Desenvolvimento de uma Aplicação Persistente com Características da Tecnologia Blockchain

José Cleanto F. Arrais Neto[1], **Leonardo O. Moreira**[1]

[1]Instituto Universidade Virtual (IUVI)
Universidade Federal do Ceará (UFC) – Fortaleza, CE – Brasil

cleantoneto2010@hotmail.com, leoomoreira@virtual.ufc.br

Abstract. *Several data models are being designed, because in some classes of applications, the relational data model does not meet some needs. Some of these diverse data models can be found in NoSQL Databases. In addition, other technologies are emerging, such as Blockchain, which store data aiming at other types of important characteristics for some types of applications. This article aims to present the development of a persistent application with features of Blockchain technology.*

Resumo. *Diversos modelos de dados estão sendo concebidos, pois em algumas classes de aplicações, o modelo de dados relacional não atende algumas necessidades. Alguns desses diversos modelos de dados podem ser encontrados em Bancos de Dados NoSQL. Ademais, estão surgindo outras tecnologias, como Blockchain, que fazem armazenamento de dados visando outros tipos de características importantes para alguns tipos de aplicações. Este artigo tem como objetivo apresentar o desenvolvimento de uma aplicação persistente com características da tecnologia Blockchain.*

1. Introdução

Um Banco de Dados pode ser conceituado como uma coleção de dados relacionados, onde o modelo relacional é um modelo de dados amplamente utilizado por aplicações de diversos domínios [Elmasri and Navathe 2011]. Este modelo de dados, o relacional, estrutura um Banco de Dados como uma coleção de uma ou mais relações (tabelas) compostas de tuplas (linhas) e atributos (colunas) [Marinho et al. 2020]. Como muitas aplicações são orientada a dados [Marinho et al. 2018], um grande volume de dados é produzido e se faz necessário um processamento adequado, visando alguns aspectos como desempenho, disponibilidade e segurança. Assim, outros modelos de dados têm sido propostos visando algumas classes de aplicações, onde o modelo de dados relacional não se mostra adequado. Outro fator que deve-se levar em consideração, é que o maior tempo gasto no processamento das requisições de aplicações está relacionada à solução de persistência de dados [Moreira 2014]. Portanto, em aplicações orientadas a dados, deve-se ter uma maior atenção com o modelo de dados utilizado e também com o componente que implementa a solução de persistência de dados [Neto et al. 2021].

Marinho et al. (2020) comentam que muitas aplicações, orientadas a dados, utilizam dados de diferentes tipos e domínios. Assim, alternativas ao modelo de dados relacional, como o *Not Only SQL* (NoSQL), estão se solidificando [Marinho et al. 2020]. MongoDB [MongoDB 2021b] e Blockchain [Moreira Neto et al. 2020] são, respectivamente,

um Banco de Dados NoSQL distribuído de uso geral baseado em documentos e uma tecnologia que permite armazenamento de dados distribuídos [Neto et al. 2021]. O MongoDB, além de ser um Banco de Dados NoSQL popular, é frequentemente apreciado em estudos comparativos devido à sua flexibilidade e facilidade de uso [Andreoli et al. 2021].

El-Hindi et al. (2019) discutem que o BigchainDB [BigchainDB 2021a] se baseia no MongoDB, assim possuindo características semelhantes. BigchainDB, além das características comuns de um Banco de Dados, acrescenta os atributos vinculados à tecnologia Blockchain [Neto et al. 2021]. Portanto, estas duas soluções de persistência de dados, MongoDB e BigchainDB, possuem algumas similaridades, mas com propostas de gestão de dados diferentes [Neto et al. 2021]. Diante disso, a principal contribuição deste artigo é o desenvolvimento de uma aplicação persistente que possua características da tecnologia Blockchain. Neto et al. (2021) confrontaram as duas soluções supracitadas de persistência de dados e discutiram quais benefícios podem trazer às aplicações orientadas a dados. Assim, nesta extensão do trabalho, iremos aplicar aspectos de desenvolvimento e aplicação para demonstrar as características da tecnologia Blockchain em uma aplicação prática.

Segundo Gil (2002), algumas regras práticas para formulação de problemas científicos são: i) deve ser estruturada como uma pergunta; ii) deve ser a mais específica possível; e iii) utilizar terminologias claras com significativo preciso. Assim, pode-se formular o problema científico que permeia este trabalho com a seguinte questão: "Como se dá o processo de desenvolvimento de uma aplicação persistente com características da tecnologia Blockchain?". Além da questão que reflete o problema científico desta pesquisa, é formulada a seguinte hipótese: "Existe a possibilidade do armazenamento NoSQL se assemelhar aos comportamentos da tecnologia Blockchain no quesito de gestão de dados".

O MongoDB foi escolhido como Banco de Dados para o desenvolvimento da aplicação em razão da sua flexibilidade e facilidade de uso e por ofertar todos os recursos necessários para a aplicação, além de conseguir armazenar e servir grandes volumes de dados [Neto et al. 2021]. A adição das características de blockchain se dá pela finalidade de alcançar algumas das garantias que são desejáveis dentro do contexto da aplicação e que são ofertadas por este tipo de estrutura, como a imutabilidade, rastreabilidade e auditabilidade. O BigchainDB, por sua vez, não foi aplicado de forma nativa no desenvolvimento por apresentar uma maior curva de aprendizado e por apresentar algumas desvantagens em tópicos como desempenho e escalabilidade [Furtado 2019].

O objetivo principal deste ensaio é apresentar o desenvolvimento de uma aplicação persistente com características da tecnologia Blockchain. Para alcançar o objetivo principal almejado, os seguintes objetivos específicos foram elencados: i) apresentar os conceitos e motivações relacionadas às tecnologias de Banco de Dados NoSQL e Blockchain; ii) elaborar uma metodologia que passeie pelos métodos científicos que apresentem as etapas do desenvolvimento da aplicação; e iii) realizar uma discussão, nos aspectos de implementação e uso da aplicação, diante requisitos dos de dados e da aplicação.

Este trabalho está dividido, além da introdução, nas seguintes seções: Seção 2 apresenta todo arcabouço teórico necessário para o entendimento, compreendendo os conceitos de Banco de Dados Relacional, Banco de Dados NoSQL e Blockchain. Ainda na

Seção 2 são descritos o MongoDB e BigchainDB como soluções, respectivamente, de Banco de Dados NoSQL e Banco de Dados com Características de Blockchain. Já a Seção 3 descreve os aspectos metodológicos adotados no trabalho. A Seção 4, por sua vez, comenta sobre os resultados obtidos e apresenta a discussão. Por fim, a Seção 5 apresenta as conclusões e comenta sobre trabalhos futuros.

2. Fundamentação Teórica

Nesta seção foi feita uma exploração teórica por meio de um levantamento bibliográfico sobre comparações entre Banco de Dados Relacional e Banco de Dados NoSQL, o Banco de Dados NoSQL MongoDB, a tecnologia Blockchain na perspectiva do armazenamento de dados e o Banco de Dados em Blockchain BigchainDB. Todo o arcabouço teórico escrito nesta seção é de suma importância para melhor compreensão dos resultados alcançados e a discussão, elementos textuais presentes na Seção 4.

2.1. Banco de Dados Relacional e Banco de Dados NoSQL

Com a popularização da Internet e do surgimento de infraestruturas como a Computação em Nuvem, surge a necessidade de utilizar Bancos de Dados que possam armazenar e processar Big Data de forma eficaz, com demanda por alto desempenho na leitura e escrita dos dados [Han et al. 2011]. Big Data, de forma resumida, pode ser utilizado para coletar, tratar, analisar e processar grandes volumes de dados, de diversos tipos, em altíssima velocidade [Neto et al. 2021]. Assim, tendo em vista o crescimento e a variedade de tipos de dados gerados a todo instante, o modelo de dados relacional nem sempre é o mais recomendado, principalmente, quando grandes volumes de dados devem ser gerenciados [Soares and Boscarioli 2013]. Portanto, um Banco de Dados Relacional enfrenta muitos desafios [Han et al. 2011] diante das situações supracitadas [Neto et al. 2021].

Um Banco de Dados que implementa o modelo de dados relacional pode ser definido como um conjunto de relações (tabelas), colunas (atributos), tuplas (linhas) e as restrições de integridade do modelo relacional [Elmasri and Navathe 2011]. O estudo conduzido por Farias (2014) destaca algumas limitações do modelo de dados relacional: i) não permite uma estrutura de dados flexível que favoreça a obtenção de um nível mais alto de escalabilidade; e ii) não possibilita a estruturação dos seus dados de acordo com as necessidades individuais [Neto et al. 2021]. Apesar do Banco de Dados Relacional ainda ser um dos mais utilizados, ele possui algumas limitações que podem impactar negativamente em certas aplicações que necessitem uma estrutura de dados flexível e processamento em larga escala [Neto et al. 2021].

Han et al. (2011) discutem que as principais vantagens dos Bancos de Dados que implementam NoSQL são as seguintes: i) leitura e escrita de dados eficiente; ii) suporte ao armazenamento em massa; iii) fácil de expandir; e iv) baixo custo. Vale ressaltar que os Bancos de Dados NoSQL também apresentam algumas desvantagens, que são justamente alguns dos pontos fortes ofertados pelo modelo relacional: i) não suportam a linguagem *Structured Query Language* (SQL) que é padrão da indústria; ii) ausência de transações, relatórios e outros recursos adicionais; e iii) não é um modelo de dados consolidado o suficiente para a maioria dos produtos [Han et al. 2011].

