não há restrições de entidades ou operações na rede, pelo que há a necessidade de criação de um mecanismo de consenso descentralizado e de criptomoedas, de modo a incentivar a manutenção do mesmo.

O *permissioned ledger*, pelo contrário, utiliza um mecanismo de consenso centralizado, uma vez que os utilizadores são conhecidos e confiáveis, utilizando protocolos de concordância de modo a assegurar a versão correta do *Blockchain*. Assim, não é necessária uma criptomoeda para incentivar o consenso. Este *Blockchain* tanto pode ser público como privado. [Bas17]

Explicada a estrutura e mecanismos, é possível fazer uma coletânea das principais caraterísticas do *Blockchain*.

A descentralização permite a eliminação de intermediários e, uma vez que o *Blockchain* é distribuído e partilhado, aumenta-se também a transparência, pois todos têm acesso à informação. Melhora-se igualmente a disponibilidade do sistema, pois, mesmo que alguns nós falhem, a rede continua com os restantes. O *Blockchain* é imutável, visto que após o consenso da rede, torna-se quase impossível a adulteração da cadeia. Por fim, é um sistema seguro pois qualquer transação é assegurada criptograficamente, o que garante integridade de informação. [KKOM17]

Esta tecnologia assume-se, cada vez mais, como uma ferramenta importante para melhorar, e até solucionar, vários problemas existentes na área da saúde. [Met16]

A adoção do *Blockchain* em diferentes áreas conduziu à estandardização de uso com a norma *ISO/TC 307*, de modo a cultivar a inovação e fornecer soluções para problemas globais, enquanto assegura a qualidade, segurança e eficiência das mesmas. [fS16]

2.2.5 *Bitcoin*

A *Bitcoin* é uma moeda digital proposta por *Satoshi Nakamoto* em 2009 [Nak09]. Esta assenta num protocolo estruturado que faz uso de vários métodos e técnicas, de modo a criar uma rede de consenso distribuído que permite efetuar transações *online*. É caracterizada como uma moeda virtual descentralizada, uma vez que não é controlada nem tutelada por uma autoridade central ou governo. Por conseguinte, as funções normalmente desempenhadas pelas instituições passam a ser feitas pela própria rede.

Uma função vital para a estabilidade e credibilidade de uma plataforma financeira consiste em garantir a autenticidade das transações. No caso concreto da *Bitcoin*, estas são validadas utilizando um algoritmo de consenso, designado de prova de trabalho, que garante que um qualquer consenso na rede é suportado por gastos computacionais suficientes, de modo a dificultar a manipulação indevida da cadeia.

No caso da *Bitcoin* os utilizadores, conhecidos como mineiros, tentam adivinhar de forma aleatória um *puzzle* criptográfico de elevada dificuldade. Uma vez resolvido, a solução e a prova de gasto computacional são apresentadas a toda a rede. No fim, a rede concorda com a solução e o vencedor recebe comissões por ter resolvido o *puzzle*. Assim, privilegia-se um utilizador que em vez de tentar adulterar a cadeia, onde terá de gastar elevados recursos computacionais, tentará validar transações pois receberá comissões que compensam os recursos computacionais gastos. Tanto a dificuldade como as comissões variam conforme a necessidade da rede.

Com esta técnica pretende-se evitar os esquemas de *double spending*, ou seja, impedir a utilização de moedas já gastas. No entanto, este protocolo apresenta uma falha de segurança, uma vez que permite o agrupamento de recursos computacionais. Isto traduz-se no facto de que se

uma determinada entidade, conhecida por *mining pool*, conseguir obter mais de 50% da capacidade computacional de toda a rede, poderá controlar a cadeia principal e, eventualmente, fazer esquemas de *double spending*. [GKW+16]

As transações são guardadas num registo público distribuído e são agrupadas em blocos consoante a sua era de criação, ou seja, a data da sua criação. Referenciam blocos anteriores através de técnicas de *hashing*, o que torna computacionalmente difícil a adulteração de transações anteriores, pois a *hash* de um bloco depende do anterior e a modificação de informação implica uma *hash* diferente. Esta cadeia de blocos é conhecida como *Blockchain*.

A *Bitcoin* é ainda pseudo anónima, pois, apesar de a identidade real dos utilizadores não ser conhecida, é utilizado um sistema de endereços que funcionam como identificadores, o que pode levar a uma identificação através da inspeção e análise dos dados públicos das transações [Mös13].

Por estas razões, a *Bitcoin* é considerada um sistema transparente e sem necessidade de confiança nos intervenientes, devido à inexistência de intermediários, e ainda, ao facto de toda a informação sobre as transações da rede estar pública e acessível a qualquer pessoa.

Atualmente, existem várias criptomoedas que são desenvolvidas e lançadas, cada uma com características diferenciadoras. Estas criptomoedas são chamadas de *altcoins*, pois a *Bitcoin* foi a sua precursora.

O protocolo sobre o qual a *Bitcoin* assenta é livre e qualquer pessoa pode contribuir para o desenvolvimento da plataforma.

2.2.6 *Ethereum*

Ao contrário da *Bitcoin*, que pertence exclusivamente ao *Blockchain 1.0* por apenas permitir transações financeiras, o *Ethereum* tanto pertence ao *Blockchain 1.0*, da mesma forma que a Bitcoin, como pertence ao *Blockchain 2.0*, tendo sido o precursor do mesmo, através da utilização de contratos sobre o *Blockchain*, permitindo mais operações financeiras para além de transações.

Deste modo, o *Ethereum* é uma alternativa ao protocolo da Bitcoin para a criação de aplicações descentralizadas. Ao contrário do protocolo da Bitcoin que é restritivo ao nível de estados do *Blockchain*, uma vez que apenas permite transações financeiras com estados e formatos pré-definidos, o *Blockchain* do *Ethereum* é uma máquina de *Turing* completa. Este permite a qualquer pessoa escrever contratos inteligentes, utilizando a linguagem *Solidity*, e criar aplicações descentralizadas definindo as próprias regras de propriedade, formato da transação e funções de estado de transação. [DWC+17]

Outra grande diferença consiste no mecanismo de consenso utilizado pois, em vez de fazer uso da prova de trabalho, faz uso da prova de participação.

A prova de participação foi proposta como alternativa à prova de trabalho e tem como vantagens uma maior segurança, menor risco de centralização e mais eficiência energética. Este algoritmo de consenso, depende da participação do mineiro na rede, isto é, na quantidade de criptomoedas que o mineiro possui. Para além da quantidade, também o intervalo de tempo em que as possui contribui para a participação na rede. [Bas17]

Tal como a Bitcoin, o *Ethereum* também tem uma criptomoeda, denominada *ether*, que é utilizada para criar contratos inteligentes e executar funções particulares. [Bas17]

2.2.7 Registo Médico Eletrónico

Segundo a norma *ISO/TR 14639*, o RME corresponde à informação relevante para a saúde e bem estar de um indivíduo, organizada segundo um modelo standard e que está estruturada num formulário passível de ser processado por um computador [fS14]. Na prática, os RMEs vieram substituir os antigos registos médicos físicos. Dois exemplos de implementações de standards de registos médicos são o *openEHR* e o *Fast Healthcare Interoperability Resources (FHIR)*.

Atualmente as instituições de saúde guardam estes RMEs nas suas bases de dados e, em alguns casos, sem a devida segurança, quer de persistência, quer de confidencialidade. Já se registaram vários incidentes de ataques informáticos a bases de dados de instituições médicas que viram os dados dos seus pacientes serem furtados e até eliminados. [AORBK17]

As características do *Blockchain* podem melhorar a utilização de RMEs: em primeiro lugar, através da descentralização, garantindo a integridade da informação assim como a sua transparência; em segundo, através da encriptação, garantindo confidencialidade e autenticidade; em terceiro lugar, permite ainda acesso sem restrições geográficas; por último, devido ao facto de a informação poder ser verificável e ainda imutável, devido aos seus mecanismos de criptografia. [Bax16]

O controlo de informação por parte do paciente é essencial, pois garante um direito em relação ao tratamento da sua própria informação e permite a partilha de informação, por exemplo, entre parentes, o que pode ajudar a prever doenças hereditárias. Outra situação, está relacionada com a possibilidade de o paciente pedir uma segunda opinião médica, que, de forma segura e descomplicada, poderá partilhar os seus RMEs de outra instituição. [AEVL16]

2.2.8 Legislação de proteção de dados na União Europeia

A Comissão Europeia propôs a reforma da proteção de dados na União Europeia em 2012, para que a Europa se pudesse adaptar melhor à era digital. Esta reforma entra em vigor em 2018. [Eur]

De acordo com estudos realizados, mais de 90% dos europeus querem o mesmo nível de proteção dos dados pessoais em toda a União, independentemente do lugar onde os dados são tratados, e ainda, segundo um inquérito Eurobarómetro, dois terços dos europeus estão apreensivos quanto ao potencial de utilização das informações fornecidas por parte das empresas. [EU2]

A proteção das pessoas singulares relativamente ao tratamento de dados pessoais é um direito fundamental. O artigo 8.o, n.o 1, da Carta dos Direitos Fundamentais da União Europeia («Carta») e o artigo 16.o, n.o 1, do Tratado sobre o Funcionamento da União Europeia estabelecem que todas as pessoas têm direito à proteção dos dados de caráter pessoal que lhes digam respeito.

Assim, entre as principais mudanças no regulamento, incluem-se o acesso mais fácil aos próprios dados e o direito de portabilidade dos mesmos, querendo isto dizer que é mais fácil transferir dados pessoais entre prestadores de serviços. Uma outra mudança do regulamento trata-se do incentivo de uso de técnicas que favorecem a privacidade, como a pseudonimização. [EU2]

2.3 Projetos e soluções *Blockchain* na saúde

Atualmente, a indústria da saúde tem sido um caso de sucesso na inovação e desenvolvimento tecnológico. Em diferentes ramos da saúde, vários métodos e técnicas são utilizadas de modo a simplificar e melhorar as soluções médicas existentes.

O *Blockchain* é uma das tecnologias que estão a ser estudadas e integradas na área da saúde, pois assegura a imutabilidade, auditabilidade e a transparência, que sistemas convencionais não conseguem. Estas características são especialmente importantes quando a indústria da saúde tem como principais problemas a privacidade, segurança e interoperabilidade dos dados médicos guardados.

Nesta secção, são apresentadas algumas plataformas que, tirando partido do *Blockchain*, tentam resolver os problemas enumerados na indústria da saúde.

2.3.1 *MedRec*

O *MedRec* é um sistema de gestão de RMEs descentralizado que utiliza a tecnologia *Blockchain* [AEVL16]. O sistema fornece um design modular de integração com bases de dados dos provedores de saúde, de modo a facilitar a interoperabilidade, conveniência e adaptabilidade. É acompanhado ainda de um sistema e interface, de modo a permitir o acesso à informação médica.

Incentivam, entre outros, investigadores, a participar na rede de *Blockchain* como mineiros, recebendo, em troca, acesso a informação médica anónima como recompensa pela mineração, de modo a sustentar e garantir a segurança da rede utilizando a prova de trabalho.

Assim, o *MedRec* tem, como principal inovação, a economia de dados, ou seja, monetização de dados anónimos.

2.3.2 *MediBchain*

O *MediBchain* é uma plataforma de gestão de dados de saúde de pacientes que usa o *Blockchain* como forma de persistência, de modo a obter privacidade [AORBK17]. O pseudo anonimato é assegurado por funções criptográficas de modo a proteger os dados do paciente.

Os dados são diretamente guardados no *Blockchain* e os pacientes têm o controlo exclusivo sobre a informação guardada em cada bloco.

Assim, o *MediBchain* tem como principal objetivo dar total controlo dos dados ao paciente, assegurando a integridade e segurança dos mesmos.

2.3.3 *OmniPHR*

O *OmniPHR* é uma plataforma que possibilita a persistência de informação, tanto por parte do prestador de serviços de saúde como dos pacientes. A arquitetura definida é algo centralizada, pois os dados dos provedores e dos pacientes estão separados em *Blockchains* diferentes. [RdCR17]

O sistema foi implementado de modo a aceitar apenas um tipo de dados, *openEHR*, e permite a persistência de dados de sensores e dispositivos móveis na rede.

O *OmniPHR* diferencia-se pela arquitetura apresentada, de separação das instituições de saúde e dos pacientes, quer a nível de persistência de dados, quer pela interface de utilização. Os pacientes têm uma interface para dispositivos móveis, enquanto que as instituições de saúde têm uma aplicação *desktop*.

2.4 Tecnologias estudadas

Após uma pesquisa sobre as diferentes plataformas existentes que utilizam *Blockchain*, filtrada com base nas diversas funcionalidades, arquiteturas, licenças de código, estado de desenvolvimento e suporte, elaborou-se uma lista, onde duas plataformas foram selecionadas como as que mais podiam contribuir para o projeto.

Nesta secção são apresentadas as duas plataformas de código aberto, das quais o *Hyperledger Fabric* (HF) foi utilizado para a implementação do protótipo proposto.

2.4.1 *Hyperledger Fabric*

O HF é uma plataforma de código aberto para o desenvolvimento de soluções de *Blockchain*. Faz parte do projeto colaborativo *Hyperledger*, mantido pela *The Linux Foundation* com o intuito de desenvolver a tecnologia do *Blockchain* aplicada em diferentes áreas e indústrias. [ABB+18]

Este apresenta uma arquitetura modular que garante confidencialidade, resiliência, flexibilidade e escalabilidade. Tal modularidade aumenta a interoperabilidade das plataformas implementadas, permitindo a utilização de diferentes tecnologias e ferramentas na plataforma.

Destaca-se por ser uma plataforma para desenvolvimento de *Blockchains* privados e com permissões. Ao contrário de *Blockchains* públicos, que permitem identidades desconhecidas e, por isso, precisam de mecanismos de consenso como a prova de trabalho, o *HF* utiliza o *Membership Service Provider* (MSP) para garantir controlo de acesso dos membros.

Um membro do HF corresponde a uma entidade singular com um certificado próprio. Os componentes da rede como o *peer* ou *client* são interligados a um membro. O *peer* corresponde ao componente responsável por manter o *ledger* e executar os *chaincodes* e pode desempenhar vários papeis como *endorsing peer* ou *anchor peer*. O *endorsing peer* tem a função de aprovar determinadas transações. Por outro lado, o *anchor peer* tem a função de informar qualquer membro de um *channel* da existência de todos os *peers*, pelo que funciona como um ponto de encontro entre diferentes membros num *channel*. O *client* representa o utilizador final e corresponde ao componente relacionado com a aplicação cliente que faz uso do *NodeJS SDK* para interagir com a plataforma.

O principal componente do HF são os *channels*. Estes simbolizam canais de comunicação restritos a determinados participantes, o que possibilita a partilha de informação privada. Existe um *ledger* por *channel* e cada *peer* pertencente ao *channel* mantém uma cópia do *ledger*. Um *peer*, pode ainda manter, concorrentemente, vários *ledgers* e, portanto, pertencer a diferentes *channels*.

