UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

João Luiz Monteiro Joaquim

Arquitetura para Data Lakes Adequada à Privacidade de Dados no Contexto da GDPR

Florianópolis
2022

João Luiz Monteiro Joaquim

Arquitetura para Data Lakes Adequada à Privacidade de Dados no Contexto da GDPR

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina para a obtenção do título de mestre em Ciência da Computação.
Orientador: Ronaldo dos Santos Mello, Dr.

Florianópolis
2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Joaquim, João Luiz
Proposta de uma Arquitetura para Data Lakes Adequada à Privacidade de Dados no Contexto da GDPR / João Luiz Joaquim ; orientador, Ronaldo Mello, 2022.
95 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, , Programa de Pós-Graduação em Ciência da Computação, Florianópolis, 2022.

Inclui referências.

1. Ciência da Computação. 2. Data Lake. 3. Privacidade de Dados. 4. General Data Protection Regulation. 5. GDPR. I. Mello, Ronaldo. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. III. Título.

João Luiz Monteiro Joaquim

Arquitetura para Data Lakes Adequada à Privacidade de Dados no Contexto da GDPR

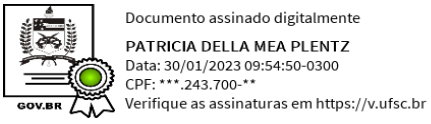
O presente trabalho em nível de mestrado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Profa. Carla Merkle Westphall, Dra.
Universidade Federal de Santa Catarina - PPGCC

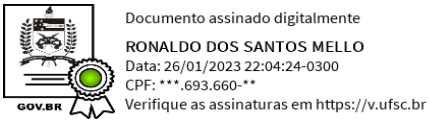
Prof. Vinicius Medina Kern, Dr.
Universidade Federal de Santa Catarina - PGCIN

Prof. Roberto Willrich, Dr.
Universidade Federal de Santa Catarina - INE

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de mestre em Ciência da Computação.



Coordenação do Programa de
Pós-Graduação



Ronaldo dos Santos Mello, Dr.
Orientador

Florianópolis, 2022.

AGRADECIMENTOS

Aos meus pais e minha avó, que me acompanharam por toda a jornada, não me abandonando em nenhum momento. Agradeço pela paciência, pelas palavras de apoio e pelo precioso suporte nos momentos mais difíceis.

Ao meu orientador, professor Ronaldo dos Santos Mello, pelos ensinamentos, conselhos e incansável apoio oferecido desde os primeiros momentos do curso de mestrado.

Ao grupo de inteligência de dados do Ciasc - Centro de Informática do Estado de Santa Catarina - pelo breve período no qual trabalhei que me ensinaram muito sobre o tema Big Data, ferramentas e como montar o cluster Hadoop para essa dissertação.

A Ahgora Sistemas S/A pelo tempo disponibilizado para frequentar as aulas, seminários e outros eventos desse mestrado.

Aos professores Carla Merkle Westphall e Frank Siqueira, meu agradecimento por aceitarem fazer parte da banca de qualificação de mestrado e pelos conhecimentos transmitidos.

Aos professores Carla Merkle Westphall, Vinicius Medina Kern e Roberto Willrich, meu agradecimento por aceitarem fazer parte da banca examinadora e pelos conhecimentos transmitidos.

Meus agradecimentos especiais à Universidade Federal do Santa Catarina (UFSC) pelo importante apoio ao meu desenvolvimento intelectual e profissional e me ajudando a alcançar boas oportunidades profissionais.

RESUMO

A utilização em massa de dispositivos eletrônicos nos últimos anos causou um aumento expressivo no volume, velocidade e variedade de geração de dados digitais sem o devido controle de privacidade. Este fenômeno fez surgir o conceito de *Big Data*, que consiste em conjuntos de dados extensos que exigem uma arquitetura computacional escalável para armazenamento, manipulação e análise eficientes. Vinculado ao conceito de *Big Data*, outro conceito recente é o de *Data Lake* (DL). Um DL é um repositório de dados heterogêneos gerados ou coletados de diversas fontes, como redes de sensores ou redes sociais, em formato nativo. Contudo, mesmo a literatura apresentando propostas de arquiteturas para DLs, elas não consideram questões relacionadas à privacidade dos dados. Logo, o objetivo dessa dissertação é estender uma arquitetura de referência bastante referenciada para que ela atenda questões relacionadas à privacidade de dados utilizando como referência a *General Data Protection Regulation* (GDPR) como legislação base sobre o assunto. Os resultados dessa dissertação possibilitam organizações estarem mais adequadas às legislações sobre privacidade de dados, tendo como benefícios o atendimento delas em camadas da arquitetura.

Palavras-chave: *Data Lake*. Privacidade de Dados; *General Data Protection Regulation*; GDPR.

ABSTRACT

The mass use of electronic devices in recent years has caused a significant increase in the volume, speed, and variety of digital data generation without proper privacy control. This phenomenon gave rise to Big Data, which consists of extensive data sets that require a scalable computational architecture for efficient storage, manipulation, and analysis. Linked to the concept of Big Data, another recent concept is the Data Lake (DL). A DL is a repository of heterogeneous data generated or collected from different sources, such as sensor networks or social networks, in native format. However, even the literature presenting proposed architectures for DLs does not consider data privacy issues. Therefore, this dissertation aims to extend a well-referenced reference architecture so that it addresses issues related to data privacy using the General Data Protection Regulation (GDPR) as a base legislation on the subject. The results of this dissertation make it possible for organizations to be adequate to the legislation on data privacy, with the benefits of meeting them in layers of the architecture.

Keywords: Data Lake. Data Privacy; *General Data Protection Regulation*; GDPR.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A utilização em massa de dispositivos eletrônicos nos últimos anos (por exemplo, *smartphones*, *tablets* e *notebooks*) em domínios de aplicação como Internet das coisas e redes sociais, causou um aumento expressivo no volume, velocidade e variedade de geração de dados digitais. Este fenômeno fez surgir o conceito de *Big Data* que, segundo Nist (2018), consiste em conjuntos de dados extensos que exigem uma arquitetura computacional escalável para armazenamento, manipulação e análise eficientes. O desenvolvimento de tal arquitetura é uma tarefa complexa que introduz novos desafios de pesquisa à comunidade de Banco de Dados (BD), como técnicas de processamento de consultas e de correlação de dados com alto desempenho.

Vinculado ao conceito de *Big Data*, outro conceito recente é o de *Data Lake* (DL). Um DL é um repositório de dados heterogêneos gerados ou coletados de diversas fontes, como redes de sensores ou redes sociais, em formato nativo. Muitas vezes denominado de repositório de *Big Data*, um DL é útil principalmente para a realização de procedimentos de análise de dados que visam revelar conhecimento útil para tomadas de decisão em uma Organização (MADERA; LAURENT, 2016). O gerenciamento de um DL é um tópico de pesquisa em aberto na área de BD e, dentre os desafios associados, um deles é o gerenciamento da privacidade dos dados.

De maneira geral, o controle de privacidade dos dados tem recebido pouca atenção na literatura considerando sua aplicação no contexto de DLs (FANG, 2015; BARANOVIC, 2018). A ausência deste controle, entretanto, pode gerar diversos problemas, como o frequente vazamento de milhões de dados pessoais no mundo, dados esses presentes em repositórios *Big Data*, como aqueles mantidos pelas redes sociais (CHAUHAN; SOOD, 2021). No contexto de DL, este problema tende a ser mais complexo, pois um dado pessoal de uma fonte *X* pode estar correlacionado a um dado de uma fonte *Y*, oferecendo mais subsídios para a descoberta de um indivíduo por agentes mal intencionados. Para minimizar esse problema, algumas leis têm surgido. Essas leis obrigam empresas de tecnologia a adotarem políticas de segurança de dados pessoais. Dentre essas leis, a que mais se destaca é a *General Data Protection Regulation* (GDPR), criada pelo Parlamento Europeu com o objetivo de garantir uma maior proteção do direito individual dos cidadãos da União Europeia (GDPR, 2019).

Ao analisar trabalhos que mencionam privacidade de dados, big data e DLs temos os trabalhos do próprio autor dessa dissertação (JOAQUIM; SANTOS MELLO, 2020), Chen, Chen e Huang (CHEN, Y.-H.; CHEN, H.-H.; HUANG, 2018) e Baranovic et. al. (BARANOVIC, 2018). O primeiro trabalho realiza uma comparação das tecnologias de confidencialidade e descreve onde as funções de segurança devem estar alocadas considerando uma arquitetura de referência. No segundo trabalho é proposto uma estrutura para que DLs públicos possam controlar a privacidade dos dados no

processo de compartilhamento dos dados. No terceiro trabalho descreve uma forma de identificar determinado dado pessoal para um caso específico. Portanto, é perceptível a necessidade de mais trabalhos com maior profundidade nesses assuntos e proponha uma arquitetura em camadas para gestão de privacidade em DL, desde a ingestão até a interação, sendo essas limitações os diferenciais desta dissertação.

Logo, o desafio de pesquisa a ser tratado nesta dissertação é: como realizar a gestão de privacidade de dados no contexto de um *DL* de acordo com normas reguladoras, em especial a GDPR? Para lidar com este desafio, essa dissertação visa propor a extensão de uma arquitetura de DL preexistente, no caso utilizamos como referência de Quix e Hai (2018), que seja adequada aos requisitos que as leis relacionadas à privacidade de dados definem, utilizando como referência a GDPR.

1.1 OBJETIVOS

O objetivo geral desta dissertação é o desenvolvimento de uma arquitetura de DL que atenda os requisitos regulatórios de privacidade de dados utilizando como referência a GDPR. Para tanto, os seguintes objetivos específicos devem ser alcançados: *(i)* Identificar os requisitos que a GDPR define para o tratamento da privacidade dos dados; *(ii)* Propor uma arquitetura de DL adequado aos requisitos da GDPR; *(iii)* Testar a proposta arquitetural em relação aos requisitos levantados através de um estudo de caso.

1.2 CONTRIBUIÇÕES

A principal contribuição científica desta dissertação é uma proposta arquitetural para um sistema de gerência de DL que leve em consideração a privacidade dos dados do usuário, utilizando como referência a GDPR. Desta forma, organizações estarão em conformidade com as normas reguladoras referentes à privacidade de dados. Além disso, em contraste com o trabalho de Baranovic (BARANOVIC, 2018), cujo foco está somente numa parte da arquitetura de DLs de Quix e Hai (2018), nessa dissertação o objetivo é ir além dessa tarefa, abordando todas as partes que uma arquitetura representa.

1.3 METODOLOGIA

Esse estudo tem por finalidade desenvolver um projeto de pesquisa aplicada, uma vez que utilizará conhecimento da pesquisa básica para resolver problemas.

Para um melhor tratamento dos objetivos e melhor apreciação desta pesquisa, observa-se que ela é classificada como pesquisa qualitativa e exploratória. Detectou-se também a necessidade da pesquisa bibliográfica. A pesquisa bibliográfica implica

que os dados e informações necessárias para realização da pesquisa sejam obtidos a partir de abordagens já trabalhadas por outros autores através de livros, artigos, *surveys* e documentos eletrônicos, entre outras fontes (DA SILVA; MENEZES, 2005), com vistas à aquisição de conhecimento sobre DLs e privacidade de dados.

A primeira etapa da metodologia aqui seguida consistiu em um levantamento do estado da arte sobre os temas DLs, segurança da informação, privacidade de dados e *GDPR*. A segunda etapa realizou um levantamento de trabalhos relacionados ao tema dessa dissertação, com foco na busca por termos e privacidade de dados. Por fim, na terceira etapa foi descrita uma proposta de melhorias em relação à privacidade de dados considerando uma arquitetura de referência estudada na primeira etapa.

1.4 ORGANIZAÇÃO DA DISSERTAÇÃO

Este trabalho está organizado em 8 capítulos. O capítulo 2 apresenta a fundamentação teórica a respeito de Big Data, DLs, Segurança da Informação, Privacidade de dados, Anonimização de Dados, GDPR e LGPD. O capítulo 3 apresenta os trabalhos relacionados a esta dissertação. O capítulo 4 descreve a proposta arquitetural de um DL adequado a GDPR. O capítulo 5 descreve os dois esquemas de metadados necessários para a dissertação. No capítulo 6, descrevemos as tecnologias escolhidas para atender a proposta arquitetural adequada a GDPR. O capítulo 7 apresenta um estudo de caso para validar a proposta arquitetural dessa dissertação. Por fim, a conclusão do trabalho, sintetizamos os principais pontos tratados nos capítulos anteriores e resumindo as contribuições proporcionadas. Além do mais, também são listados os trabalhos futuros referente a arquitetura do DLMS para atendimento da GDPR.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos utilizados para fundamentar a proposta desta dissertação. Inicialmente, aborda-se DL, detalhando as características e arquiteturas de um sistema de gerência de DL. Além disso, aborda-se conceitos relacionados à segurança e privacidade de dados. Por fim, apresenta-se a GDPR.

2.1 DATA LAKE

Com o surgimento de novas tecnologias de informação e comunicação, como dispositivos móveis, redes sociais, entre outros, levou ao aumento do volume de dados gerados em grande escala, criando demandas por novas estratégias de gerenciamento de dados. Neste contexto, *Big Data* surge como uma linha de pesquisa destinada a oferecer novas maneiras de armazenar, manipular e extrair valor das informações. Para Cavanillas, Curry e Wahlster (2015), a capacidade de gerenciar *Big Data* é vista, por parte da indústria, como uma vantagem competitiva essencial para os seus negócios, incluindo a extração de conhecimento útil a partir dos dados e *insight* de novos negócios.

Sobre a definição do termo *Big Data*, Cavanillas, Curry e Wahlster (2015) ressaltam que existem diversas propostas. Para essa dissertação, consideramos que *Big Data* corresponde a dados com alto volume, alta velocidade de consumo e alta variedade, requerendo novas formas de processamento para permitir tomada de decisão, descoberta de *insights* e otimização de processos (LANEY, 2001).

Por sua vez, um DL pode ser considerado um repositório para Big Data. O termo *Data Lake (DL)* foi cunhado por James Dixon (2010) para distinguir as abordagens de armazenamento do Apache Hadoop de data warehouses (DWs):

> Se você pensa em um DW como uma água engarrafada para venda - limpa, embalada e estruturada para facilitar o consumo - o DL é um grande corpo de água em um estado mais natural. O conteúdo do DL flui de uma ou mais fontes de água para encher o lago, e vários usuários do lago podem examinar, mergulhar ou coletar amostras.

Durante os anos subsequentes muitas discussões definiram formalizações para esse assunto.

Fang (2015) e Madera e Laurent (2016) estenderam o entendimento de Dixon. No caso de Fang, ele usa uma definição de DL mais alinhada com uma metodologia, enquanto Madera definem apenas DL como uma visão lógica dos dados. A metodologia de Fang concentra-se em dois pontos importantes: nível de maturidade e melhores práticas de desenvolvimento. O nível de maturidade considera vários subníveis, começando pelo nível sem o Apache Hadoop, que ele usa como base para armazenamento do DL, até a integração completa de todos os sistemas de informação ao DL. As melhores práticas, por sua vez, consideram quatro atividades: *(i)* capturar o máximo de

dados possível; *(ii)* aprimorar a capacidade de transformar e analisar dados; *(iii)* espalhar os resultados analíticos para fazer DLs e DWs trabalharem juntos; e *(iv)* adicionar recursos corporativos ao DL.

Nos anos seguintes, as obras de Quix e Hai (2018), Couto et al. (2019) e Ravat e Zhao (2019) convergem para um entendimento semelhante de DL. Entre as obras citadas anteriormente, baseamos na ideia de Couto et al. (2019) no qual definem que DL é um sistema para armazenamento, processamento e análise de dados no qual permite armazenar uma grande quantidade de dados, numa variada gama de formatos disponíveis e serem consultados apenas quando necessário.

Outro ponto a ser esclarecido, conforme destacado na definição de Dixon, são as diferenças entre DL e DW. Fang discute melhor essas diferenças através de comparação conforme mostra a Tabela 1.

Características	**DW**	**DL**
Carga de Trabalho	-Centenas a milhares de usuários -Otimização do processamento de carga de trabalho	-Processamento de lote de dados em escala -Melhoria contínua da capacidade
Schema	- *schema-a-priori*	- *schema-a-posteriori*
Escala	-Grandes volumes de dados a um custo moderado	-Volumes extremos de dados a baixo custo
Acesso	-SQL e ferramentas de BI -*Seek method*	-Programas criados pelos desenvolvedores -*Scan method*
Data	-Limpo -Homogêneo	-Bruto -Heterogêneo
Custo	-Eficiente uso de CPU/IO	-Baixo custo de armazenamento e processamento
Complexidade	- Uniões (*joins*) complexos	- Processamento complexo

Tabela 1 – Comparação de DW e DL (adaptado de (FANG, 2015)).

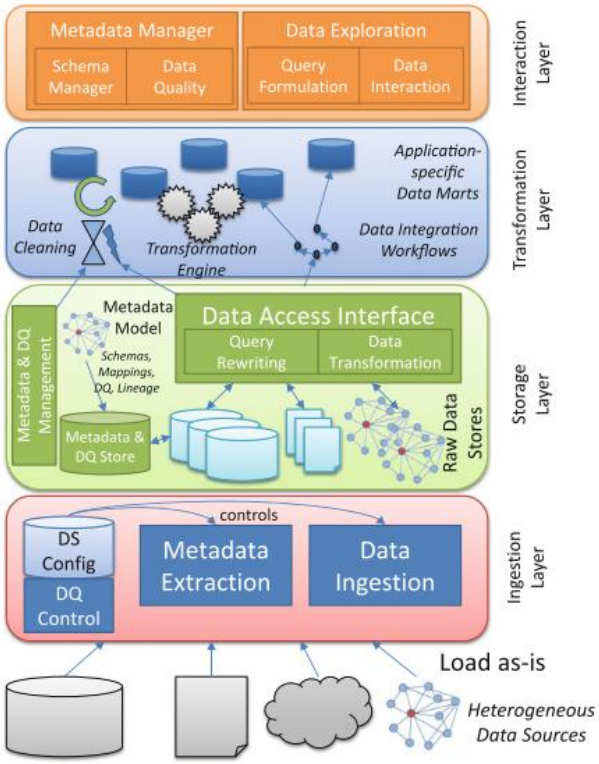
Conforme ilustra a Tabela 1, o objetivo de um sistema de gerenciamento de DL é trabalhar com alto volume, velocidade e variabilidade de dados com baixo custo de armazenamento. Já os DWs são sistemas onde os dados são modelados *a priori*, suportam cargas de trabalho em lote e são construídos para uso simultâneo por centenas a milhares de usuários simultâneos que geram relatórios ou tarefas analíticas.

2.1.1 Arquitetura de um sistema de gerência de data lake

Arquiteturas de sistemas de gerência de DLs (SGDLs) descrevem como os dados e os componentes de gerenciamento são organizados visando definir as responsabilidades de cada componente e quais dados são manipulados por eles (GIEBLER

et al., 2019). Esta dissertação utiliza como arquitetura de referência a definida por Quix e Hai (2018), apresentada na Figura 1, pelo fato dela ser uma arquitetura detalhada e também pela experiência dos autores, que possuem diversos trabalhos relacionados a DLs já publicados.

Figura 1 – Arquitetura de referência para SGDL



Fonte: (QUIX; HAI, 2018)

Conforme mostra a Figura 1, a arquitetura é dividida em quatro camadas: *Ingestão, Armazenamento, Transformação* e *Visualização*. A camada de ingestão é responsável por importar dados de fontes heterogêneas para o DL e extrair seus metadados. Nessa camada, os principais componentes são o *datasource config* (DS) e *data quality control* (DQ), sendo o DS responsável pelas configurações de acessos às fontes de dados externas e o DQ por definir controle de qualidade de dados prévio para a ingestão dos dados dentro do DL. Um de seus principais objetivos deste componente é o esforço mínimo para ingerir e carregar dados no DL. Devido a isso, ele costuma armazenar os dados em sua forma original para deixá-los livres para qualquer transformação e análise (QUIX; HAI, 2018).

A camada de armazenamento tem como objetivo armazenar e gerenciar os dados e metadados obtidos da camada de ingestão. De acordo com a Figura 1, seus principais componentes são o repositório de metadados e também os repositórios de dados nativos. O repositório de metadados armazena todos os metadados DL que foram eventualmente coletados da camada de ingestão ou extraídos por um pro-

cessamento específico. Os armazenamentos de dados brutos são o núcleo do DL, considerando o volume. Como a camada de ingestão traz os dados em sua forma original, esse componente deve armazenar os mais diversos tipos de dados (QUIX; HAI, 2018). Além disso, ao observar a posição dessa camada, entre a ingestão e transformação, os dados brutos vêm pela camada de ingestão e são colocados numa região específica, ao contrário dos dados transformados, que são mantidos em regiões que atendam suas específicas necessidades.

A camada de Transformação é responsável por limpar, processar e converter os dados de sua forma natural para um novo formato mais acessível aos usuários. Para modificar os dados brutos na estrutura de destino desejada, o DL precisa oferecer mecanismos de transformação de dados. Um exemplo de transformação pode ser a padronização e limpeza de dados de data e hora, utilizando, por exemplo, uma sequência ano-mês-dia hora:minuto:segundo (2021-01-01 00:00:00) (QUIX; HAI, 2018).

Por fim, a camada de Interação fornece ferramentas para acessar dados e metadados para atender às necessidades do usuário. Conforme mostrado na Figura 1, esta camada possui dois componentes principais: gerenciador de metadados e exploração de dados. O gerenciador de metadados permite que os usuários saibam quais tipos de dados estão disponíveis para serem pesquisados e analisados. O componente de exploração de dados permite ao usuário definir consultas que expressam seus requisitos de informação (QUIX; HAI, 2018).