Além das desvantagens do NoSQL supracitadas, Sadalage e Fowler (2013) discutem quais transações em Banco de Dados Relacional permitem a manipulação de

qualquer combinação de linhas e quaisquer tabelas em uma única transação. Tais transações implementam as propriedades chamadas de Atomicidade, Consistência, Isolamento e Durabilidade (ACID) [Neto et al. 2021]. ACID é um acrônimo, onde propriedades como a atomicidade significa que muitas linhas por muitas tabelas são atualizadas em uma única operação. Ou a operação tem sucesso total, ou desfaz totalmente os efeitos produzidos. Operações concorrentes ficam isoladas umas das outras em um ambiente de múltiplos usuários, de modo que não possam ver uma atualização parcial [Sadalage and Fowler 2013]. Note que as transações ACID aumentam a integridade e confiabilidade dos dados, alguns pontos fortes dos Banco de Dados Relacional [Neto et al. 2021].

Em busca por soluções para problemas pontuais, as implementações de Banco de Dados NoSQL acabam se distanciando umas das outras, o que gera diferentes estruturas [Freitas et al. 2015]. As estruturas são associadas aos modelos de dados, que no NoSQL são categorizados em chave-valor, colunas, documentos e grafos. Freitas et al. (2015) fazem um síntese sobre estes modelos:

- **Chave-valor**. É o modelo que possui a estrutura mais simples, onde sua representação estrutural é constituída de uma lista de pares de itens compostos por uma chave e um valor [Freitas et al. 2015]. O modelo chave-valor pode ser comparado à estrutura de dados Tabela *Hash*, onde os valores são indexados às chaves de busca e permitem um rápido acesso aos seus valores. Por ser o mais simples, o modelo chave-valor é o que suporta maior volume de dados e potenciais para escalabilidade [Farias 2014]. No entanto, o modelo chave-valor não permite que consultas sejam realizadas sobre os dados, todo acesso é feito por meio de chaves de busca. Além disso, o modelo chave-valor não agrupa dados por instâncias, todos os dados estão armazenados em uma estrutura simples que é similar a um dicionário, não existindo a possibilidade de formular consultas mais complexas, como subconsultas [Freitas et al. 2015]. São exemplos de banco de dados que utilizam este modelo: OracleNoSQL, Riak, BerkeleyDB e Redis.
- **Colunas**. É o modelo que mais se assemelha ao modelo relacional, pois sua organização é baseada em linhas e colunas [Freitas et al. 2015]. Segundo Freitas et al. (2015) esse modelo de dados não trata os dados de forma normalizada e, geralmente, não visa a consistência das informações. O modelo em colunas facilita a execução de consultas em um subconjunto de dados, mas não permite consultas onde se faz necessário a junção de famílias de colunas, pois este modelo não faz uso de recursos de chaves estrangeiras [Freitas et al. 2015]. São exemplos de banco de dados que utilizam este modelo: BigTable, Cassandra, Hbase.
- **Documentos**. Da mesma forma que o modelo chave-valor, o modelo baseado em documentos também faz uso de associações entre pares de chaves e valores, mas os dados não estão dispostos em uma única estrutura de dados. No modelo de documentos os dados estão agrupados em documentos que podem seguir as regras estruturais do *eXtensible Markup Language* (XML) e *JavaScript Object Notation* (JSON), onde a última é a mais adotada [Freitas et al. 2015]. Um documento pode ser caracterizado como um conjunto de chaves e valores que estão relacionados a uma instância de dados. Assim, vários documentos pertencentes a um mesmo domínio são armazenados em uma coleção de documentos, da mesma forma como as tuplas (linhas) são armazenadas em tabelas (relações) no modelo

relacional [Freitas et al. 2015]. Diferentemente dos demais modelos NoSQL, o modelo baseado em documentos suporta referências e garante a integridade referencial [Freitas et al. 2015]. São exemplos de banco de dados que utilizam este modelo: MongoDB, CouchDB e RavenDB.

- **Grafos**. Em comparação aos modelos NoSQL anteriores, o modelo em grafos é o que mais difere dos demais. Os modelos de dados anteriores visam estratégias de armazenamento de dados, o modelo em grafos se destaca pelos relacionamentos que ocorrem entre entidades de sua base [Freitas et al. 2015]. O modelo em grafo toma como base a teoria dos grafos, onde segue o uso de três tipos de elementos: nós, arestas e propriedades. Os nós representam as instâncias, as arestas correspondem aos relacionamentos entre instâncias e as propriedades dizem respeito aos valores de dados contidos nas instâncias, os quais podem ser, por exemplo, lógicos, inteiros, caracteres e conjuntos de valores [Freitas et al. 2015]. Freitas et al. (2015) discutem que diferente dos outros modelos, a estrutura baseada em grafos suporta o conceito de referências, garantindo a integridade referencial, o que assegura que o nó de entrada sempre faça referência ao nó de saída. São exemplos de banco de dados que utilizam este modelo: Amazon Neptune, Neo4J e ArangoDB.

Apesar das diferentes estruturas implementadas em Banco de Dados NoSQL, existem características que se manifestam de forma predominante em todos estes bancos [Kokay 2015]. Segundo Kokay (2015) existem cinco atributos comuns aos bancos NoSQL que os tornam mais distintos dos bancos relacionais:

- **Escalabilidade Horizontal**. De acordo com o crescimento do volume de dados, evidencia-se a necessidade de melhorias na escalabilidade e no desempenho. Dentro deste contexto, a escalabilidade horizontal se mostra como sendo opção a mais viável. No entanto, esta escalabilidade exige a utilização de diversas *threads* ou a criação e divisão de determinados processos de uma tarefa. Dessa forma, o uso de um banco de dados relacional não se manifesta como sendo a alternativa mais adequada, pois em circunstâncias nas quais diversos processos se conectam simultaneamente em um mesmo conjunto de dados, ocorre uma geração de uma alta concorrência por estas informações, consequentemente aumentando tempo necessário para que seja possível o acesso às tabelas. Os bancos NoSQL, por sua vez, são beneficiados pela ausência destas limitações, dado que eles não são afetados pelo aumento da concorrência, o que possibilita a escalabilidade horizontal com uma maior facilidade e eficiência. Essa possibilidade ocorre devido ao *Sharding*, processo presente nos bancos NoSQL, que divide os dados em múltiplas tabelas, rompendo a cadeia de relacionamentos e realizando o armazenamento destas tabelas de forma distribuída.
- **Ausência ou flexibilidade de esquema**. O aspecto mais perceptível dos bancos NoSQL é a ausência parcial ou total de um esquema que define a estrutura de dados, sendo este traço o fator que facilita uma alta escalabilidade disponibilidade, em detrimento da garantia de integridade dos dados, fenômeno que não ocorre, por exemplo, nos modelos relacionais.
- **Suporte nativo a replicação**. A possibilidade de replicação de dados de forma nativa também é um fator que possibilita a redução do tempo necessário para

recuperar informações, resultando em uma fonte alternativa para prover maior disponibilidade e escalabilidade.

- **API simples para acesso ao banco de dados**. Nos bancos NoSQL, o foco não está na forma em que os dados são armazenadas, mas sim na eficiência existente durante o processo de recuperação destas informações. Portanto, é essencial a existência de API's simples que permitam o fácil acesso aos dados, possibilitando uma utilização rápida e prática do banco de dados.
- **Consistência eventual**. Um dos atributos mais particulares dos bancos NoSQL é a ausência de garantia na consistência dos dados. Este fenômeno tem como base o o Teorema *Consistency, Availability e Partition Tolerance* (CAP) que afirma que nenhuma estrutura de armazenamento de dados é capaz de oferecer, de forma simultânea, as 3 garantias: Consistência, Disponibilidade e Tolerância ao Particionamento [Brito 2010]. Portanto, o foco dos bancos NoSQL em oferecer um acesso rápido à um grande volume de dados ao mesmo tempo em que uma alta disponibilidade é mantida, pode incorrer em prejuízos no que se refere à consistência dos dados.

É possível ainda ressaltar as diferenças existentes entre bancos relacionais e bancos NoSQL, ao comparar seus paradigmas. Os bancos relacionais seguem o paradigma ACID, que contrasta fortemente com o paradigma *Basically Available, Soft State, Eventual Consistency* (BASE), utilizado pelos bancos NoSQL [Pritchett 2008]. O modelo BASE pode ser destrinchado em 3 atributos: 1) Basicamente disponível, estado em que o sistema se apresenta como funcional constantemente. 2) Estado leve, que define uma condição na qual o sistema não precisa ser consistente a todo momento. 3) Eventualmente Consistente, que afirma o sistema se tornará consistente em algum momento.

2.2. MongoDB

O MongoDB[1] é um popular Banco de Dados NoSQL de código-aberto, que implementa o modelo de dados orientado a documentos e frequentemente apreciado em estudos comparativos devido à sua flexibilidade e facilidade de uso, ao mesmo tempo que oferece todos os recursos necessários para atender aos requisitos complexos de aplicativos modernos [Andreoli et al. 2021]. Andreoli et al. (2021) destacam que MongoDB possui uma capacidade de armazenar e servir grandes volumes de dados, distinguindo-se de Banco de Dados Relacionais que não são capazes de alcançar facilmente esse tratamento eficaz [Neto et al. 2021]. Inclusive no trabalho conduzido por Andreoli et al. (2021) é comentado, por meio do estudo conduzido por Rubio et al. (2020), que a velocidade do MongoDB é 10 vezes maior do que a do Banco de Dados Relacional MySQL, quando o volume de dados ultrapassa 50 GB.