O HF permite ainda a utilização de contratos inteligentes, chamados de *Chaincode*, que asseguram a lógica de negócio do sistema. Atualmente podem ser programados em *Go* ou *NodeJS*.

Esta plataforma possibilita também a atualização de praticamente todos os seus componentes dinamicamente, ou seja, sem ter de desmantelar a rede. Isto é importante pois permite que as plataformas desenvolvidas tenham sempre as funcionalidades mais recentes sem prejudicar o funcionamento e disponibilidade do sistema. [lf]

2.4.1.1 *Chaincode*

O *chaincode* é o componente HF que corresponde à implementação de um contrato inteligente. Este corre dentro de um contentor *Docker* separado do processo *peer*. O *chaincode* inicializa e gere o estado do *ledger* através de invocações feitas pelas aplicações *client*.

Para um *peer* executar um *chaincode* é necessário que este seja primeiro instalado e instanciado. A instalação consiste em indicar o caminho da definição do *chaincode* e identificá-lo com um nome e versão. A instanciação consiste em iniciar o *chaincode* num contentor e executar uma operação inicial definida no *chaincode*. Ao instanciar são ainda definidas as políticas de aprovação, ou seja, definem-se quais as entidades que podem executar operações do *chaincode*.

Um *peer* pode ter várias instâncias de diferentes *chaincodes*.

2.4.1.2 *Ledger*

O *ledger* consiste num registo, ordenado e resistente a adulterações, de todas as transações efetuadas na rede. Estas transações resultam de invocações de *chaincodes*, submetidas pelos intervenientes na rede. Cada transação consiste num registo chave/valor.

O *ledger* é constituído por um *Blockchain*, chamado de *chain*, que serve para guardar as transações em blocos imutáveis e sequenciais, e de uma base de dados com os estados atuais das transações, chamada de *state database*.

São os *peers*, pertencentes ao *channel*, que mantêm o *Chain* através da persistência dos registos sob a forma de pares chave/valor numa base de dados *LevelDB*, interna ao processo do próprio *peer*.

2.4.1.3 *Chain*

A *chain* é uma cadeia de blocos que corresponde a um registo de transações. O cabeçalho do bloco inclui um registo temporal da sua criação, uma *hash* da transação do bloco, assim como, a *hash* do cabeçalho do bloco anterior. Desta forma, todas as transações são sequenciais e criptográficamente interligadas.

Isto faz com que seja computacionalmente impossível adulterar informação e fazer com que seja aceite, pois, para alterar a informação num bloco é necessário alterar igualmente a informação dos blocos que vêm a seguir. Quantos mais blocos forem adicionados à cadeia, mas difícil se torna adulterar a informação.

Assim, a *hash* do último bloco adicionado representa todas as transações que foram anteriormente adicionadas, o que assegura que todos os *peers* estão num estado de consenso. A *chain* é guardada no sistema de ficheiros do *peer*.

2.4.1.4 Base de dados de estados

O estado mais atual do *ledger* corresponde ao último valor de todas as chaves guardadas na *chain*.

As invocações dos *chaincodes* são executadas em relação ao estado mais atual da *chain*, pelo que, de modo a tornar este processo mais eficiente, os últimos valores das chaves são guardados na base de dados de estados, chamada de *state database*.

Para o *state database* pode-se optar por duas tecnologias. A primeira consiste em utilizar a base de dados *LevelDB*, usada por defeito e que é interna ao *peer*, isto é, os dados são guardados localmente. A segunda opção consiste na base de dados *CouchDB* que é externa ao *peer*, ou seja, corre paralelamente a este num contentor *Docker*, e suporta *rich queries* estilo *MongoDB* quando o conteúdo é modelado em *JavaScript Object Notation (JSON)*.

2.4.1.5 Fluxo de transações

O fluxo de transação consiste numa proposta de transação por parte de uma aplicação cliente para um determinado grupo de *endorsing peers*. Estes *peers* verificam a assinatura do cliente e executam a respetiva operação do *chaincode* para simular a transação. O resultado consiste no par chave/valor do que foi lido pelo *chaincode* assim como no par do que foi escrito pelo mesmo. Este resultado é enviado para o cliente acompanhado de uma assinatura de aprovação dos *peers*. O *peer* agrega os resultados numa transação para o serviço de ordenação que, por sua vez, envia transações de ordenação como blocos para todos os *peers* no *channel*. Os *peers* vão verificar o par chave/valor do que foi lido de modo a assegurar a integridade da informação e proteger contra esquemas de *double spending*. O HF garante que nenhuma transação altere a informação lida por outra transação através de um sistema de controlo que funciona paralelamente em cada *peer* que recebeu a transação. Caso tenha havido modificação da informação, a transação é marcada como inválida e não é concluída. Caso não haja problemas, a transação é aceite e é efetuada na *state database*.

Este processo pode ser melhor compreendido através do esquema da Figura 2.2.

2.4.1.6 Sistema de consenso

De modo a que toda a rede HF tenha o mesmo estado do *ledger*, é necessário implementar um mecanismo de consenso entre todos os membros. O componente HF com esta função é designado de *Orderer* e, graças à arquitetura modular do HF é possível adotar um qualquer sistema de ordenação, desde que seja compatível com o mesmo [Hyp17].

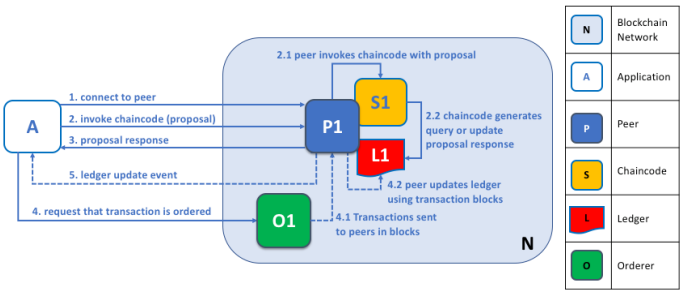

Figura 2.2: Esquema do fluxo de transações HF, retirado da documentação oficial.

O HF fornece duas implementações: *solo* e *Kafka*. O primeiro não é tolerante a falhas pelo que apenas é aconselhado em ambientes de desenvolvimento. O segundo é tolerante a falhas não bizantinas e foi implementado para ambientes de produção.

Por outro lado, o HF não disponibiliza nativamente um algoritmo de consenso tolerante a falhas bizantinas, apesar de estar previsto o seu desenvolvimento. No entanto, possibilita a utilização de um qualquer módulo que forneça esse mesmo mecanismo, como é o exemplo do algoritmo *BFTSmart* [SBV17].

2.4.1.6.1 *Apache Kafka e Zookeeper*

O *Kafka* [KNR11] comporta-se como um sistema de processamento de mensagens através do modelo *Publish/Subscribe* em que as mensagens recebidas, publicadas pelos chamados *producers* num determinado tópico, são guardadas e separadas em diferentes partições dentro de um tópico. Numa partição as mensagens são ordenadas e indexadas com um registo temporal correspondente à instância em que foram recebidas. Os subscritores, chamados de *consumers*, podem ler as mensagens contidas nas partições dos tópicos que subscreveram. Cada partição é atribuída a apenas um *consumer* para assegurar que a mensagem é lida apenas uma vez.

O *Kafka* funciona num *cluster* de vários servidores chamados de *brokers*, pelo que as partições de todos os tópicos são distribuídas pelos diferentes nós do *cluster*. Esta arquitetura torna o *Kafka* tolerante a falhas, ou seja, caso um nó falhe o sistema continua ativo através da utilização de sistemas líder/seguidor, em que quando um nó falha, o seguidor torna-se no novo líder. Os *consumers* têm de tomar conhecimento desta nova mudança e, para isso, a coordenação dos *consumers* é conseguida através da utilização do *Zookeeper* que consiste num sistema para fornecer serviços de distribuição de informação de configuração e sincronização de serviços. O *Zookeeper* é igualmente um sistema tolerante a falhas.

Na Figura 2.3 é apresentada a arquitetura da plataforma *Apache Kafka*.

2.4.1.6.2 *Kafka* aplicado no HF

No HF o consenso é conseguido através da utilização de serviços de ordenação distribuídos em conjuntos de nós. Estes nós utilizam o *Kafka* de forma a tornar o serviço de ordenação tolerante a

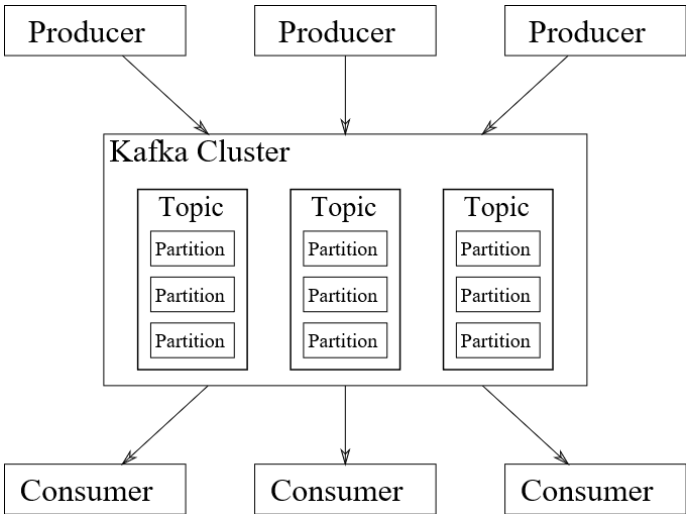

Figura 2.3: Arquitetura da plataforma Apache Kafka.

falhas.

Cada *channel* é mapeado para um tópico na rede Kafka. Quando um nó do sistema de ordenação recebe transações via *broadcast Remote Procedure Call* (RPC), verifica as permissões de escrita do cliente no *channel* e, caso se verifiquem, redireciona as transações para a partição correta no *Kafka*. Esta partição é ainda consumida pelos nós do serviço de ordenação, que agrupam as transações recebidas em blocos. De seguida, enviam as transações para os devidos clientes *deliver RPC*.

2.4.1.7 *Fabric Certificate Authority*

De modo a identificar todos os intervenientes na plataforma, para garantir a autenticidade da entidade e integridade das operações, o HF disponibiliza uma *Application Programming Interface* (API) especificamente desenvolvida para utilização na plataforma, que funciona como uma autoridade certificadora: *Fabric Certificate Authority (Fabric CA)*.

Esta permite o registo de entidades e a emissão, renovação ou revogação de certificados standard X.509.

O *Fabric CA* consiste tanto num componente servidor como cliente e todas as comunicações entre estes e o HF utilizam o protocolo TLS.

2.4.1.8 *Membership Service Provider*

O MSP consiste num componente para gestão de membros que participam numa rede HF. Este abstrai os mecanismos criptográficos e os protocolos, tanto para a emissão e validação de certificados como para a autenticação dos utilizadores, e define ainda as identidades, como por exemplo *peer* e *client*, e as regras de como as validar e autenticar. Qualquer membro da rede HF

tem de ser identificado por um MSP, de modo a poder ser validado e autenticado e assim poder interagir com a rede.

O MSP possui um nome e um identificador único e a sua configuração tem de ser especificada localmente em cada *peer*, *orderer* e *channel*. No caso dos *peers* e dos *orderers*, serve para habilitar a assinatura dos mesmos. No caso do *channel*, serve para habilitar a validação de identidades *client*, *orderer* e *peer* e das respetivas assinaturas por todos os membros do *channel*.

2.4.2 *Ethermint*

O *Ethermint* é uma plataforma, reconhecida pela sua fiabilidade e performance, que estende o *Ethereum* e permite executar aplicações feitas para o mesmo.

Assim como o *Ethereum*, o *Ethermint* considera um sistema de consenso baseado na prova de trabalho, mas utiliza o algoritmo *Tendermint core*, o que melhora o desempenho, quer ao nível do processamento de transações, quer ao nível da utilização de contratos inteligentes [AIB].

O *Tendermint* é tolerante a falhas bizantinas e assegura o normal funcionamento da rede, incluindo comportamentos maliciosos dos participantes, mesmo que um terço da rede falhe arbitrariamente.

Esta plataforma é constituída por dois componentes: o algoritmo de consenso para *Blockchain* e a interface gráfica. O motor de consenso assegura que cada transação seja replicada por todos os nós e pela mesma ordem. A interface gráfica, chamada de *Application Blockchain Interface*, possibilita a criação de contratos inteligentes em várias linguagens de programação.

Capítulo 3

Requisitos e arquitetura

O protótipo desenvolvido chama-se *gChain* e consiste numa prova de conceito de uma plataforma *web* para a gestão e partilha de informação médica, sob a forma de RMEs, e que utiliza a tecnologia *Blockchain* na sua arquitetura interna.

Existem dois tipos de utilizadores nesta plataforma: instituições de saúde e pacientes. A *gChain* é constituída por duas aplicações *web*, uma por cada tipo de utilizador, que faz uso de uma API que permite interagir e executar ações através do acesso às diferentes bases de dados da plataforma.

Neste capítulo, estão expostas as funcionalidades e requisitos, a arquitetura e as metodologias utilizadas para a implementação da plataforma *gChain*.

3.1 Funcionalidades e requisitos

Nesta secção são apresentadas as funcionalidades e os requisitos da plataforma desenvolvida e dos seus diferentes módulos.

3.1.1 Aplicação para pacientes

Esta aplicação *web* foi desenvolvida para os pacientes. É constituída por quatro páginas diferentes: página de registo da figura E.1, página de *login* da figura E.2, página para visualização dos registos clínicos disponíveis da figura E.3 e página para gestão de permissões da figura E.4.

A página de registo é o primeiro contacto do paciente com a plataforma *gChain* e é através desta página que o paciente se regista na plataforma. A página consiste num formulário para o preenchimento da informação necessária, como primeiro e último nome, *email* e *password* de acesso, para a utilização da plataforma. O paciente pode ainda adicionar uma imagem de perfil.

Após o registo, o utilizador é redirecionado para a página de *login*, onde é necessário utilizar as credenciais de acesso definidas anteriormente, que consistem no par *email/password*.

Cumpridas as regras necessárias para o *login* na plataforma, o paciente é redirecionado para a página de visualização dos registos clínicos. Nesta página o paciente pode ver os seus dados por instituição, adicionar uma nova instituição à sua conta *gChain*, eliminar os seus registos e ainda ter

acesso à página de gestão de permissões. Nesta última página, o paciente pode gerir as permissões de acesso aos seus registos, ou seja, pode fornecer ou remover permissões de partilha de dados originados numa instituição com uma outra.

Na aplicação é ainda possível aceitar ou recusar o *upload* dos seus registos clínicos na plataforma *gChain* por parte de uma instituição de saúde.

Como utilizador não registado, o paciente pode:

- registar-se na plataforma com os seus dados pessoais.

Como utilizador não autenticado, o paciente pode:

- autenticar-se com as credenciais de acesso.