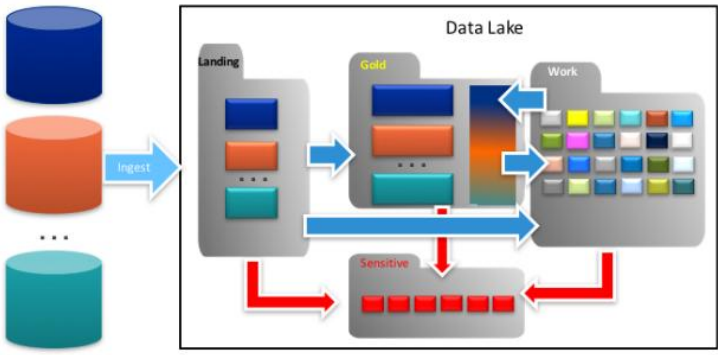
2.1.2 Arquitetura baseada em dados

Outra forma de organizar uma arquitetura de DLs diz respeito a como os dados são inseridos, tratados, armazenados e disponibilizados para consultas e análises. Sob essa perspectiva, existem duas categorias: arquiteturas em zonas (*zone architectures*) e arquitetura em lagos(*pond architectures*) (GIEBLER *et al.*, 2019).

As arquiteturas em zonas tem como ideia principal posicionar os dados em zonas de acordo com o tipo de processamento realizado sobre eles. Quando os dados são inseridos no SGDL, eles são armazenados na zona crua (*raw*). A vantagem dessa arquitetura é que mesmo se o dado está disponível em um formato já tratado, ele pode estar também disponível em sua forma bruta e armazenado na zona *raw*.

Diversas propostas de arquiteturas em zonas são encontradas na literatura. A Figura 2 mostra um exemplo dessa arquitetura conforme proposto por Gorelik (GORELIK, 2019). Nesta proposta, a zona *raw* denomina-se *landing*. Na sequência, os dados são limpos e colocados na zona *gold*. Adicionalmente, Gorelik definiu duas zonas com propósitos bem definidos. A zona *work* tem como objetivo habilitar que os usuários mais técnicos, analistas, cientistas e engenheiros de dados, possam realizar seus trabalhos de análise. Já a zona *sensitive* mantém todos os dados sensíveis relacionados, por políticas de acesso, gestão e armazenamento diferenciado.

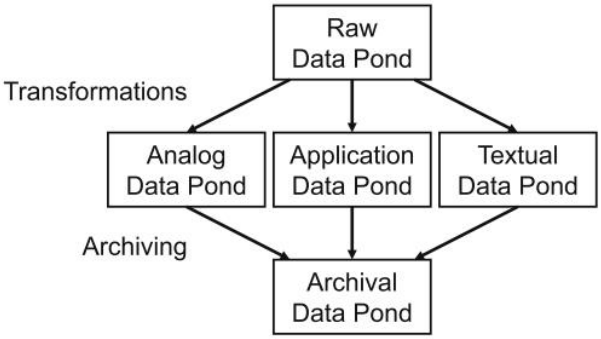
Figura 2 – Arquitetura de dados em *zones*



Fonte – (GORELIK, 2019)

As arquiteturas *pond* armezenam os dados em cinco *ponds* diferentes. Entretanto, ao contrário da arquitetura em zonas, os dados estão disponíveis em um único *pond* em determinado momento, conforme mostra a Figura 3.

Figura 3 – Arquitetura de dados em *ponds*



Fonte – (INMON, 2016)

Os dados são injetados no *raw pond*. Após a ingestão, os dados são direcionados para os demais *ponds*, como dados analógicos (*analog*), aplicativos (*application*) e textuais (*textual*). Para qual *pond* determinado dado deve ir depende da sua característica. O *data pond* analógico contém dados de medição, como arquivos de registro ou dados utilizados pela Internet das Coisas (*IoT*). No *data pond* de aplicativo, todos os dados gerados por aplicativos são armazenados. Já o *data pond* textual contém dados de texto. Por fim, quando os dados não são mais usados, eles deixam seu respectivo *data pond* e se movem para o *data pond* de arquivo (INMON, 2016).

Embora existam várias alternativas para arquiteturas de DLs, segundo Giebler et. al (2019), não há uma abordagem geralmente aceita. Embora as arquiteturas *zones* sejam mencionadas com mais frequência na literatura do que as outras variantes, as

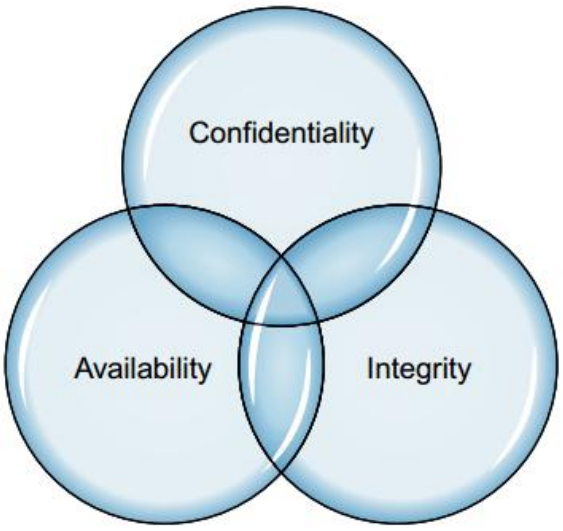
definições das várias zonas diferem muito em alguns casos e por cada autor. Além disso, Giebler et. al (2019) destacam que não existem trabalhos no momento avaliando ou comparando as arquiteturas, deixando essa avaliação como uma questão em aberto para futuras pesquisas.

2.2 SEGURANÇA DE DADOS

Embora a tecnologia nos permita ser mais produtivos com esforço menor, ela também carrega uma série de problemas de segurança. Se nossas informações presentes nos sistemas usados por empresas ou bancos ficarem expostas, por exemplo, as consequências podem ser imensuráveis. Ou pior, empresas podem perder muito dinheiro, processos legais e danos a reputação. Logo, na atualidade, segurança da informação tornou-se um tema muito importante e estratégico para empresas e governos (ANDRESS, 2014).

A segurança dos dados é definida como a proteção das informações e de seus sistemas contra acesso, uso, divulgação, interrupção, modificação ou destruição não autorizada. Em essência, significa proteger seus dados (onde quer que estejam) e os ativos do sistema de informações daqueles que tentariam usá-los incorretamente (ANDRESS, 2014). A Figura 4 mostra os três conceitos primários envolvendo segurança de dados (*confidencialidade, integridade* e *disponibilidade*), também conhecido pelo acrônimo *CID*.

Figura 4 – Tríade Segurança da Informação



Fonte – (ANDRESS, 2014)

Confidencialidade é a capacidade de proteger nossos dados daqueles que não estão autorizados a visualizá-los e manipulá-los. Várias técnicas podem ser empregadas para garantir a confidencialidade, entre elas estão a *autenticação* e *autorização*.

Autenticação, segundo a RFC 4949, é definido como o processo de verificação de quem uma entidade ou recurso possui determinado valor de atributo. Isso garante que o usuário relata com segurança quem ele é. Por sua vez, a autorização determina se o usuário tem permissão para acessar o recurso solicitado. No entanto, é importante destacar a relação entre confidencialidade e privacidade. Eles são conceitos semelhantes, mas não a ponto de serem considerados sinônimos (TORRA, 2017). Privacidade de dados geralmente significa a capacidade de um indivíduo de controlar o acesso as informações que ele/ela gera.

Integridade refere-se à capacidade de evitar que dados sejam alterados de uma maneira não autorizada ou indesejável. Por fim, *disponibilidade* refere-se à habilidade de acessar dados quando necessário (ANDRESS, 2014).

2.2.1 Privacidade de Dados

Nos dias atuais, dados são coletados numa quantidade e variedade nunca antes percebida por diversos motivos. Solove (2010) afirma que, quando a análise desses dados somente é realizada dentro da própria empresa que coletou, o perigo de divulgação de informações confidenciais é limitado. Entretanto, quando é realizada por terceiros ou por atores não diretamente relacionados à análise (desenvolvedores de software, por exemplo), a privacidade se torna uma questão muito mais relevante.

Privacidade de dados, segundo Torra (2017), significa a capacidade de um indivíduo de controlar as informações que ele/ela geram e quais informações terceiros podem revelar de um indivíduo. Um exemplo típico é uma assinatura de serviço pela Internet. Neste contexto, o contrato de prestação de serviço descreve quais dados do usuário serão utilizados e, em função da política de privacidade vigente, o usuário pode aceitar ou não disponibilizar esses dados.

Com relação aos dados, existem diversas formas de representação que podem ser utilizadas, como por exemplo, o formato tabular, no qual cada coluna representa um atributo e cada linha um registro. Segundo Fung et al. (2010), a forma mais básica que podemos classificar esses atributos são:

- *Identificadores*: atributos que identificam uma pessoa, como nome, e-mail e CPF;
- *Semi-identificadores*: atributos que não identificam a pessoa de forma explícita, porém, com a utilização de técnicas de agrupamento podem potencialmente identificar o indivíduo. Exemplos nessa categoria são data de nascimento e endereço;
- *Sensíveis*: atributos que são informações sensíveis do indivíduo, como por exemplo, salário e doença;
- *Não-sensíveis*: atributos que não se encaixam nas categorias anteriores.

Conforme descrito por Brito e Machado (2017), uma violação de privacidade ocorre por meio de um ataque ou quando um usuário consegue associar o titular de um dado a um registro de determinado conjunto de dados. Logo, proteger dados é uma atividade complicada pois quando se disponibiliza dados para fins de pesquisa, estatística, ou testes, é necessário aplicar técnicas de privacidade de dados para evitar a descoberta de dados pessoais ou sensíveis por parte de usuários que os utilizam para fins maliciosos. Portanto, como forma de manter a privacidade do indivíduo, o detentor dos dados precisa evitar que eventuais descobertas de informações não ocorram no conjunto de dados disponibilizado.

As três principais técnicas para contornar os problemas supracitados, ou seja, técnicas de controle de privacidade, são: *(i)* criptografia, *(ii)* tokenização e *(iii)* anonimização (BRITO; MACHADO, 2017). A criptografia utiliza um algoritmo capaz de embaralhar matematicamente os dados, gerando substitutos ilegíveis. Para que esses dados ilegíveis voltem a seus valores originais é necessário a utilização de uma chave de acesso. Portanto, tem como uma de suas vantagens possibilitar que dados importantes possam trafegar numa rede de computadores insegura somente com os interessados que possuem as chaves saberem o conteúdo original.

A tokenização é uma técnica utilizada principalmente quando é necessário proteger dados confidenciais já armazenados ou em movimentação na rede. Ela gera aleatoriamente um valor sem formatação específica, chamado *token*, a partir de um registro original, e armazena o mapeamento desse token para o seu respectivo valor original em uma base de dados separada. Dessa forma, tokens não podem ser revertidos aos seus valores originais sem o devido acesso à este mapeamento. Tokenização e criptografia apresentam finalidades similares, ou seja, trocar dados originais por outros sem nenhuma conexão com o original. Entretanto, na primeira técnica o token somente é utilizado no ambiente de aplicação local, sendo completamente inútil em outro contexto.

Por fim, a anonimização é constituída por um conjunto de métodos que modificam dados originais de tal forma que os dados anonimizados não se assemelham aos dados originais, mas ambos possuem semântica e sintaxe bastante semelhantes. O termo anonimato representa o fato do sujeito não ser unicamente caracterizado dentro de um conjunto de sujeitos, tendo como objetivo a possibilidade de compartilhar informações com outras entidades, as quais poderão utilizá-las para diversas finalidades, sem que haja violação de privacidade, como para fins estatísticos, pesquisa ou de testes. Entretanto, realizar modificações implicam em perda de informação e, consequentemente, em diminuição da utilidade dos dados. Portanto, o grande desafio da anonimização é balancear entre manter a privacidade do indivíduo sem perda da utilidade dessa mesma informação. Mais detalhes sobre técnicas de anonimização são apresentados na Seção 2.5.

2.3 GENERAL DATA PROTECTION REGULATION

O direito à privacidade já fazia parte da convenção europeia desde 1950. Entretanto, com o progresso da tecnologia e a invenção da Internet, a União Europeia (UE) reconheceu que os meios para garantir essa proteção deveriam se modernizar. Então, em 1995 foi publicado o *European Data Protection Directive* (EDPD). Ele estabelece padrões mínimos de segurança e privacidade, nos quais cada Estado signatário baseou sua própria lei de implementação. Entretanto, com o passar do tempo, a Internet se popularizou tanto no número de usuários quanto nas possibilidades de utilização e os meios, tornando a legislação obsoleta. Com isso, surgiu a *General Data Protection Regulation* (GDPR), que entrou em vigor em 2016 após a aprovação do Parlamento Europeu e, em 25 de maio de 2018, todas as organizações foram obrigadas a estar em conformidade com ela (GDPR, 2019).

A *General Data Protection Regulation* (GDPR) é uma lei que aborda questões relacionadas à privacidade de dados no âmbito da União Europeia (UE). Além disso, seu conteúdo é dividido em 11 capítulos que agrupam 99 artigos. Dentro desses capítulos, aborda-se aqui as partes relevantes para essa dissertação: *(i)* disposições gerais; *(ii)* princípios; *(iii)* direitos; e *(iv)* controladores e processadores.

Sobre o capítulo disposições gerais, no qual abrange os artigos *i* até o *iv*, descreve o assunto e os objetivos dessa legislação, estabelecendo regras para a proteção no que tange ao tratamento de dados pessoais. Além disso, define o escopo material, o escopo no sentido territorial, e define diversos termos da área com o objetivo de evitar ambiguidades no momento da interpretação.

O capítulo princípios descreve os princípios que orientam na aplicação dessa legislação em relação ao processamento de dados pessoais. Logo, os controladores e processadores devem, entre outros princípios: (*i*) ser processado conforme as leis, justo e transparente em relação ao titular dos dados (licitude, lealdade e transparência); (*ii*) coleta dos dados conforme a finalidade especificada para o usuário de forma explícita (limitação das finalidades); (*iii*) adequado, relevante e limitado às necessidades em relação aos propósitos definidos (minimização dos dados); (*iv*) acurado e, sempre que necessário, mantido atualizado (exatidão); (*v*) mantido na forma que permita a identificação dos titulares por um período não superior ao necessário (limitação da conservação); (*vi*) processado de maneira que garanta segurança apropriada (integridade e confidencialidade). Além das questões referente ao processamento de dados pessoais, nesse capítulo também menciona a questão do consentimento do titular, no qual nos casos em que o processamento dos dados é baseado em consentimento, o responsável deve demonstrar que o titular aceitou processar seus dados pessoais. Além do mais, o titular tem o direito de retirar, a qualquer momento, os direitos do responsável.

Outro capítulo importante está relacionado aos direitos dos titulares dos dados.

Sobre os direitos, dentro do capítulo referente ao assunto, ela divide em transparência e modalidades, informação e acesso aos dados pessoais, retificação e remoção, direito de contestar e tomada de decisão individual automatizada, e restrições. Referente a transparência e modalidades, o titular tem o direito do responsável reportar as informações solicitadas pelo titular ou por autoridade competente de forma concisa, transparente, inteligível e facilmente acessível, além de definir um prazo de um mês para responder, a não ser que justifique previamente a possível demora. Sobre informação e acesso aos dados pessoais, o titular dos dados tem direito a saber quando seus dados pessoais são coletados, propósitos do processamento, período que esses dados serão armazenados. Sobre retificação e remoção, o titular dos dados tem o direito de retificá-los, apagá-los e de definir restrições ao processamento. Por fim, em contestação, o titular tem o direito de se opor, por motivos relacionados a sua situação específica, a qualquer momento, ao tratamento de dados pessoais que lhe digam respeito. Logo, o responsável pelo tratamento deixará de processar os dados pessoais, a menos que demonstre motivos legítimos imperiosos para o tratamento que se sobreponham aos interesses, direitos e liberdades do titular dos dados ou para o estabelecimento, exercício ou defesa de ações judiciais.

Por fim, no capítulo controladores e processadores descreve seus respectivos direitos e deveres. Este capítulo é dividido nas seguintes seções: obrigações gerais, segurança dos dados pessoais, avaliação do impacto da proteção de dados e consulta prévia, *data protection officer* (DPO), código de conduta e certificação. Na seção obrigações, define que o controlador deve implementar medidas técnicas e organizacionais adequadas para garantir e demonstrar que o processamento é realizado em conformidade com o presente regulamento, como por exemplo aplicação e a minimização de dados para proteger os dados dos titulares. Na seção segurança dos dados pessoais, os controladores e processadores devem mitigar riscos ao processar dados dos titulares, utilizando sempre e de forma adequada: pseudoanonimização e criptografia de dados pessoais; capacidade de garantir a confidencialidade, integridade, disponibilidade e resiliência contínuas dos sistemas e serviços de processamento; capacidade de restaurar a disponibilidade e o acesso a dados pessoais em tempo hábil no caso de um incidente físico ou técnico; um processo para testar regularmente e avaliar a eficácia das medidas técnicas e organizacionais para garantir a segurança do processamento.

Na seção avaliação do impacto da proteção de dados e consulta prévia, do capítulo controladores e processadores, em relação a avaliação de impactos, define-se que, quando determinado processo aplicado a dados pessoais resultem em um alto risco para os direitos dos titulares, os responsáveis pelo tratamento devem realizar uma avaliação de impacto dessas operações considerando os casos descritos nesse artigo. Além disso, na parte de consulta prévia, determina que o responsável pelo

tratamento deve consultar a autoridade de supervisão antes do tratamento, se uma avaliação de impacto na proteção de dados, indicar que o tratamento resultaria em um risco elevado na ausência de medidas tomadas pelo responsável pelo tratamento para atenuar o risco. Na seção DPO, define o termo como o profissional encarregado de cuidar das questões referentes à proteção de dados da organização. Por fim, na seção código de conduta e certificação, em relação ao código de conduta, define que os estados membros, as autoridades de supervisão, o Conselho de Administração e a Comissão incentivam a elaboração e monitoramento de códigos de conduta destinados a contribuir com a correta aplicação da GDPR. Em relação a certificação, define que os Estados-Membros, as autoridades de supervisão, o Conselho de Administração e a Comissão incentivem a criação de mecanismos de certificação de proteção de dados e de selos e marcas de proteção de dados.

2.4 LEI GERAL DE PROTEÇÃO DOS DADOS (LGPD)

Conforme França, Nogueira e Antunes (2019), a Lei Geral de Proteção de Dados (LGPD) é a lei Brasileira que dispõe sobre a coleta, tratamento e armazenamento de dados pessoais e privacidade de dados, inspirada na respectiva lei europeia GDPR. A Tabela 2 compara as diferenças em ambas as legislações de forma resumida.

Tabela 2 – Comparação entre a LGPD e GDPR. (Extraído de (CÁTEDRA, s.d.))

LGPD	**GDPR**
Princípios de Tratamento e Privacidade	
1. Finalidade 2. Adequação 3. Necessidade 4. Livre acesso 5. Qualidade dos dados 6. Transparência 7. Segurança 8. Prevenção 9. Não discriminação 10. Responsabilização	1. Licitude 2. Lealdade 3. Transparência 4. Limitação das finalidades 5. Minimização dos dados 6. Exatidão 7. Limitação da conservação 8. Integridade e confiabilidade 9. Responsabilidade
Bases Legais para Tratamento	
Estabelece 10 bases legais	Estabelece 6 bases legais
Relação Entre Controlador de Dados e Operador de Dados	
Requer que o operador execute o tratamento dos dados conforme orientação do controlador	Exige um contrato entre controlador e operador de dados que explicite o tratamento dos dados

Transferências Internacionais de Dados Pessoais	
Impõe restrições, mas a Autoridade Nacional de Dados (ANPD) ainda deve estabelecer regras de transferências	Impõe restrições à transferência de dados pessoais para países terceiros. São necessários acordos e ajustes específicos para tal compartilhamento
Registro de Tratamentos de Dados	
Exige registro de tratamento dos dados pessoais	Exige o registro de tratamento de dados pessoais e especifica as informações sujeitas à manutenção de registros
Avaliação de Impacto sobre a Proteção de Dados	
Exige que o controlador de dados realize uma avaliação de impacto para avaliar os riscos de certas atividades de tratamento. Contudo, deixa a cargo da ANPD determinar quando essa avaliação é necessária	Exige que o controlador de dados realize uma avaliação de impacto para avaliar os riscos e detalha quando requer tal avaliação e o que exatamente as avaliações devem cobrir
Encarregado de Dados (DPO)	
Exige que o controlador de dados nomeie um DPO	Exige que o controlador e o operador de dados pessoais nomeiem um DPO. A GDPR explicita quando os DPOs não são necessários.
Segurança e Violações de Dados	
Exige que o controlador de dados implemente medidas de segurança de dados. A LGPD determina que a ANPD emitirá orientações e que a ANPD seja informada, assim como o titular do dado, em caso de ocorrência de evento	Exige que o controlador de dados implemente medidas de segurança de dados. A GDPR normatiza as medidas e determina que a comunicação com a autoridade de dados ocorra em até 72 horas em caso de ocorrência de evento, e dispensa essa comunicação de acordo com a severidade do evento
Penalidades e Sanções	
Define multas, sanções e processos civis a controladores e operadores, de acordo com o tipo de evento e severidade	Define multas, sanções e processos civis a controladores e operadores, de acordo com o tipo de evento e severidade

Conforme demonstrado na Tabela 2, mesmo a LGPD contendo mais bases legais comparada à GDPR, pode-se observar que a GDPR é mais exigente em aspectos

relacionados à responsabilização entre controlador e operador de dados, exigindo um contrato entre eles explicitando o tratamento dos dados. No caso de transferências internacionais, ambas impõem restrições, contudo, no caso Brasileiro, a ANPD estabelece as regras para transferências comparado a GDPR, que define somente em acordos específicos. Por fim, em relação à segurança, violação de dados, penalidade e sanções, a GDPR é mais restritiva e punitiva que a LGPD.