O MongoDB faz a gestão de documentos em um formato chamado *binary JSON* (BSON), uma estrutura de serialização codificada em JSON otimizado para velocidade, espaço e flexibilidade [Neto et al. 2021]. Possui uma linguagem de consulta rica e expressiva que permite filtrar e classificar por qualquer campo, não importa o quão aninhado ele possa estar em um documento. As próprias consultas são JSON e, portanto, facilmente combináveis [MongoDB 2021b]. É válido ressaltar que o MongoDB armazena os dados no formato BSON internamente e pela rede, mas isso não significa que não se possa

[1]MongoDB. Disponível em: `https://www.mongodb.com/`. Acessado em: 10 de junho de 2021.

pensar no MongoDB como um Banco de Dados JSON. Qualquer coisa que você possa representar em JSON pode ser armazenado nativamente no MongoDB e recuperado com a mesma facilidade em JSON. Arquivos grandes, como imagens e vídeos que ultrapassam o limite de 16 *megabytes* do formato BSON, também podem ser armazenados através da especificação *GridFS*. [MongoDB 2021a].

Os documentos em MongoDB podem ser armazenados em coleções, onde serão efetuadas operações de busca (*queries*) e de indexação. As *queries* são transmitidas na sintaxe JSON e enviadas ao MongoDB como objetos BSON através de um *driver* de conexão ao banco. A indexação, por sua vez, fornece formas eficientes de acessar e analisar os dados presentes no banco [MongoDB 2021b].

Todos os documentos gerados no MongoDB contêm uma chave única de identificação especial, gerada de forma automática e denominada _id que permite identificar um documento de forma global dentro de uma coleção [Hecht and Jablonski 2011]. Na Figura 1 abaixo, é apresentado um exemplo de um documento no MongoDB, ligeiramente semelhante aos documentos criados durante o desenvolvimento do produto proposto, simulando as características de um determinado patrimônio, de forma que os campos foram especificados como: _id e patrimônio, este último que contém os campos internos nomeados como nome, código, descrição e localização:

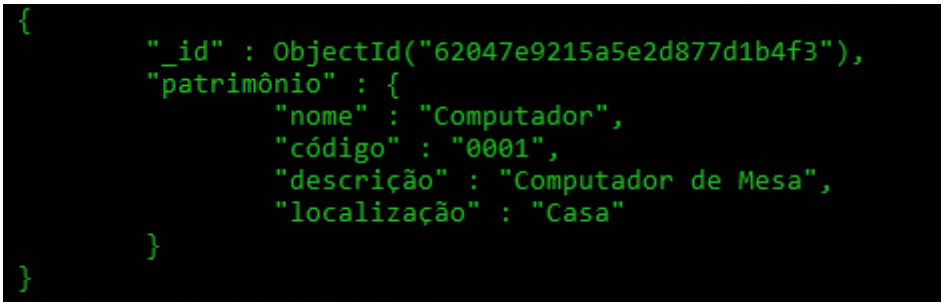

Figura 1. Exemplo de um Documento MongoDB (Fonte: Elaborado pelo Autor)

O esquema flexível, como uma das principais características do MongoDB, permite modificar completamente a estrutura do documento acima, a qualquer momento e da forma que for exigida pela aplicação, sem causar nenhuma inconsistência ao banco de dados, mesmo que esta modificação torne este documento diferente dos demais documentos presentes na coleção ao qual ele pertence [Lóscio et al. 2011].

Nas versões mais recentes do MongoDB, adicionou-se o suporte a transações ACID distribuídas de vários documentos com *Snapshot Isolation*, onde por definição as transações fornecerão uma visão globalmente consistente dos dados e forçarão a execução de tudo ou nada para manter a integridade dos dados [MongoDB 2021b]. O MongoDB suporta redundância e alta disponibilidade por meio da implantação de um conjunto de réplicas e um grupo de instâncias que mantém o mesmo conjunto de dados. Em resumo, o MongoDB permite que os desenvolvedores de aplicações orientadas a dados implementem com facilidade, adaptem-se rapidamente e escalem de forma confiável.

2.3. Armazenamento de Dados em Blockchain

A tecnologia Blockchain fornece suporte a transações distribuídas, confiáveis e seguras aos participantes de um rede *peer-to-peer* (P2P) em larga escala

[Moreira Neto et al. 2020]. Uma transação é o registro de informações de algum ativo na rede. Um ativo pode representar qualquer objeto físico ou digital que contém dados que são imutáveis. Este registro é composto pelo ativo, junto com dados sobre quem enviou, quem recebeu, horário da transação e dentre outros [Neto et al. 2021]. A Figura 2 ilustra um exemplo de arquitetura de redes P2P com a Blockchain. Cada nó computacional possui uma réplica do livro-razão, comumente conhecido como *ledger*, que armazenada todas as transações que ocorreram na Blockchain e todos os nós se comunicam entre si por meio desta arquitetura [Neto et al. 2021].

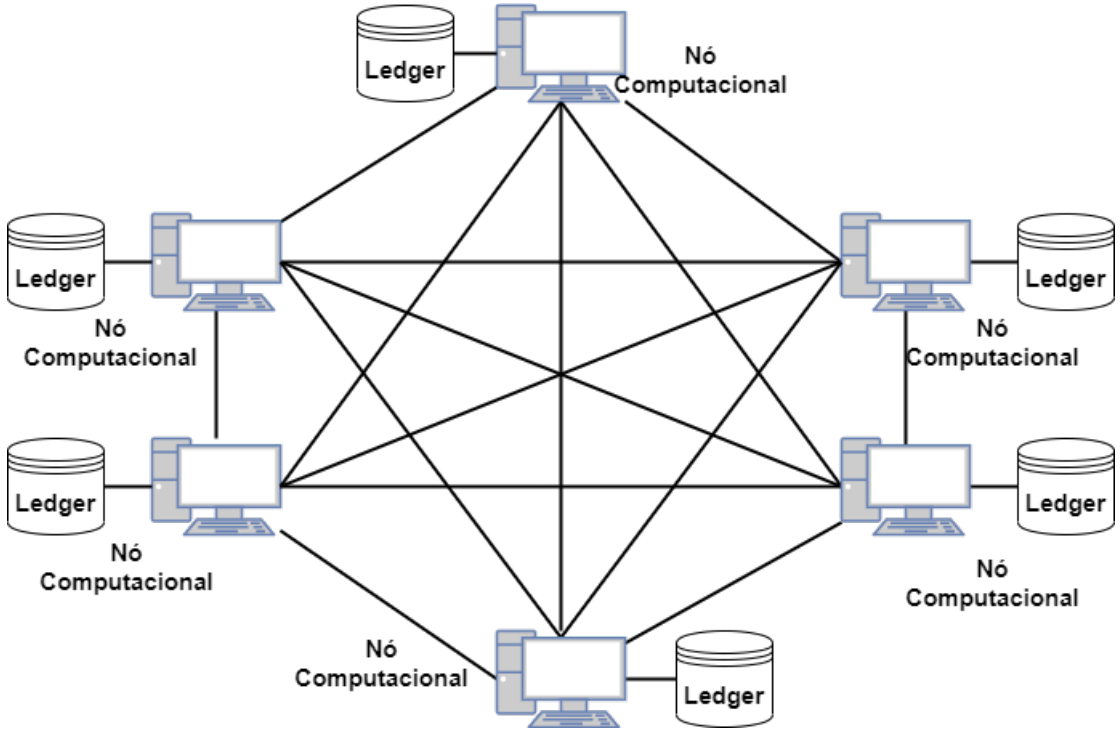

Figura 2. Arquitetura de Redes P2P da Blockchain (Fonte: Adaptado de [Braga et al. 2017, Neto et al. 2021])

A Blockchain é uma tecnologia disruptiva, pois os nós computacionais da rede não confiam necessariamente uns nos outros e não se tem um elemento central para intermediar as transações que ocorrem na rede [Chen et al. 2018], o que lhe distingue de outros sistemas que dependem de entidades centralizadoras como bancos e cartórios para intermediar as transações. A Blockchain é um sistema descentralizado e com alta disponibilidade, pois estará disponível mesmo se alguns nós computacionais estejam desconectados da rede [Yaga and Mell 2018]. Uma das características da Blockchain é a capacidade de auditar as transações realizadas na rede [Chen et al. 2018]. Todas as transações são registradas no livro-razão, e por serem visíveis a todos os membros da rede, é possível inspecioná-los [Yaga and Mell 2018]. Além disso, as transações da Blockchain são imutáveis, ou seja, uma vez que a transação é validada pela rede e adicionada ao *ledger*, não pode ser alterada [Moreira Neto et al. 2020]. A Figura 3 apresenta um exemplo estrutural de uma Blockchain genérica. Cada bloco da Blockchain contém um conjunto de transações e um ponteiro (*hash*) que aponta para o bloco anterior. Esta estrutura de dados é replicada em todos os nós da rede [Neto et al. 2021].