Como utilizador autenticado, o paciente pode:

- visualizar, em forma de lista e por instituição, os seus registos clínicos pessoais das várias instituições de saúde;
- marcar instituições de saúde como entidades seguras a serem utilizadas na plataforma;
- gerir permissões de acesso aos seus registos clínicos, ou seja, fornecer ou remover autorizações de acesso de uma instituição a registos noutras instituições;
- aceitar ou recusar registos de dados pelas instituições na plataforma;
- eliminar os seus registos da *gChain*.

3.1.2 Aplicação para instituições de saúde

Consiste numa aplicação *web* para as instituições de saúde. É constituída por quatro páginas: página de registo da figura E.5, página de *login* da figura E.6, página de gestão da figura E.7 e página de visualização de registos clínicos da figura E.8.

A página de registo serve para registar uma instituição na plataforma. Para isso é necessário preencher um formulário com os dados da instituição e ainda um par utilizador/*password* para a autenticação na plataforma. Registada na plataforma, a instituição tem de se autenticar com as credenciais de acesso através da página de *login*. Após o *login*, a instituição é redirecionada para a página principal de gestão. Nesta página, a instituição pode criar as associações paciente/instituição, ter acesso à lista de utilizadores, registados e associados à instituição na plataforma, e ainda fazer *upload* de registos clínicos de pacientes.

Através da seleção de um paciente, a instituição é redirecionada para a página de visualização de registos clínicos, onde pode ter acesso aos registos clínicos já adicionados e ainda a registos de outras instituições de saúde.

Como utilizador não registado, a instituição pode:

- registar-se na plataforma com os dados da instituição.

Como utilizador não autenticado, a instituição pode:

- autenticar-se com os dados de registo da plataforma.

Como utilizador autenticado, a instituição pode:

- associar um paciente à instituição;

- visualizar, em forma de lista, os pacientes associados;
- visualizar, em forma de lista, os registos clínicos dos pacientes da sua instituição ou de outras, caso tenha permissões dadas pelo paciente;
- fazer *upload* de registos clínicos de pacientes, caso tenha permissões.

3.1.3 Rede *Blockchain*

Esta rede Blockchain foi implementada através da plataforma HF.

Para a rede *Blockchain* foram definidos os seguintes requisitos:

- existência de quatro instituições de saúde;
- um *peer* por instituição;
- um *peer* central, para pacientes;
- um CA para os pacientes;
- um CA por cada instituição de saúde;
- sistema de ordenação tolerante a falhas;
- utilização de *Transport Layer Security* (TLS) para as comunicações na rede e para as aplicações *web*;
- implementação de um *chaincode* para controlo de acesso aos dados.

Com estes requisitos pretende-se simular o funcionamento da plataforma de forma a poder verificar a sua escalabilidade, performance e aplicabilidade relativamente à tecnologia *Blockchain*.

3.1.4 Application Programming Interface

A API consiste num conjunto de métodos que permitem a validação das operações e o acesso às bases de dados da plataforma. Esta corresponde ao módulo central da plataforma, uma vez que qualquer operação passa obrigatoriamente pela API.

Para a API da plataforma, foram definidos os seguintes requisitos:

- validação das instituições para fazer *upload* de registos clínicos;
- validação das instituições para acesso aos registos clínicos;
- validação dos pacientes para acesso aos registos clínicos;
- validação do paciente para a adição ou alteração de permissões de acesso;
- registo de associações paciente/instituição;
- encriptação dos registos clínicos;
- registo dos utilizadores da plataforma;
- autenticação dos utilizadores da plataforma.

3.1.5 Plataforma *gChain*

Relativamente à plataforma *gChain*, foi definido como requisito estar em cumprimento com a nova legislação de proteção de dados na União Europeia.

Para isso, qualquer ação na plataforma que envolva os dados do paciente implica a sua aprovação. Isto inclui a aprovação do *upload* dos registos clínicos na plataforma, assim como as permissões de acesso aos mesmos.

A plataforma tem ainda de apresentar uma arquitetura que faça uso de técnicas que privilegiem a segurança da plataforma e dos seus utilizadores, bem como a pseudonimização relativamente às operações efetuadas e aos dados guardados na plataforma.

3.2 Arquitetura

A plataforma *gChain* é constituída por vários módulos: interface gráfica, base de dados *Blockchain* com autorizações de acesso, base de dados com mapeamento de ids, base de dados de registos clínicos e a API para exposição de serviços e interligação de módulos. De um modo geral, o utilizador através da interface gráfica tem acesso à API desenvolvida, de forma a poder utilizar a plataforma e, por conseguinte, aceder à informação contida nas diversas bases de dados com os registos clínicos.

Na Figura 3.1 está ilustrado o diagrama de arquitetura da plataforma *gChain*.

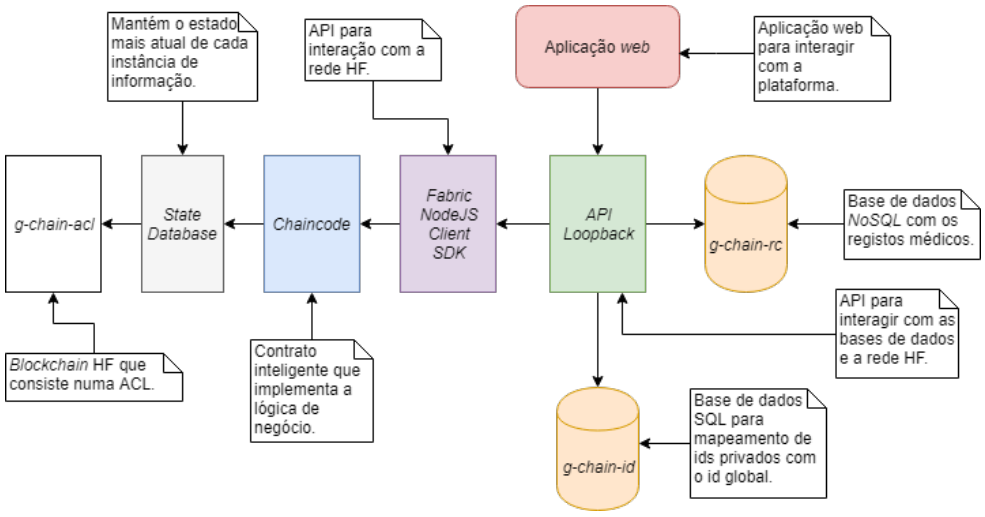

Figura 3.1: Diagrama de arquitetura da plataforma gChain.

Na arquitetura da plataforma *gChain*, os registos clínicos são guardados numa base de dados *NoSQL*, tratada internamente como *g-chain-rc*. Para aceder à *g-chain-rc* é necessário ter acesso à API da plataforma *gChain*, estar devidamente autenticado e cumprir com as permissões de acesso aos dados. No caso das instituições de saúde, as suas permissões de acesso a dados clínicos dependem dos pacientes, uma vez que são estes quem gere as permissões de acesso aos seus dados.

Estas permissões são mantidas no *Blockchain* HF, tratado internamente como *g-chain-acl*. Por outro lado, o paciente tem acesso aos seus registos clínicos assim que for criada a associação entre o mesmo e a instituição de saúde e esta adicione os dados do paciente na *g-chain-rc*. Entre o módulo *g-chain-acl* e o *g-chain-rc* existe outro, tratado internamente como *g-chain-id*, que mantém o registo dos pacientes, das instituições, as respetivas associações paciente/instituição e ainda a associação paciente/*token*. O *g-chain-id* tem como principal funcionalidade permitir o mapeamento entre os dados que estão na *g-chain-acl* e os dados na *g-chain-rc*. Este processo, assim como todos os módulos, está explicado detalhadamente nas secções seguintes.

3.2.1 Interface gráfica

A interface gráfica subdivide-se em interface para instituições de saúde e para pacientes e corresponde à camada mais alta da plataforma, servindo como porta de acesso aos serviços disponibilizados globalmente pela plataforma.

A interface gráfica foi desenvolvida em *React* e utiliza a API da plataforma para chamar métodos e serviços.

3.2.2 Application Programming Interface

Para a implementação da API foi utilizada a *framework NodeJs* de APIs *Loopback*, desenvolvida e mantida pela *Strongloop*. Esta *framework* utiliza *models* para a representação de origens de dados *backend* como bases de dados ou serviços *Representational State Transfer* (REST) [loo]. Foram criados três *models* em que cada um representa o acesso às bases de dados e serviços implementados: *g-chain-acl*, *g-chain-id* e *g-chain-rc*.

3.2.3 *g-chain-acl*

O *g-chain-acl* é o módulo que corresponde à implementação da rede *Blockchain* HF. Para interagir com este módulo são precisos três outros: *state database*, *chaincode* e HF *NodeJS Client Software development kit (SDK)*. Estes funcionam como uma dependência sequencial, desde a API até ao próprio registo *Blockchain*. Cada registo existente na *g-chain-acl* corresponde a um objeto *JSON*, necessariamente identificado em relação ao paciente, à instituição e correspondente estado da autorização. Isto traduz-se no par chave/valor, em que a chave corresponde ao id da relação entre o paciente e a instituição e o valor corresponde ao *JSON* com a lista de permissões. A listagem 3.1 corresponde à estrutura *JSON* de um registo de permissões e a listagem 3.2 a um exemplo do mesmo.

Inicialmente, a arquitetura consistia num registo entre o id global do paciente e a lista de permissões, em que estava explicitamente descrito que uma instituição tinha acesso a outra. No entanto, devido ao problema de privacidade relativamente à identificação do paciente em relação às instituições pelo seu id, esta informação teria de ser encriptada.

Optou-se por não encriptar os dados na *g-chain-acl*. Esta decisão prende-se com um possível problema do *Blockchain*, a longo prazo, em que o método de criptografia usado para encriptar a

informação se torna obsoleto, ou seja, deixa de ser criptograficamente seguro. Essencialmente, o problema surge uma vez que os dados no *Blockchain* não podem ser alterados, ou seja, não é possível eliminar ou modificar registos e estados anteriores. Neste caso, toda a informação mais antiga, que tenha sido adicionada antes da atualização da rede, está comprometida. A única solução seria eliminar toda a cadeia e voltar a criar uma nova com os dados encriptados com o novo método criptográfico seguro.

Obviamente, esta solução é pouco aceitável, por motivos de exigência computacional e temporal no processo descrito, pecando ainda pela falta de eficiência, na medida em que se tinha que garantir o processo para toda a rede.

Assim, de forma a ultrapassar este possível problema, torna-se imperativa a anonimização da informação guardada para evitar a necessidade de encriptação. De modo a fazer valer esta funcionalidade, existe ainda uma base de dados com o principal objetivo de fazer mapeamentos de ids: *g-chain-id*.

```
[{
    "access": "institution_id",
    "autorization": true
  },
  {
    "access": "institution_id",
    "autorization": false
  }
]
```

Listagem 3.1: Estrutura JSON da permissão.

```
[{
    "access": "04c62230-5f7b-11e8-9
        ebe-190abb0d1c5c",
    "autorization": true
  },
  {
    "access": "05c11g30-5f7b-13e2-1
        ebe-180ahh0r1c3j",
    "autorization": false
  }
]
```

Listagem 3.2: Exemplo de permissão.

3.2.4 *g-chain-id*

O *g-chain-id* corresponde a uma base de dados relacional *MySQL*. Esta é constituída por quatro tabelas: *patient*, *organization*, *patient_organization* e *token*.

A primeira contém a informação dos pacientes; a segunda contém a informação das instituições de saúde; a terceira corresponde à associação entre as duas anteriores; e, a última consiste na associação entre o paciente e o *token* gerado para a associação paciente/instituição. Na Figura 3.2, é possível ver o modelo relacional desta base de dados.

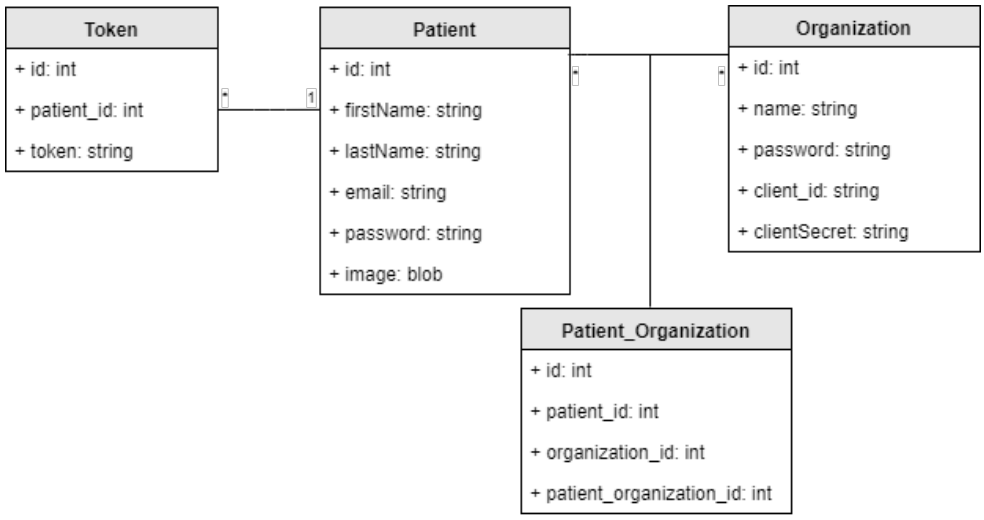

Figura 3.2: Modelo relacional da base de dados *g-chain-id*.

A tabela de associação paciente/instituição é crítica para o funcionamento da plataforma, pois referencia tanto o id do paciente como o id da instituição e ainda um id que representa a associação entre eles. Este id de associação é usado como um id privado pelo paciente relativamente às instituições. É ainda usado para identificar o registo na *g-chain-acl* em detrimento do id global identificativo do paciente. Isto é necessário para garantir a privacidade de identidade. É importante referir que o id de associação é encriptado pela a API antes de ser inserido na base de dados, de modo a não ser possível interligar o id de associação com o paciente e, portanto, com as permissões na *g-chain-acl*.

Na verdade, o paciente é identificado por dois ids: um global para a plataforma e outro por cada relação existente entre ele e uma instituição de saúde. Desta forma, nenhum outro paciente ou instituição, que tenha acesso à *g-chain-acl*, consegue interligar a informação a alguma entidade, ou seja, é impossível relacionar um paciente a diferentes instituições de saúde, pois, aos olhos das instituições, apenas conseguem identificar os ids dos pacientes que foram associados entre elas e a plataforma *gChain* e, aos olhos dos pacientes, apenas conseguem identificar os seus ids privados nas diferentes instituições associadas.

Esta arquitetura pretende solucionar um problema, que é facilmente visualizado, através de um exemplo que envolve uma instituição ou paciente afeto a uma seguradora e outro qualquer paciente, em que ambos têm acesso à *g-chain-acl*. Admitindo que o paciente tem registos numa instituição oncológica, através da leitura dos dados, a entidade asseguradora conseguiria inferir que aquele paciente ia àquela instituição, pois conseguiria ver a autorização de partilha de informação para a instituição, uma vez que o id de registo para a seguradora e para as outras instituições era o mesmo, e daí, alterar o seu comportamento em relação ao paciente, ou seja, não o assegurar.