2.5 ANONIMIZAÇÃO DE DADOS

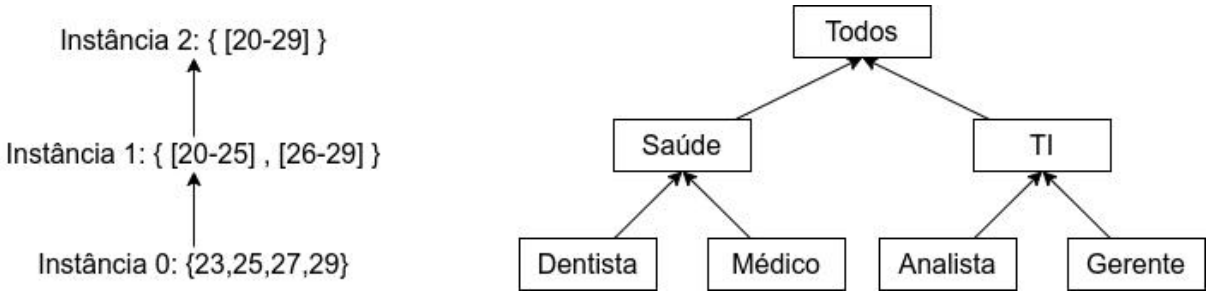
Esta seção apresenta as principais técnicas para anonimização de dados, entre elas a generalização, anatomização e perturbação randômica. Posteriormente, aborda-se os principais algoritmos dos modelos de privacidade de dados sintéticos e os diferenciais. Cabe ressaltar aqui que, dentre as técnicas de privacidade existentes, optamos por enfatizar anonimização de dados devido à *GDPR* mencionar principalmente essa técnica e propostas de melhoria para utilizá-la de forma mais efetiva.

2.5.1 Técnicas

A publicação dos dados ao público sem nenhum tratamento prévio pode ocasionar sérios riscos a privacidade. Logo, para evitar problemas ao disponibilizar os dados, existem técnicas para anonimizar os dados. Segundo Fung et. al (2010), existem três categorias de técnicas de anonimização de dados: generalização e supressão; anatomização; e perturbação randômica.

As técnicas de generalização tem como objetivo aumentar a incerteza de um atacante que tenta associar um indivíduo a seu registro baseado no conjunto de dados publicado. Nesta categoria de técnica, os valores são substituídos por outros semanticamente semelhantes classificados numa hierarquia de domínio pré-definida. Cada nível da hierarquia representa os dados de forma menos específica (BRITO; MACHADO, 2017). A Figura 5 mostra exemplos de aplicação dessa técnica.

Figura 5 – Exemplos de aplicação de técnicas de generalização



Fonte – Adaptado de (BRITO; MACHADO, 2017)

No exemplo à esquerda temos a instância 0 representando os dados iniciais 23, 25, 27 e 29. Ao aplicar a técnica, os valores são generalizados baseado em valores aproximados entre eles, sendo gerada a instância 1 com intervalos 20 até 25 e 26 até 29. Posteriormente é gerada uma nova instância com outro intervalo representando o conjunto total de dados. O exemplo à direita aplica a generalização em campos categóricos. Em cada nível o cargo é agrupado em categorias mais gerais aos quais o cargo pertence. Vale ressalvar que ao aplicar a técnica de generalização deve ser realizada uma ponderação mínima necessária para que a relação privacidade e utilidade sejam autênticas (BRITO; MACHADO, 2017).

A técnica de supressão de dados, assim como a generalização, é outra estratégia utilizada para compartilhamento e publicação dos dados preservando a veracidade semântica dos mesmos. Ela consiste em substituir ou remover um ou mais valores de um conjunto de dados por algum símbolo especial. Um exemplo poderia ser numa coluna onde estão registrados todos os salários de funcionários de determinada empresa. Caso o valor atenda determinada condição ele é substituído por "*"(FUNG *et al.*, 2010).

As técnicas de anatomização, ao contrário da generalização e supressão, modificam os atributos semi-identificadores e sensíveis, realizando uma desassociação entre eles (FUNG *et al.*, 2010). A Figura 6 demonstra o processo considerando como base uma tabela com dados de pacientes.

Figura 6 – Exemplos da técnica de anatomização sendo aplicada

Idade	Sexo	Doença
30	Masculino	Hepatite
30	Masculino	Hepatite
30	Masculino	HIV
32	Masculino	Hepatite
32	Masculino	HIV
32	Masculino	HIV
36	Feminino	Gripe
38	Feminino	Gripe
38	Feminino	Coração
38	Feminino	Coração

Idade	Sexo	Código Grupo
30	Masculino	1
30	Masculino	1
30	Masculino	1
32	Masculino	1
32	Masculino	1
32	Masculino	1
36	Feminino	2
38	Feminino	2
38	Feminino	2
38	Feminino	2

Código Grupo	Doença	Quantidade
1	Hepatite	3
1	HIV	3
2	Gripe	2
2	Coração	2

Fonte – Adaptado de (BRITO; MACHADO, 2017)

Conforme demonstrado na Figura 6, a primeira tabela de pacientes apresenta três colunas: idade, sexo e um atributo sensível doença. A técnica de anatomização gera as tabelas à direita. A primeira contém os registros dos pacientes, porém os valores do atributo doença foram substituídos por um identificador de uma terceira tabela que vincula a doença a um identificador de grupo.

Por fim, as técnicas de perturbação randômica são utilizadas com mais frequência para análises estatísticas devido a simplicidade, eficiência e capacidade de balancear a relação entre privacidade de dados e uma menor perda de qualidade comparada as técnicas de generalização e supressão. Segundo Brito e Machado (2017), essa técnica tem como base a substituição dos valores de atributos semi-identificadores por valores fictícios, porém permitindo que as informações estatísticas não se diferenciem significativamente dos valores originais.

2.5.2 Modelos de Privacidade Sintéticos

Os modelos de privacidade sintéticos estabelecem que os dados devem satisfazer determinadas condições após um processo de anonimização. Para que as condições sejam atendidas, esses modelos utilizam na maioria das vezes as técnicas de generalização e/ou supressão nos dados (BRITO; MACHADO, 2017). Os modelos mais conhecidos são κ-anonimato, *l*-diversidade, *t*-proximidade e δ-presença.

No modelo κ-anonimato, um conjunto de dados original contendo informações pessoais pode ser transformado de forma que seja difícil para um intruso determinar a identidade dos indivíduos. Um conjunto de dados κ-anônimos tem a propriedade de que cada registro é semelhante a pelo menos outros κ-1 registros nas variáveis de identificação potencial. Por exemplo, se κ = 5 e as variáveis potencialmente identificadoras são idade e sexo, então um conjunto de dados anonimizado κ tem pelo menos 5 registros para cada combinação de valor de idade e sexo. As implementações mais comuns de k-anonimato usam técnicas de transformação, como generalização, recodificação global e supressão (EL EMAM; DANKAR, 2008). κ-anonimato pode ser descrito como uma garantia de "esconder-se na multidão": se um indivíduo faz parte de um grupo maior, então qualquer um dos registros neste grupo pode corresponder a uma única pessoa. A Tabela 3 mostra um exemplo com dados antes e depois da aplicação do modelo para as colunas código ZIP e idade, tornando as informações dos pacientes genéricas e dificultando sua identificação.

Contudo, segundo Brito e Machado (2017), o modelo κ-anonimato tem suas desvantagens, entre elas ser vulnerável a ataques de ligação de atributo. Ataques de ligação de atributo torna o atacante capaz de inferir atributos sensíveis baseado no conjunto de valores sensíveis associados ao grupo no qual pertence. Logo, como forma de prover proteção contra ataques de ligação de atributos, o modelo denominado *l*-diversidade foi proposto. Essa técnica considera que deve haver "*L*" valores bem

Sem anonimização				Com anonimização		
Código	Código ZIP	Idade	Doença	Código ZIP	Idade	Doença
1	57677	29	Cardíaca	576**	2*	Cardíaca
2	57602	22	Cardíaca	576**	2*	Cardíaca
3	57678	27	Cardíaca	576**	2*	Cardíaca
4	57905	43	Pele	5790*	>40	Pele
5	57909	52	Cardíaca	5790*	>40	Cardíaca
6	57906	47	Câncer	5790*	>40	Câncer
7	57605	30	Cardíaca	576**	3*	Cardíaca
8	57673	36	Câncer	576**	3*	Câncer
9	57607	32	Câncer	576**	3*	Câncer

Tabela 3 – Exemplo da utilização do modelo κ-anonimato.

Fonte – Adaptado de (RAO; KRISHNA; KUMAR, 2018)

representados para o atributo sensível (a coluna Doença no exemplo da Tabela 3) em cada classe de equivalência. Todavia, alguns pontos não são cobertos por essa técnica, entre elas o *Skewness Attack*. Esse tipo de ataque consiste no atacante ter conhecimento prévio e descobrir tanto a classe de equivalência de um indivíduo quanto a distribuição dos atributos sensíveis, o que pode ser obtido analisando a tabela publicada (BRITO; MACHADO, 2017).

Para contornar a lacuna do modelo *l*-diversidade em relação a *Skewness Attack* foi proposto o *t*-proximidade. Esse modelo tem como objetivo garantir que a distribuição dos dados de um atributo sensível em cada classe de equivalência seja próxima à sua distribuição global. A distância máxima entre a distribuição global e as classes é definida pelo atributo *t* (FUNG *et al.*, 2010).

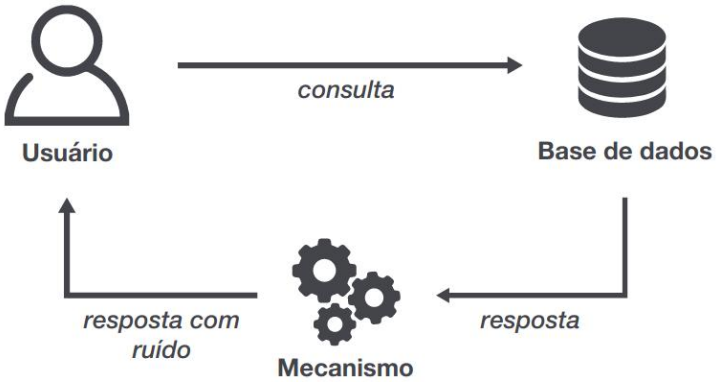
Por fim, o modelo δ-presença visa solucionar o ataque de ligação à tabela. Este tipo de ataque tem como objetivo inferir a presença ou não da vítima nos dados publicados. O modelo d-presença define um limite "*d*" para a probabilidade de determinado adversário inferir a presença de um indivíduo num conjunto de dados publicados. Com isso, esse modelo previne tanto ataques de ligações ao registro quanto ao atributo, visto que um atacante possui no máximo d% de confiança de que o registro da vítima está presente na tabela publicada (BRITO; MACHADO, 2017).

2.5.3 Modelo de Privacidade Diferencial

Os modelos de privacidade diferencial, propostos por Dwork (2006), realizam a anonimização de determinado conjunto de dados no momento que são consultados, em contraponto aos modelos sintéticos que atua nos dados tabulados. Além disso, eles evitam os ataques do tipo probabilísticos, definidos por Brito e Machado (2017) como um tipo de ataque no qual o atacante foca em determinar probabilisticamente determinado indivíduo após ter acessado o conjunto de dados publicado. No caso da

privacidade diferencial, para que ela funcione geralmente deve satisfazer determinados algoritmos denominados *mecanismos*.

Figura 7 – Fluxo de interação entre os atores na privacidade diferencial



Fonte – Extraído de (BRITO; MACHADO, 2017)

A Figura 7 demonstra como a privacidade diferencial atua de forma interativa através de três atores principais: Usuário, Base de Dados e Mecanismo. O fluxo de interação começa pelo Usuário, que submete uma consulta à uma Base de Dados, e no retorno o Mecanismo atua para anonimizar os dados a serem retornados ao Usuário. Cada conjunto de dados retornado para uma mesma consulta após o Mecanismo atuar é diferente e baseado no valor de um parâmetro ε. Portanto, para cada saída, um atacante não poderá distinguir entre todos os conjuntos de dados retornados na resposta fornecida pelo mecanismo. O parâmetro ε é explicado mais adiante.

Entre os mecanismos disponíveis, pode-se citar o mecanismo de *Laplace*, que é o mais comum e simples para alcançar a privacidade diferencial. Para entender como ele funciona, consideramos como exemplo a Figura 8, que representa o conjunto de imóveis que determinado indivíduo declarou.

Figura 8 – Exemplo do conjunto de dados original

ID	Nome	Nº Imóveis
1	Roney	4
2	André	2
3	Leo	7
4	Bruno	1

Fonte – Extraído de (BRITO; MACHADO, 2017)

Suponha uma consulta *f* que retorne a soma de todos os imóveis de todos os indivíduos. Para aplicar o mecanismo de *Laplace*, inicialmente devemos calcular cada conjunto vizinho a partir do conjunto de dados original. O conjunto de dados vizinhos são todos os possíveis conjuntos de dados gerados por *f* a partir da remoção

de determinado registro de um conjunto de dados original. A Figura 9 apresenta os possíveis conjuntos de dados vizinhos gerados a partir dos dados da Figura 8.

Figura 9 – Exemplos de conjuntos de dados vizinhos

ID	Nome	Nº Imóveis
2	André	2
3	Leo	7
4	Bruno	1

$f(D_1) = 2 + 7 + 1 = 10$

ID	Nome	Nº Imóveis
1	Roney	4
3	Leo	7
4	Bruno	1

$f(D_2) = 4 + 7 + 1 = 12$

ID	Nome	Nº Imóveis
1	Roney	4
2	André	2
4	Bruno	1

$f(D_3) = 4 + 2 + 1 = 7$

ID	Nome	Nº Imóveis
1	Roney	4
2	André	2
3	Leo	7

$f(D_4) = 4 + 2 + 7 = 13$

Fonte – Extraído de (BRITO; MACHADO, 2017)

Com os conjuntos de dados vizinhos calculados, e para que o modelo de privacidade diferencial garanta a privacidade do indivíduo, é necessário calcular a variação máxima que a ausência do indivíduo provoca no resultado de *f*. No conjunto de dados apresentado anteriormente, o registro que causa a maior variação no resultado é o indivíduo com identificador 3. Assim sendo, o próximo passo é calcular a *sensibilidade da consulta* aplicada ao conjunto de dados sobre a qual a consulta está sendo realizada. O calculo da sensibilidade pode ser definida pela maior diferença entre o resultado da consulta no conjunto de dados original e no conjunto de dados vizinho que tem a maior variação. Neste exemplo, o resultado de *f* sobre o conjunto de dados original é 14 e no conjunto de dados sem o indivíduo 3 é 7, tendo como diferença 7. Por fim, o ruído a ser adicionado deve ser igual a *Laplace*(0; e 7/ε).

O parâmetro ε é um valor definido pelo detentor dos dados e tem como função controlar o nível de diferença que cada conjunto de dado vizinho é calculado. Quanto menor o valor de ε maior é a garantia da privacidade. A recomendação é utilizar valores como 0,01, 0,1 ou logarítmicos naturais. A Figura 10 mostra a aplicação do parâmetro ε com valor 1 para cinco exemplos de ruídos, respostas e probabilidades de ocorrências após a aplicação do mecanismo de *Laplace*.

Conforme demonstrado na Figura 10, após a aplicação do mecanismo de *Laplace* o valor de ruído de -4,58 possui probabilidade de ocorrência de 3,7% sobre o conjunto de dados original, resultando num valor anonimizado de resposta à consulta de 9,42 imóveis.

Figura 10 – Exemplo do conjunto de dados vizinhos

Ruído	*f(D) + ruído*	*Pr(f(D) + ruído)%*
-4,58	9,42	3,70
-0,15	13,85	6,98
12,15	26,15	1,25
-6,43	7,57	2,85
2,89	16,89	4,72

Fonte – (BRITO; MACHADO, 2017)

2.5.4 Considerações Finais

Com relação à privacidade de dados e DLs podemos levar em consideração dois pontos. O primeiro ponto refere-se as especifidades de cada modelo de privacidade de dados apresentados anteriormente, no caso os modelos sintéticos e os diferenciais. Os modelos sintéticos são aplicados na divulgação de dados anonimizados de forma estática. Além disso, esse modelo conta com diversos algoritmos disponíveis devido a ser um campo de pesquisa ativo tanto na indústria quanto na academia. Já os modelos diferenciais, conforme descrito nesse capítulo, apresentam garantias maiores de privacidade e sua aplicabilidade é dinâmica, ou seja, quando existe uma consulta a determinado conjunto de dados e que seja necessária a anonimização.

O segundo ponto refere-se a aplicabilidade da privacidade de dados em uma arquitetura DLMS. Nesse caso, deve observar em quais momentos a privacidade de dados é necessária, qual o processamento será feito para determinados domínios de dados sem afetar o trabalho dos interessados nos dados do DLMS. Por exemplo, o modelo diferencial, por atuar diretamente no resultado da consulta sendo feita, pode interferir no resultado de uma análise onde seja necessário calcular determinados valores que requerem uma alta precisão.

3 TRABALHOS RELACIONADOS

Este capítulo descreve os trabalhos relacionados presentes na literatura. Desde o advento do conceito de DL, tem havido um interesse crescente da indústria e da academia em pesquisar e desenvolver soluções de gestão voltadas ao assunto. Algumas dessas pesquisas estão relacionadas a este trabalho, pois também propõem o levantamento ou análise de algumas questões sobre o gerenciamento de DLs e, em especial, à segurança e à privacidade de dados.

Iniciamos pelo trabalho de Couto et al. (2019) que apresenta uma revisão sistemática do conceito de DL, bem como possíveis arquiteturas. Eles utilizam uma metodologia de revisão sistemática bem conhecida (KITCHENHAM; CHARTERS, 2007), que culmina com uma nova definição do termo DL. Apesar de uma discussão conceitual aprofundada, este trabalho carece de justificativa e de uma proposta de arquitetura para DL. Os autores mencionam ferramentas que realizam tarefas de gerenciamento específicas e as agrupam em categorias: *ingestão, armazenamento, processamento, apresentação* e *segurança*. Entretanto, em termos de segurança, os autores não descrevem as técnicas adotadas.

Outras pesquisas apresentam uma visão geral de uma arquitetura de um DLMS (GIEBLER *et al.*, 2019; RAVAT; ZHAO, 2019; QUIX; HAI, 2018; PANWAR; BHATNAGAR, 2020). Eles discutem alguns problemas (componentes arquiteturais, modelagem e metadados), bem como desafios e questões em aberto. No entanto, a metodologia de pesquisa considerada para a produção do trabalho não é apresentada.

Com relação a trabalhos relacionados à privacidade de dados, o artigo de Rao, Krishna e Kumar (2018) realiza um levantamento do estado da arte de técnicas de preservação de privacidade em análise de *big data*. Além disso, os autores propõem um modelo para armazenamento de dados em DLs que preserva a privacidade dos dados. Contudo, os autores, ao proporem uma solução para a privacidade de dados dentro de DLs, somente recomendam a utilização de uma inteligência artificial (IA) para classificar se determinado dado é sensível sem entrar em detalhes sobre esse assunto. O trabalho de Giebler et. al (2021) apresentam um framework para para construção arquitetural de DLs. O trabalho afirma que um dos aspectos a serem levantados é a segurança e cita a *GDPR*, mas sem entrar em detalhes de como aplicá-la em DLs.

Durante o projeto de pesquisa dessa dissertação, o trabalho desse autor e Mello (2020) realizam uma revisão sistemática referente à confidencialidade em DLs. Nesse trabalho, os autores comparam as tecnologias utilizadas e propõem onde as funções de segurança devem estar alocadas considerando uma arquitetura de referência. Entretanto, o trabalho não aborda questões de privacidade de dados e nem a consideração de uma legislação de base à respeito de privacidade de dados.

A Tabela 4 mostra os trabalhos mencionados separados por assuntos: arqui-

tetura de um DLMS (no geral), privacidade, segurança de dados e legislação base. Percebe-se que nenhum deles aborda todos os assuntos ao mesmo tempo, nem consideram uma legislação de base para privacidade de dados para embasar a solução proposta para controle de privacidade em um DLMS. Portanto, esta dissertação tem como diferencial abordar os assuntos DLs, privacidade de dados, segurança de dados e legislação de privacidade de dados, no caso a *GDPR*. Ao abordar esses assuntos tem-se, como contribuição principal, uma arquitetura de DLMS aderente à essa legislação.

	Arquitetura DLMS	Priv.	Seg.	Legis. base
(RAO; KRISHNA; KUMAR, 2018)	X	X		
(COUTO *et al.*, 2019)	X			
(GIEBLER *et al.*, 2019)	X			
(RAVAT; ZHAO, 2019)	X			
(JOAQUIM; SANTOS MELLO, 2020)	X		X	
(PANWAR; BHATNAGAR, 2020)	X			
(GIEBLER *et al.*, 2021)	X			
Propostas desta dissertação	X	X	X	X

Tabela 4 – Trabalhos relacionados à dissertação

4 PROPOSTA

Os DLMSs possuem como principais funções gerenciar dados de diferentes naturezas, desde não estruturados, como vídeos e imagens, até dados estruturados, como tabelas relacionais e planilhas, tendo como uma de suas vantagens a capcidade de armazenar volumes de dados com altas taxas de crescimento, de ingestão em lotes, e até mesmo processados em tempo real. Apesar dessa vantagem e outras apresentadas na Seção 2, uma grande desafio está relacionada ao controle de privacidade de dados, que não é um requisito em muitas implementações. Assim sendo, este trabalho propõe uma arquitetura de DLMS adequada à privacidade de dados, tendo como diferencial a adequação à *GDPR*.

As próximas seções deste capítulo detalham a proposta. Inicialmente, são apresentados os requisitos legais que devem ser observados na arquitetura conforme descrito na Seção 2.3. Na sequência, descreve-se uma arquitetura utilizando como base os trabalhos de Christoph Quix e Rihan Hai (2018) e Joaquim e Mello (2020), porém, atendendo os requisitos mencionados anteriormente.