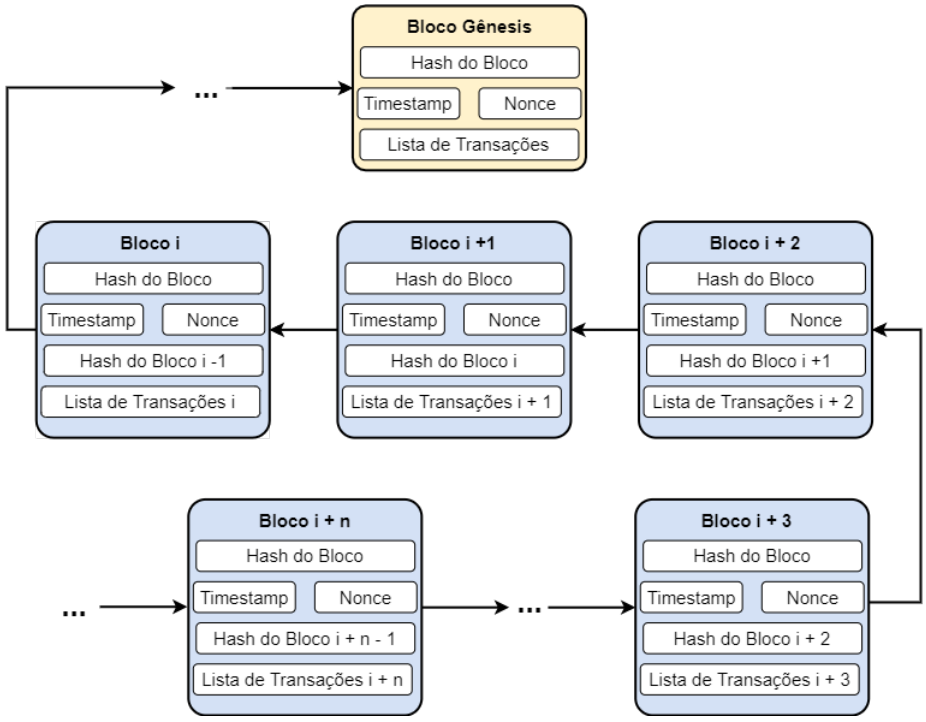

Figura 3. Exemplo de uma Estrutura Blockchain (Fonte: Adaptado de [Nofer et al. 2017, Neto et al. 2021]

De uma forma resumida, seguem as características de uma Blockchain:

- **Disponibilidade e Integridade**. Os nós possuem os dados replicados do *ledger*, o que garante a integridade e a consistência dos dados armazenados na Blockchain;
- **Transparência e Auditabilidade**. Todas as informações contidas no *ledger* são públicas e podem ser analisadas e auditadas;
- **Imutabilidade e Irrefutabilidade**. As informações registradas no *ledger* não podem ser alteradas e as atualizações somente são possíveis a partir da geração de novas transações. O usuário ou cliente da Blockchain, ao inserir uma transação na Blockchain, não pode informar que não foi o responsável pela transação, implementando a característica de irrefutabilidade;
- **Privacidade e Anonimidade**. Greve et al. (2018) enfatizam que cada nó armazena uma parte criptografada de dados dos usuários. A anonimização dos dados transacionados são parciais, já que a visibilidade da transação é pública;
- **Desintermediação**. Greve et al. (2018) comentam que a Blockchain possibilita a integração entre diversos sistemas de forma direta e eficiente; e
- **Cooperação e Incentivos**. O modelo de negócios de uma Blockchain é baseada em incentivos [Greve et al. 2018]. Quando um nó consegue validar um bloco, uma parte do ativo é fornecido a este nó.

2.3.1. Algoritmos de Consenso

A Blockchain utiliza algoritmos de consenso para validar a inserção de um novo bloco [Fan et al. 2020]. Existem diversos algoritmos de consenso, sendo os mais comuns o Prova de Trabalho (Do inglês, *Proof of Work*, PoW) e a Prova de Aposta (Do inglês,

Proof of Stake, PoS) [Yaga and Mell 2018]. A PoW é um mecanismo de consenso descentralizado que exige que os nós computacionais que constituem a rede se esforcem para resolver um problema matemático arbitrário [Yang et al. 2019]. Entretanto, o mecanismo de PoW é computacionalmente oneroso e exige um gasto de energia muito alto dos nós para minerar um único bloco da rede. O mecanismo de PoS surgiu como uma alternativa a PoW [Neto et al. 2021]. O mecanismo de PoS seleciona os validadores por meio da proporção da quantidade de participações deste nó na Blockchain [Niya et al. 2019]. A PoS tem um gasto de energia menor do que o mecanismo PoW [Niya et al. 2019].

2.3.2. Níveis de Acesso

As Blockchains podem ser divididas em dois tipos de acordo com o nível de acesso: Blockchain pública e Blockchain privada [Neto et al. 2021]. A Blockchain pública, também conhecida como Blockchain *permissionless*, é caracterizada pela ausência de permissão e com acesso aberto. Qualquer usuário pode entrar e participar de uma Blockchain pública, além de permitir a leitura e gravação de transações [Aleksieva et al. 2020]. Em contrapartida, as Blockchains privadas, também conhecida como Blockchain *permissioned*, impedem que usuários não autorizados acessem a rede [Behl et al. 2020] [Aleksieva et al. 2020]. Estas Blockchains utilizam controles de acesso por meio de uma entidade central, que é um nó computacional responsável por autorizar o acesso aos usuários, ou seja, apenas usuários que são autorizados a compor a rede podem ler e gravar transações [Aleksieva et al. 2020].

2.3.3. Contratos Inteligentes

Os contratos inteligentes, presentes em Blockchains, são programas, escritos em linguagens de programação, implantados e executados em cada um dos nós computacionais da rede Blockchain [Braga et al. 2017]. Segundo Braga et al. (2017), um contrato inteligente consiste em programas seguros e imparáveis que representam acordos executáveis e exigíveis automaticamente. Os contratos inteligentes resolvem questões ou desafios que necessitam de acordos (consensos) servindo-se de uma mínima confiança entre os nós participantes de sistemas distribuídos [Braga et al. 2017]. Braga et al. (2017) comentam que o programa executável (binário), que representa o contrato inteligente, é implantado e executado por todos os nós computacionais da rede P2P de uma Blockchain e sua execução correta é garantida pelos mecanismos de consenso.

2.4. BigchainDB

O BigchainDB[2] é um Banco de Dados Distribuído, que implementa o gerenciamento de grandes conjuntos de dados ao mesmo tempo que adiciona características de Blockchain, possuindo um controle descentralizado, imutabilidade e transferência de ativos digitais [BigchainDB 2021b]. Um ativo pode representar qualquer objeto físico ou digital que contém dados que são imutáveis [Neto et al. 2021]. Segundo El-Hindi (2019), o BigchainDB se baseia no Banco de Dados NoSQL MongoDB, o que pode ajudar na análise comparativa entre estas duas soluções. O BigchainDB é uma solução que pode ser utilizada por desenvolvedores e organizações que procuram um Banco de Dados consultável com características de Blockchain e a capacidade de tratar qualquer coisa armazenada

[2]BigchainDB. Disponível em: `https://www.bigchaindb.com/`. Acessado em: 10 de junho de 2021.

no Banco de Dados como um ativo [Neto et al. 2021]. Quer sejam átomos, *bits* ou *bytes* de valor [BigchainDB 2021b]. BigchainDB (2021b) sumariza algumas características do BigchainDB:

- **Descentralização**. Não existe um ponto único de controle, assim nenhum ponto único de falha. O controle descentralizado por meio de uma federação de nós de votação contribui para uma rede P2P;
- **Imutabilidade**. É mais do que apenas resistente à violação. Uma vez armazenados, os dados não podem ser alterados ou excluídos;
- **Tolerante a Falhas Bizantinas**. Até um terço dos nós na rede pode estar experimentando falhas arbitrárias e o resto da rede ainda chegará a um consenso no próximo bloco;
- **Consultas**. Escreve e executa qualquer consulta MongoDB para pesquisar o conteúdo de todas as transações, ativos, metadados e blocos armazenados. Funcionalidade desenvolvida pelo próprio MongoDB;
- **Suporte nativo de múltiplos ativos**. Sem moeda nativa no BigchainDB, qualquer ativo, *token* ou moeda pode ser emitido;
- **Baixa latência**. Uma rede global leva cerca de um segundo para chegar a um consenso sobre um novo bloco. Em outras palavras, a finalização da transação acontece rapidamente;
- **Customizável**. Projeta sua própria rede privada com a personalização de ativos, transações, permissões e transparência;
- **Permissão rica**. Permite a definição de permissões no nível da transação para garantir uma separação clara de funções e impor acesso seletivo;
- **Código aberto**. Código aberto para a comunidade para que todos possam usá-lo e construir seus próprios aplicativos a partir dele; e
- **Público ou Privado**. Existe a possibilidade de implementar suas próprias redes públicas ou privadas para casos de uso específicos da indústria.

Apesar do BigchainDB mostrar que pode fornecer um desempenho mais alto do que as soluções de Blockchain nativas, está constantemente sendo criticado por não fornecer as mesmas garantias de confiança e modelo de tolerância a falhas que as Blockchains nativos [El-Hindi et al. 2019].

3. Metodologia

Um objetivo de pesquisa, frequentemente, comporta uma ou mais hipóteses de trabalho. Segundo Wazlawick (2009) um bom objetivo de pesquisa, normalmente, terá a forma de demonstrar que a hipótese elaborada é verdadeira. Para alcançar tal objetivo nesta pesquisa, em seus aspectos metodológicos, envolveram-se os passos discutidos por Wazlawick (2009): i) elaborar um tema de pesquisa que determina uma área de conhecimento na qual se deseja trabalhar; ii) realizar uma revisão bibliográfica ou revisão do estado da arte; e iii) definir o objetivo da pesquisa. Esta primeira etapa da metodologia foi de grande importância na identificação do nível de relevância do tema deste ensaio, tal característica foi contemplada no trabalho preliminar [Neto et al. 2021]. Além disso, ajudou no estabelecimento do escopo da pesquisa por meio das hipóteses e dos objetivos geral e específicos, na presente extensão do trabalho preliminar os objetivos geral e específicos foram para um contexto mais técnico na pesquisa.