A tabela de associação paciente/*token* tem o intuito de fazer o mapeamento entre *tokens* e os pacientes. Esta associação corresponde ao par id global do paciente e o *token* gerado pela API a pedido do paciente. Este *token* serve para fornecer a uma instituição de saúde de modo a proceder

ao estabelecimento da associação paciente/instituição. Uma vez com o *token*, a instituição através da API, cria a associação e recebe um id privado do paciente que corresponde à identificação do paciente para o hospital relativamente à plataforma *gChain*.

3.2.5 g-chain-rc

O *g-chain-rc* corresponde à base de dados com os registos clínicos da plataforma *gChain*. Esta base de dados não relacional foi desenvolvida em *MongoDB* e é constituída por três *collections*: *Patient*, *Prescription* e *Diagnosis*.

Cada registo de qualquer *collection* corresponde a um objeto JSON FHIR encapsulado por um cabeçalho mostrado na listagem 3.3, que contém o id global do paciente correspondente e da instituição que originou o registo. A parte do registo correspondente ao objeto *FHIR* é encriptada no momento da adição à *g-chain-rc* pela API. Na listagem 3.4 está representada a estrutura do objeto FHIR, enquanto que na listagem 3.5 é possível ver um exemplo de um registo.

Optou-se pela utilização do *standard FHIR* por ser mais utilizado na indústria, em comparação ao *openEHR*, e por ser uma evolução do *HL7*, que é um standard amplamente utilizado nos sistemas *Glintt*.

Optou-se por guardar os registos clínicos numa base de dados *NoSQL*, em detrimento de uma base de dados *Blockchain*, de forma a evitar, essencialmente, um problema de escalabilidade, pois a persistência de elevadas quantidades de dados num *Blockchain* é custosa ao nível de armazenamento pois, sempre que há uma alteração de um registo clínico, o registo inicial mantém-se e um novo é adicionado. Isto acontece sempre que há alterações, portanto, em vez de haver apenas uma instância do registo, existem várias instâncias, sendo que apenas a última é considerada a correta e as demais tratadas como histórico de informação. Para além disso, toda a informação teria ainda de ser replicada por todos os participantes na rede.

```
1 {
2   "globalId": "patient_id",
3   "sourceId": "institution_id",
4   "recordObject":
        clinicalRecordObject
5 }
```

Listagem 3.3: Estrutura do cabeçalho para encapsular registos clínicos.

```
{
  "resourceType" : "[Resource Type]",
  "property1" : "<[primitive]>",
  "property2" : { [Data Type] },
  "property3" : {
    "propertyA" : { CodeableConcept
        },
  },
  "property4" : [{
    "propertyB" : { Reference(
        ResourceType) }
  }]
}
```

Listagem 3.4: Estrutura *FHIR* retirada da documentação oficial.

```
{
  "resourceType": "Patient",
  "identifier": "04c62230-5f7b-11e8-9
      ebe-190abb0d1c5c",
  "name" : "patient1",
  "deceasedBoolean": true,
  "gender" : "male"
}
```

Listagem 3.5: Exemplo de um possível registo clínico.

3.2.6 *Hyperledger Fabric NodeJS Software Development Kit*

Esta API subdivide-se em dois módulos principais: *fabric-client* e *fabric-client-ca*.

O *fabric-client* corresponde à instância e respetivos métodos para interagir com as redes *Blockchain* HF existentes ou para estabelecer novas. O *fabric-client-ca* corresponde à instância e respetivos métodos para interação com os servidores *fabric-ca*.

Para além destes dois módulos, esta API apresenta ainda um módulo secundário de utilização facultativa, mas pertinente para criar novas funcionalidades à plataforma HF. As classes existentes são a *CryptoSuite*, *Key* e *KeyValueStore*. A classe *CryptoSuite* fornece um conjunto de algoritmos criptográficos para assinaturas digitais e operações de encriptação/desencriptação e *hashing*. As chaves utilizadas na plataforma são assimétricas, baseadas no algoritmo de assinatura digital de curvas elípticas. A classe *KeyValueStore* é usada para guardar as chaves e os certificados. A classe *Key* fornece métodos para gerar e interagir com chaves criptográficas.

Todos estes módulos foram utilizados na implementação da rede *Blockchain* HF.

3.2.7 *Chaincode*

Este é utilizado de modo a aceder à *g-chain-acl* e possibilitar a leitura e escrita da mesma.

O *chaincode* corresponde a um módulo de programação na linguagem *Go* que tem de ser instalado nos *peers* intervenientes na rede HF, de modo a capacitar a interação com a rede para além da persistência da informação localmente.

O *chaincode* desenvolvido para a plataforma *gChain* permite essencialmente duas operações: *invoke* e *query*.

A operação *invoke* traduz-se numa nova transação para a rede. É necessário passar um par chave/valor, de modo a adicionar a transação num bloco a ser incluído no *Blockchain*.

A operação *query* corresponde à leitura de informação do *Blockchain*. É necessário passar a chave do registo a que se quer ter acesso, de modo a ser retornado o valor deste. Nos *chaincodes* implementados existe ainda um método que permite fazer *rich queries* estilo *MongoDB*, ou seja, pesquisar por combinações ou regras para além da chave.

Capítulo 4

Realização do projeto

4.1 *Hyperledger Fabric* e abordagem ao problema

No decorrer da Unidade Curricular de Preparação para a Dissertação, foi elaborada uma lista de plataformas de desenvolvimento em *Blockchain*, de forma a chegar a uma conclusão sobre qual seria a mais proveitosa para a implementação da plataforma *gChain*. Esta lista foi elaborada tendo em conta os requisitos necessários previamente delineados para a *gChain*, assim como, as características das próprias plataformas de desenvolvimento.

Concluiu-se que o HF era a melhor opção pelas seguintes razões:

1. apresenta uma arquitetura adequada ao problema, na medida em que fornece os atores e funcionalidades necessárias para cumprir as regras de negócio. Isto é possível devido à existência de papeis como *peer*, *client*, a estruturação em *channels* e, por fim, à existência de *chaincodes*;

2. utiliza tecnologias e metodologias *standard*, amplamente utilizadas e reconhecidas como boas práticas. O principal exemplo está nos algoritmos criptográficos utilizados pela plataforma;

3. tem uma boa documentação, manuais e tutoriais úteis ao desenvolvimento;

4. possui diversos meios de informação, como repositórios e *chats* que possibilitam uma aproximação entre desenvolvedores;

5. possui uma comunidade em crescimento e interessada, pronta a aconselhar e ajudar;

6. é uma plataforma gratuita, de código aberto, com um nível de atividade elevado, incluindo planeamento alargado de desenvolvimento de novas funcionalidades, versões e correções de erros.

7. é uma plataforma com provas dadas, pois estão disponíveis inúmeros projetos desenvolvidos, e em desenvolvimento, que atestam a usabilidade da plataforma.

8. uma vez que foi focada a nível empresarial contém todas as caraterísticas necessárias, quer a nível de negócio, quer a nível de segurança e performance.

9. é modular, pois, para além das ferramentas disponibilizadas na plataforma, o HF permite a utilização de outras tecnologias de modo a desempenhar diversas funcionalidades.

Para a implementação do *Blockchain* HF foi utilizada a versão 1.1 do HF por ser a versão estável mais recente, já com suporte e várias novas funcionalidades.

A rede HF é constituída por apenas um *channel* em que os intervenientes são os pacientes e as instituições de saúde. Desta forma, existe apenas um *Blockchain* que está replicado em cada *peer* da rede. Este *Blockchain* contém o registo de permissões de partilha de informação médica, por isso, é caracterizado como uma ACL. Uma vez que a cadeia é imutável, funciona como um histórico de registos de permissões.

Esta arquitetura de ACL em *Blockchain* apresenta quatro vantagens importantes, descriminadas em baixo.

A primeira está relacionada com o facto do *Blockchain* se tornar um histórico de alterações, o que pode ser útil para processos de consultadoria, monitorização ou integração com a própria plataforma.

A segunda vantagem relaciona-se com a integridade da informação, pois numa base de dados tradicional, um qualquer utilizador legítimo ou não com acesso à mesma, pode alterar com facilidade a informação guardada. No *Blockchain*, pelo contrário, é necessário ter acesso a certificados e respetivas chaves de entidades válidas, de modo a ser validado pelo sistema de ordenação. Mesmo que haja uma alteração indevida, é sempre possível ver os estados anteriores. Para além disso, não é possível alterar estados anteriores sem criar novos, uma vez que os blocos mais recentes referenciam os mais antigos.

A terceira consiste no facto da informação na cadeia ser ininteligível, devido à utilização de técnicas de anonimização, pelo que apenas os visados conseguem obter a informação importante, apesar da cadeia estar disponível para todos os participantes.

Por fim, a quarta vantagem trata-se do facto de garantir segurança, pois caso um *peer* perca a informação, por exemplo por falha, a informação da rede continua guardada e replicada por todos os outros intervenientes na rede. O mesmo se passa no serviço de ordenação que é igualmente tolerante a falhas, pelo que, até um determinado número de nós que falham, a integridade da informação está assegurada.

4.1.1 Arquitetura interna *Hyperledger Fabric*

A arquitetura da rede HF implementada consiste num *channel* que persiste permissões de acesso aos registos clínicos.

Este *channel* é replicado em todos os *peers* pertencentes à plataforma. Na plataforma existe um *peer* central e ainda um *peer* por cada instituição de saúde participante. Foi adotada esta

arquitetura de modo a descentralizar a informação em cada instituição e garantir a disponibilidade e consistência da informação. Cada *peer* tem ainda acesso a uma base de dados de estados *CouchDB*, pelo que existem nove instâncias desta.

Existem nove organizações, oito representam instituições de saúde enquanto que uma representa os pacientes.

Por predefinição, cada instituição de saúde comunica com o *peer* designado para a instituição e os pacientes comunicam com o *peer* central. Com esta abordagem pretende-se remover qualquer requisito obrigatório de *hardware* do paciente, uma vez que o esforço computacional e de armazenamento está distribuído entre a plataforma e as instituições.

Como requisito, a plataforma tinha de garantir a coexistência de quatro instituições de saúde, por isso, foram utilizadas oito de modo a testar a escalabilidade da plataforma.

Apenas os pacientes podem fazer transações para o *Blockchain*, que se traduz na adição ou atualização de permissões. O estado do *Blockchain* é assegurado pelo *orderer* da rede. Numa rede HF podem coexistir vários nós *orderers*, nesta implementação foram utilizados apenas dois, devido à restrição de recursos. Estes fazem ainda uso de quatro *brokers Kafka* e três nós *zookeeper*.

De forma a registar as entidades, é utilizado o *Fabric CA*. Existem nove instâncias desta, uma por cada membro da plataforma, ou seja, oito relativas às instituições e uma relativa aos pacientes.

Foi implementado apenas um *chaincode* que é instanciado por cada *peer* pertencente à rede, havendo, assim, nove instâncias.

Por fim, existe um *cli container* que tem como objetivo gerar os artefactos da rede e fazer as inicializações necessárias, como a criação do *channel*.

Assim, relativamente aos artefactos que compõem a rede HF, existem:

- nove organizações;
- dois *orderers*;
- quatro *brokers Kafka*;
- três instâncias *zookeeper*;
- nove CAs;
- nove *peers*;
- nove instâncias *chaincode*;
- nove instâncias *CouchDB*;
- um *cli container*;

4.1.1.1 *Orderer*

O serviço de ordenação, designado pelo HF como *Orderer*, utilizado na *gChain* foi o *Kafka*. A escolha recaiu neste por ser o algoritmo de consenso aconselhado numa rede HF em ambiente de produção uma vez que é tolerante a falhas e apresenta elevada performance e fiabilidade.

De modo a garantir tolerância a falhas o *Kafka* precisa, no mínimo de quatro nós para implementar o *cluster*, ou seja, com quatro *brokers* é possível que um deles falhe e que a rede continue

a funcionar corretamente. Relativamente ao *zookeeper* é necessário um número ímpar de nós para evitar situações de *split-brain*. De forma a garantir a tolerância a falhas é necessário haver mais do que um nó. Optou-se, então, por implementar três nós *zookeeper* [hfk].

O serviço de ordenação *solo* foi descartado uma vez que não é tolerante a falhas, por apresentar apenas um nó constituinte.

Já o *BFTSmart* não foi utilizado por necessitar de um maior tempo de desenvolvimento para uma integração total com a solução implementada, uma vez que apresenta algumas limitações em relação ao *Kafka*, como por exemplo, a utilização do módulo *NodeJS Client SDK* não ser suportada.

4.1.2 Módulos e tecnologias internas ao *Hyperledger Fabric*

4.1.2.1 *Docker*

O *Docker* permite a virtualização, ao nível do sistema operativo, através do isolamento de recursos em forma de *containers*. Estes funcionam como uma máquina independente e podem correr em qualquer plataforma desde que tenha o *Docker* instalado.

Através do *Docker compose* é possível lançar vários *containers* ao mesmo tempo e definir possíveis dependências e outras configurações relativas a cada *container*.

Apesar de ser facultativo, a utilização de *Docker* no HF é fundamental por permitir minimizar o tempo de desenvolvimento, pois basta fazer *download* das imagens *Docker* pré-configuradas relativas a artefactos da rede e usá-las para configurar *containers*. Para interagir com estes é possível executar comandos previamente definidos na linha de comandos.

O *Docker* permite ainda, com facilidade, manter e garantir versões quer da rede quer de artefactos através da manutenção de versões de imagens *Docker*. Outra vantagem está na possibilidade de simulação de ambientes de produção, uma vez que um *container* funciona como uma máquina independente, mas identificável e acessível.

4.1.2.2 *Fabric Certificate Authority*

Tal como é referido na documentação oficial do HF, a utilização de um CA para cada organização simula o que acontece na realidade, onde cada instituição poderá optar por um qualquer CA para gerar os seus certificados.

Para cada organização existe um CA que consiste num servidor que utiliza bases de dados *SQLite* de modo a guardar as identidades, certificados e respetivas chaves públicas.

Relativamente ao cliente, para interagir com o servidor *Fabric CA* é utilizado o HF *NodeJS client SDK*. Primeiramente é necessário registar uma entidade inicial com o papel de administrador e, de seguida, é possível registar novas entidades com o papel de *client* para poderem usufruir do certificado e chave de assinatura. Estes são gerados a partir de ficheiros de configuração do servidor que contém uma secção chamada de *Certificate Signing Request (CSR)*.

4.1.2.3 *Membership Service Provider* e políticas

Ao instanciar o *chaincode* instalado nos *peers*, é possível definir políticas de aprovação que restringem a execução do mesmo relativamente a membros da rede.