4.1 REQUISITOS LEGAIS E FUNCIONAIS

Os requisitos legais são as premissas que a arquitetura deve atender. Revisitando o que está descrito na Seção 2.3, inicia-se pelo artigo *v*. Nesse artigo, a arquitetura proposta deve atender aos princípios da *licitude, lealdade, transparência, limitação das finalidades, minimização dos dados, exatidão, limitação da conservação, integridade, confidencialidade* e *responsabilidade*.

Considerando o item *I* do artigo *v* (*licitude, lealdade* e *transparência*), o DLMS deve ter a capacidade de, ao tratar os dados pessoais para seguir a lei, ter mecanismos capazes de demonstrar como esses dados são processados de forma fiel ao executado na prática. Os itens *II* (*limitação das finalidades*) e *III* (*minimização dos dados*) definem que o DLMS, ao inserir os dados e processá-los, deve ter um propósito bem definido e utilizar somente os dados necessários. Contudo, vale ressaltar que, nos casos de DLMSs com vários domínios de dados, eles devem ser delimitados e mapeados com o objetivo de não ferir esse princípio devido ao DLMS suportar o armazenamento de variedade de dados descritos anteriormente. O item *IV* (*exatidão*) exige do DLMS que, em caso de haver dados inexatos, aqueles que estão fora da finalidade previamente definida sejam apagados ou retificados. O item *V* (*limitação da conservação*) requer que os dados armazenados no DLMS permitam a identificação dos titulares apenas durante o período necessário para as finalidades definidas. Por fim, o item *VI* (*integridade* e *confidencialidade*), exige que o DLMS garanta a segurança dos dados contra acessos não autorizados e mantenha a sua integridade.

Um tema que as legislações de dados evidenciaram, principalmente a GDPR,

foi a questão do consentimento. O consentimento é o ato do titular em aceitar que seus dados possam ser processados por um ou mais terceiros, desde que se respeite todos os pontos que a GDPR define, como a transparência e a finalidade, entre outros já citados nessa dissertação. Esse tema aparece em diversos trechos da GDPR, entretanto, o ato do consentimento consideramos como uma premissa para a função do DL. Contudo, nosso DLMS deve ter ferramentas para que o titular dos dados saiba onde estão seus dados e quais processamentos foram realizados.

Outro ponto de destaque é que na GDPR existem algumas categorias de dados que merecem tratamento especial ou considerações importantes. O mais importante a se destacar está escrito no artigo *ix*. Ele descreve que é proibido revelar a origem racial ou étnica, opiniões políticas, convicções religiosas ou filosóficas, filiação sindical, dados genéticos, biométricos, relativo a saúde, vida sexual ou orientação sexual de uma pessoa. Entretanto, o item *II* descreve as exceções em relação ao item anterior.

Portanto, considerando as observações descritas anteriormente na Seção 2.3, podemos agrupar os requisitos legais em requisitos funcionais. Por requisitos funcionais entende-se como um conjunto de funcionalidades que determinado serviço obrigatoriamente deve fornecer (SOMMERVILLE, 2015). Os requisitos funcionais, propostos pelo autor dessa dissertação, estão organizados conforme segue:

- *RF01*: Classificar o dado conforme a GDPR (pessoal, sensível, especial, entre outros) após a sua ingestão;
- *RF02*: Aplicar técnicas de privacidade de dados (criptografia, tokenização, anonimização) baseado na política de segurança relacionado ao dado classificado;
- *RF03*: Rastrear onde os dados estão sendo armazenados e processados;
- *RF04*: Permitir ao titular do dado saber onde suas informações estão armazenadas;
- *RF05*: Permitir ao titular poder alterar qualquer informação armazenada;
- *RF06*: Remover informações vinculadas ao titular do dado, caso ele/ela deseje.

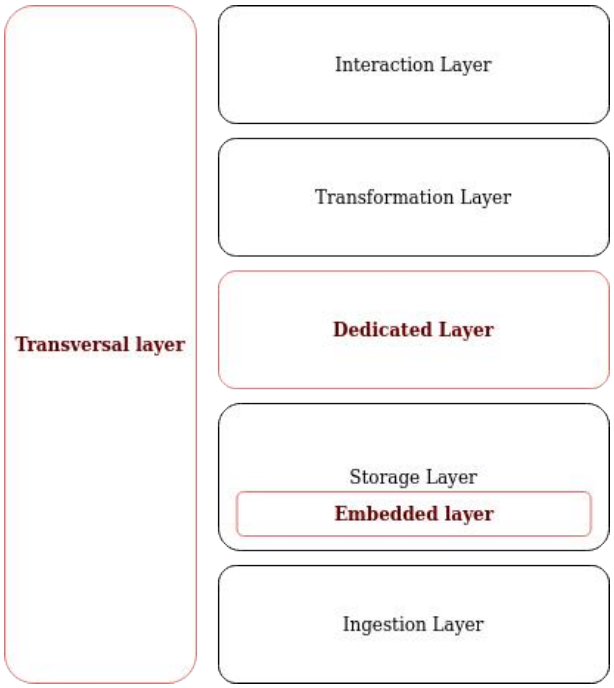
Conforme descrito nesta lista, diversos requisitos funcionais agrupam diversos artigos da GDPR com o objetivo de facilitar a validação para os experimentos realizados nessa dissertação.

4.2 ARQUITETURA

A arquitetura proposta nessa dissertação tem como base inicial a proposta de Christoph Quix e Rihan Hai (2018), utilizada como uma referência na organização dos componentes e responsabilidades de cada camada do DL conforme demonstrado na

Figura 1. Além disso, utilizamos o trabalho de Joaquim e Mello (2020) nos aspectos relacionados a segurança de dados. Este trabalho classifica, conforme apresentado na Figura 11, onde as funções de confidencialidade são organizadas levando em consideração a arquitetura proposta por Quix e Hai (QUIX; HAI, 2018).

Figura 11 – Classificação das camadas onde as funções de confidencialidade são posicionadas.



Fonte – Extraído de (JOAQUIM; SANTOS MELLO, 2020)

Conforme é observado na Figura 11, o trabalho de Joaquim e Mello (JOAQUIM; SANTOS MELLO, 2020) apresenta três classificações, representadas pelas linhas vermelhas, de onde o componente de segurança pode ser posicionado considerando a arquitetura de Quix e Hai (**2018**): *(i) transversal*, *(ii) dedicada* e *(iii) embutida*. Na classificação transversal, o controle de confidencialidade é suportado por uma camada específica, e está posicionado de forma transversal na arquitetura, ou seja, interage com algumas ou mesmo todas as outras camadas do DMLS. Na classificação dedicada, o controle de confidencialidade é suportado por uma camada específica que é posicionada entre outras camadas na pilha de camadas do DMLS. Na classificação embutida, o controle de confidencialidade é suportado por uma camada do DLMS existente, ou seja, não há uma camada específica para confidencialidade.

Portanto, seguindo as recomendações do trabalho de Joaquim e Mello (JOAQUIM; SANTOS MELLO, 2020), nossa proposta arquitetural é criar um componente de segurança que atenda os requisitos funcionais levantados anteriormente e posicioná-lo dentro da camada de armazenamento, seguindo as recomendações dos autores. A Figura 12 mostra onde o componente de segurança posicionado dentro da arquitetura

de referência.

Figura 12 – Classificação das camadas onde as funções de confidencialidade são posicionadas.

Fonte – Elaborada pelo autor

Conforme é demonstrado na Figura 12, o componente de segurança, posicionado na nossa arquitetura, funciona como o responsável por definir e aplicar as políticas de segurança a determinados conjuntos de dados baseado nas regras definidas pelo gestor do dado, que observa a legislação vigente. Além disso, para que a aplicação das políticas de segurança funcione de fato, o componente de segurança necessita acessar o catálogo de metadados, gerenciado pelo componente de mesmo nome, pois esse componente possui uma visão mais completa dos dados armazenados dentro do DLMS. Nas próximas subseções iremos detalhar nossas propostas organizando para cada camada da arquitetura: *ingestão*, *armazenamento*, *transformação* e *interação*.

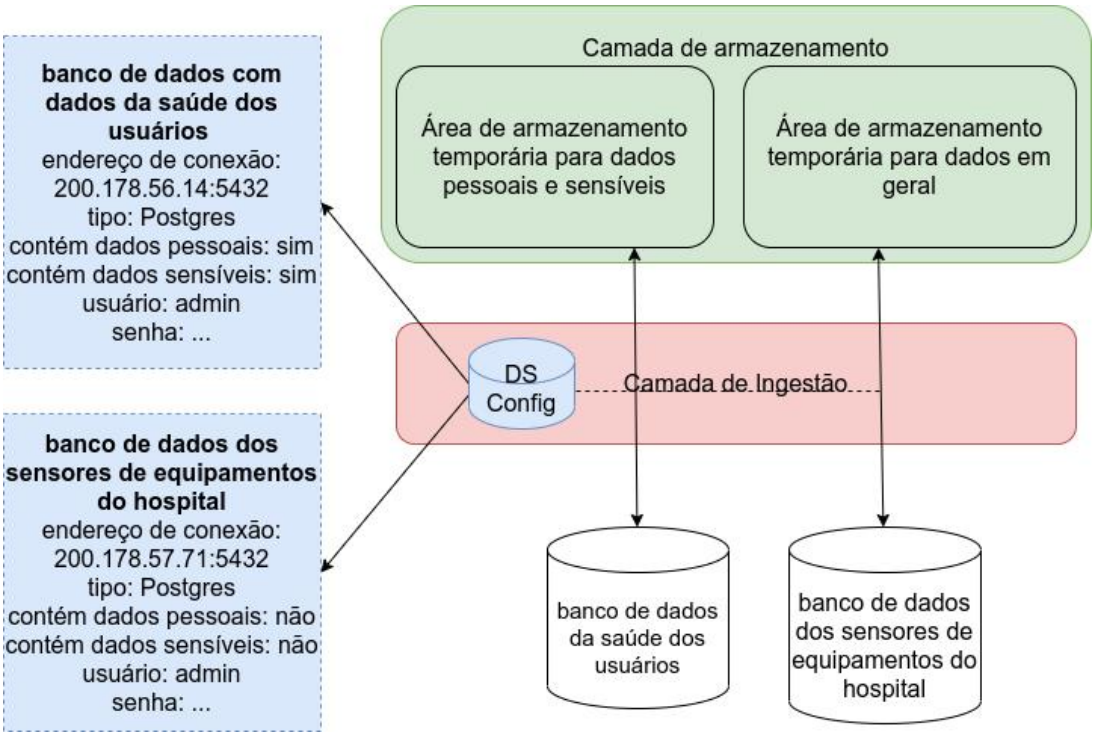
4.2.1 Ingestão

A camada de ingestão tem como principal função organizar todos os procedimentos relacionados a entrada de dados, independentemente do tipo ou tamanho. A proposta para essa camada envolve melhorias no componente *DS config*, apresentado na Seção 2.1, voltadas à segurança dos dados: no cadastro das fontes de dados a serem inseridas no DLMS, se está descrito que determinada fonte contém dados pessoais, sensíveis, ou outro que a legislação defina que deva ter tratamento diferenciado, ele deve sinalizar para a camada de armazenamento que os dados da base devem ser guardados em uma área que atenda os requisitos desse tipo de dado, como por

exemplo, se determinado tipo de dado precisa ser criptografado, anonimizado, entre outros. Para sinalizar à camada de armazenamento que determinada fonte de dados contém algum dado que precisa de tratamento diferenciado é possível proceder de dois modos, sendo a primeira manualmente, via um usuário especialista, e a segunda é a utilização de um componente que realize uma inferência nos dados que estão sendo inseridos no DLMS e decida se é um dado que necessite tratamento diferenciado.

Para exemplificar essa contribuição, a Figura 13 mostra um cenário considerando um DLMS com dados da área médica. Na parte inferior temos como exemplo dois BDs: o primeiro referente aos dados da saúde dos usuários e o segundo com registros dos sensores dos equipamentos hospitalares. No primeiro caso, quando essa base é inserida no DLMS, representado pela linha que liga a camada de armazenamento e o BD, o componente *DS Config* sinaliza para a camada de armazenamento (representado pela linha tracejada), que o BD contém dados pessoais e sensíveis. Logo, a camada de armazenamento guarda esses dados em uma área temporária dedicada para eles. No segundo caso, por não haver dados pessoais ou sensíveis, o *DS Config* sinaliza que esse dado pode ser armazenado numa área genérica temporária. Em ambos os casos, devido aos dados serem inseridos direto no DLMS sem nenhum tratamento realizado nesse momento, ambos foram colocados desde o início em áreas separadas e também temporárias para que a camada de armazenamento realize posteriormente os devidos tratamentos necessários.

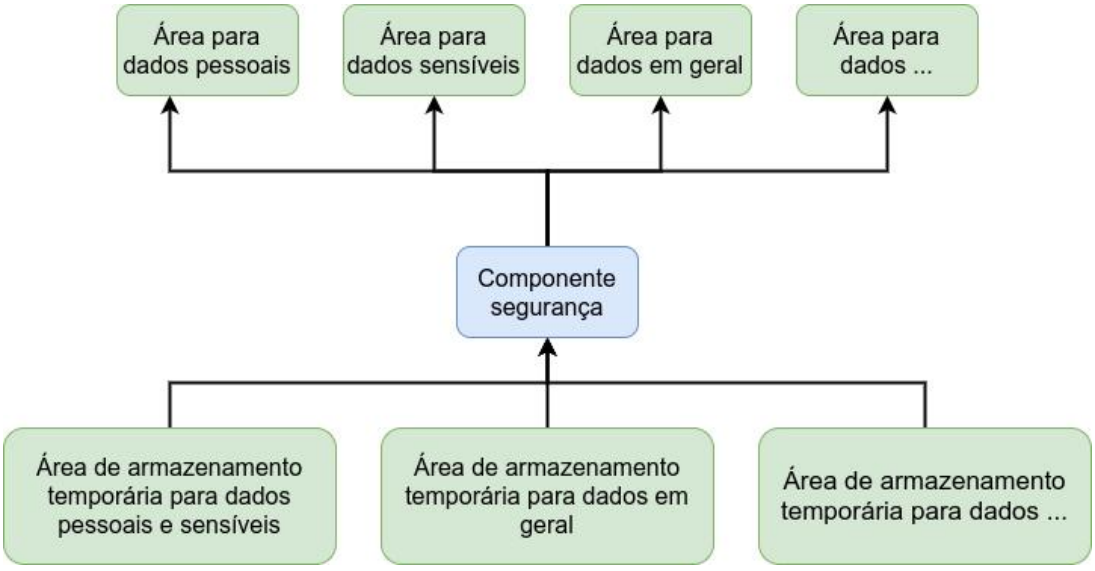
Figura 13 – Proposta do processo de ingestão no contexto de dados da área da saúde



4.2.2 Armazenamento

A camada de armazenamento de um DLMS tem como principal função armazenar e gerenciar todos os dados e metadados do DL. A primeira contribuição para a camada de armazenamento envolve a extensão do componente de segurança para que seja possível classificar os dados dentro do DLMS. Classificar (ou *tagear*) no contexto dos DLMSs significa a possibilidade de demarcar determinado conjunto de dados baseado em critérios previamente definidos. Contudo, a questão de como realizar a classificação, no contexto de legislações de privacidade de dados, é um tema em aberto em DLs. Assim sendo, a classificação pode ser feita manualmente (um usuário especialista analisa e realiza esse procedimento), ou automatizada (via utilização de IA ou qualquer outra tecnologia que auxilie nesta tarefa). Além disso, a contribuição da camada de ingestão, melhoria no componente *DS Config* pode auxiliar na classficação dos dados tanto se for feita manualmente ou automatizada. Ainda com relação a ausência de pesquisas sobre esse problema em específico, os critérios sobre como organizar os dados dentro do DLMS deixamos a cargo da equipe técnica para atender melhor cada cenário em específico que estão lidando. A Figura 14 mostra um exemplo do funcionamento dessa contribuição, classificando os dados que chegam nele e redirecionando para áreas específicas de dados.

Figura 14 – Exemplo de classificação do dado dentro do DLMS



Fonte – Elaborada pelo autor

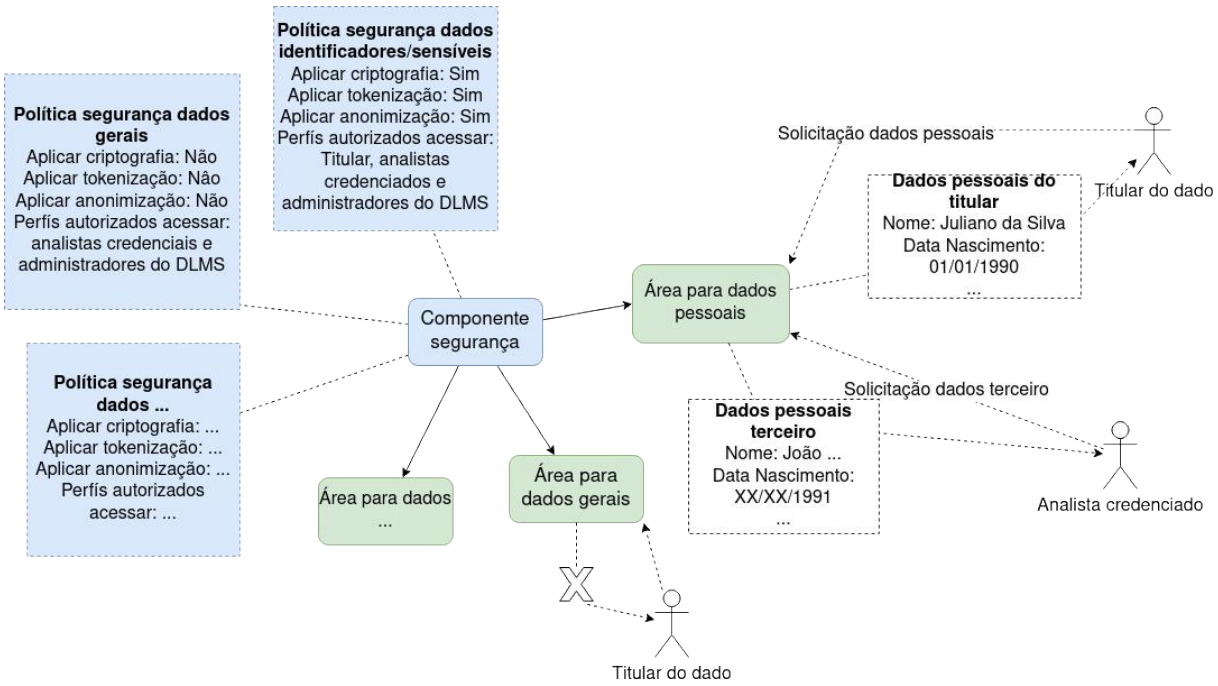
Conforme mostra a Figura 14, temos na parte inferior as áreas de armazenamento temporárias, nas quais os dados foram inseridos diretamente no DLMS sem nenhum tratamento ou classificação aplicada. Nesse caso, optamos por separar previamente os dados pessoais e sensíveis em um área temporária dos dados em geral. Na sequência, o componente de segurança analisa os dados em cada área temporária,

os classifica e os move para áreas específicas, conforme critérios definidos.

Um exemplo de critério que podemos utilizar está na própria GDPR (artigo *xxxii*): ela define que, na medida do possível, devem ser mitigados os riscos ao processar os dados dos titulares. Portanto, para alcançarmos o máximo possível dessa afirmação, nessa contribuição o primeiro passo é classificar e separar os dados dos titulares, que podem ser pessoais ou sensíveis, dependendo do contexto no qual o DLMS é aplicado, em lugares separados. Com os dados separados reduzimos os riscos de segurança relacionados à exposição dos dados dos titulares. Além disso, essa contribuição deixa em aberto a criação de novas áreas temporárias e áreas específicas, representado pelas reticências, para atender necessidades específicas.

A segunda contribuição envolve a extensão do componente de segurança para possibilitar a aplicação de técnicas de privacidade de dados. Podemos citar, como exemplo, a utilização de criptografia, a anonimização de dados, ou qualquer outro que o administrador do DLMS julgar necessário para atender as legislações de privacidade de dados, no caso dessa dissertação, a GDPR. A Figura 15 mostra o componente de segurança aplicando políticas de privacidade de dados em cada área de dados cadastrada, além de registrar e validar o acesso dos usuários a essa área de dados.