Em termos de caracterização, a presente pesquisa se comporta como pesquisa exploratória e experimental. Pesquisas exploratórias têm como objetivo principal o aprimoramento de ideias ou a descoberta de intuições [Gil 2002]. Segundo Gil (2002) grande parte das pesquisas exploratórias envolvem: i) levantamento bibliográfico; ii) entrevistas com pessoas que tiveram experiências práticas com o problema pesquisado; e iii) análise de exemplos que estimulem a compreensão. No trabalho preliminar e herdado pelo presente trabalho, a natureza da pesquisa exploratória foi utilizada para apresentar os conceitos básicos necessários para que se possa entender o problema e o direcionamento da pesquisa, além de descrever, com maiores detalhes, as soluções de persistência de dados escolhidas e os motivos das suas escolhas. Portanto, a segunda etapa da metodologia, herdada pelo trabalho preliminar [Neto et al. 2021], se serviu da pesquisa exploratória para reunir todo arcabouço teórico e os conhecimentos necessários para o entendimento do presente estudo.

A terceira parte da pesquisa, diferentemente do trabalho preliminar que se caracterizava como não-experimental [Neto et al. 2021], o presente trabalho se caracteriza, quanto ao método, em uma pesquisa experimental. Segundo Wazlawick (2009) uma pesquisa experimental implica que o investigador, de forma sistemática, fará alterações no ambiente ou objeto a ser pesquisado de forma a observar se cada intervenção produz os resultados esperados. Sendo assim, o método foi utilizado para desenvolvimento, implementação e validação da aplicação desenvolvida para mostrar a aplicabilidade das características da tecnologia Blockchain nos aspectos de gestão de dados. A partir do desenvolvimento e análise da aplicação, é possível observar como as técnicas de programação que possam implementar características da tecnologia Blockchain mesmo em armazenamentos que não possuem tais características nativas. Portanto, a terceira e última etapa da pesquisa se serviu da pesquisa experimental para reunir informações necessárias para discussão técnica dos aspectos de implementação e das características Blockchain em armazenamentos que não são Blockchain nativos no quesito gestão de dados.

4. Resultados

Nesta seção são apresentados os detalhes do desenvolvimento da aplicação persistente com características de Blockchain implementadas em um armazenamento não nativo na tecnologia Blockchain. Para isso, são apresentados os contextos de uso da aplicação, seus principais requisitos, aspectos de implementação dos requisitos e aspectos funcionais por meio de um guia de telas da aplicação em sua versão final.

A aplicação Gestão Patrimonial[3] tem como finalidade auxiliar uma empresa a realizar o gerenciamento de todo tipo de patrimônio de forma segura, rastreável e auditável. Dessa forma, existem dois públicos a serem atendidos pela aplicação: administradores e usuários públicos. Os administradores podem criar novos patrimônios e também atualizar os existentes, além de alternar a visibilidade desses patrimônios entre público ou privado. Patrimônios definidos como privados não podem ser vistos por usuários públicos. A funcionalidade de privar um item tem por finalidade ocultar patrimônios sigilosos e também possibilita mascarar a exibição de patrimônios ociosos, de forma que se mantenha o seu registro no sistema, a fim de garantir a auditabilidade de todos os itens, uma vez que a

[3]Vídeo demonstrativo disponível em: `https://www.youtube.com/watch?v=ZkEt1uH97Po`

exclusão de um patrimônio pode prejudicar a rastreabilidade do sistema.

Os usuários públicos, por sua vez, não necessitam de *login* e podem visualizar todos os patrimônios públicos presentes no sistema através da página *Home*. Os principais casos de uso e seus respectivos atores podem ser visualizados no diagrama presente na figura 4.

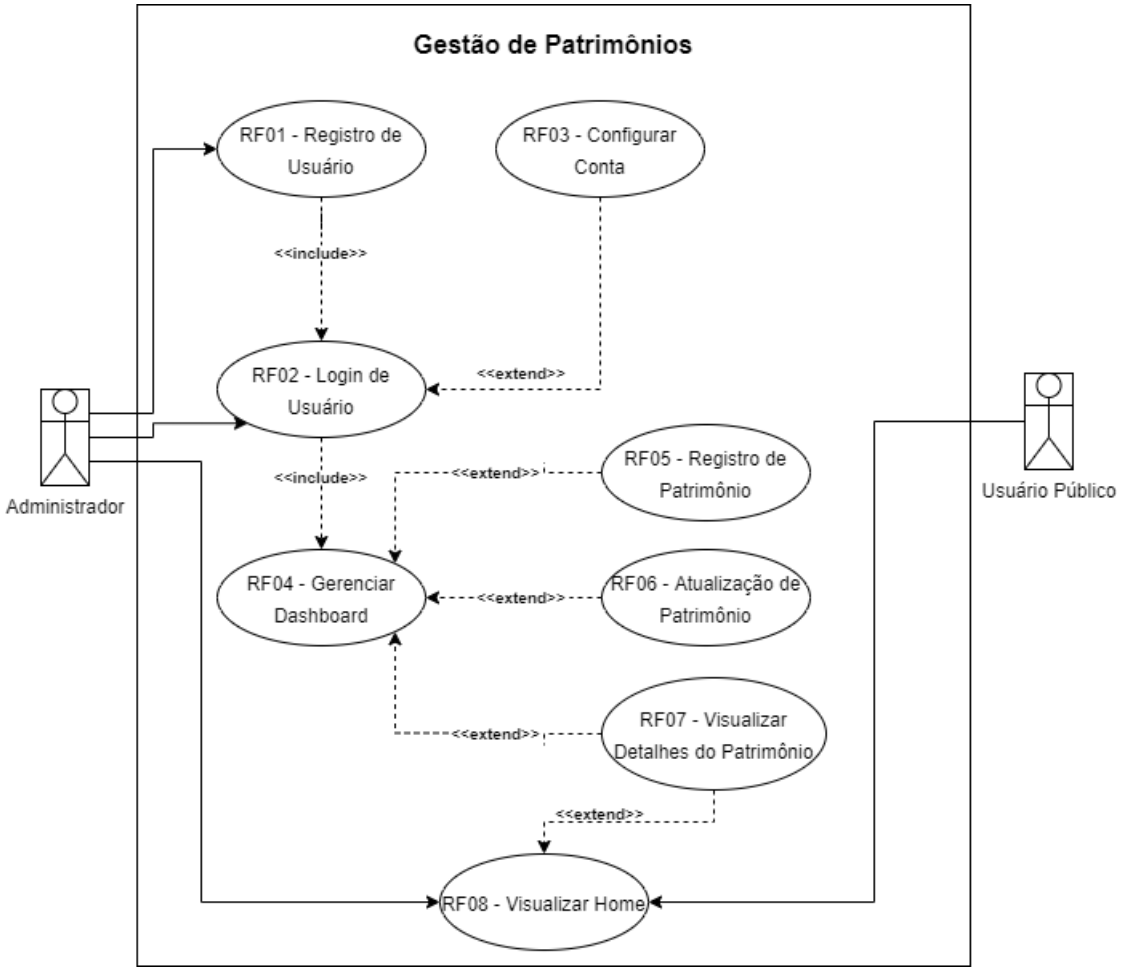

Figura 4. Diagrama de Casos de Uso da Aplicação (Fonte: Elaborado pelo Autor)

4.1. Requisitos da Aplicação de Gestão de Patrimônios

Nesta subseção são descritos os principais requisitos funcionais da aplicação, além de seus pré-requisitos, validações e diferentes comportamentos.

(RF01) Requisito Funcional 1 – Registro de Usuário

Não há nenhum pré-requisito para este requisito. A realização deste requisito funcional permite o cadastramento de um usuário capaz de criar e gerenciar patrimônios na aplicação. Para isso, solicita-se a definição de um nome de usuário, e-mail, senha e a confirmação da senha. Ademais possui 3 validações: i) verifica se o e-mail digitado já está cadastrado na aplicação; ii) verifica se as senhas digitadas nos campos de senha e confirmação de senha são idênticas; e iii) verifica se qualquer um dos campos possui

valor nulo.

(RF02) Requisito Funcional 2 – Login de Usuário

Na realização deste requisito funcional, há a necessidade do pré-requisito descrito no RF01. A realização deste requisito funcional permite o login de um usuário previamente existente. Para isso, solicita-se o preenchimento do campo de e-mail e do campo de senha. Além disso, o requisito faz 3 validações: i) verifica se o e-mail informado consta registrado na aplicação; ii) verifica se a senha inserida é correta; e iii) verifica se qualquer um dos campos possui valor nulo.

(RF03) Requisito Funcional 3 – Configurações de Usuário

Neste requisito, em termos de realização, há somente o pré-requisito que é a necessidade do usuário estar identificado e autenticado (RF02). A realização deste requisito permite que o usuário gerencie as informações da sua conta, possibilitando a atualização de nome, e-mail e senha. Ademais, possui 2 validações: i) caso o e-mail seja atualizado, verifica se o novo e-mail já existe na aplicação; e ii) verifica se os campos de senha e confirmação de senha são idênticos.

(RF04) Requisito Funcional 4 – *Dashboard*

Neste requisito também, em termos de realização, há somente o pré-requisito que é a necessidade do usuário estar identificado e autenticado (RF02). A realização deste requisito permite a visualização, em formato de tabela, de todos os patrimônios públicos e privados que existem registrados na aplicação, além de possibilitar a criação de um novo patrimônio e a atualização ou a exibição detalhada de um patrimônio existente. Ademais, possui 2 comportamentos: i) caso existam patrimônios cadastrados, é exibida uma tabela com os patrimônios; e ii) caso não existam patrimônios cadastrados, exibe uma mensagem que sugere a criação de um novo patrimônio.