Neste caso, foram definidas políticas para limitar a execução da operação *invoke* a membros do MSP relativo aos pacientes. Ou seja, a operação *invoke* pode ser executada tanto por entidades *client* como *peer*, pertencentes ao MSP dos pacientes. Pelo contrário, relativamente às instituições de saúde, apenas a entidade *peer* pode executar a operação *invoke*. Assim, é possível controlar o acesso mantendo o estado do *channel* atualizado.

Por outro lado, a operação *query* está disponível para qualquer membro MSP da rede.

4.1.2.4 Ferramentas para gerar artefactos

De forma a iniciar uma rede HF é preciso gerar os artefactos necessários para estabelecer os *channels*, assim como, o devido material criptográfico para identificar as entidades participantes e componentes. O HF fornece duas ferramentas que facilitam a criação destes artefactos, como o *Configtxgen* e o *Cryptogen*.

O *Configtxgen* serve para gerar os artefactos necessários para a criação de um *channel*. O *Cryptogen* serve para gerar o material criptográfico para uso na rede, como certificados e chaves privadas. Tanto um como o outro geram os artefactos a partir de ficheiros de configuração.

Para além destas ferramentas, o HF disponibiliza ainda o *Configtxlator* que possibilita a reconfiguração de redes já estabelecidas. Esta ferramenta é especialmente importante, pois permite a adição de novos participantes na rede, dinamicamente e sem interrupções. É desta forma que são adicionadas novas instituições de saúde à plataforma *gChain*.

4.1.3 Integração e interoperabilidade

A plataforma *gChain* foi desenhada para ser uma plataforma de integração com outras plataformas e serviços *Glintt*. Assim, a ideia passou por criar uma associação entre o paciente e a instituição relativamente a esta plataforma. Isto foi conseguido através da solução descrita que consiste em gerar, por cada paciente, um id relativo à associação instituição/paciente para a *gChain*.

É expectável que uma determinada instituição de saúde já tenha o registo e outros dados dos seus pacientes, pelo que apenas é necessário fazer o mapeamento da informação já guardada e o id gerado para utilização dos serviços da plataforma.

4.1.4 Segurança e integridade

Para além das regras e políticas definidas na rede HF que asseguram a restrição de acesso e operações à rede HF do módulo *g-chain-acl*, também houve necessidade de restringir o acesso aos módulos *g-chain-id* e *g-chain-rc*, acedidos a partir da API.

Para isso, foi implementado um servidor *oAuth* 2.0 que é um protocolo standard para autorizações. Este consiste num servidor à parte da plataforma *gChain* mas apenas utilizada por esta.

Sempre que um utilizador, quer paciente quer organização, faz *login*, é feito um pedido ao servidor *oAuth* para gerar um *token* de sessão para o utilizador. O utilizador ao fazer um pedido à API passa o *token* e esta, através do servidor *oAuth*, verifica a validade do *token* e, portanto, do utilizador. Sem um *token* válido não é possível fazer pedidos à API.

No caso de o utilizador ser um membro de uma instituição de saúde é preciso ainda passar outros dois parâmetros: *client_id* e *clientSecret*. Este processo consiste numa boa prática implementada pelo *oAuth*, de modo a assegurar que um determinado utilizador se está a autenticar a partir de uma aplicação relativa a uma instituição de saúde especifica.

O servidor *oAuth* tem acesso à base de dados *g-chain-id*, de forma a verificar a existência do paciente. Os *tokens* gerados são do tipo *JSON Web Token*. Estes têm um tempo de expiração e é necessária uma chave secreta, tanto para gerar como para verificar a validade do *token*. Esta chave secreta é restrita ao servidor *oAuth* da plataforma.

A segurança da comunicação entre os diferentes artefactos da rede *Blockchain*, como os *peers*, CAs, *orderers* e aplicações cliente, é assegurada pela utilização do protocolo de segurança TLS. Esta funcionalidade está nativamente disponível na plataforma HF, em que é apenas necessário configurar os respetivos certificados e chaves.

Tanto para a encriptação dos ids de associação como para os registos clínicos, foi utilizado o módulo *Crypto*, disponibilizado em *NodeJS*, de modo a implementar o algoritmo *Advanced Encryption Standard* com 256 *bits*.

Para guardar as *passwords* de autenticação dos utilizadores de forma segura, foi utilizado o módulo *Bcryt*, também disponibilizado em *NodeJS*, de modo a implementar uma função de *hashing* baseada no algoritmo *BlowFish* [PM99].

4.2 Fluxo de utilização

No início, o paciente regista-se na plataforma *gChain*. A partir da página principal da figura E.3, o paciente pode executar quatro ações: adicionar uma instituição à sua conta *gChain*, gerir as permissões de partilha dos seus registos, ver todos os registos ou eliminar os seus registos.

Uma instituição registada na plataforma pode, a partir da sua página principal da figura E.7 executar quatro ações: visualizar os pacientes associados, associar novos pacientes, ver os registos dos pacientes ou adicionar novos registos de pacientes.

4.2.0.1 Registo e *login* do paciente

O processo de registo de um novo paciente é autónomo, ou seja, é o próprio paciente que faz o seu registo na plataforma *gChain*. Para isto, tem de aceder à aplicação *web* para pacientes da figura E.1 através de um *browser*.

Para o registo é necessário preencher um formulário com os dados pessoais como o primeiro e último nome, *email* e uma *password* pessoal para acesso à plataforma *gChain*. Este pode ainda adicionar uma imagem de perfil, para uso na plataforma.

Este registo é guardado na base de dados *g-chain-id*, através da adição de uma linha na tabela *Patient*.

Após registo, o paciente é redirecionado para a página de *login* da figura E.2, onde tem de se autenticar para utilizar a plataforma.

4.2.0.2 Página pessoal do paciente

Após o *login*, o paciente tem acesso à sua página pessoal da figura E.3, que se encontra vazia. Para ter acesso aos seus registos clínicos existentes nas instituições de saúde pertencentes à plataforma *gChain*, o paciente tem, primeiro, que se associar a instituições de saúde, para isso, deve utilizar a opção "Associar instituição de saúde".

Já com associações estabelecidas, o paciente tem acesso aos seus registos clínicos guardados na base de dados *g-chain-rc* e pode, a qualquer altura, decidir eliminá-los através da opção "Eliminar registos".

4.2.0.3 Página de gestão da instituição

Uma instituição registada e autenticada na plataforma, tem acesso à sua página de gestão da figura E.7. A partir desta página, a instituição, através da opção "Associar novo paciente", pode criar uma associação com um paciente, desde que este a tenha solicitado.

Para além da associação, a instituição tem acesso a uma lista dos pacientes já associados. Para cada paciente, a instituição pode ver os registos desse paciente, através da opção "ver registos", ou fazer *upload* de novos registos, através da opção "Adicionar registos".

4.2.0.4 Associação paciente e instituição

O paciente, ao executar a ação "Associar instituição de saúde", gera um *token* identificativo da sua vontade em se associar a uma determinada instituição.

A instituição de saúde recebe uma notificação com o *token* e executa a opção "Associar novo paciente". Uma vez que o *token* está associado ao paciente, é criada a associação entre o paciente e a instituição.

Esta associação é registada na base de dados *g-chain-id*, através da adição de uma linha na tabela *patient_organization*.

4.2.0.5 Gestão de permissões pelo paciente

Já na página de gestão de permissões da figura E.4, o paciente tem uma lista das instituições a que está associado e, para cada uma delas, pode decidir partilhar os seus registos clínicos de outras instituições, para isso, tem de executar a ação "Alterar". Através desta ação, o paciente pode fornecer ou remover as permissões de partilha dos seus registos clínicos.

Qualquer ação anteriormente descrita, é possível através de uma nova transação para a base de dados *g-chain-acl*.

4.2.0.6 Visualização de registos pela instituição

Através da opção "Ver registos", disponível na página de gestão da figura E.7, a instituição é redirecionada para a página de visualização de registos da figura E.8 relativa ao paciente escolhido.

Nesta pagina, a instituição tem acesso aos dados adicionados pela mesma na secção "Registos adicionados", e aos registos partilhados pelo paciente na secção "Outros registos".

A instituição pode, novamente, executar a opção "Adicionar registos".

4.2.0.7 *Upload* de registos clínicos pela instituição

Através da opção "Adicionar registos", a instituição pede, primeiro, permissão ao paciente, através do envio de uma notificação de solicitação de permissão para fazer a adição e, só depois de aceite, é que pode fazer o *upload* dos registos para a base de dados *g-chain-rc*.

Neste processo, o paciente recebe a notificação com a solicitação da instituição e pode optar por aceitar ou recusar a mesma.

Capítulo 5

Testes e resultados

Neste capítulo estão expostos os resultados obtidos após utilização e teste à plataforma desenvolvida. Na primeira secção são apresentadas as metodologias de teste à plataforma e os resultados obtidos. Na última secção são discutidos os resultados apresentados na secção anterior.

5.1 Metodologias usadas

Para além da utilização da tecnologia *Blockchain*, a *gChain* pode ser considerada uma plataforma convencional uma vez que faz uso de tecnologias já bastante utilizadas e testadas como *MySQL* e *MongoDB*.

Assim, foi dado especial foco aos testes para o módulo *g-chain-acl* que consiste na implementação da rede *Blockchain* HF.

De modo a prever a escalabilidade e performance da plataforma, foram definidos os seguintes testes:

- estado do *channel*;
- tolerância a falhas do serviço de ordenação;
- performance sobre as operações executadas no *channel*;

O ambiente de desenvolvimento e teste foi uma máquina virtual com o sistema operativo *Ubuntu* 17.10, utilizando quatro núcleos de um processador *i7-4790*, 12GB de memória RAM e 60GB de armazenamento num disco rígido. Foi utilizado *port forwarding* de modo a possibilitar a comunicação entre a máquina virtual e outras máquinas em rede. Isto foi essencial de modo a simular a utilização da API em rede.

É importante referir que, apesar da utilização de *Docker* containers, que simulam uma máquina isolada, as comunicações dentro da rede *Blockchain* HF foram feitas em *localhost*.

5.1.1 Estado do *channel*

Este teste tem o intuito de averiguar o estado da cadeia, ou seja, verificar que todos os *peers* têm a mesma informação e pela mesma ordem.

Para isso foram utilizadas duas metodologias:

- verificação da *state database*;
- verificação do *ledger*;

A primeira consistiu em aceder, através da interface web disponibilizada pelo *CouchDB*, à base de dados de estados de cada *peer*, de modo a verificar se os últimos estados eram iguais para todos, ou seja, se todos os *peers* tinham as mesmas chaves e o mesmo valor para cada uma delas.

A segunda consistiu em aceder ao *cli container* e obter o último bloco da cadeia, através da execução de comandos nativos. Depois, através do *configtxlator* foi possível descodificar o bloco para leitura e verificar o conteúdo do mesmo.

Através destas metodologias foi possível comprovar, de forma prática, o consenso da rede e o funcionamento do sistema de ordenação.

5.1.2 Tolerância a falhas do serviço de ordenação

Com este teste pretende-se verificar o nível de tolerância a falhas do sistema de ordenação. Para isso, procedeu-se a vários testes que consistiram em parar e reativar a execução dos *Docker containers* relacionados com o serviço de ordenação, através dos comandos expostos na listagem 5.1.

```
1 docker stop id_container
2 docker start id_container
```

Listagem 5.1: Comandos para parar e recomeçar um *Docker container*.

Através deste teste verificou-se que é possível remover um nó do serviço de ordenação, continuando o outro a funcionar, e que é igualmente possível remover uma instância *Kafka*. Caso se remova mais do que uma instância *Kafka*, o serviço de ordenação fica indisponível.

5.1.3 Teste de performance sobre as operações na rede

Este teste tem como objetivo testar a performance da rede *Blockchain*, de modo a prever a evolução do desempenho da plataforma.

Todas as operações sujeitas ao teste foram chamadas a partir da API da plataforma *gChain*.

Uma vez que o HF utiliza uma base de dados com os estados mais atuais, faz sentido conduzir o teste de modo a testar o acesso à informação nesses moldes. Por isso, não interessa apenas o número de transações e blocos na rede, mas sim a variedade da informação.

Este teste consistiu em simular transações feitas por mil pacientes. Cada paciente fez, pelo menos, dez transações sobre a rede. De seguida, um paciente foi selecionado de modo a preencher o *Blockchain* com mais informação, permitindo verificar-se a escalabilidade da rede.

Este teste foi dividido em três fases relativamente ao número de transações efetuadas para a rede.

A primeira fase consistiu em fazer dez mil transações para a rede, enquanto na segunda fase se realizaram transações, de modo a perfazer cinquenta mil e, por fim, na terceira alcançaram-se as oitenta mil.

No final de cada fase, as operações *query* e *invoke* foram executadas da seguinte forma:

- dez mil *queries* com concorrência de cem;
- mil transações com concorrência de vinte;

Uma vez que todos os artefactos da rede HF estão isolados em *Docker containers*, o teste consistiu em monitorizar, tanto o desempenho dos *containers*, como do próprio *Blockchain* HF. Para o *Docker*, foram utilizados os comandos expostos na listagem 5.2. Já para testar a plataforma foi utilizado o *package loadtest*. Este foi usado para fazer chamadas à API da *gChain*, com várias opções, como por exemplo, definições de concorrência, de modo a monitorizar várias métricas de performance.

```
1 docker stats nome_container --format "{{.Param}}" --no-stream >> nome_ficheiro
2 docker ps --format "{{.Size}}"
3 docker system df
```

Listagem 5.2: Comandos para monitorização do *Docker* e dos respetivos *containers*

Através do primeiro comando é possível monitorizar a percentagem de utilização do *Central Processing Unit* (CPU) e a percentagem de utilização da memória RAM. É importante referir que o valor de utilização de CPU vai desde os 0% até aos 400%, uma vez que existem quatro núcleos de processador.

De modo a automatizar a monitorização destes dados, foi utilizado um *script*. O processo consistiu em iniciar o *script* antes de os testes serem executados, de modo a capturar pelo menos uma iteração, e pará-lo apenas depois da conclusão das operações.

Para os testes mencionados anteriormente, foram monitorizados os seguintes *containers*:

- *peer0.org1.example.com*
- *peer0.pl1.example.com*
- *couchdb1*
- *dev-peer0.org1.example.com-mycc-1.0*
- *dev-peer0.pl1.example.com-mycc-1.0*

O primeiro *container* corresponde a um *peer* pertencente a uma instituição de saúde. O segundo é relativo ao *peer* central da plataforma para acesso dos pacientes. O terceiro consiste na base de dados de estados do primeiro *peer*. Por último, o quarto e quinto *containers* são, respetivamente, as instâncias do *chaincode* do *peer* da instituição de saúde e do *peer* central.

Para além dos anteriores, o serviço de ordenação e todos os *containers* dependentes deste foram monitorizados:

- *orderer0.example.com*
- *orderer1.example.com*
- *kafka0.example.com*
- *kafka1.example.com*
- *kafka2.example.com*
- *kafka3.example.com*
- *zookeeper0.example.com*
- *zookeeper1.example.com*
- *zookeeper2.example.com*

De forma a simular corretamente uma situação de teste à plataforma, foi utilizado para a operação *invoke* o *peer* central da plataforma *gChain*, pois este é utilizado pelos pacientes no acesso à plataforma e é a partir deste que podem fazer transações para o *Blockchain g-chain-acl*, ou seja, adicionar ou alterar permissões.