Figura 15 – Exemplo de aplicação de técnicas de privacidade dentro do DLMS



Fonte – Elaborada pelo autor

Conforme ilustra a Figura 15, temos duas áreas de dados: uma para dados pessoais/sensíveis e outras para dados gerais. No caso da área para dados pessoais/-sensíveis, o componente de segurança mantém uma lista de políticas de segurança

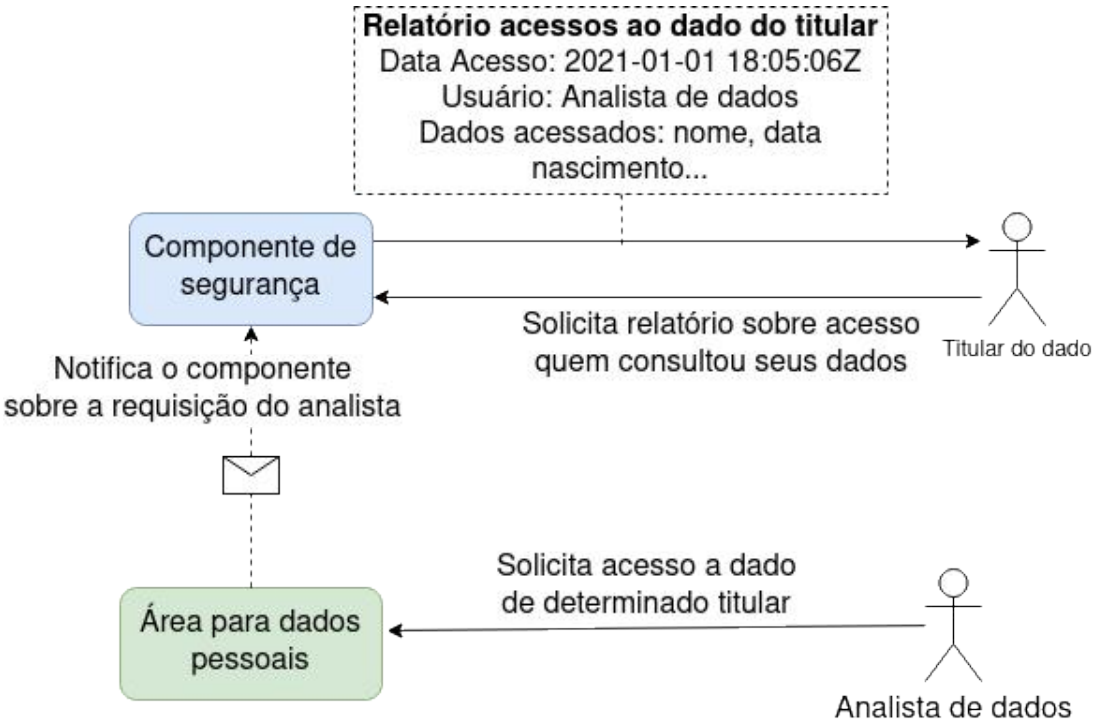
que deve ser aplicada, além dos perfis que tem permissão de acesso. Na área de dados gerais pode-se definir outra política de segurança sem necessidade de criptografia ou anonimização de dados. Todavia, todas as áreas devem ter uma lista de perfis com permissão de acesso. Portanto, o importante dessa contribuição está no componente registrar as políticas de segurança para cada área e aplicá-las conforme a necessidade. Além disso, essa contribuição deixa em aberto a criação de novas políticas de segurança, representado pelos quadrados azuis com reticências, para cada área de dados representado pelos retângulos verdes com reticências, com o objetivo de atender necessidades específicas em cada caso a ser aplicado.

Referente a políticas de segurança cadastradas para cada área deve observar que nas áreas que contém dados pessoais ou sensíveis devem obedecer a alguns tratamentos diferenciados. No caso da contribuição demonstrada na Figura 15, para proteger a privacidade do titular, deve-se aplicar técnicas de privacidade de dados, conforme elucidadas na Subseção 2.2.1. No caso de dados pessoais e sensíveis, recomenda-se, baseado na *GDPR*, que os dados sejam anonimizados. Contudo, a *GDPR* não restringe o controlador ou processador de dados de somente utilizar técnicas de anonimização, podendo combinar outras técnicas desde que o objetivo seja aumentar a segurança dos dados. Por exemplo, no caso da área com dados sensíveis, recomendamos que, para os dados que explicitam indivíduos, como números de cartões de crédito, números de seguro social, seja aplicada a tokenização, enquanto que para os dados semi-identificadores se aplique a anonimização, e para os demais tipos de dados seja aplicada a criptografia.

Outro ponto é o controle de acesso dos usuários a determinadas áreas do DLMS. Além das políticas de segurança e as funções típicas dela, como controle de autenticação e autorização, estendemos o comportamento do controle de autorização para que os titulares dos dados tenham apenas permissão de acessar seu próprio dado, sem ter privilégios para consultas genéricas ou extração de informação dentro da área que solicitou acesso. Isso está ilustrado na Figura 15 quando o titular de determinado dado solicita acesso e tem suas informações retornadas com sucesso. Contudo, ao solicitar acesso a área de dados gerais tem sua requisição negada. Vale observar que todas as solicitações de acesso são registradas e monitoradas com o objetivo de auditoria e aumento da segurança em casos de vazamentos ou indícios de acessos indevidos.

A terceira contribuição para a camada de armazenamento é a extensão do componente de segurança para o rastreamento do ciclo de vida do dado dentro do DLMS. Este rastreamento, considerando o contexto de um DLMS, aborda desde a gestão, manipulação, acesso até o arquivamento. Esta contribuição visa rastrear todos os acessos a dados pessoais/sensíveis para que o titular do dado saiba quando sua informação foi consultada e quem acessou. A Figura 16 mostra um exemplo.

Figura 16 – Exemplo de rastreabilidade do dado dentro do DLMS



Fonte – Elaborada pelo autor

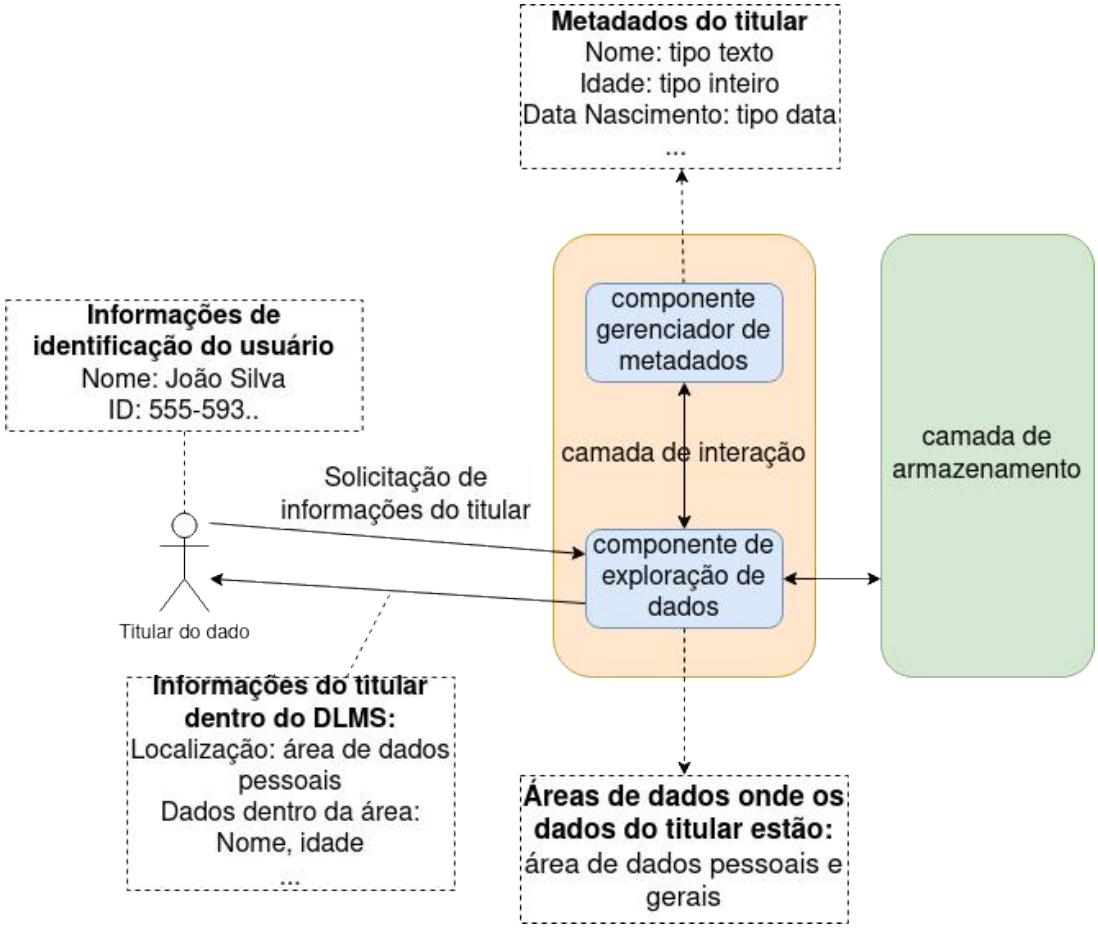
Nesse exemplo, um determinado analista acessa dados pessoais/sensíveis de um titular. Ao acessar a área na qual está o dado, a camada de armazenamento registra essa requisição e informa o componente de segurança. Além disso, o componente de segurança realiza o controle de autenticação e autorização dentro do DLMS. Desta forma, o DLMS torna-se mais transparente, disponibilizando ao titular dos dados relatórios sobre o acesso aos seus dados, quando este solicitar. Essa contribuição atende os artigos da seção transparência da GDPR (artigos *xiii* até o *xv*).

4.2.3 Transformação e Interação

A camada de transformação de um DLMS tem como principal função manipular os dados que chegam da camada de ingestão para estruturas de dados mais adequadas para análise de um cientista de dados, por exemplo. Nossas propostas para essas camadas são a extensão dos componentes *metadata manager* e *data exploration* voltadas para a privacidade de dados. A primeira contribuição para a camada de interação é a extensão dos componentes *data exploration* e *metadata manager* de modo a permitir que o titular do dado possa encontrar suas informações no DLMS. Essa contribuição é importante para atender a GDPR pois, segundo o princípio da privacidade, o titular do dado tem o direito de saber onde estão os seus dados, solicitar acesso aos seus dados e também retificar ou remover seus dados. A Figura 17 apresenta um exemplo do titular do dado solicitando suas informações pessoais do DLMS e ele retornando

onde elas estão.

Figura 17 – Exemplo de interação do usuário titular com o DLMS requisitando informações pessoais



Fonte – Elaborada pelo autor

Conforme mostra a Figura 17, o usuário, ao solicitar suas informações pessoais para o DLMS, ele aciona os componentes *metadata manager* e *data exploration*. Com a requisição do titular e as informações de identificação do mesmo, os componentes procuram neles as informações prévias cadastradas do titular. Contudo, como forma de garantir que todos os dados sejam retornados ao titular, ambos os componentes iniciam uma varredura por toda a camada de armazenamento atrás de suas informações. Com o término da procura, ambos atualizam essas informações nos seus próprios componentes e retornam ao titular todos os repositórios e quais dados estão nesses repositórios. De posse dessas informações, o titular do dado também pode solicitar alterações e exclusões.

4.3 CONSIDERAÇÕES FINAIS

DLMSs possuem como característica armazenar grande quantidade de dados independente do tipo de dado armazenado, desde estruturados até não estruturados.

Apesar de suas vantagens, a falta de uma governança que suporte a privacidade dos dados e que esteja aderente às respectivas legislações sobre o tema, em especial a *GDPR*, pode acarretar em uma sérias falhas de segurança. Desta forma, torna-se pertinente o desenvolvimento de uma proposta de arquitetura que habilite que os dados inseridos dentro de um DLMS tenham um nível de segurança satisfatório.

Este capítulo aborda propostas para cada camada de uma arquitetura de referência para um DLMS. Com base nas propostas e no levantamento de requisitos descritos na Seção 4.1, a Tabela 5 relaciona o requisito a qual camada a proposta de solução está envolvida.

Tabela 5 – Relação entre requisitos legais com a camada da arquitetura.

	Ingestão	Armazenamento	Transformação e Interação
RF01	X		
RF02		X	
RF03		X	X
RF04			X
RF05			X
RF06			X

Fonte – Elaborada pelo autor

O requisito *RF01* é cumprido com base na proposta apresentada na camada de ingestão, pois a classificação da fonte de dados com base nas categorias que a GDPR descreve, facilita para a camada de armazenamento gerenciar cada domínio de dado. Os requisito *RF02* é contemplado na camada de armazenamento com a utilização do componente de segurança. No caso da *RF03*, tanto a camada de armazenamento quanto de transformação e interação cumprem esse requisito, pois o armazenamento e manipulação do dado cobre todas essas camadas, cada camada tendo sua função específica. Por fim, as *RF04*, *RF05* e *RF06* são tratadas nas propostas feitas para as camadas de transformação e interação.

5 ESQUEMA DE METADADOS

Este capítulo apresenta os esquemas de metadados para DLs voltados à privacidade de dados propostos nesta dissertação. Dois esquemas de metadados são descritos: um para manter definições de políticas de privacidade voltados aos dados pessoais e sensíveis, e outro para manter todo o rastreamento do ciclo de vida do dado dentro do DLMS. Os esquemas foram definidos através do modelo Entidade-Relacionamento (ER) estendido, definidos por Batini, Ceri e Navathe (1991), que é um padrão para projeto de bancos de dados em nível conceitual. As propriedades de entidade e relacionamentos não são apresentadas para facilitar a compreensão dos esquemas. Além disso, por serem propostas em alto nível de abstração, optamos por deixar as propriedades em aberto para permitir definições mais customizadas.

5.1 POLÍTICAS DE PRIVACIDADE DOS DADOS PESSOAIS E SENSÍVEIS

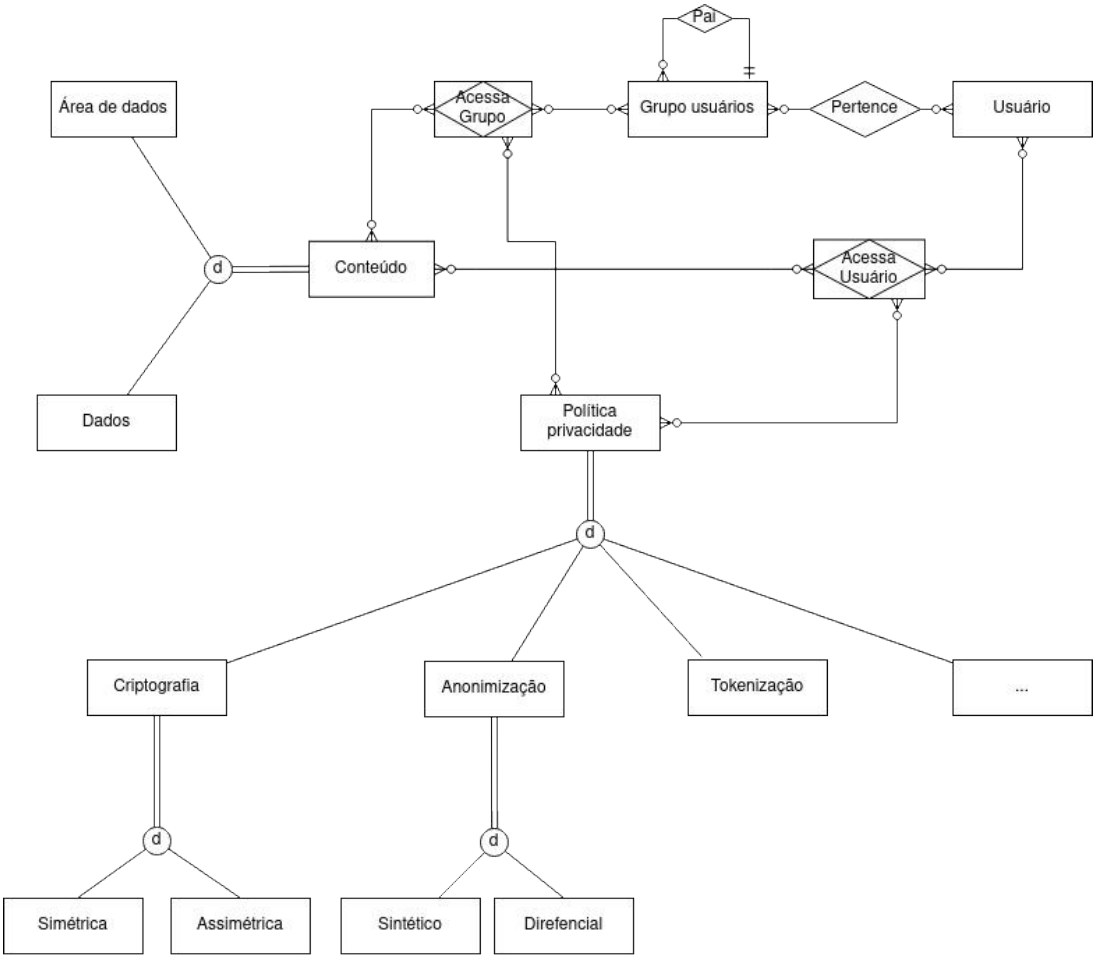
Este esquema indica quais dados, com suas respectivas políticas de privacidade, um determinado conjunto de usuários pode acessar. Seu objetivo é tornar possível, no contexto dos DLs, a aplicação de técnicas de privacidade de dados conforme o grupo de usuários. A Figura 18 apresenta o esquema proposto.

A entidade *Usuário* representa pessoas e sistemas que acessam o DL. Como propriedades básicas pode-se ter um identificador único, podendo ser uma chave inteira auto incrementável ou identificadores externos, como o CPF (Cadastro Pessoa Física) ou CNPJ (Cadastro Nacional Pessoa Jurídica), além do nome e sobrenome. Esta entidade se relaciona com a entidade *Grupo usuários*, cuja função é vincular um usuário a um ou mais grupos para facilitar a gestão de acesso aos respectivos dados ou às áreas de dados.

A entidade *Grupo usuários* representa agrupamentos de características em comum definidas pro DL em termos de privacidade. Somente usuários gestores ou administradores podem criar, alterar e excluir dados dessa entidade. Além do relacionamento com *Usuário*, mencionado anteriormente, outros dois relacionamentos são definidos. O relacionamento *Pai* tem como objetivo criar uma relação de hierarquia entre os grupos, facilitando a gestão de grupos com as mesmas características gerais. Sua cardinalidade é definida como 0 a *N*, ou seja, um grupo pode ser uma generalização de vários outros grupos. O relacionamento *Acessa* descreve quais áreas de dados que determinado grupo, ou grupos, tem permissão de acesso. Sua cardinalidade permite que um grupo acesse vários conteúdos e um conteúdo possa ser acessado por vários grupos.

A entidade *Conteúdo* representa os dados do DL. Ela possui duas especializações do tipo disjunta (mutuamente exclusiva): *Dados* e *Área de Dados*. A entidade *Dados* indica os dados armazenados no DL, podendo ter nome, tipo e localização

Figura 18 – Esquema para políticas de privacidade de dados pessoais e sensíveis



Fonte – Elaborada pelo autor

como suas propriedades. Um exemplo da entidade *Dados* seria um arquivo chamado *pacientes.csv*, do tipo *CSV* e localizado no diretório */healthcare/sensitive* no DL. A entidade *Área de Dados* representa um local onde um conjunto de *Dados* tem políticas de privacidade semelhantes, tendo como exemplo o diretório */healthcare/sensitive* no qual todos os dados vinculados ao diretório tem como política de privacidade a utilização de criptografia e anonimização. Além do relacionamento *Acessa*, o relacionamento *Contém* vincula o conteúdo a políticas de privacidade definidas pelos administradores do DL. Neste caso, um conteúdo pode apresentar várias políticas de segurança. Ao acessar o conteúdo o usuário do DL terá as informações das políticas de privacidade aplicadas ao arquivo, além de saber se tem permissão para acesso.

Por fim, temos a entidade *Política privacidade* e suas especializações. No esquema proposto foi definida uma entidade especializada representada por uma reticência, que deixa em aberto a definição de mais técnicas de privacidade de dados. As entidades especializadas, como é o caso de *Criptografia*, *Tokenização* e *Anonimização*, também possuem especializações visando detalhar o tratamento da privacidade a ser

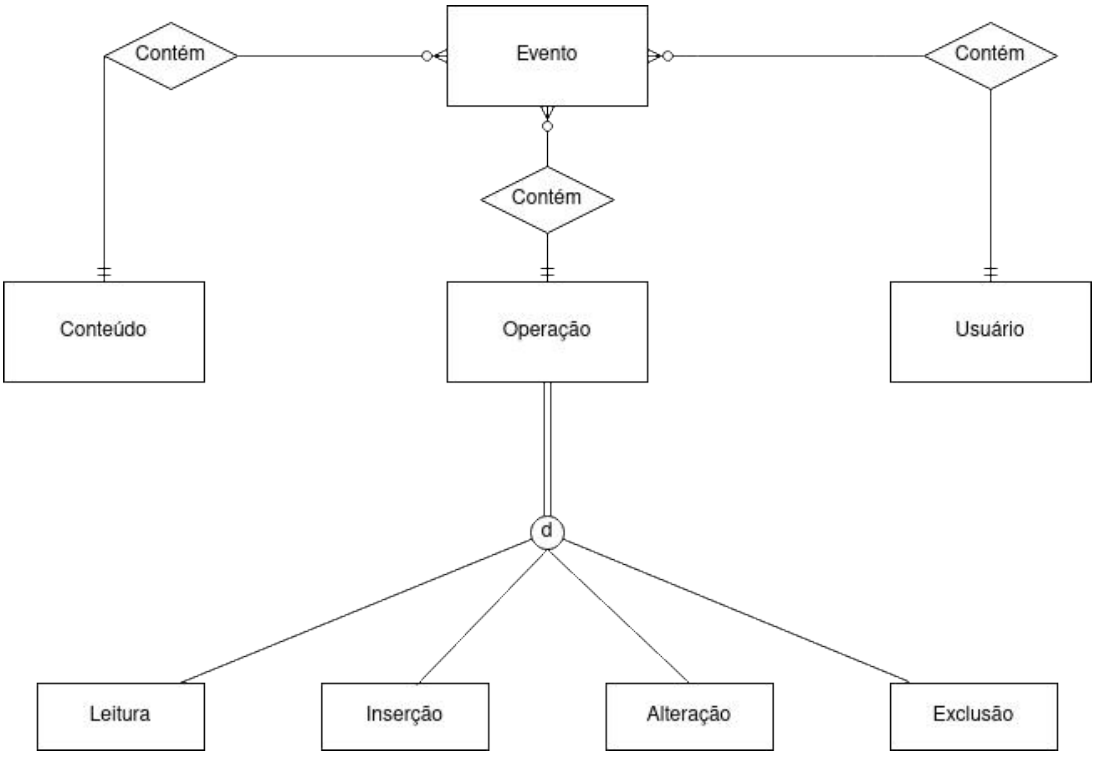
considerado no contexto do DL.

Este esquema mostrado na Figura 18 é uma proposta de solução para a RF02, além de ser um esquema de alto nível, simples e flexível para futuras alterações. Optou-se também por deixar em aberto as políticas de privacidade devido a existência de várias técnicas com suas peculiaridades, não sendo o objetivo dessa dissertação exaurir todos os cenários possíveis.

5.2 RASTREAMENTO DO CICLO DE VIDA DO DADO

Este esquema diz respeito à rastreabilidade de dados no contexto dos DLs, permitindo atividades como auditoria dos dados, visando atender a *RF03* descrita na Seção 4.1. A Figura 19 apresenta este esquema.