(RF05) Requisito Funcional 5 – Registro de Patrimônio

Neste requisito também, em termos de realização, há somente o pré-requisito que é a necessidade do usuário estar identificado e autenticado (RF02). A realização deste requisito permite a criação de um novo patrimônio. Para isso, é solicitado, de forma obrigatória, as definições de um nome de patrimônio, descrição, código de patrimônio, data de movimentação e local de movimentação. Opcionalmente, podem ser submetidas imagens do patrimônio e sua definição como um patrimônio privado ou público através de um *checkbox*. Ademais, possui 2 comportamentos: i) verifica se o código de patrimônio informado já existe no aplicativo; e ii) verifica se um ou mais campos obrigatórios foram informados como nulos.

(RF06) Requisito Funcional 6 – Atualização de Patrimônio

Neste requisito também, em termos de realização, há somente o pré-requisito que

é a necessidade do usuário estar identificado e autenticado (RF02). A realização deste requisito permite a visualização dos dados de um patrimônio e atualização das informações de local do patrimônio e data de movimentação do objeto, que são adicionadas a uma lista junto com os dados anteriores. Também permite a definição do objeto como privado ou público. Não permite adicionar ou sobrescrever novas informações sobre o patrimônio, além do local e data de movimentação.

(RF07) Requisito Funcional 7 – Visualização Detalhada de Patrimônio

Neste requisito também, em termos de realização, há somente o pré-requisito que é a necessidade do usuário estar identificado e autenticado (RF02), caso o patrimônio seja privado. A realização deste requisito permite a visualização das informações de um patrimônio, como: nome, descrição e código de patrimônio, além de um histórico contendo as datas e os locais para onde o patrimônio foi movimentado.

(RF08) Requisito Funcional 8 – *Home* (Início)

Não há pré-requisitos para realização deste requisito. A realização deste requisito permite uma visualização geral de todos os patrimônios públicos do sistema, exibindo uma foto do patrimônio, nome e código de patrimônio. Também permite o redirecionamento para a exibição detalhada de um patrimônio selecionado. Ademais, possui 2 comportamentos: i) caso existam patrimônios, exibe todos os patrimônios públicos registrados na aplicação; e ii) caso não existam patrimônios públicos, exibe uma mensagem que não há patrimônios existentes.

4.2. Implementação das Características de Blockchain no MongoDB na Aplicação Gestão de Patrimônios

Nesta subseção é explorada a arquitetura e a implementação dos requisitos funcionais da aplicação de gestão de patrimônios, aprofundando-se no *back-end* da aplicação e na utilização das ferramentas e comandos do MongoDB.

Arquitetura

A aplicação[4] se caracteriza como uma aplicação web e segue uma arquitetura básica baseada na divisão entre *front-end* e *back-end*. O *front-end* utiliza o *Vue.js* como *framework* para desenvolver as interfaces utilizadas pelo usuário. O backend utiliza a linguagem *JavaScript* em conjunto com o *MongoDB Node Driver* para possibilitar a comunicação com o banco de dados. Além disso, foram utilizadas algumas dependências para facilitar o desenvolvimento do projeto:

- **Bcrypt**: É uma biblioteca para *Node.js* que utiliza um método de criptografia do tipo hash nas senhas presentes na aplicação.
- **Body-Parser**: É um módulo que permite a conversão do *body* das requisições para JSON, formato que é utilizado para realizar a comunicação com o *MongoDB*.

[4]Código fonte disponível em:`https://github.com/cleantoneto/GestaoPatrimonial.git`

- **Express**: É um *framework* utilizado em conjunto com o *Node.js* que permite a criação de diversas rotas e o tratamento de exceções na aplicação.
- **CORS**: É utilizado de forma complementar ao *Express* para garantir que as solicitações requisitadas pelo cliente tenham um funcionamento adequado.
- **JSON Web Token**: É um método utilizado para realizar autenticação entre duas partes por meio de um token assinado que autentica uma requisição web.
- **Mongoose**: É um módulo do *NodeJS* desenvolvido para conectar-se ao *MongoDB* e que permite modelar os dados da aplicação.
- **Multer**: É um *middleware* que é utilizado para realizar o *upload* das imagens presentes na aplicação.

Implementação do (RF01) – Registro de Usuário

Os dados recebidos pelo *front-end* passam por validações e enviam mensagens de erro caso alguma validação não seja atendida. O e-mail informado é verificado através do comando de busca "findOne" do MongoDB, que busca o valor inserido no campo no banco de dados e verifica, em seguida, a pré-existência deste e-mail no banco de dados, enviando uma mensagem de erro caso ele já exista. Em caso de sucesso de um registro de usuário, a senha do usuário é criptografada através de uma função *hash*, por meio da ferramenta *Bcrypt*. Um *hash*, que também é utilizado na blockchain, é uma versão criptografada de uma *string* da qual é impossível derivar a *string* original [Di Pierro 2017]. Também é gerado um *token* em formato JSON, através do *Jsonwebtoken*, para autenticar o usuário em futuras operações. Por fim, os dados recebidos são armazenados em uma estrutura baseada no modelo de dados do usuário, que posteriormente é salva no banco de dados através do método "save" do MongoDB.

Implementação do (RF02) – Login de Usuário

É verificado se o e-mail informado consta no sistema através do comando "findOne" do MongoDB, retornando uma mensagem de erro caso não exista. Em seguida a senha inserida é criptografada e comparada com a senha presente no banco de dados. Caso as senhas sejam iguais, o login é efetuado e o *token* do usuário é recuperado.

Implementação do (RF03) – Configurações de Usuário

Os dados a serem atualizados fornecidos no *front-end* são recebidos pelo *back-end*, sendo posteriormente armazenados em uma estrutura de dados. O id do usuário é recuperado através do *token* de usuário e os dados são atualizados através do comando "findOneAndUpdate" e do operador "set" do MongoDB, onde são passados como parâmetros o id do usuário a ser atualizado e a estrutura contendo os novos dados que serão armazenados.

Implementação do (RF04) – *Dashboard*

Os patrimônios públicos e privados cadastrados pelo usuário a serem exibidos são recuperados por meio do id de usuário associado a cada um deles, através do *token* de usuário. Em seguida, utilizando o comando "find", são listados todos os patrimônios

públicos e também os privados que contêm o id do usuário associado, que posteriormente são inseridos em uma lista e exibidos em forma de tabela no *front-end*.

Implementação do (RF05) – Registro de Patrimônio

Os dados fornecidos pelo *front-end* passam por validações para assegurar que nenhum deles é nulo. Em seguida, é verificado se o código de patrimônio já existe na tabela. O código de patrimônio é definido como único para cada patrimônio, através da criação de índices do MongoDB, por meio do comando "createIndex", que possibilita a restrição de valores únicos para um determinado campo de um documento. Posteriormente, todos os dados passados pela requisição são armazenados em uma estrutura baseada no modelo de dados do patrimônio, que então é inserida no banco através do comando "save" do MongoDB. Caso sejam submetidas imagens durante a criação do patrimônio, o *Multer* é acionado para realizar o armazenamento delas em uma pasta do disco.

Implementação do (RF06) – Atualização de Patrimônio

Os dados fornecidos pelo *front-end* passam por validações para assegurar que nenhum deles é nulo. Em seguida, os dados passados pela requisição são armazenados em uma estrutura baseada no modelo de dados do patrimônio. Posteriormente, ocorre o processo de atualização, utilizando o comando "updateOne" do MongoDB. Para assegurar a rastreabilidade dos dados que não devem ser sobrescritos ou apagados, é utilizado o operador "push" do MongoDB, que seleciona exclusivamente os campos de Local (*locations*) e Data de movimentação (*patrimonyDate*) do patrimônio e adiciona os novos valores passados, através de uma operação de adição em lista, sem sobrescrever os valores antigos. Na figura 5, é possível observar os campos de informação de um patrimônio denominado 'computador' antes de uma operação de atualização.


```
{
        "_id" : ObjectId("622187d133267050280ffdfb"),
        "locations" : [
                "Casa"
        ],
        "patrimonyDate" : [
                "02/02/2022"
        ],
        "photos" : [
                "public\\img\\1646364625870.png"
        ],
        "title" : "Computador",
        "description" : "Computador Pessoal",
        "patrimonyCode" : "0001",
        "privacy" : false,
        "userId" : ObjectId("622186e833267050280ffde7"),
        "__v" : 0
}
```


Figura 5. Exemplo de Patrimônio antes de uma atualização (Fonte: Elaborado pelo Autor)

Os campos essenciais de identificação do patrimônio, como Nome, Descrição e Código do Patrimônio têm sua edição bloqueada no formulário, pois são definidos

Figura 6. Exemplo de Patrimônio depois de uma atualização (Fonte: Elaborado pelo Autor)

como campos restritos a leitura no próprio código *HTML* que constitui o formulário de atualização. Além disso, o comando "updateOne" utilizado não seleciona os campos de identificação para atualização no banco de dados, assegurando a imutabilidade destas informações.

Após a atualização, as novas informações dos campos de Local de Movimentação e Data de Movimentação enviadas na requisição foram adicionadas, ao mesmo tempo que as informações anteriores foram preservadas, gerando um histórico de movimentação e permitindo a rastreabilidade do patrimônio, como observado na Figura 6.

Por fim, o operador padrão de atualização "set" do MongoDB fica reservado para atualizar o nível de permissão do patrimônio, que poderá ser definido como público ou privado. Dessa forma, é garantido que os campos de identificação essenciais do patrimônio não irão sofrer qualquer alteração que possa excluir ou sobrescrever informações.

Implementação do (RF07) – Visualização Detalhada de Patrimônio

É recuperado o id de um patrimônio selecionado no *front-end* por meio do seu endereço único (URL) que também contém o seu id. Logo, é utilizado o comando "findOne" do MongoDB, que realiza a busca do patrimônio através do id do patrimônio e do id do usuário que criou o patrimônio, que é recuperado por meio do seu *token*. Por fim, esta busca retorna todos os campos do patrimônio com o id correspondente, que são exibidos ao usuário pelo *front-end*.