Para a operação *query*, foi utilizado um *peer* correspondente a uma instituição de saúde, nomeadamente o *peer0.org1.example.com*.

5.2 Discussão de resultados

Através da leitura dos gráficos A.1, A.2, A.3, A.4, A.5, A.6, A.7 e A.8 verifica-se que para o *peer0.org1* há uma maior utilização do CPU na operação *query*, visto que é este quem executa a operação, no entanto, também há utilização considerável na operação *invoke*, uma vez que este também tem de atualizar o *ledger*, mantido localmente, devido às operações *invoke* feitas na rede. Verifica-se ainda uma maior necessidade de memória consoante as fases. Para o *peer0.pl1* assiste-se a uma maior utilização do processador na operação *invoke*, por estar a fazer transações para a rede. Tal como para o *peer* anterior, este também usa mais memória no decorrer das diferentes fases.

Nos gráficos B.1, B.2, B.3 e B.4 verifica-se uma maior utilização do processador para a operação *query* em comparação com a *invoke*. A utilização de memória é constante em ambas as operações para as três fases. Como era de esperar, a base de dados de estados é crucial para fazer *queries* na plataforma, uma vez que são estes *containers* que têm as transações mais recentes da rede.

Observando os gráficos C.1, C.2, C.3, C.4, C.5, C.6, C.7 e C.8 verifica-se que os contratos inteligentes utilizam pouca memória e de forma constante, independentemente da operação e da fase. Verifica-se, ainda, um maior esforço de processamento para a operação *query*. Quando não são executadas operações nos contratos, a utilização do CPU é nula.

Pelo estudo dos gráficos D.1, D.2, D.3, D.4, D.5, D.6, D.7, D.8, D.9, D.10, D.11 e D.12 é possível confirmar o esforço acumulado de todos os componentes do serviço de ordenação, através de gráficos de linhas empilhadas. Verifica-se uma maior utilização tanto do CPU como da memória RAM em relação às fases testadas, o que significa que a performance da rede diminui com o aumento do número de transações.

Na tabela 5.1 contemplam-se os resultados das várias etapas e testes feitos à plataforma. O parâmetro *tTotal* corresponde à duração total da execução em segundos, o parâmetro *Requests Per Second* (RPS) corresponde ao número de pedidos à API por segundo, e os *lmed*, *lmax* e *lmin* correspondem, respetivamente, aos valores de latência média, máxima e mínima em segundos.

Teste	**Nº pedidos**	**Nº erros**	**tTotal**	**RPS**	**lmed**	**lmax**	**lmin**
Primeiro registo	1	0	2.17	0	2173.9	2173	2173
Até 10k	9000	109	1267.02	17	582.9	605	561
query 10k	10000	0	53.40	187	531.4	2608	154
invoke 10k	1000	0	70.96	14	1059.7	5244	503
Até 50k	39000	844	3449.68	11	1326.4	20322	436
query 50k	10000	61	63.82	157	634.3	5526	11
invoke 50k	1000	15	83.34	12	1241	3320	537
Até 80k	29000	680	2686.29	11	1111.1	5264	402
query 80k	10000	69	67.14	149	662.4	5834	3
invoke 80k	1000	0	84.90	12	1009.7	2740	572

Tabela 5.1: Tabela de resultados de execução da API.

Através da leitura da tabela anterior, é possível verificar uma diminuição de performance para ambas as operações, relativamente ao evoluir das fases, o que resulta numa diminuição do valor de RPS, aumento de erros verificados e consequente aumento dos valores de latência.

Outra observação consiste na duração do primeiro registo por um utilizador, que consiste na primeira transação feita para a rede, que é consideravelmente mais lenta do que as restantes, uma vez que neste processo inicial é necessário comunicar com o CA e este tem de gerar e persistir as chaves criptográficas do utilizador.

A partir destes dados foram traçados os gráficos de tendência da operação *query* 5.2 e da *invoke* 5.1, de modo a prever a evolução do valor de RPS para as duas operações. Foram estimados os resultados, tanto para um estado de rede com cem mil transações, como para duzentas mil. A vermelho tracejado é possível ver as funções de tendência.

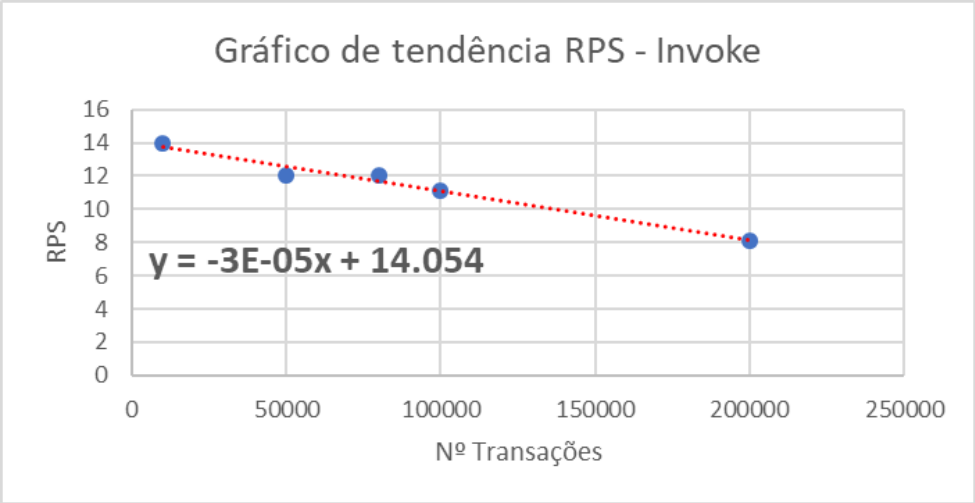

Figura 5.1: Gráfico de tendência para operação *invoke*.

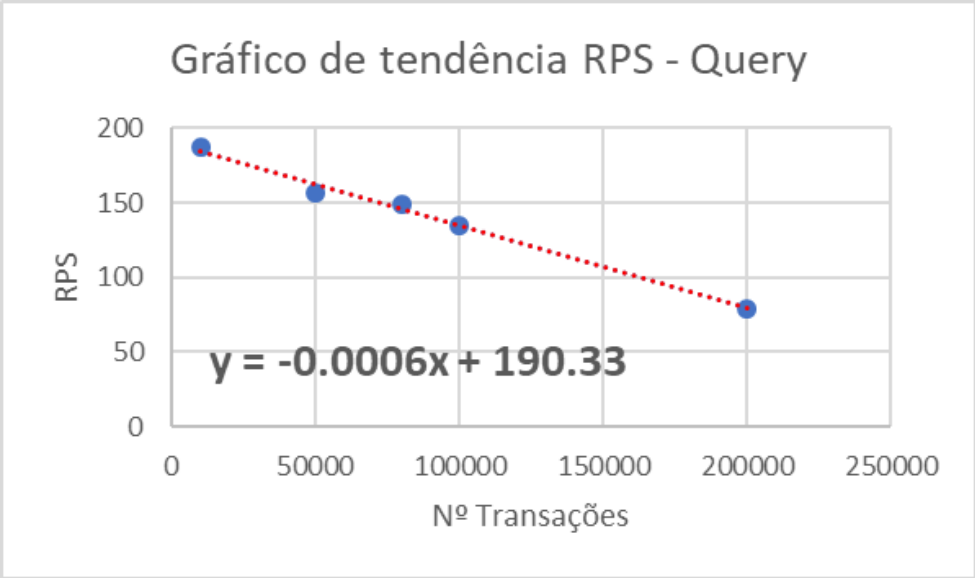

Figura 5.2: Gráfico de tendência para operação *query*.

Como era de prever, os valores de RPS diminuem conforme o aumento de transações na rede uma vez que a função tendência, para ambas as operações, apresenta um declive negativo.

No final da execução destes testes o tamanho ocupado pelo *ledger* em cada *peer* era de 394,9MB. Já o tamanho da base de dados de estados ocupava 2,4GB em cada instância *CouchDB* associada a um *peer*.

Foi ainda testado o nível de concorrência da plataforma. Para isso, foram feitos vários pedidos à API, com diferentes valores de concorrência. Os resultados do teste para a operação *query* estão na tabela 5.2, enquanto que na tabela 5.3 estão os resultados para a operação *invoke*.

Tipo	Nº pedidos	Nº erros	tTotal	RPS	lmed	lmax	lmin
100 concorrência	10000	0	81.17	123	808.2	1650	259
200 concorrência	10000	0	78.27	128	1548.9	2665	633
500 concorrência	10000	708	77.32	129	3715.9	9655	3
10000 concorrência	10000	1065	93.77	107	8823.1	22541	4

Tabela 5.2: Tabela de resultados de concorrência para *query*.

Tipo	Nº pedidos	Nº erros	tTotal	RPS	lmed	lmax	lmin
20 concorrência	1000	20	84.92	12	1690.3	4340	904
40 concorrência	1000	864	85.34	12	3348.2	4313	1008
100 concorrência	1000	990	50.53	20	4848.6	7074	2793

Tabela 5.3: Tabela de resultados de concorrência para *invoke*.

Verifica-se que, para ambas as operações, o aumento da concorrência de pedidos traduz-se num aumento de erros. Para além disso, verifica-se que a operação *invoke* reage pior ao aumento da concorrência. Isto deve-se à incapacidade do serviço de ordenação em responder a todos os pedidos feitos. Isto leva ao não processamento dos pedidos, ou seja, são descartados.

Assim, os comportamentos demonstrados pela plataforma aos testes realizados não são desejáveis.

Uma possível solução passará por, primeiro, fazer uma atualização ao *hardware* onde a plataforma vai correr. E, em segundo lugar, deverá implementar-se um sistema de filas com o objetivo de não se perder pedidos à rede por parte dos utilizadores. Seguidamente, deve implementar-se um sistema de balanceamento de carga na plataforma, por forma a controlar e distribuir os acessos à plataforma.

Uma vez implementadas as soluções anteriores, outra possível solução consiste em aumentar o número de *peers* na plataforma central, uma vez que já haveria um balanceador a controlar os pedidos à plataforma.

Por fim, uma outra solução passaria pela expansão do serviço de ordenação, ou seja, aumentar o número de nós do serviço, incluindo as dependências *Kafka*, de modo a diminuir o número de tarefas por cada nó e assim diminuir a carga computacional de cada um.

Após várias iterações de testes foi possível verificar que qualquer ação executada na máquina virtual interferia com os resultados obtidos. O próprio mecanismo de teste, que consistiu em guardar os resultados em ficheiros de texto, também interferiu no desempenho da plataforma, por utilizar o disco.

Os testes realizados permitiram validar a fiabilidade da plataforma, no entanto, existem custos de performance associados. Para além disso, também se pode concluir que o desempenho da plataforma depende diretamente dos recursos onde está alojado.

Capítulo 6

Conclusões e desenvolvimentos futuros

Esta dissertação tinha como propósito criar uma plataforma que resolvesse o problema da dispersão de RMEs e, ao mesmo tempo, cumprisse com as novas normas sobre a proteção de dados na União Europeia.

Assim, foi desenvolvido um novo modelo de partilha de RMEs que resultou na plataforma *gChain*. Esta plataforma tem a particularidade de incluir na sua arquitetura a tecnologia *Blockchain*, de forma a atingir os objetivos e requisitos propostos.

Uma vez que o *Blockchain* é, ainda, uma tecnologia pouco explorada na área da saúde, foi dado foco ao estudo da sua arquitetura, vantagens e limitações de modo a aplicar da melhor forma esta tecnologia.

Foi, por isso, escolhida a plataforma HF para o desenvolvimento do módulo *Blockchain* da *gChain*. O HF mostrou ser a escolha certa devido à sua arquitetura modular focada nas necessidades empresariais, o que implica fiabilidade e segurança da plataforma.

Verificou-se que o *Blockchain* não é a melhor solução para guardar grandes quantidades de informação, como registos clínicos, devido à sua arquitetura de persistência de dados e replicação, pelo que a solução encontrada passou, então, pela persistência de permissões de partilha dos registos clínicos. Desta forma, foi possível tirar partido das principais vantagens do *Blockchain* como o histórico de alterações e consequente imutabilidade, descentralização e autenticidade da informação.

O *Blockchain*, aliado a uma arquitetura focada na privacidade, fez com que a *gChain* atingisse os objetivos propostos: cumpre as novas normas Europeias sobre a proteção de dados, garante a segurança de operações, transparência e privacidade. Para além disso, garante ainda o total controlo aos pacientes sobre os seus dados e oferece uma solução que agrega todos os RMEs do paciente, independentemente da instituição de origem.

De modo a medir a performance, fiabilidade e escalabilidade da plataforma desenvolvida, foram executados alguns testes.

A partir dos testes conclui-se que a plataforma perde performance com o aumento do número de transações na rede, isto deve-se à maior exigência computacional de recursos como processamento e memória.

Foram, por isso, identificadas algumas soluções de forma a combater estes problemas de escalabilidade, que passam por utilizar um ambiente de desenvolvimento melhorado a nível de processamento e memória, pela implementação de um mecanismo de balanceamento de carga, de modo a gerir a distribuição dos pedidos, pelo desenvolvimento de um mecanismo de filas, para que nenhum pedido à API seja descartado, e, por fim, pelo aumento do número de nós de serviços de ordenação.

Com estas soluções pretende-se corrigir os problemas de escalabilidade da plataforma, de modo a conseguir uma performance aceitável para ambientes de produção.

Para além dos desenvolvimentos feitos, foram ainda identificadas funcionalidades como trabalho futuro: implementação das soluções anteriores, de forma a colmatar os problemas de escalabilidade; implementação de um sistema de notificações interno à plataforma, de modo a notificar e controlar as ações na plataforma; implementação da funcionalidade para ver o histórico de alterações de permissões; e, aumentar as opções de partilha dos registos do paciente, ou seja, em vez de partilhar todos os dados de uma instituição, selecionar apenas os registos, ou períodos de registos, desejáveis.

Após desenvolvimentos e maturação, a plataforma *gChain* será integrada no conjunto de soluções oferecidas pela *Glintt*.

Referências

[ABB+18] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolić, Sharon Weed Cocco e Jason Yellick. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. 2018. URL: http://arxiv.org/abs/1801.10228{%}0Ahttp://dx.doi.org/10.1145/3190508.3190538, arXiv:1801.10228, doi:10.1145/3190508.3190538.

[Adv16] UK Government Chief Scientific Adviser. Distributed ledger technology: beyond block chain. Report, Government Office for Science, 2016.

[AEVL16] A. Azaria, A. Ekblaw, T. Vieira e A. Lippman. Medrec: using blockchain for medical data access and permission management. pages 25 – 30, Los Alamitos, CA, USA, 2016//. URL: http://dx.doi.org/10.1109/OBD.2016.11.