Figura 19 – Esquema para rastreabilidade as operações realizadas no DL



Fonte – Elaborada pelo autor

As entidades *Usuário* e *Conteúdo* são as mesmas definidas no esquema da seção anterior. A entidade *Operação* representa as ações de manipulação de dados realizadas sobre o DL. Suas especializações indicam essas possíveis ações: *Leitura, Inserção, Alteração* e *Exclusão*.

A entidade *Evento*, por sua vez, vincula todas as entidades anteriores para fins de rastreabilidade. Ela indica qual usuário ou grupo de usuários realizou determinada operação sobre um determinado conteúdo do DL.

Este esquema é uma proposta de solução para RF03 e RF04, apresentadas na Seção 4.1. Com o suporte a estes requisitos torna-se possível saber onde os dados estão sendo armazenados, processados e quem consulta as informações, atendendo, assim, os requisitos da GDPR descritos na Seção 2.3.

6 TECNOLOGIAS SUGERIDAS

Este capítulo descreve as tecnologias sugeridas nesta dissertação para serem utilizadas nas camadas da arquitetura de referência mostrada na Figura 1. Ele detalha essas tecnologias, suas responsabilidades e como elas atendem os requisitos funcionais da Seção 4.1. As tecnologias sugeridas para cada camada são mostradas na Figura 20.

Figura 20 – Proposta arquitetural com sugestão de tecnologias para cada camada

Na camada de ingestão foi escolhida a ferramenta *Apache Sqoop*[1]. O motivo da escolha é o fato dela ser de código aberto, amplamente aceita e fácil de integrar com as outras ferramentas que fazem parte do ecossistema Hadoop, suportado e mantido pela *Apache Software Foundation*[2]. Esta ferramenta permite a ingestão de dados de BDs relacionais para o ambiente Hadoop.

Para a camada de armazenamento foram escolhidas as ferramentas *Apache Hadoop* (Hadoop)[3] como solução para armazenamento dos dados dentro do DL, *Apache Atlas* (Atlas)[4] como o componente responsável pela gestão dos metadados, e *Apache Ranger* (Ranger)[5] para gestão de segurança. Estas ferramentas foram selecionadas pelos mesmos motivos informados anteriormente para a *Apache Sqoop*.

Para a camada de transformação e interação foi selecionada a *Apache Hive*[6] como ferramenta de acesso e manipulação de arquivos mantidos na camada de armazenamento. A vantagem da sua utilização está no uso do padrão SQL (*Structured*

[1] *https://sqoop.apache.org*
[2] *https://www.apache.org*
[3] *https://hadoop.apache.org*
[4] *https://atlas.apache.org*
[5] *https://ranger.apache.org*
[6] *https://hive.apache.org*

Query Language) para realizar análises de dados em diversos tipos de arquivos que ele suporta.

Um ponto a se observar em relação à arquitetura de referência proposta por Quix e Hai (2018) é que não é necessário seguir fielmente a organização arquitetural. Além disso, vale ressaltar que a escolha de uma única ferramenta para representar determinada camada não implica que somente esta ferramenta deva ser utilizada para essa finalidade. Nosso intuito é utilizar as ferramentas selecionadas apenas para testes de validação dessa dissertação, deixando em aberto futuras escolhas de outras ferramentas com as mesmas capacidades. As próximas subseções detalham cada camada da arquitetura.

6.1 CAMADA DE INGESTÃO

Conforme descrito na Seção 2.1, a camada de ingestão tem como objetivo inserir dados, desde estruturados até não estruturados, dentro do DL. Esta seção detalha o componente *Apache Sqoop* como exemplo de ferramenta que auxilia a ingestão de dados que estão em BDs relacionais para o DL. Devido ao fato comentado anteriormente da natureza de ingestão dos dados suportar dados estruturados até não estruturados, seria inviável para essa dissertação descrever todas as ferramentas disponíveis. Portanto, restringimos a utilização para a ferramenta utilizada no capítulo Estudo de Caso (ver capítulo 7).

O *Apache Sqoop* (Sqoop) é uma ferramenta projetada para facilitar a transmissão de dados entre BDs relacionais e o Apache Hadoop. Além disso, também suporta a importação de dados localizados em *mainframes*. Por último, permite o caminho inverso, ou seja, exportar dados do DL para BDs relacionais. O diferencial dessa ferramenta está em automatizar a maior parte desse processo, desde consultar os dados, transformar e exportar para o destino, somente disponibilizando ao usuário uma interface em linha de comando. Para que esse diferencial seja executado, o Sqoop utiliza o *Apache Hadoop MapReduce*, que realiza todo o processamento de forma paralela e com tolerância a falhas (SQOOP, s.d.). A Figura 21 apresenta as funcionalidades presentes no Sqoop informadas pelo comando *sqoop help*.

Entre as funcionalidades disponíveis apresentadas anteriormente focaremos na importação dos dados de BDs relacionais para o DL, disponibilizado pelo comando *sqoop import*. Com este comando, o sqoop controla diversos aspectos do processo de importação dos dados, como paralelismo, tipo de importação, total ou incremental, entre outros. Além de colocar os dados na camada de armazenamento, o *sqoop import* pode também criar as tabelas e colunas nos componentes *Apache Hive*[7], *Apache*

[7] *https://hive.apache.org*

Figura 21 – Funcionalidades do Sqoop mostradas pelo comando *sqoop help*

```
$ sqoop help
usage: sqoop COMMAND [ARGS]

Available commands:
  codegen            Generate code to interact with database records
  create-hive-table  Import a table definition into Hive
  eval               Evaluate a SQL statement and display the results
  export             Export an HDFS directory to a database table
  help               List available commands
  import             Import a table from a database to HDFS
  import-all-tables  Import tables from a database to HDFS
  import-mainframe   Import mainframe datasets to HDFS
  list-databases     List available databases on a server
  list-tables        List available tables in a database
  version            Display version information

See 'sqoop help COMMAND' for information on a specific command.
```

Fonte – (SQOOP, s.d.)

HBase[8] e *Apache Accumulo*[9], conforme solicitado pelo usuário. Com isso o dado já fica disponível para consultas nos respectivos componentes (SQOOP, s.d.). Como forma de demonstrar a utilização do comando de importação, a Figura 22 apresenta dois exemplos de importação dos dados: a primeira somente colocando os dados no DL e a segunda criando uma tabela no *Apache Hive*.

Figura 22 – Exemplos de utilização do comando *sqoop import*

```
$ sqoop import --connect jdbc:mysql://database.example.com/healthcare \
--username aaron --password 12345 \
--query 'SELECT * FROM patients WHERE $CONDITIONS' \
--target-dir /user/aaron/healthcare/patients

$ sqoop import --connect jdbc:mysql://database.example.com/healthcare \
--username aaron --password 12345 \
--query 'SELECT * FROM patients WHERE $CONDITIONS' \
--hive-import --create-hive-table --hive-table patients
```

Fonte – (SQOOP, s.d.)

Os dois exemplos mostram, além do *sqoop import*, os parâmetros básicos de conexão ao banco de dados, como endereço, usuário e senha, o comando SQL a ser executado no BD relacional mais o parâmetro *$CONDITIONS*, que controla a execução pelo lado do sqoop. Aa diferença entre os exemplos, entretanto, está no destino que os dados terão dentro do DL. No primeiro exemplo, o parametro *target-dir* sinaliza ao sqoop que os dados importados serão colocados no diretório */user/aroon/healthcare/patients*. Já no segundo exemplo, os parâmetros *hive-import create-hive-table hive-table* sinalizam ao sqoop que os dados importados serão enviados para o componente *Apache Hive* e deverá ser criada uma tabela chamada *patients* dentro do componente.

Portanto, considerando as funcionalidades apresentadas anteriormente, podemos concluir que o *Apache Sqoop* tem como objetivo importar dados de BDs relacionais, em forma de lote (*batch*), para o DL. Contudo, devido a natureza do DL de

[8] *https://hbase.apache.org*
[9] *https://accumulo.apache.org*

aceitar desde dados estruturados até não estruturados, de importação em lote (*batch import*) ou importação em fluxo (*streaming import*), ele atende parcialmente o *RF01*, apresentado na Seção 4.1, pois somente importa dados estruturados em lote. Nessa dissertação escolhemos somente uma ferramenta para essa camada devido a vasta quantidade de ferramentas disponíveis para tipos de importação e cenários diferentes e, no nosso caso, iremos apresentar no próximo capítulo a validação com um cenário de uso utilizando como fonte de dados um BD relacional.

6.2 CAMADA DE ARMAZENAMENTO

Conforme descrito na Seção 2.1, a camada de armazenamento armazena e gerencia os dados e metadados obtidos da camada de ingestão. Esta seção detalha os componentes de armazenamento (*Apache Hadoop*), segurança (*Apache Ranger*) e metadados (*Apache Atlas*) e como cada um deles atende os requisitos funcionais levantados na Seção 4.1.

6.2.1 Armazenamento

O componente de armazenamento de um DLMS mantém todos os dados inseridos de forma que seja escalável, disponível e atenda dados desde estruturados até não estruturados. Entre as ferramentas disponíveis para armazenamento de dados no contexto dos DLs, o Hadoop se mostra adequado a esse cenário, pois é um framework aberto, escalável e tolerante a falhas, executando tanto em redes de computadores tradicionais quanto em ambientes em nuvem e desde máquinas simples até máquinas com grande poder de processamento (WHITE, 2015).

O Hadoop é composto por duas partes principais: *Hadoop Distributed File System (HDFS)*, que é um sistema de arquivos do tipo distribuído e *Hadoop MapReduce (MapReduce)*, que é um modelo para processamento de dados de forma distribuída. Além dessas partes, o Hadoop possui alguns módulos de apoio, como o *Hadoop Common*, que contém um conjunto de bibliotecas e utilitários comuns a todos os módulos e outros componentes que se integram ao Hadoop, e o *Hadoop Yarn*, que realiza a gestão dos recursos do cluster sobre o qual o Hadoop está executando (WHITE, 2015).

Por fim, vale ressaltar que o Hadoop possui diversos subprojetos, formando um ecossistema que oferece serviços complementares. Entre os subprojetos estão o *Apache Hive*, *Apache HBase* e o *Apache Ranger*. As próximas subseções explicitam as duas partes principais do Hadoop: o HDFS e o Hadoop MapReduce.

6.2.1.1 HDFS

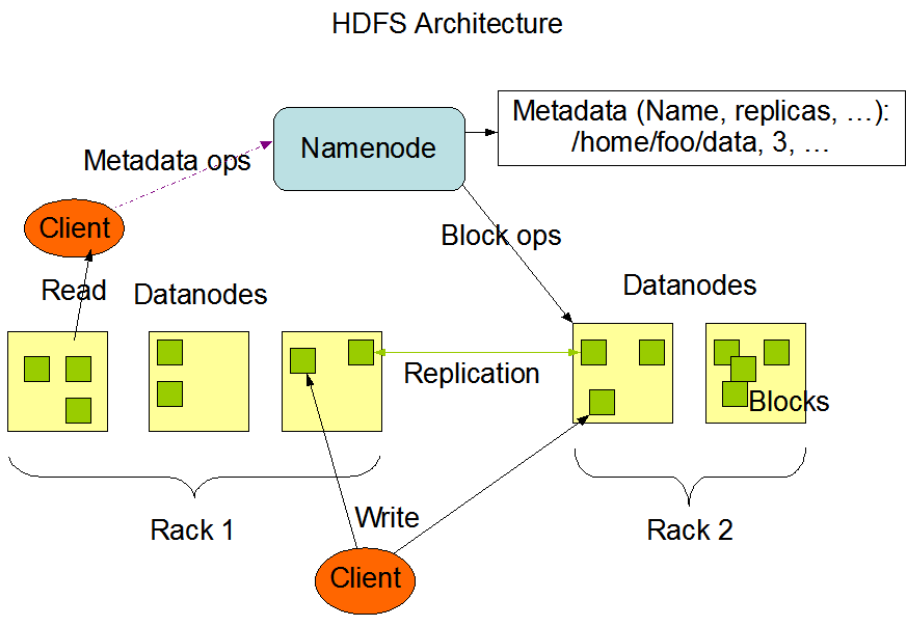
O HDFS é um sistema de arquivos distribuído escalável, tolerante a falhas e projetado para ser implementado em hardware de baixo custo. Além disso, o HDFS

fornece acesso de alta taxa de transferência aos dados do aplicativo e é adequado para aplicativos que possuem grandes conjuntos de dados (HADOOP, s.d.[a]).

Sobre a organização dos dados, por ser pensada para ambientes distribuídos e o suporte a arquivos muito grandes, temos duas características importantes a mencionar: os *blocos de dados* e a *replicação dos dados*. Os blocos de dados suportam o modo de escrita uma única vez e várias leituras (*write-once-read-many*) e os arquivos são divididos, por padrão, em blocos de 128 MB. O modo *write-once-read-many* define que um sistema de armazenamento permite que várias aplicações podem ler o mesmo bloco ao mesmo tempo, contudo para escrita somente permite uma aplicação por vez, evitando possíveis conflitos. Referente à replicação de dados, o HDFS replica os blocos de dados, por padrão, em três partes entre as instâncias do cluster (HADOOP, s.d.[a]).

O HDFS tem uma organização arquitetural do tipo mestre/escravo (*master/slave*). Nessa organização arquitetural, quem tem o papel de *master* é o nó chamado *NameNode*, cuja responsabilidade é gerenciar o sistema de arquivos, guardando os metadados dos dados e regulando os acessos aos arquivos pelos clientes. Já os *slaves*, denominados *DataNodes*, tem como responsabilidade gerenciar o armazenamento dos dados repassados pelo *NameNode* (HADOOP, s.d.[a]). A Figura 23 demonstra a organização arquitetural do HDFS.

Figura 23 – Exemplo da arquitetura *HDFS*



Fonte – (HADOOP, s.d.[a])

A Figura 23 também mostra a comunicação entre os clientes, *NameNode* e *DataNodes*. No primeiro caso, o cliente solicita os metadados referentes a determinado dado ao *NameNode*, que retorna o nome e caminho do dado, quantas réplicas desse

dado existem, dentre outras informações. No segundo caso, o cliente realiza uma operação de escrita e, baseado na configuração de réplica do HDFS, o dado é replicado entre alguns *DataNodes*. Vale ressaltar que um *NameNode* pode ter um ou vários *DataNodes* vinculados, e um *DataNode* está vinculado a apenas um *NameNode*.

Além disso, o HDFS implementa criptografia de ponta a ponta, nos dados, de forma transparente ao usuário. Uma vez configurados os diretórios no HDFS, os dados são criptografados e descriptografados de forma transparente, sem exigir alterações no código do aplicativo do usuário. Além disso, o HDFS nunca armazena ou tem acesso a dados não criptografados ou chaves de criptografia de dados não criptografados (HADOOP, s.d.[b]).

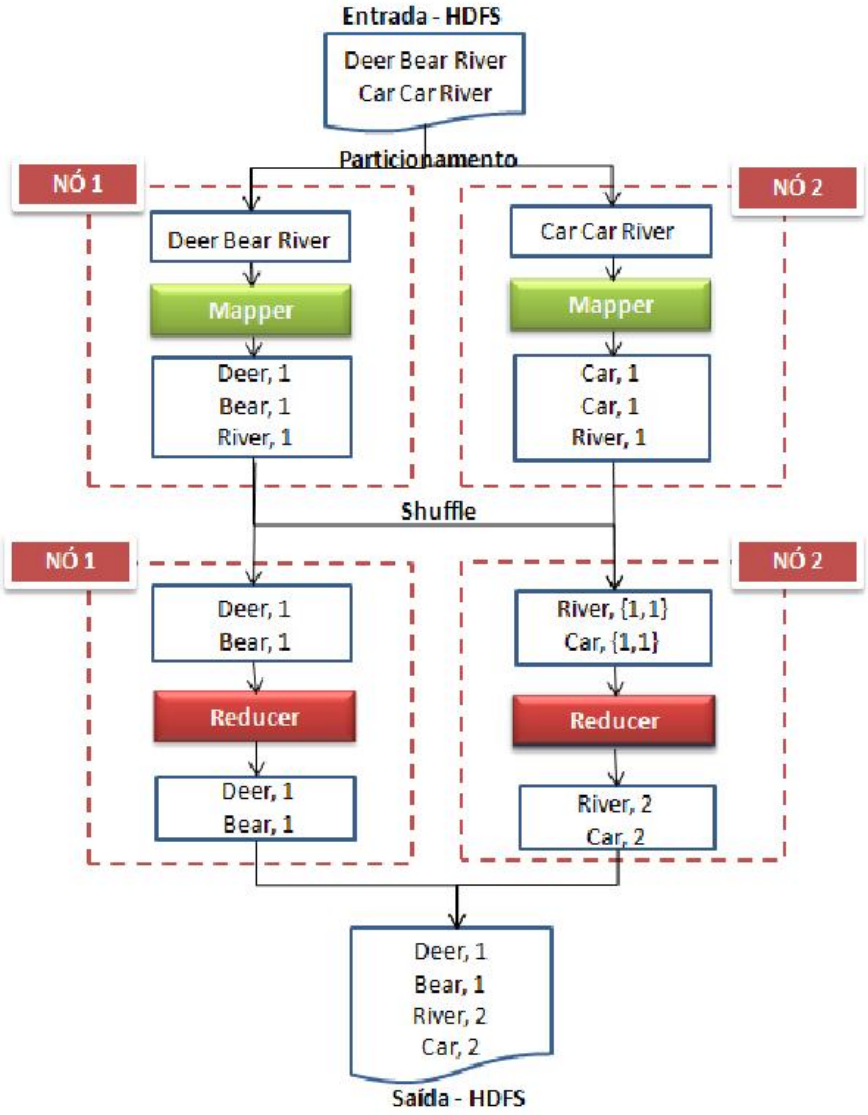
6.2.1.2 Hadoop MapReduce

O MapReduce é um *software* que facilita a escrita de aplicações para realizar processamento em grandes quantidades de dados de forma paralela em grandes redes de computadores. Os programas que rodam no MapReduce são chamados de *jobs* e devem implementar obrigatoriamente duas funções: um método chamado *Map*, que organiza os dados num formato chave/valor, também conhecido por mapa (*map*), e um método chamado *reduce*, que recebendo as informações do método anterior realiza os respectivos processamentos dos dados (WHITE, 2015).

No início de um *job*, cada nó do cluster executa tarefas *Map* e lê os blocos de dados de entrada que estão localmente disponíveis para o nó. Após a fase *Map* tem-se a *Shuffle*, em que a saída da fase *Map* é classificada e agrupada de acordo com a sua chave. Posteriormente ocorre a fase *Reduce*, que recebendo os dados prontos da fase anterior realiza os respectivos processamentos implementados e ao final do *job* todos os valores são escritos no HDFS. A Figura 24 mostra um exemplo de execução do *job* (NUNES, 2016).

A Figura 24 mostra um exemplo de um programa que realiza a contagem de palavras em determinado arquivo. Devido ao arquivo estar armazenado no HDFS, diversas instâncias da fase *Map* são criadas em cada nó do *cluster*. Em cada instância de execução e para cada palavra encontrada, a tarefa *Map* gera um par cuja a chave é a palavra e o inteiro "1"é o valor, gerando como saída o par (palavra, 1). Após as execuções das tarefas *Map*, os pares são organizados e agrupados pelas suas respectivas chaves na fase *Shuffle*. Posteriormente, na face *Reduce*, cada tarefa *Reduce* processa a lista de valores associados a uma palavra específica, garantindo que cada palavra seja processada por uma única tarefa Reduce. A lista de valores é uma lista de valores "1"s. A tarefa Reduce soma estes valores gerando uma contagem final associada a uma palavra, emitindo como saída final o par (palavra, contagem), que é gravado em um arquivo de saída armazenado no HDFS (NUNES, 2016).

Figura 24 – Exemplo do funcionamento do *MapReduce*



Fonte – Extraído de (NUNES, 2016)

6.2.2 Segurança

O componente de segurança no DLMS tem como responsabilidade realizar a gestão de todo o controle de confidencialidade, que inclui autenticação, autorização e auditoria dos dados (JOAQUIM; SANTOS MELLO, 2020). Para a gestão de segurança no DLMS foi escolhida a ferramenta *Apache Ranger*. Segundo a página oficial da ferramenta, o *Apache Ranger* é um framework que habilita, monitora e gerencia a segurança de dados na plataforma Hadoop.

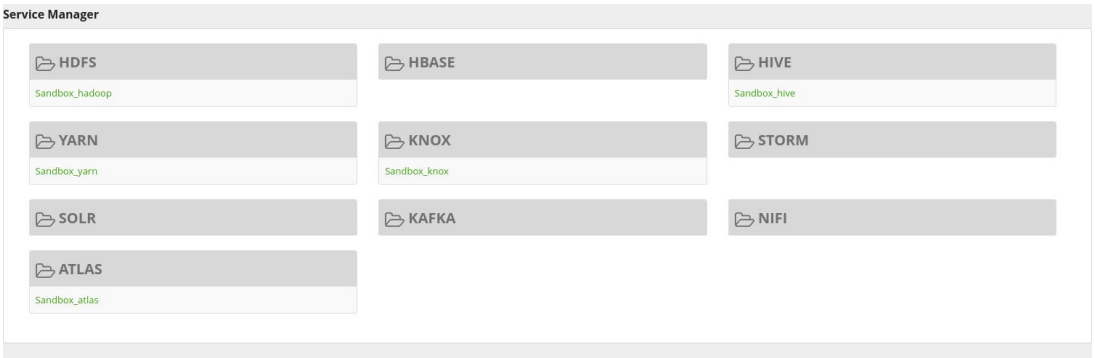
Os principais objetivos do Apache Ranger são: (*i*) todas as funções relacionadas à segurança devem estar disponíveis para serem administradas de forma centralizada em uma interface visual na Web ou através da *API REST*[10]; (*ii*) gestão de autoriza-

[10] API REST, também chamada de API RESTful, é uma interface de programação de aplicações (API

ção a operações ou ações nos componentes que o Ranger estiver integrado deve ser realizada pela central de administração citada no item anterior; (*iii*) padronização do método de autorização em todos os componentes da plataforma *Hadoop* integradas a ele; (*iv*) suporte a diversos métodos de autorização, como o controle de acesso baseado em papéis (*role-based access control (RBAC)*) e o controle de acesso baseado em atributos (*attribute-based access control (ABAC)*); (*v*) auditoria de acesso dos usuários aos componentes do *Hadoop*, bem como dados dentro destes componentes, integradas a ele. Com relação aos métodos RBAC e ABAC, o primeiro concede os direitos de acesso a determinado recurso baseado no papel que ele assume. Por exemplo, um usuário com papel de médico pode acessar o prontuário de determinado paciente enquanto que o paciente somente acessa seus respectivos dados. Já o ABAC controla o acesso com base em quais operações são permitidas ao usuário. Um exemplo seria um usuário que tem permissão de leitura e escrita nos arquivos de prontuários dos pacientes de uma UBS e outro usuário somente pode ler os respectivos arquivos.