Implementação do (RF08) – *Home* (Inicio)

Através do comando "find" do MongoDB são buscados e filtrados todos os pa-

trimônios cuja privacidade foi definida como pública. Posteriormente, estes patrimônios são inseridos em uma lista que é recuperada no *front-end* para exibição ao usuário.

4.3. Aspectos Funcionais da Aplicação Gestão de Patrimônios

Nesta subseção são apresentadas as principais telas da aplicação desenvolvida e também demonstrados seus comportamentos funcionais.

A Figura 7 apresenta a tela de registro de um novo usuário na aplicação. Caso o e-mail do usuário já esteja cadastrado, é exibido uma mensagem de erro, solicitando um e-mail que não esteja cadastrado. O e-mail é um identificador único do usuário na aplicação. A aplicação faz a validação de máscara de e-mail antes do cadastramento, exibindo um erro caso o e-mail não esteja corretamente formulado. Outro aspecto importante de ser destacado é com relação às senhas, onde a aplicação pede uma confirmação da senha a ser cadastrada. Caso o usuário passe por todas as validações de campos do formulário, uma mensagem de sucesso é exibida e sua conta é registrada na aplicação.

Figura 7. Registro de Usuário - Tela de Registro de Usuário na Aplicação

A ação de efetuar login na aplicação é ilustrada pela Figura 8. Nesta ação, o usuário deve estar registrado para que ele possa ter acesso à aplicação. A tela exibida na figura, implementa validação dois campos pela máscara de e-mail e presença obrigatória da senha. Caso o usuário passe por todas as validações de campos do formulário e tenha informado as credenciais corretas, uma mensagem de sucesso é exibida e o usuário é redirecionado para a página após login.

A tela ilustrada na Figura 9 permite que o usuário possa visualizar e alterar seus dados de perfil. Para isso, é necessário que o usuário possua um *token* de autenticação. O *token* de autenticação é obtido quando o usuário efetua o login corretamente na aplicação. Caso o usuário passe por todas as validações de campos do formulário, uma mensagem de sucesso é exibida ao usuário.

Na Figura 10 é mostrada a tela após a identificação e autenticação do usuário. Na tela é mostrada o comportamento onde não existem patrimônios registrados na aplicação. Já na Figura 11 mostra o comportamento da mesma tela que existem patrimônios registrados na aplicação. Ainda nesta tela é possível ver as opções de atualizar os registros de cada patrimônio registrado, por meio do botão “Atualizar”, e a possibilidade de cadastrar um novo patrimônio por meio do botão “Cadastrar Patrimônio”

Figura 8. Login - Tela de Login do Usuário na Aplicação

Figura 9. Configurações - Tela de Visualização e Alteração dos Dados do Perfil do Usuário

Figura 10. *Dashboard* - Tela de Listagem de Patrimônios Públicos e Privados do Usuário

A tela representada pela Figura 12 é o conteúdo apresentado quando o usuário, identificado e autenticado na aplicação, deseja registrar um novo patrimônio. Na tela, o usuário precisa entrar com dados de identificação do patrimônio, como por exemplo:

Figura 11. *Dashboard* - Tela de Listagem de Patrimônios Públicos e Privados do Usuário

título, descrição, código, data de movimentação, local de movimentação, imagens do patrimônio e se é um patrimônio privado.

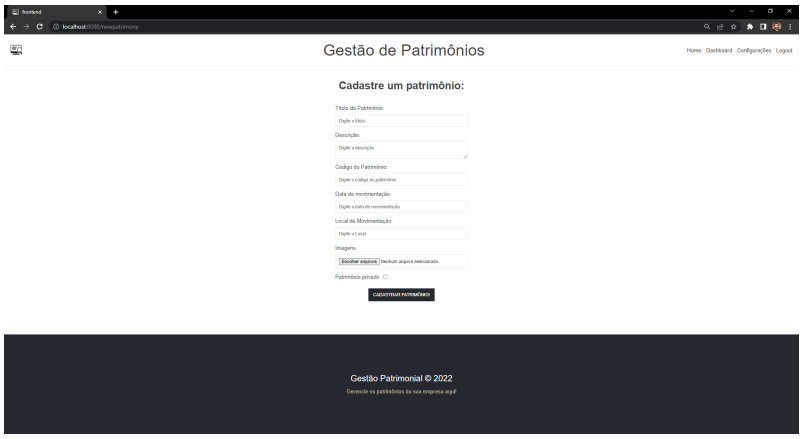

Figura 12. Registro de Patrimônio - Tela que Permite o Cadastro de Patrimônios

Já a tela ilustrada na Figura 13 representa a ação de atualizar os dados de um patrimônio existente na aplicação. Na tela é possível alterar os dados apresentados na tela da Figura 12, mantendo os dados de identificação. Por meio da atualização, é possível saber a localização atual do patrimônio e quando ele foi realocado. Assim, mantendo o histórico de localizações.

A Figura 14 exibe a tela que apresenta os detalhes do patrimônio por meio dos dados cadastrados e atualizados. Como é possível perceber, a tela mantém o histórico dos locais onde patrimônio esteve alocado e também as respectivas datas de realocações. É válido ressaltar a importância desta tela, pois para fins de aspectos de auditabilidade de patrimônios de uma instituição, sabe-se onde ele esteve e o período que ficou alocado em determinado local.

A tela apresentada na Figura 15 apresenta os patrimônios recentes que foram registrados na aplicação. Nesta tela é possível ver uma imagem que representa o patrimônio, seu título, código e um botão “Ver Mais” para ter acesso a todos os dados associados ao

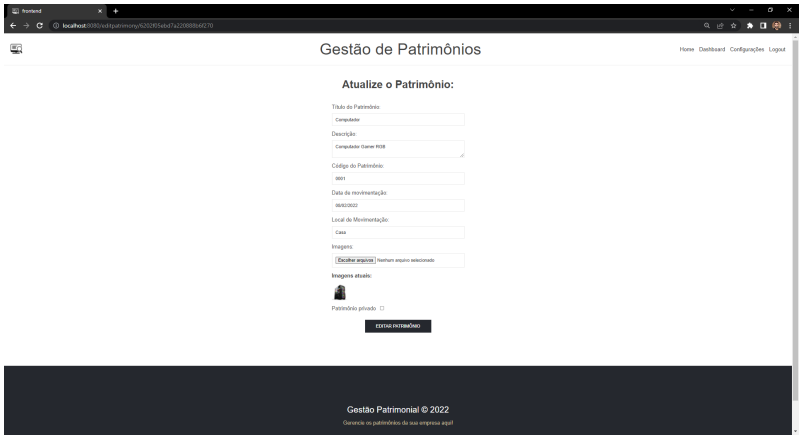

Figura 13. Atualização de Patrimônio - Tela que Permite a Atualização de Patrimônios

Figura 14. Visualização de Patrimônio - Tela que Permite a Visualização de Patrimônios

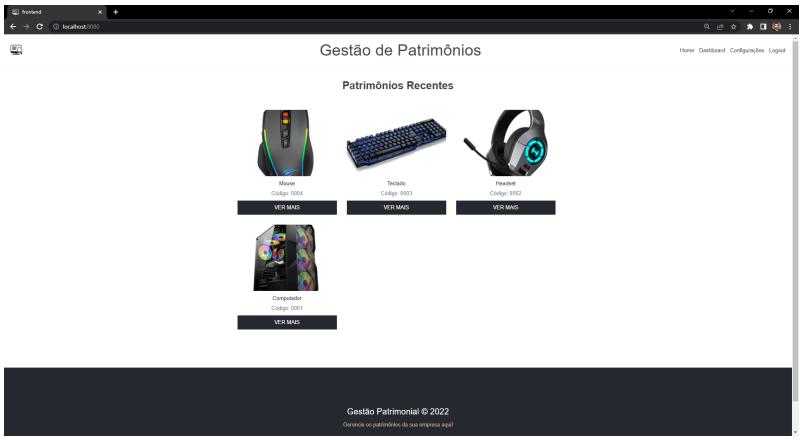

Figura 15. *Home* (Início) - Visualização de Patrimônios Cadastrados

tal patrimônio.

Por fim, a tela ilustrada na Figura 16 demonstra a situação, comentada na tela da Figura 15, onde não existem patrimônios registrados na aplicação.

Figura 16. *Home* (Início) - Quando não Existe Patrimônio Cadastrado

5. Conclusão

Este trabalho apresentou o desenvolvimento de uma aplicação persistente com características Blockchain em armazenamento não Blockchain nativo. Para isso, foram apresentados conceitos teóricos pertinentes ao modelos de dados Relacional, modelo de dados NoSQL e a tecnologia Blockchain. Além disso, foram destacados e detalhados o Sistema Gerenciador de Banco de Dados MongoDB e o armazenamento de dados em Blockchain por meio do BigchainDB. Em seguida, a metodologia do presente trabalho foi destacada e os métodos de pesquisa foram justificados. Por fim, na Seção 4 foi destacado objetivo da aplicação desenvolvida, seus aspectos funcionais e detalhes de implementação que se assemelharam as características da tecnologia Blockchain na perspectiva da gestão de dados.