[AIB] Inc All In Bits. Tendermint. Disponível em https://tendermint.readthedocs.io/en/master/, acedido a última vez em 7 de fevereiro de 2018.

[AORBK17] Abdullah Al Omar, Mohammad Shahriar Rahman, Anirban Basu e Shinsaku Kiyomoto. Medibchain: A blockchain based privacy preserving platform for healthcare data. In Guojun Wang, Mohammed Atiquzzaman, Zheng Yan e Kim-Kwang Raymond Choo, editors, *Security, Privacy, and Anonymity in Computation, Communication, and Storage*, pages 534–543, Cham, 2017. Springer International Publishing.

[Bas17] Imran Bashir. Mastering Blockchain, 2017.

[Bax16] Gareth Baxendale. Can blockchain revolutionise eprs? *ITNOW*, 58(1):38–39, 2016. URL: +http://dx.doi.org/10.1093/itnow/bww017, doi:10.1093/itnow/bww017.

[DWC+17] Tien Tuan Anh Dinh, Ji Wang, Gang Chen, Rui Liu, Beng Chin Ooi e Kian-Lee Tan. BLOCKBENCH A framework for analyzing private blockchains. *CoRR*, abs1703.04057, 2017. URL: httparxiv.orgabs1703.04057, arXiv:1703.04057.

[EU1] Eur-lex access to european union law. Disponível em http://eur-lex.europa.eu/legal-content/PT/TXT/?uri=celex%3A32016R0679, acedido a última vez em 7 de fevereiro de 2018.

[EU2] Acordo sobre reforma da proteção de dados na ue proposta pela comissão estimula mercado único digital. Disponível em http://europa.eu/rapid/press-release_IP-15-6321_pt.htm, acedido a última vez em 7 de fevereiro de 2018.

[Eur] Comissão Europeia. Regulamento (ue) 2016/679 — proteção das pessoas singulares no que diz respeito ao tratamento de dados pessoais e à livre circulação desses dados. Disponível em http://publications.europa.eu/resource/cellar/e899a0de-be12-405c-84ae-34e8a22d6acd.0020.02/DOC_1, acedido a última vez em 7 de fevereiro de 2018.

[fS14] ISO International Organization for Standardization. Health informatics — capacity-based ehealth architecture roadmap — part 2: Architectural components and maturity model. Disponível em https://www.iso.org/obp/ui/#iso:std:iso:tr:14639:-2:ed-1:v1:en, acedido a última vez em 7 de fevereiro de 2018, 2014.

[fS16] ISO International Organization for Standardization. Blockchain and distributed ledger technologies. Disponível em https://www.iso.org/committee/6266604.html, acedido a última vez em 7 de fevereiro de 2018, 2016.

[GKW+16] Arthur Gervais, Ghassan O. Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf e Srdjan Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 3–16, New York, NY, USA, 2016. ACM. URL: http://doi.acm.org/10.1145/2976749.2978341, doi:10.1145/2976749.2978341.

[GL06] S. Gilbert e N. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51 – 9, 2002/06/. URL: http://dx.doi.org/10.1145/564585.564601.

[hfk] Bringing up a kafka-based ordering service¶. Disponível em =http://hyperledger-fabric.readthedocs.io/en/release-1.1/kafka.html, acedido a última vez em 15 de fevereiro de 2018.

[Hyp17] Hyperledger Architecture Working Group. Hyperledger Architecture , Volume 1. *Hyperledger.org*, 1:15, 2017. URL: https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger{_}Arch{_}WG{_}Paper{_}1{_}Consensus.pdf.

[KKOM17] Tsung-Ting Kuo, Hyeon-Eui Kim e Lucila Ohno-Machado. Blockchain distributed ledger technologies for biomedical and health care applications. *Journal of the American Medical Informatics Association*, 24(6):1211–1220, 2017. URL: +http://dx.doi.org/10.1093/jamia/ocx068, doi:10.1093/jamia/ocx068.

[KNR11] Jay Kreps, Neha Narkhede e Jun Rao. Kafka: a Distributed Messaging System for Log Processing. *ACM SIGMOD Workshop on Networking Meets Databases*, page 6, 2011. URL: http://research.microsoft.com/en-us/um/people/srikanth/netdb11/netdb11papers/netdb11-final12.pdf.

[lf] The linux foundation. Hyperledger fabric. Disponível em https://hyperledger-fabric.readthedocs.io/en/release/blockchain.html, acedido a última vez em 7 de fevereiro de 2018.

[loo] Loopback core concepts. Disponível em =http://loopback.io/doc/en/lb3/LoopBack-core-concepts.html, acedido a última vez em 15 de junho de 2018.

[Men92] Belden Menkus. *Introduction to computer security*, volume 11. 1992. arXiv:arXiv:1011.1669v3, doi:10.1016/0167-4048(92)90036-Q.

[Met16] M. Mettler. Blockchain technology in healthcare: The revolution starts here. In *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pages 1–3, Sept 2016. doi:10.1109/HealthCom.2016.7749510.

[Mös13] Malte Möser. Anonymity of bitcoin transactions an analysis of mixing services. University of Münster, 2013.

[Nak09] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2009.

[PM99] Niels Provos e David Mazieres. A future-adaptable password scheme. *USENIX Annual Technical Conference, . . .* , pages 1–12, 1999. URL: https://www.usenix.org/legacy/event/usenix99/full{_}papers/provos/provos.pdf.

[RdCR17] Alex Roehrs, Cristiano Andre da Costa e Rodrigo da Rosa Righi. OmniPHR: A distributed architecture model to integrate personal health records. *JOURNAL OF BIOMEDICAL INFORMATICS*, 71:70–81, JUL 2017. doi:10.1016/j.jbi.2017.05.012.

[SBV17] João Sousa, Alysson Bessani e Marko Vukolić. A Byzantine Fault-Tolerant Ordering Service for the Hyperledger Fabric Blockchain Platform. 2017. arXiv:1709.06921, doi:10.1145/3152824.3152830.

[Swa15] Swan M. *Blockchain Blueprint for a New Economy*. Number O'REILLY. 2015. arXiv:arXiv:1011.1669v3, doi:10.13140/RG.2.1.2350.8568.

[TS06] Andrew S. Tanenbaum e Martin Van Steen. *Distributed Systems:Principles and Paradigms*. Pearson Education. Inc, 2006.

REFERÊNCIAS

Anexo A

Gráficos de carga para *Peers*

A.1 *Peer0.org1*

A.1.1 *Invoke*

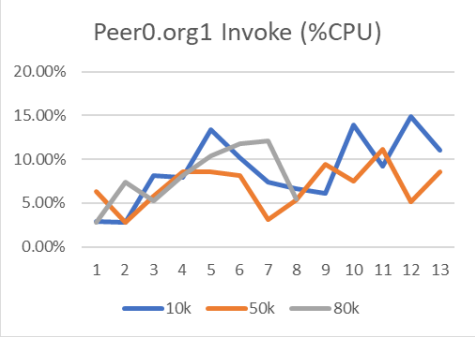

Figura A.1: Medição de CPU para a operação *invoke*.

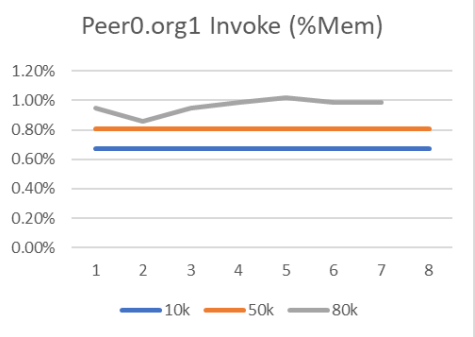

Figura A.2: Medição de RAM para a operação *invoke*.

A.1.2 *Query*

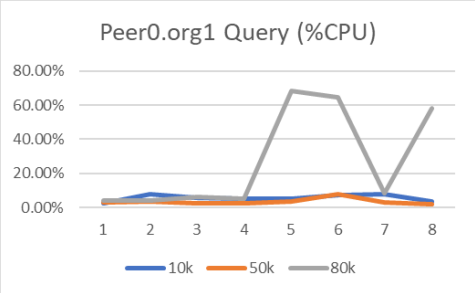

Figura A.3: Medição de CPU para a operação *query*.

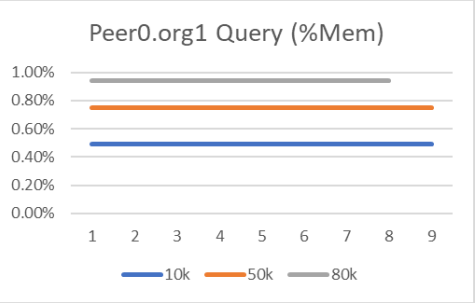

Figura A.4: Medição de RAM para a operação *query*.

A.2 *Peer0.pl1*

A.2.1 *Invoke*

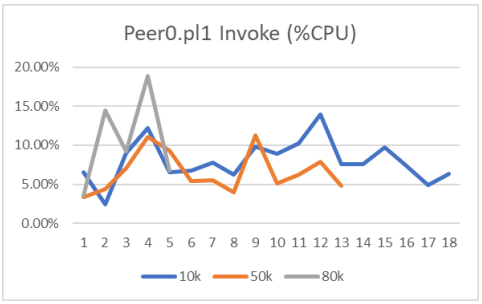

Figura A.5: Medição de CPU para a operação *invoke*.

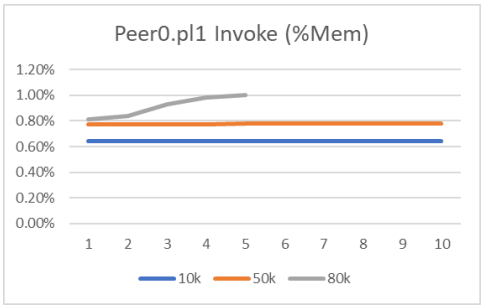

Figura A.6: Medição de RAM para a operação *invoke*.

A.2.2 *Query*

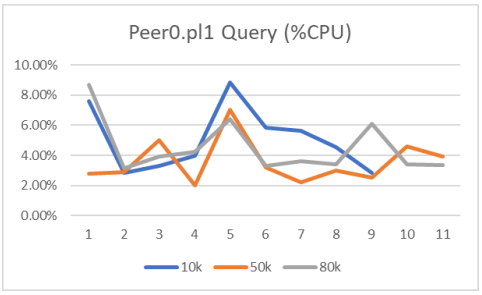

Figura A.7: Medição de CPU para a operação *query*.

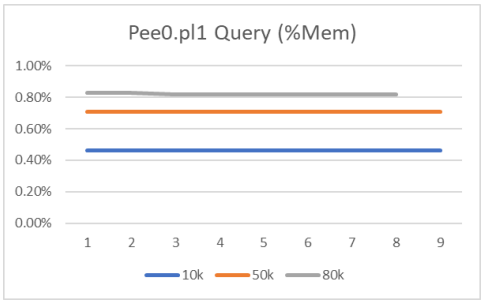

Figura A.8: Medição de RAM para a operação *query*.

Anexo B

Gráficos de carga *CouchDB*

B.1 *CouchDb1*

B.1.1 *Invoke*

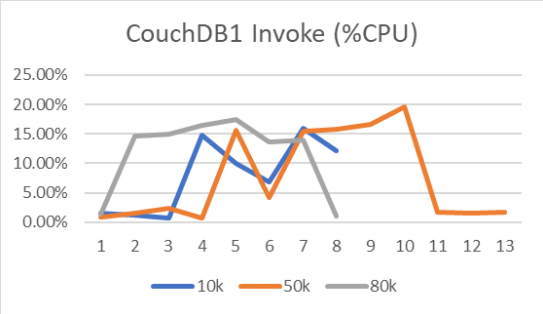

Figura B.1: Medição de CPU para a operação *invoke*.

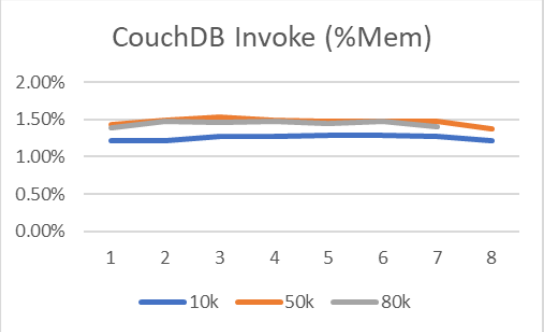

Figura B.2: Medição de RAM para a operação *invoke*.

B.1.2 *Query*

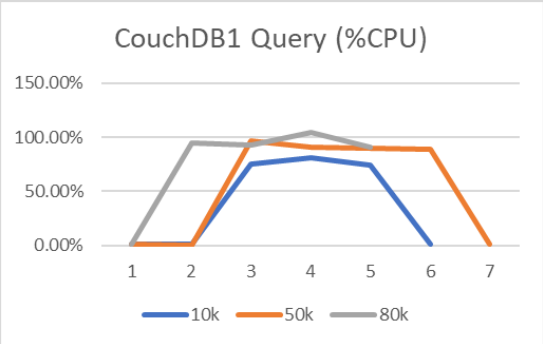

Figura B.3: Medição de CPU para a operação *query*.

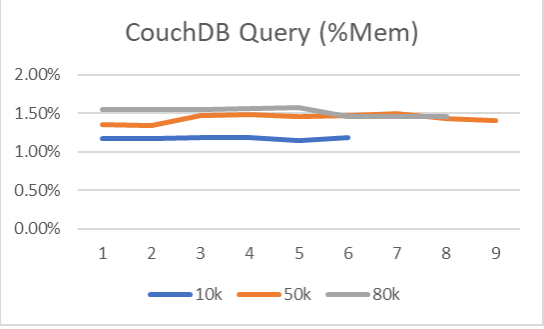

Figura B.4: Medição de RAM para a operação *query*.

Anexo C

Gráficos de carga *Chaincode*

C.0.1 *peer0.org1*

C.0.1.1 *Invoke*

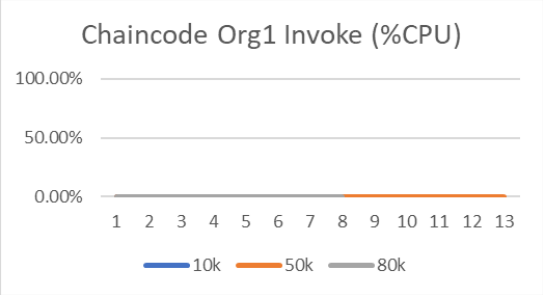

Figura C.1: Medição de CPU para a operação *invoke*.

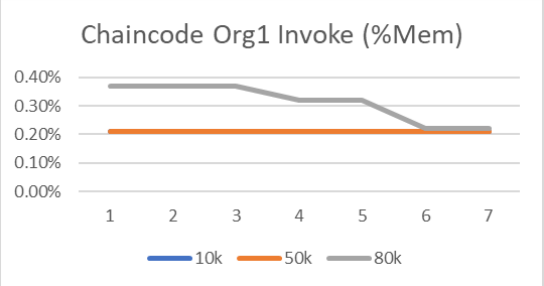

Figura C.2: Medição de RAM para a operação *invoke*.