O *Apache Ranger* fornece uma estrutura de segurança centralizada para gerenciar o controle de acesso na plataforma *Hadoop*. Através desta estrutura é possível gerenciar as políticas de acesso a determinados recursos (por exemplo, um arquivo, pasta, banco de dados, tabela ou coluna) para um determinado usuário ou grupos de usuários. Além disso, ele também permite o rastreamento de auditoria e análise de política para um controle mais refinado dentro do DL (RANGER, s.d.[a]). A Figura 25 apresenta a tela inicial após o usuário se autenticar dentro do Ranger.

Figura 25 – Tela inicial do Apache Ranger



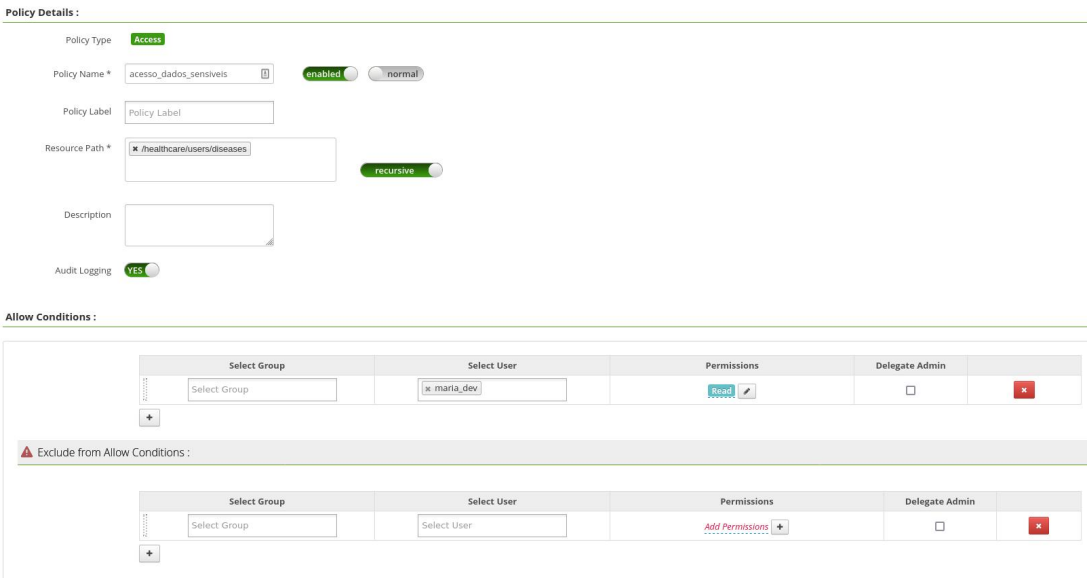
Conforme mostra a Figura 25, é possível visualizar todos os componentes integrados com o Ranger e ir para a tela cadastral de um componente. O Ranger também permite o registro de componentes a serem integrados ao DL, disponibilizando um *plug-in* necessário para que a comunicação entre ambos de fato funcione. Com a integração funcionando entre ambos, o Ranger consegue enviar ao respectivo com-

ou API web) que está em conformidade com as restrições do estilo de arquitetura REST, permitindo a interação com serviços web RESTful. REST é a sigla em inglês para "Representational State Transfer", que em português significa tansferência de estado representacional. Exemplo é realizar uma chamada REST do tipo GET para o endereço www.google.com.br

ponente as políticas de segurança cadastradas, e o componente integrado envia as informações de acesso dos usuários, facilitando a gestão centralizada de autoria de acesso (RANGER, s.d.[a]).

Com relação ao controle de acesso, o Ranger centraliza nele essa responsabilidade e permite a utilização de vários tipos de controle, como o RBAC e o ABAC. As políticas de acesso são cadastradas no Ranger e os componentes integrados a ele recebem essas políticas para serem executadas. As políticas de acesso são gerenciadas pelo próprio Ranger. Elas definem quais ações (por exemplo leitura, escrita ou execução) determinados usuários ou grupos podem realizar nos componentes especificados (RANGER, s.d.[a]). Para exemplificar essa funcionalidade, a Figura 26 mostra a tela de cadastro de política de acesso para que determinado usuário somente possa acessar determinada pasta dentro do HDFS.

Figura 26 – Tela cadastro de política de acesso considerando um usuário para determinado componente



Na parte de auditoria, os componentes enviam ao Ranger os eventos de auditoria (por exemplo, a execução de determinada ação por determinado usuário) com o objetivo de centralizar a inspeção e possíveis auditorias no DL (RANGER, s.d.[a]). A Figura 27 apresenta um exemplo de operação realizada na camada de armazenamento que determinado usuário solicitou a criação de pastas dentro do DLMS. Nela, pode-se observar dados importantes para auditoria, como a data e hora do evento, quem realizou a operação, qual operação, qual serviço alvo da operação, endereço do serviço, entre outros.

Além das funcionalidades disponíveis para autenticação, autorização e auditoria, o Ranger também realiza a gestão de chaves criptográficas a partir do módulo *Ranger Key Management Service (Ranger KMS)*. O Ranger KMS é baseado no Hadoop KMS, originalmente desenvolvido pela comunidade Apache. Mas, ao contrário do Hadoop

Figura 27 – Exemplo de operação realizada dentro do DLMS e apresentada pelo Apache Ranger

Policy ID	Policy Version	Event Time	Application	User	Service (Name / Type)	Resource (Name / Type)	Access Type	Permission	Result	Access Enforcer	Agent Host Name	Client IP	Cluster Name	Zone Name	Event Count	Tags
1	--	06/22/2021 03:54:41 PM	ozone	hadoop	storage ozone	gdpr-raw volume	create	create	Allowed	ranger-acl	om	172.19.0.12			1	--
1	--	06/22/2021 03:54:05 PM	ozone	hadoop	storage ozone	raw volume	create	create	Allowed	ranger-acl	om	172.19.0.12			1	--

KMS, que armazena chaves em um keystore Java baseado em arquivo por padrão, o Ranger KMS estende a funcionalidade nativa do Hadoop KMS, permitindo que você armazene as chaves em um BDs seguro. O Ranger realiza o gerenciamento de chaves por meio do seu portal de administração (RANGER, s.d.[b]).

Considerando as funcionalidades apresentadas anteriormente, podemos concluir que o Apache Ranger é uma tecnologia adequada à gestão de segurança de um DLMS. Além disso, considerando os requisitos funcionais levantados na Seção 4.1, o Ranger atende a *RF02* com a funcionalidade de edição de políticas de acesso, o Ranger possibilita aplicação das técnicas de privacidade de dados dependendo de cada componente que se integra a ele.

6.2.3 Metadados

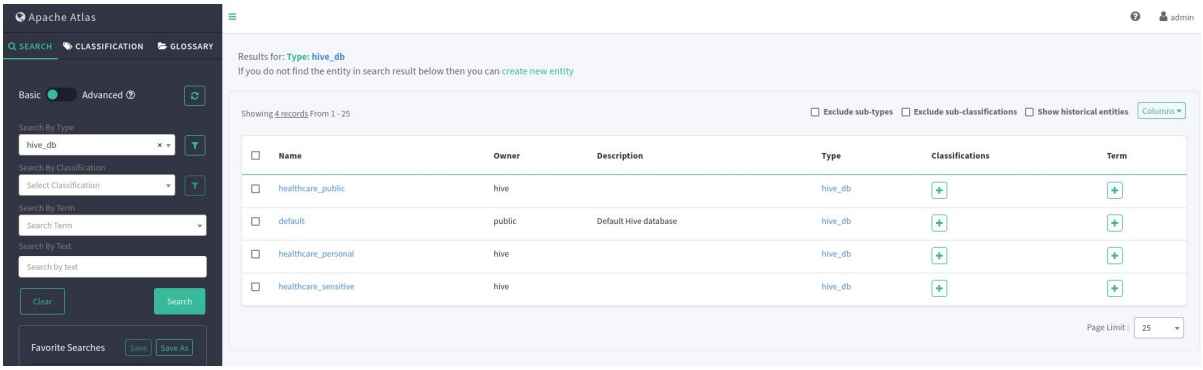
O componente de metadados do DLMS tem como objetivo armazenar todos os metadados que foram parcialmente coletados pela camada de ingestão ou serão posteriormente adicionados manualmente durante a curadoria ou uso do DLMS. Além disso, ela auxilia os gestores do DLMS a ter uma visão geral dos dados armazenados, disponibilizando um catálogo dos metadados (QUIX; HAI, 2018). Entre essas e outras responsabilidades desse componente, mais as ferramentas disponíveis para essa função nos DLs, optou-se pela utilização do *Apache Atlas* por ser um framework aberto, escalável, extensível e integrado a plataforma Hadoop (ATLAS, s.d.).

As principais funcionalidades do Atlas são: (*i*) definição de tipos de metadados nos dados armazenados na plataforma Hadoop; (*ii*) classificação dos dados. Na área da saúde, por exemplo, dados do prontuário do paciente podem ser classificados como sensíveis, pois contêm dados de identificação do paciente e as comorbidades que ele sofre; (*iii*) rastreabilidade dos dados; (*iv*) disponibilização de uma interface para o usuários pesquisarem sobre os tipos de dados, metadados e classificações; (*v*) controle de acesso aos dados e metadados baseado nas classificações (ATLAS, s.d.).

O *Apache Atlas* (Atlas) fornece recursos de governança e gerenciamento de metadados para que as organizações criem um catálogo de seus ativos de dados, classifiquem e governem esses ativos e forneçam recursos de colaboração em torno desses ativos de dados para cientistas de dados, analistas e a equipe de governança de dados. A Figura 28 apresenta a tela inicial após o usuário se autenticar no Atlas.

A tela inicial do Atlas mostra a funcionalidade de pesquisa de metadados. Con-

Figura 28 – Tela inicial do Apache Atlas



forme é observado na Figura 28, optou-se por pesquisar todos os metadados do tipo "hive_db", que representa todos os BDs criados no *Apache Hive*. Além disso, após o resultado da pesquisa, é possível classificar o metadado conforme a necessidade.

O Atlas, além de ser o componente principal para gestão de metadados, também permite que outros componentes de um DL possa enviar informações relativas aos metadados, disponibilizando um *plug-in* para que a comunicação entre ambos de fato funcione. No caso do Atlas, esse *plug-in* é chamado *hook*. Os *hooks* permitem que o componente integrado envie informações dos metadados ao Atlas quando ocorre um determinado evento nele, como a criação de uma tabela ou uma inserção de dados, com o objetivo de registrar e centralizar essas informações (ATLAS, s.d.).

Entre as integrações que são possíveis no Atlas, a que mais se destaca em relação ao tema dessa dissertação é a integração com o Ranger. Esta integração permite autorização/mascaramento de dados no acesso a dados com base em classificações associadas a entidades no Atlas. Por exemplo, quem pode acessar dados classificados como pessoais ou sensíveis, ou usuários de atendimento ao cliente só podem ver os últimos 4 dígitos das colunas classificadas como cartão de crédito (ATLAS, s.d.).

Considerando as funcionalidades apresentadas anteriormente, podemos concluir que o Atlas é uma tecnologia adequada à gestão de metadados do DLMS. Considerando os requisitos funcionais levantados na Seção 4.1, o Atlas dá apoio ao Ranger no cumprimento das RF02, RF05 e RF06. Além disso, ele atende o RF03 sendo um dos seus objetivos principais a rastreabilidade dos dados. Por fim, atende a RF04 por disponibilizar aos usuários um catálogo dos metadados.

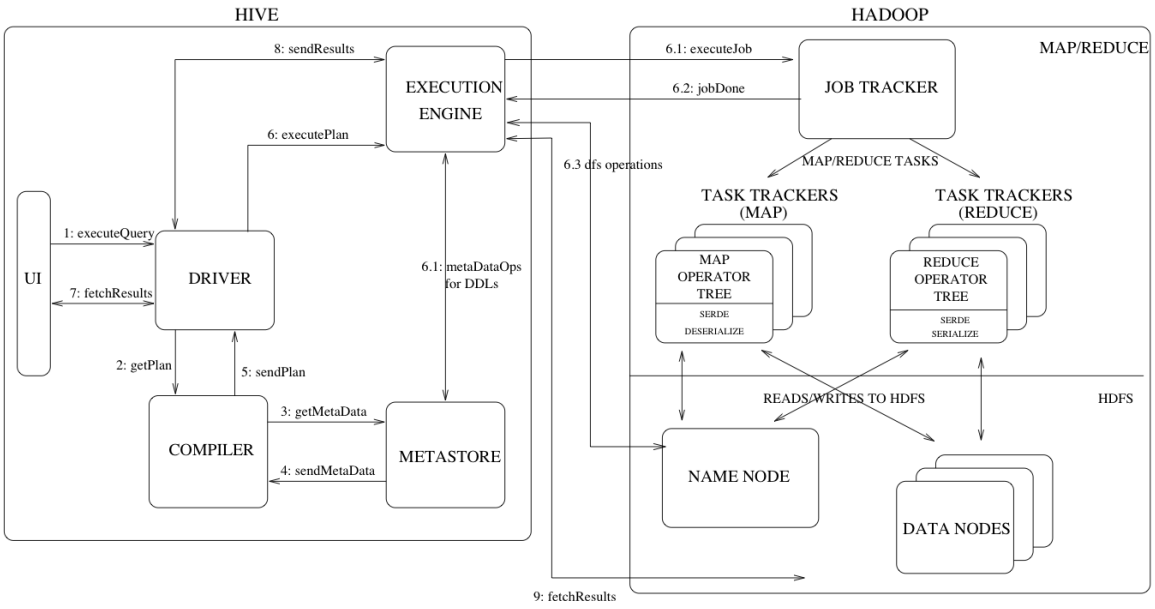
6.3 CAMADA DE TRANSFORMAÇÃO E INTERAÇÃO

Conforme descrito na Seção 2.1, a camada de transformação tem como objetivo limpar, processar e converter os dados de sua forma natural para um novo formato mais acessível aos usuários. Já a camada de interação fornece ferramentas para acessar dados e metadados para atender as necessidades do usuário. Para representar ambas as camadas nessa dissertação foi escolhido o *Apache Hive* (Hive).

O Hive é um software que facilita a leitura, escrita e gerenciamento de grandes conjuntos de dados armazenados no HDFS. Algumas de suas principais características são: (*i*) Permite aos usuários do DLMS acessar dados utilizando SQL, possibilitando geração de relatórios e análise de dados; (*ii*) Disponibiliza mecanismo para estruturar os dados considerando a variedade de formatos dentro de um DLMS; (*iii*) Acesso aos arquivos armazenados no HDFS ou em outros SGBDs, como por exemplo o *Apache HBase*; (*iv*) Possibilita a execução das consultas via outros componentes da plataforma Hadoop, como o *Apache Tez*, *Apache Spark* ou *Hadoop MapReduce* (HIVE, s.d.[a]).

O Hive possui diversos componentes com responsabilidades definidas: (*i*) *User Interface (UI)*: interface para que os usuários submetam as consultas e outras operações ao sistema, podendo ser enviados via *web* ou linha de comando; (*ii*) *Driver*: responsável por receber as consultas, gerenciar as sessões abertas pelos usuários e disponibilizar uma interface padronizada para comunicação; (*iii*) Compilador (*Compiler*): analisa as consultas, realiza análise semântica e gera um plano de execução da consulta; (*iv*) *Metastore*: armazena todas as informações de estrutura das várias tabelas e partições, incluindo informações de coluna e tipo de coluna; (*v*) *Execution Engine*: executa o plano de execução criado pelo compilador (HIVE, s.d.[b]). A Figura 29 mostra os componentes descritos anteriormente.

Figura 29 – Exemplo da arquitetura *Hive*



Fonte – (HIVE, s.d.[b])

A Figura 29 mostra também como os componentes do Hive interagem para execução de uma consulta. A UI ativa a interface de execução para o *Driver* (etapa 1). O Driver cria um identificador de sessão para a consulta e envia a consulta ao compilador para gerar um plano de execução (etapa 2). O compilador obtém os metadados necessários do metastore (etapas 3 e 4). Esses metadados são usados para realizar

validações referentes ao modelo de dados. O plano gerado pelo compilador (etapa 5) é um grafo acíclico (DAG) para ser executado pelo Hadoop MapReduce no HDFS. O mecanismo de execução submete esses estágios aos componentes apropriados (etapas 6, 6.1, 6.2 e 6.3). Para consultas, o conteúdo do arquivo temporário é lido pelo mecanismo de execução diretamente do HDFS como parte da chamada de busca do Driver (etapas 7, 8 e 9) (HIVE, s.d.[b]). Além disso, como forma de exemplificar a utilização do Hive, a Figura 30 demonstra a utilização dele utilizando a interface via linha de comando.

Figura 30 – Exemplo da arquitetura *Hive*

```
[hive@sandbox-hdp ~]$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/hdp/3.0.1.0-187/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/hdp/3.0.1.0-187/hadoop/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Connecting to jdbc:hive2://sandbox-hdp.hortonworks.com:2181/default;password=hive;serviceDiscoveryMode=zooKeeper;user=hive;zooKeeperNamespace=hiveserver2
22/10/12 21:59:25 [main]: INFO jdbc.HiveConnection: Connected to sandbox-hdp.hortonworks.com:10000
Connected to: Apache Hive (version 3.1.0.3.0.1.0-187)
Driver: Hive JDBC (version 3.1.0.3.0.1.0-187)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 3.1.0.3.0.1.0-187 by Apache Hive
0: jdbc:hive2://sandbox-hdp.hortonworks.com:2> create table pessoa(id bigint, nome string, data_nascimento date);
INFO  : Compiling command(queryId=hive_20221012215931_4f98de4a-814a-46f6-bc7f-2bd10a950932): create table pessoa(id bigint, nome string, data_nascimento date)
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Returning Hive schema: Schema(fieldSchemas:null, properties:null)
INFO  : Completed compiling command(queryId=hive_20221012215931_4f98de4a-814a-46f6-bc7f-2bd10a950932); Time taken: 0.192 seconds
INFO  : Executing command(queryId=hive_20221012215931_4f98de4a-814a-46f6-bc7f-2bd10a950932): create table pessoa(id bigint, nome string, data_nascimento date)
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=hive_20221012215931_4f98de4a-814a-46f6-bc7f-2bd10a950932); Time taken: 0.706 seconds
INFO  : OK
No rows affected (1.815 seconds)
0: jdbc:hive2://sandbox-hdp.hortonworks.com:2>
```

Fonte – Elaborado pelo autor

Conforme é demonstrado na Figura 30, é observado via linha de comando o usuário hive executando o comando *hive*, no qual conecta a interface do Hive localmente. Após isso, o usuário cria uma tabela chamada pessoa com três atributos e tendo um retorno da ferramenta de sucesso.

6.4 CONSIDERAÇÕES FINAIS

Este capítulo descreve as tecnologias sugeridas por esta dissertação para a arquitetura proposta, visando atender os requisitos funcionais da Seção 4.1. A Tabela 6 relaciona os requisitos e as tecnologias sugeridas para as camadas.

Tabela 6 – Relação entre requisitos legais e tecnologias sugeridas.

	Sqoop	HDFS/MapReduce	Ranger	Atlas	Hive
RF01	X				
RF02		X	X		
RF03				X	
RF04				X	
RF05					X
RF06					X

Fonte – Elaborada pelo autor

Para o *RF01*, o Sqoop atende esse requisito de forma parcial nos cenários de ingestão de dados relacionais e de forma em lote. Para o *RF02*, o HDFS/MapReduce e Ranger realizam a aplicação das políticas de segurança dependendo de qual técnica de privacidade será aplicada. O Atlas atende completamente os requisitos *RF03* e *RF04* ao rastrear os dados e permitir o titular acessar os dados dentro do DLMS. Por fim, os requisitos *RF05* e *RF06* são atendidos via Apache Hive.

7 PROVA DE CONCEITO

Este capítulo descreve uma prova de conceito para validar a arquitetura e as tecnologias propostas nos capítulos anteriores. Ele está dividido em três seções: *contextualização*, onde apresentamos o domínio escolhido (saúde pública) e um *software* utilizado usado como base; *implementação*, onde definimos um ambiente de teste com as tecnologias sugeridas no capítulo 6 e as validamos com base nos requisitos funcionais levantados na seção 4.1; e as *considerações finais*.