A pesquisa conseguiu responder a questão descrita na Seção 1: A pergunta consistia em saber como se dá o processo de desenvolvimento de uma aplicação persistente com características da tecnologia Blockchain. Por meio da Seção 4, por meio da descrição da aplicação, destaque dos requisitos e os aspectos técnicos e funcionais, pode-se perceber que, mesmo utilizando um armazenamento Blockchain não nativo, algumas características como rastreabilidade e auditabilidade foram preservadas. O princípio da imutabilidade é parcialmente atendido pela aplicação, tendo em vista que as operações de atualização resultam necessariamente na modificação de alguns campos de informações (data e local de movimentação) dos patrimônios, mesmo que não ocorra perda ou sobrescrita dos dados anteriores. Já os outros campos de identificação do patrimônio, como nome, descrição e código de patrimônio se mantêm consistentemente imutáveis. Por fim, a hipótese do presente trabalho foi possível de ser comprovada por meio dos resultados da Seção 4, onde uma aplicação de gestão de patrimônio foi desenvolvida, requisitos identificados e aspectos de técnicas de aplicação mostraram que, mesmo não utilizando um armazenamento Blockchain nativo, algumas características de Blockchain foram preservadas.

Em trabalhos futuros, almeja-se:

- refatorar a aplicação de gestão de patrimônios e fazer sua implementação em uma infraestrutura de gestão de dados Blockchain por meio do BigchainDB, fazendo uma avaliação comparativa na perspectiva do desenvolvedor;

- realizar uma avaliação de desempenho diante dos cenários do crescimento de usuários simultâneos, crescimento do *dataset* e variação da taxa de transações de escrita; e
- explorar alternativas ao BigchainDB para que se possa avaliar outras formas de armazenamento de dados na tecnologia Blockchain.

Referências

Aleksieva, V., Valchanov, H., and Huliyan, A. (2020). Smart contracts based on private and public blockchains for the purpose of insurance services. In *2020 International Conference Automatics and Informatics (ICAI)*, pages 1–4.

Andreoli, R., Cucinotta, T., and Pedreschi, D. (2021). RT-MongoDB: A NoSQL Database with Differentiated Performance. In Helfert, M., Ferguson, D., and Pahl, C., editors, *Proceedings of the 11th International Conference on Cloud Computing and Services Science (CLOSER)*, pages 77–86. SciTePress.

Behl, D., Kodeswaran, P., Ramakrishna, V., Sen, S., and Vinayagamurthy, D. (2020). Trusted data notifications from private blockchains. In *2020 IEEE International Conference on Blockchain (Blockchain)*, pages 53–61.

BigchainDB (2021a). BigchainDB - The Blockchain Database. Disponível em: https://www.bigchaindb.com/. Acessado em 06 de junho de 2021.

BigchainDB (2021b). Features & Use Cases - BigchainDB. Disponível em: https://www.bigchaindb.com/features/. Acessado em 12 de junho de 2021.

Braga, A., Marino, F., and Santos, R. (2017). *Segurança de Aplicações Blockchain Além das Criptomoedas*, chapter 3, pages 99–148. SBC. Minicursos do XVII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg).

Brito, R. W. (2010). Bancos de dados nosql x sgbds relacionais: Análise comparativa. Disponível em: https://xdocs.com.br/doc/27-05-s4-1-68840-bancos-de-dados-nosql1-j9874megmw8z. Acessado em 10 de fevereiro de 2022.

Chen, W., Xu, Z., Shi, S., Zhao, Y., and Zhao, J. (2018). A survey of blockchain applications in different domains. In *Proceedings of the 2018 International Conference on Blockchain Technology and Application*, ICBTA 2018, page 17–21, New York, NY, USA. Association for Computing Machinery.

Di Pierro, M. (2017). What is the blockchain? *Computing in Science Engineering*, 19(5):92–95.

El-Hindi, M., Binnig, C., Arasu, A., Kossmann, D., and Ramamurthy, R. (2019). BlockchainDB: A Shared Database on Blockchains. *Proc. VLDB Endow.*, 12(11):1597–1609.

Elmasri, R. and Navathe, S. B. (2011). *Sistemas de Banco de Dados*. Pearson Education Brasil, 6a. edition.

Fan, C., Ghaemi, S., Khazaei, H., and Musilek, P. (2020). Performance evaluation of blockchain systems: A systematic survey. *IEEE Access*, 8:126927–126950.

Farias, F. A. d. M. (2014). Avaliação de Desempenho entre Bancos de Dados Relacionais e NoSQL. Monografia de Graduação. Bacharel em Sistemas de Informação, Centro de Ciências Aplicadas à Educação, Universidade Federal da Paraíba, Rio Tinto, Brasil.

Freitas, M. C., Souza, D. Y., and Salgado, A. C. (2015). Mapeamentos conceituais entre os modelos relacional e nosql: Uma abordagem comparativa. *Revista Principia - Divulgação Científica e Tecnológica do IFPB*, 1(28):37–50.

Furtado, F. R. (2019). L7sp: Serviços para otimizar o gerenciamento e o desempenho de blockchains privados. Master's thesis, Universidade do Vale do Rio dos Sinos.

Gil, A. C. (2002). *Como Elaborar Projetos de Pesquisa*. Atlas, São Paulo, 4a. edition.

Greve, F., Sampaio, L., Abijaude, J., Coutinho, A., Valcy, I., and Queiroz, S. (2018). *Blockchain e a Revolução do Consenso sob Demanda*, chapter 5, pages 1–52. SBC. Minicursos do XXXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC).

Han, J., E, H., Le, G., and Du, J. (2011). Survey on nosql database. In *2011 6th International Conference on Pervasive Computing and Applications*, pages 363–366.

Hecht, R. and Jablonski, S. (2011). Nosql evaluation: A use case oriented survey. In *2011 International Conference on Cloud and Service Computing*, pages 336–341.

Kokay, M. C. (2015). Banco de dados nosql: Um novo paradigma. *Revista SQL Magazine*. Disponível em: https://www.devmedia.com.br/banco-de-dados-nosql-um-novo-paradigma-revista-sql-magazine-102/25918. Acessado em 10 de fevereiro de 2022.

Lóscio, B., Oliveira, H. R., and Pontes, J. C. d. S. (2011). *Blockchain e a Revolução do Consenso sob Demanda*, pages 1–17. SBC. Minicursos do VIII Simpósio Brasileiro de Sistemas Colaborativos (SBSC).

Marinho, C. S. S., Costa Filho, J. S., Moreira, L. O., and Machado, J. C. (2020). Using a Hybrid Approach to Data Management in Relational Database and Blockchain: a Case Study on The E-health Domain. In *2020 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 114–121.

Marinho, C. S. S., Moreira, L. O., Coutinho, E. F., Costa Filho, J. S., Sousa, F. R. C., and Machado, J. C. (2018). Labareda: A predictive and elastic load balancing service for cloud-replicated databases. *Journal of Information and Data Management*, 9(1):94–106.

MongoDB (2021a). JSON and BSON - MongoDB. Disponível em: https://www.mongodb.com/json-and-bson/. Acessado em 11 de junho de 2021.

MongoDB (2021b). The most popular database for modern apps - MongoDB. Disponível em: https://www.mongodb.com/. Acessado em 06 de junho de 2021.

Moreira, L. O. (2014). *Abordagem para Qualidade de Serviço em Banco de Dados Multi-Inquilinos em Nuvem*. PhD thesis, Doutorado em Ciência da Computação, Centro de Ciências, Universidade Federal do Ceará, Fortaleza, Brasil.

Moreira Neto, M., Marinho, C. S. S., Coutinho, E. F., Moreira, L. O., Machado, J. C., and Souza, J. N. (2020). Research Opportunities for E-health Applications with DNA Sequence Data using Blockchain Technology. In *2020 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 95–102.

Neto, J. C. F. A., Araújo, F. L. N., Neto, M. M., Paillard, G. A. L., and Moreira, L. O. (2021). Um Estudo Exploratório entre Banco de Dados NoSQL e Armazenamento de Dados em Blockchain. *Revista Sistemas e Mídias Digitais (RSMD)*, 6(1).

Niya, S. R., Schiller, E., Cepilov, I., Maddaloni, F., Aydinli, K., Surbeck, T., Bocek, T., and Stiller, B. (2019). Adaptation of proof-of-stake-based blockchains for iot data streams. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 15–16.

Nofer, M., Gomber, P., Hinz, O., and Schiereck, D. (2017). Blockchain. *Business and Information Systems Engineering*, 59(3):183–187.

Pritchett, D. (2008). Base an acid alternative. *ACM Queue*, 6:48–55.

Rubio, F., ., P. V., and Reyes Ch, R. P. (2020). Nosql vs. sql in big data management: An empirical study. *KnE Engineering*, 5(1):40–49.

Sadalage, P. J. and Fowler, M. (2013). *NoSQL Essencial: Um Guia Conciso para o Mundo Emergente da Persistência Poliglota*. Novatec.

Soares, B. E. and Boscarioli, C. (2013). Modelo de banco de dados colunar: Características, aplicações e exemplos de sistemas. In *IX Escola Regional de Banco de Dados (ERBD)*, pages 1–10.

Wazlawick, R. S. (2009). *Metodologia de Pesquisa para Ciência da Computação*. Elsevier Editora, Rio de Janeiro, 1a. edition.

Yaga, D. and Mell, P. (2018). *NISTIR 8202: Blockchain Technology Overview*, chapter 1, pages 1–68. National Institute of Standards and Technology.

Yang, X., Chen, Y., and Chen, X. (2019). Effective scheme against 51% attack on proof-of-work blockchain with history weighted information. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 261–265.

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

A797d Arrais Neto, José Cleanto Feitosa.
Desenvolvimento de uma Aplicação Persistente com Características da Tecnologia Blockchain / José Cleanto Feitosa Arrais Neto. – 2022.
26 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Instituto UFC Virtual, Curso de Sistemas e Mídias Digitais, Fortaleza, 2022.
Orientação: Prof. Dr. Leonardo Oliveira Moreira.

1. Banco de Dados. 2. NoSQL. 3. Blockchain. I. Título.

CDD 302.23