C.0.1.2 Query

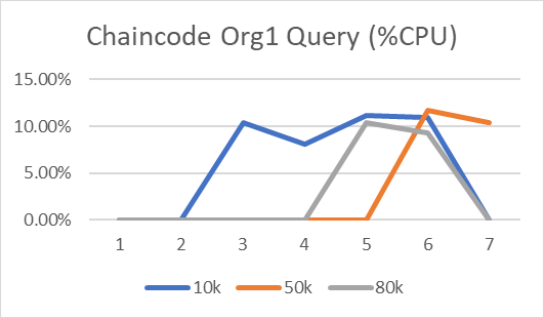

Figura C.3: Medição de CPU para a operação *query*.

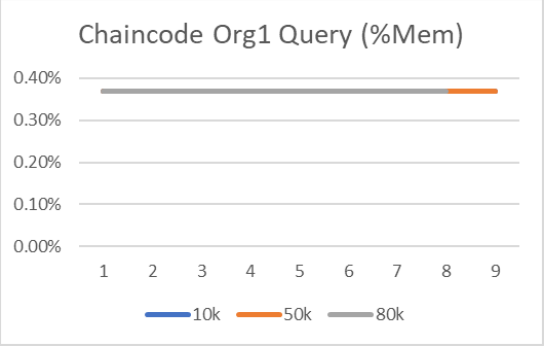

Figura C.4: Medição de RAM para a operação *query*.

C.0.2 *peer0.pl1*

C.0.2.1 *Invoke*

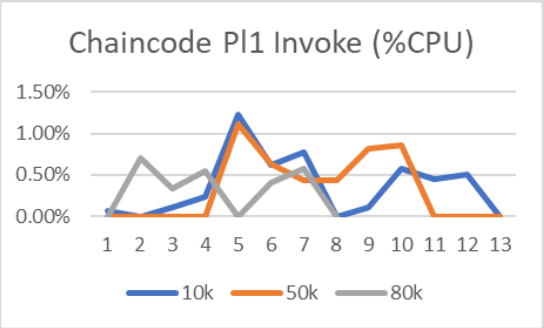

Figura C.5: Medição de CPU para a operação *invoke*.

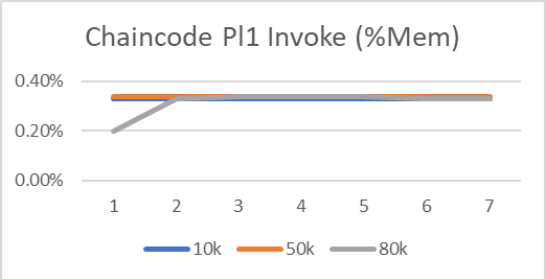

Figura C.6: Medição de RAM para a operação *invoke*.

C.0.2.2 *Query*

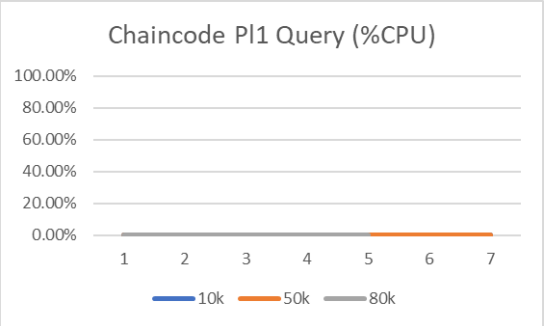

Figura C.7: Medição de CPU para a operação *query*.

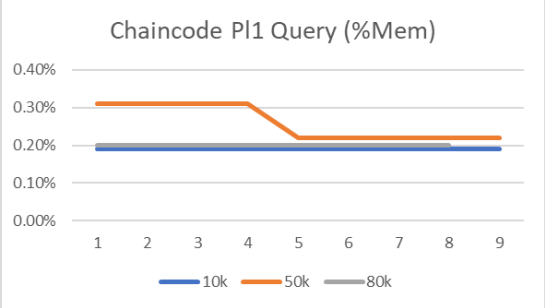

Figura C.8: Medição de RAM para a operação *query*.

Anexo D

Gráficos de carga do serviço de ordenação

Para os gráficos apresentados, as séries 1, 2, 3, 4 ,5, 6 e 7 correspondem, respetivamente aos *Docker container orderer1.example.com, orderer0.example.com, kafka1.example.com, kafka3.example.com, kafka2.example.com, kafka0.example.com, zookeeper1.example.com, zookeeper2.example.com e zookeeper0.example.com*.

D.1 *Invoke*

D.1.1 CPU

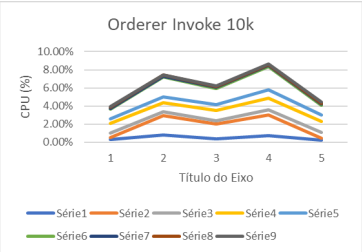

Figura D.1: Medição de CPU para a operação *invoke* para 10k transações.

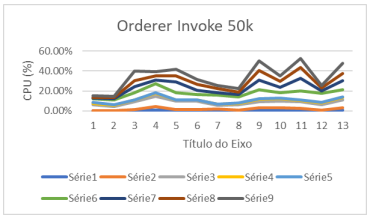

Figura D.2: Medição de CPU para a operação *invoke* para 50k transações.

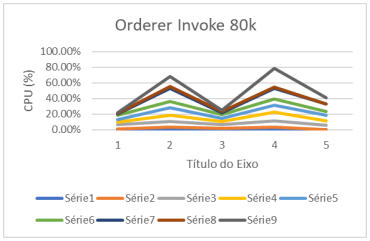

Figura D.3: Medição de CPU para a operação *invoke* para 80k transações.

D.1.2 Memória RAM

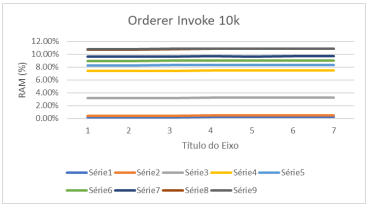

Figura D.4: Medição de RAM para a operação *invoke* para 10k transações.

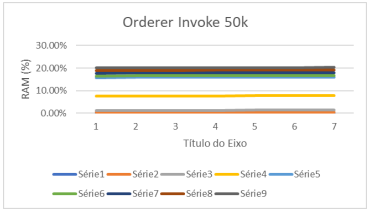

Figura D.5: Medição de RAM para a operação *invoke* para 50k transações.

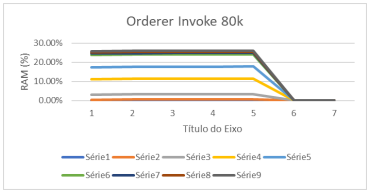

Figura D.6: Medição de RAM para a operação *invoke* para 80k transações.

D.2 *Query*

D.2.1 CPU

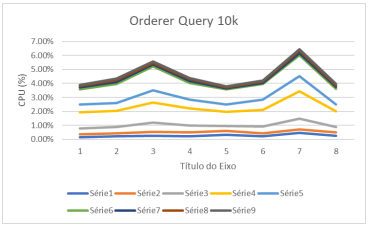

Figura D.7: Medição de CPU para a operação *query* para 10k transações.

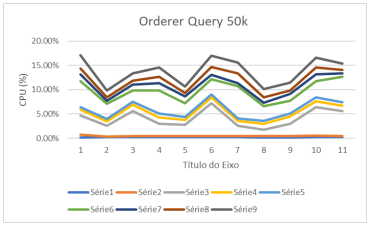

Figura D.8: Medição de CPU para a operação *query* para 50k transações.

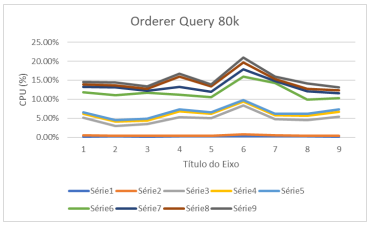

Figura D.9: Medição de CPU para a operação *query* para 80k transações.

D.2.2 Memória RAM

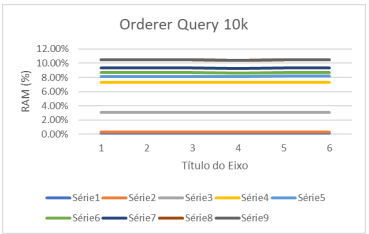

Figura D.10: Medição de RAM para a operação *query* para 10k transações.

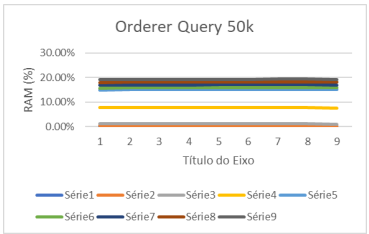

Figura D.11: Medição de RAM para a operação *query* para 50k transações.

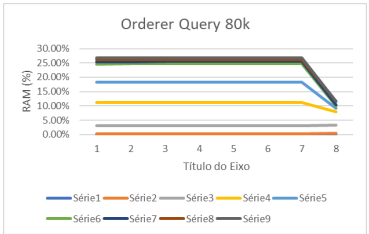

Figura D.12: Medição de RAM para a operação *query* para 80k transações.

Anexo E

Interfaces das aplicações

E.1 Aplicação para pacientes

E.1.1 Páginas de registo e *login*

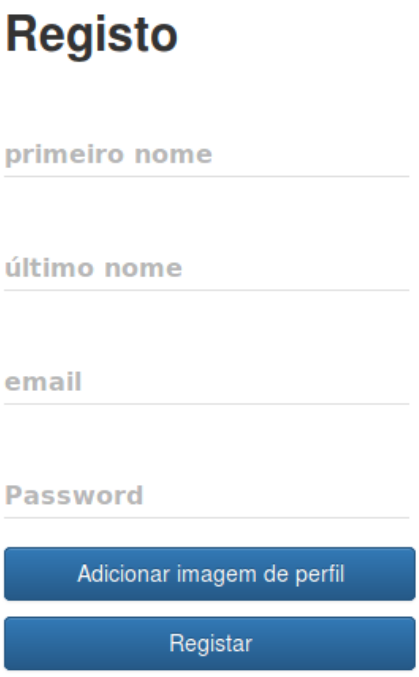

Figura E.1: Pormenor da página de *login* para pacientes.

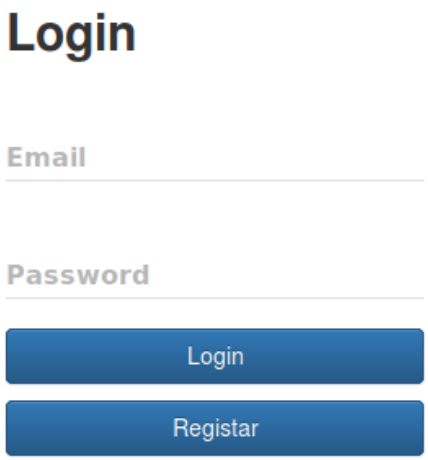

Figura E.2: Pormenor da página de registo para pacientes.

E.1.2 Página para visualização dos registos clínicos

Figura E.3: Página para visualização dos registos clínicos.

E.1.3 Página para gestão de permissões

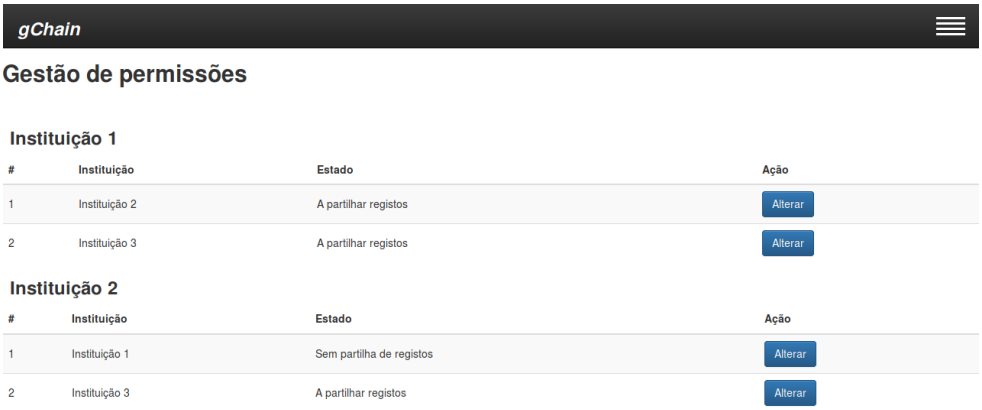

Figura E.4: Página para gestão de permissões

E.2 Aplicação para instituições de saúde

E.2.1 Páginas de registo e *login*

Figura E.5: Pormenor da página de *login* para instituições.

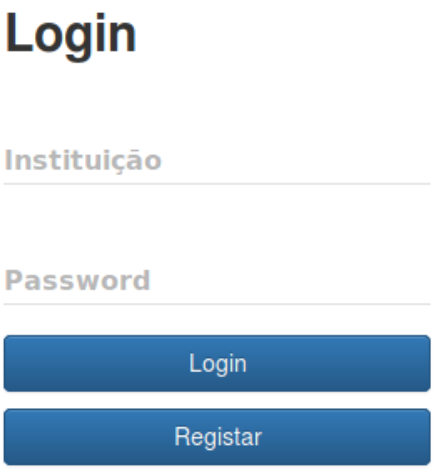

Figura E.6: Pormenor da página de registo para instituições.

E.2.2 Página de gestão

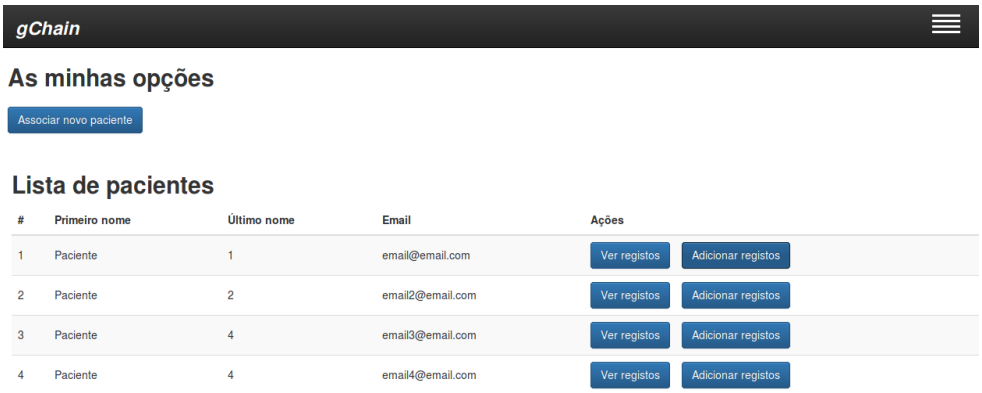

Figura E.7: Página de gestão das instituições.

E.2.3 Página de visualização de registos clínicos

Figura E.8: Página de visualização de registos clínicos pelas instituições.