7.1 CONTEXTUALIZAÇÃO

Esta prova de conceito é aplicada através de um estudo de caso no domínio da área da Saúde pelo fato desta apresentar dados relevantes em termos de controle de privacidade. No Brasil, o Sistema Único de Saúde (SUS) é um amplo sistema de saúde, abrangendo desde o simples atendimento para avaliação da pressão arterial, por meio da atenção primária, até o transplante de órgãos, garantindo acesso integral, universal e gratuito para toda a população do país. Além disso, o SUS proporciona o acesso à saúde para toda a população, sem discriminação. Para que a saúde seja atendida em todo o território, o SUS é composto pelo ente federal, representado pelo Ministério da Saúde (MS), pelas secretarias estaduais e pelas secretarias municipais de saúde. No caso do Ministério da Saúde, o gestor da unidade atua como o Gestor Nacional do SUS, tendo como atribuição formular, normatizar, fiscalizar, monitorar e avaliar políticas e ações, em articulação com o Conselho Nacional de Saúde (CNS) (SAÚDE, s.d.[e]). Além disso, na sua estrutura hierárquica, o MS é composto por assessorias, diretorias e secretarias, como por exemplo, a Secretaria de Atenção Especializada à Saúde (SAES), Secretaria de Vigilância em Saúde (SVS) e Secretaria de Atenção Primária à Saúde (SAPS) (SAÚDE, s.d.[d]).

Neste estudo de caso escolhemos o projeto *e-SUS Atenção Primária (e-SUS APS)*. O e-SUS APS é uma estratégia do Departamento de Saúde da Família (DESF) para estruturar a APS em nível nacional. O e-SUS APS possui dois *softwares* e dois aplicativos para coleta de dados. Um dos softwares é o *Sistema com Coleta de Dados Simplificada (CDS)*, que apoia o processo de coleta de dados relativos aos pacientes, domicílios, atendimento, vacinação, entre outros. O outro software é o *Sistema de Prontuário Eletrônico do Cidadão (PEC)*, que apoia o processo de informatização das Unidades Básicas de Saúde (UBS), como cadastro dos funcionários, alocação de atendimento dos profissionais aos pacientes e definição do horário de atendimento. Com relação aos aplicativos, temos o *e-SUS Território*, que auxilia os Agentes Comunitários de Saúde (ACS), Agentes de Combate às Endemias (ACE) e os Agentes de Ação Social (AAS) no cadastro de informações relativas à saúde e atendimentos dos cidadãos, e o *e-SUS Atividade Coletiva (e-SUS AC)*, responsável pela coleta de dados

das atividades coletivas realizadas pela equipe fora da UBS. Dentre esses sistemas, escolhemos o e-SUS PEC pelo fato dele possuir dados comuns a outros sistemas, como cadastro das UBS, além de dados pessoais e sensíveis (SAÚDE, s.d.[c]).

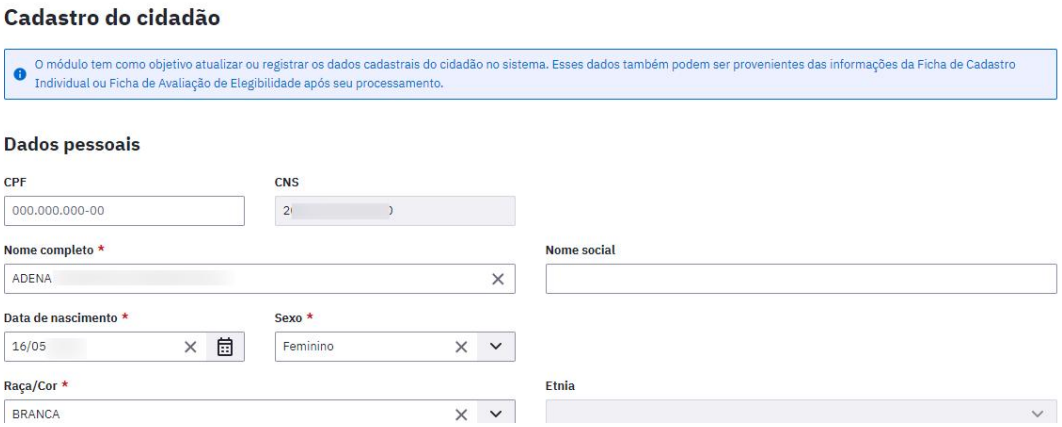
O PEC é um software público, de responsabilidade do DESF e da SAPS, sendo o desenvolvimento realizado em cooperação com a Universidade Federal de Santa Catarina (UFSC). Ele auxilia, dentre outras coisas, no: (*i*) Gerenciamento das Unidades de APS, em especial, agendas e processos de trabalhos; (*ii*) Registro e organização dos prontuários eletrônicos para os profissionais de saúde; (*ii*) Registro e organização dos dados referentes às fichas eletrônicas; (*iv*) Monitoramento e avaliação das ações de saúde no território. Para realizar essas ações, o PEC organiza o software internamente por meio de módulos (SAÚDE, s.d.[c]).

Neste estudo de caso iremos focar nos módulos *Cidadãos* e *Atendimentos*. O primeiro é o responsável por questões relacionadas ao gerenciamento do cadastro do cidadão: *(i)* Busca pelo cidadão; *(ii)* Visualizar dados do cidadão; *(iii)* Adicionar cidadão na base local; *(iv)* Adicionar ou alterar cidadão na base nacional; *(v)* Preenchimento do formulário de cadastro. Na funcionalidade *(i)*, a busca pelo cidadão se dá por diversos critérios, entre eles o número do CPF ou do Cartão Nacional de Saúde (CNS), além de buscar tanto no cadastro local do PEC como na base nacional. Após uma busca bem sucedida, ele apresenta as informações do cidadão encontrado utilizando a funcionalidade *(ii)*, que também mostra o prontuário do cidadão com os agendamentos. A funcionalidade *(iii)* adiciona o cidadão na base local, pois esses dados são necessários para que o cidadão possa ser atendido na UBS com o PEC instalado, que pode vir de três fontes: da base nacional, da integração com o software CDS ou direto na base local via funcionalidade *(v)*. A funcionalidade *(iv)* torna possível o cidadão ir a qualquer UBS com o PEC instalado, atualizar suas informações e sincronizar elas com a base nacional. A Figura 31 mostra um exemplo da funcionalidade *(v)* referente ao cadastro do cidadão na base local do PEC (SAÚDE, s.d.[a]). O usuário pode cadastrar dados pessoais e sensíveis na base local do PEC, como nome completo, CPF, data de nascimento e sexo.

O módulo de *Atendimentos* é responsável por garantir a ordenação do fluxo de atendimento que ocorre nas UBS. Suas funcionalidades são: *(i)* Lista de Atendimentos; *(ii)* Escuta Inicial; *(iii)* Realizar Vacinação; *(iv)* Atender - Prontuário do Cidadão; *(v)* Atendimento/Acompanhamento Específico; *(vi)* Registro Tardio de Atendimento.

A funcionalidade *(i)* auxilia na organização das ações realizadas aos cidadãos que já entraram no fluxo de atendimento, seja por um atendimento agendado, seja por uma demanda espontânea. A funcionalidade *(ii)* representa o primeiro atendimento realizado ao cidadão em demanda espontânea na UBS, coletando dados subjetivos e medições objetivas, além de classificar o risco/vulnerabilidade de acordo com a avaliação do risco biológico e da vulnerabilidade subjetivo-social do indivíduo. A fun-

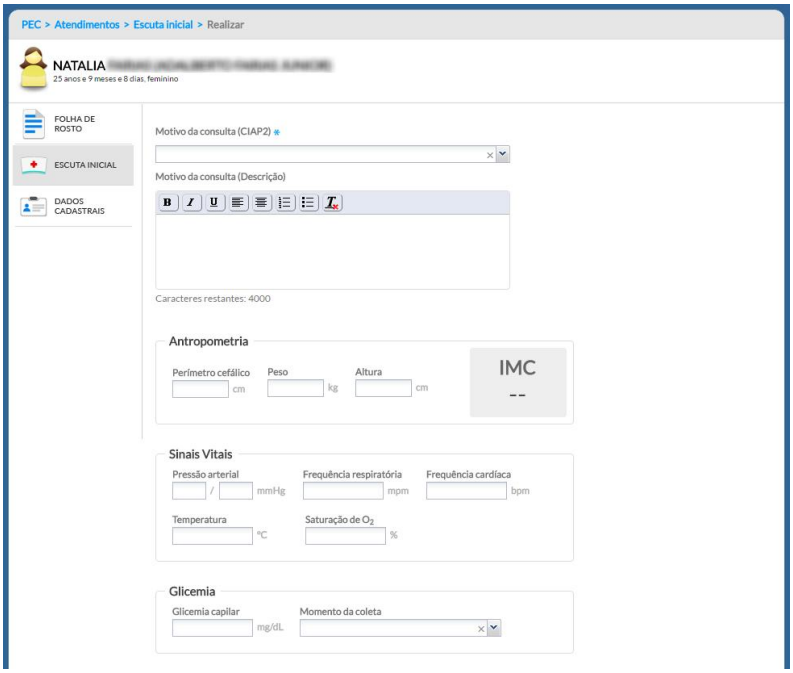
Figura 31 – Cadastro do cidadão na base local do PEC



Fonte – (SAÚDE, s.d.[a])

cionalidade *(iii)* possibilita a organização de todo o processo relacionado ao assunto. As funcionalidades *(iv)* e *(v)* funcionam como telas iniciais para atender o cidadão, reunindo todos os links necessários para o atendimento, desde cadastro e consulta ao histórico do cidadão. Por fim, a funcionalidade *(vi)* possibilita ao usuário a transcrição dos atendimentos que não foram registrados no momento em que de fato ocorreu a consulta, como por exemplo, os atendimentos individuais realizados fora da UBS ou aqueles nos quais o sistema estava indisponível por qualquer motivo.

Figura 32 – Cadastro do atendimento inicial na base local do PEC



Fonte – (SAÚDE, s.d.[b])

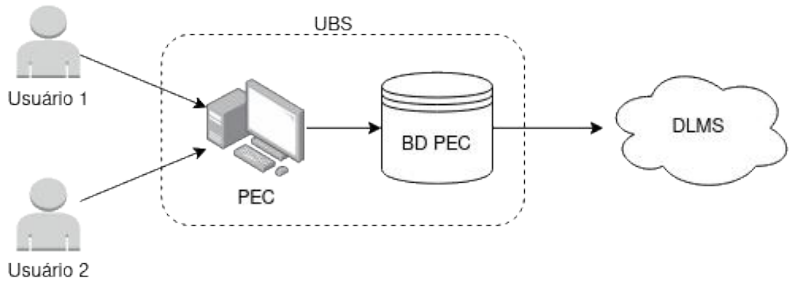
A Figura 32 mostra um exemplo da funcionalidade *(ii)* referente ao cadastro do atendimento inicial do cidadão na base local do PEC. O usuário, ao atender o cidadão e

registrar dados do atendimento inicial, cadastra dados pessoais e sensíveis do cidadão na base local do PEC (SAÚDE, s.d.[b]).

7.2 IMPLEMENTAÇÃO

O estudo de caso proposto se baseia no cenário apresentado na Figura 33. Ele considera a utilização do sistema PEC em uma Unidade de Pronto Atendimento (UPA). Na parte esquerda temos dois usuários interagindo com o sistema PEC. O *software*, representado pelo ícone de um computador e um BD, computa e armazena os dados repassadas pelos usuários. Por fim, os dados no BD são enviados para o DLMS, representado por um ícone de nuvem.

Figura 33 – Cenário de uso com exportação de dados para o DLMS



Fonte – Elaborado pelo autor

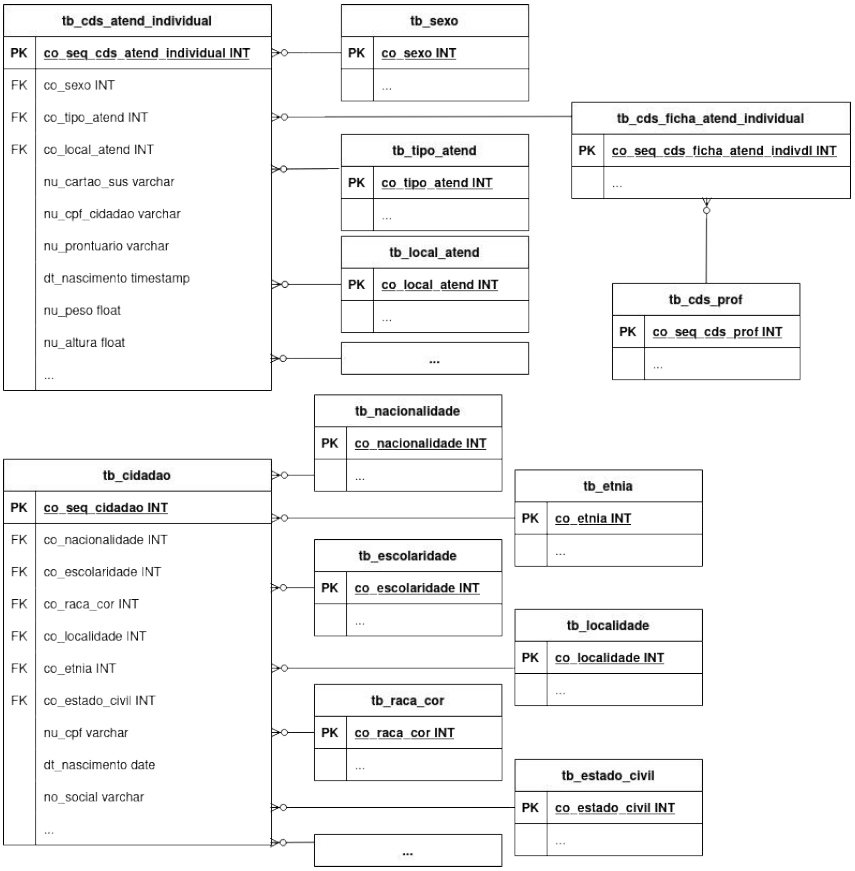
O BD relacional do PEC, na sua versão 4.5.5, conta com mais de 800 tabelas. Devido ao grande volume de dados, para este estudo de caso escolhemos duas tabelas como referência, que armazenam dados pessoais e sensíveis, mostradas na Figura 34: *tb_cds_atend_individual* e *tb_cidadao*. A primeira mantém dados sobre atendimentos realizados e a segunda armazena dados de cidadãos.

Conforme apresentado na Figura 34, além das tabelas descritas anteriormente, existem tabelas que complementam os dados das tabelas de referência, como nacionalidade (*tb_nacionalidade*) e etnia (*tb_etnia*). Portanto, no diagrama temos dados desde comuns até pessoais e sensíveis. Ao transpor esses dados para o DLMS deverá ser realizado todo o tratamento dos mesmos conforme o objetivo dessa dissertação.

Considerando o cenário de uso da Figura 33, iniciamos a implementação do ambiente de teste com a inserção dos dados no BD do PEC. Devido a indisponibilidade de acesso a dados reais da aplicação, criamos um *script*, disponível no Anexo A, que gera dados sintéticos e os insere nas tabelas necessárias para no domínio do PEC.

Após a geração dos dados sintéticos, é necessário definir quais tabelas serão enviadas para o DLMS e em quais áreas serão armazenadas. A Figura 35 indica quais tabelas apresentadas na Figura 34 são exportadas para o DLMS e em quais áreas os seus dados serão armazenados. A tabela *tb_cidadao* será enviada para a área de dados pessoais, a tabela *tb_cds_atend_individual* para a área de dados sensíveis,

Figura 34 – Esquema das tabelas tb_cds_atend_individual e tb_cidadao

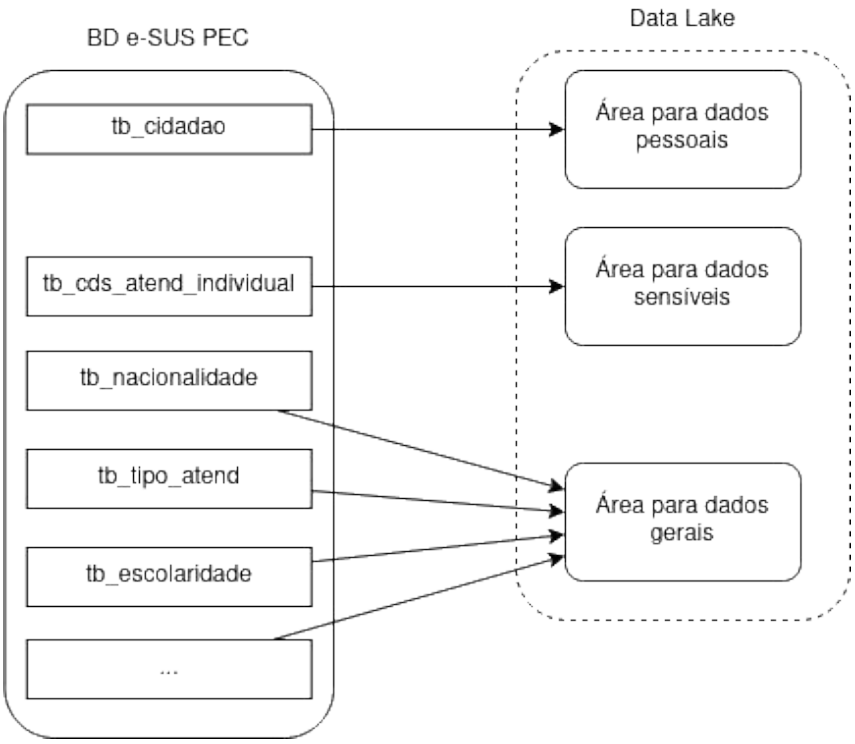


Fonte – Elaborado pelo autor

e as demais tabelas para a área de dados gerais. Essa divisão dos dados acontece devido a cada tipo de dado precisar de tratamento diferenciado, conforme explicado nos capítulos anteriores dessa dissertação.

Após o BD do PEC estar preenchido com dados válidos, mapeados para onde devem ser armazenados no DLMS e seguindo o fluxo dos dados na Figura 33, o próximo passo é a instalação de um DLMS com as respectivas tecnologias (ferramentas) citadas no Capítulo 6. Contudo, devido a complexidade de montagem de um DLMS de forma manual considerando as ferramentas citadas no Capítulo 6, optamos por utilizar um *software* terceiro que facilita esse trabalho. O *software* escolhido foi o *Hortonworks Data Platform* (HDP) 3.0. O HDP é uma plataforma que permite a instalação, configuração e monitoramento dos *softwares* necessários para montar um DLMS conforme as necessidades, podendo ser instalado em um servidor físico ou até em um ambiente em nuvem. Além disso, por gerenciar diversos programas da Apache, ele disponibiliza atualizações nos softwares suportados por ele para correções de erros e novas funcionalidades, dentre outras. Para fins de exemplificação, a Figura 36 mostra a tela inicial do Apache Ambari, um dos *softwares* que o HDP utiliza para monitoramento de todas as ferramentas da plataforma.

Figura 35 – Tabelas do PEC a serem exportadas para o DLMS e em quais áreas de dados



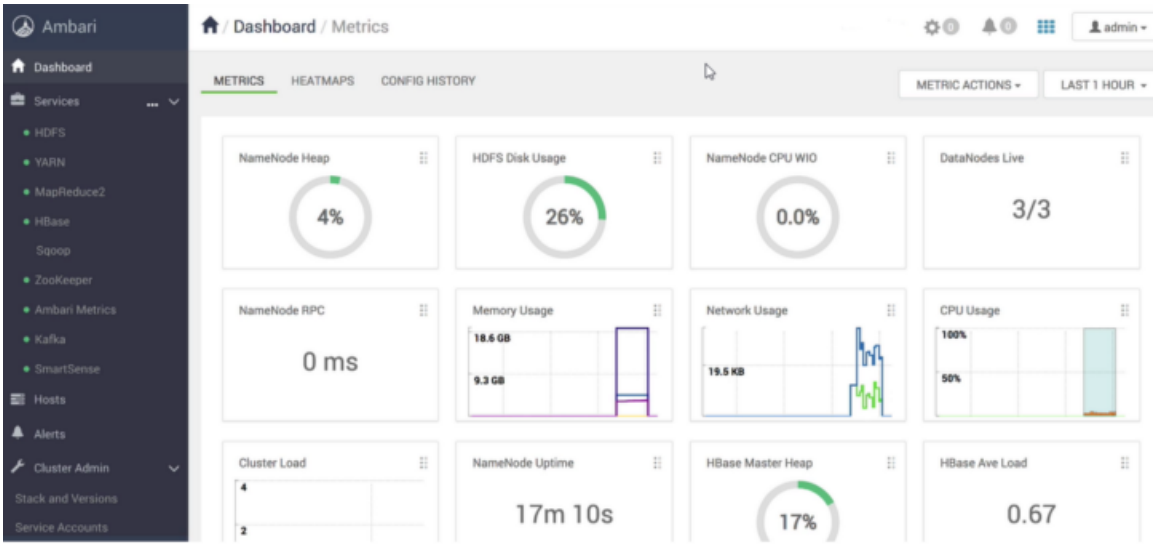
Fonte – Elaborado pelo autor

Conforme apresentado na Figura 36, o HDP, via Apache Ambari, disponibiliza ao administrador os programas necessários para implementação do DLMS. Nesse caso, podemos observar na parte da esquerda os softwares instalados pelo administrador, como o HDFS representando a camada de armazenamento, Apache HBase (SGBD colunar) e se determinado programa está ativo ou não (representados pelo botão de cor verde ou vermelha ao lado do nome do programa). Além disso, na parte central da interface com o usuário pode-se observar o monitoramento dos serviços, como o espaço em disco utilizado, tempo que o DLMS está disponível, entre outros.

Com a instalação e configuração das ferramentas selecionadas para a montagem de um DLMS, torna-se possível a importação dos dados do BD do PEC. Para realizar essa atividade, foi desenvolvido um *script* apresentado no Anexo B. Este *script* utiliza o *Sqoop*, que implementa a Camada de Ingestão apresentada na Seção 2.1. Além disso, para os dados auxiliares (tabelas *tb_cidadao* e *tb_cds_atend_individual*), optamos também por criar as tabelas no *Hive* e inserir os dados nelas como forma de facilitar a implementação do ambiente de teste. Contudo, para a tabela *tb_cidadao*, que possui dados pessoais, salvamos os dados em um diretório com as regras de segurança para esse tipo de dado. Já para a tabela *tb_cds_atend_individual* persistimos os dados em outro diretório com suas respectivas regras (ver Figura 35).

Após a execução do *script* mencionado anteriormente, as tarefas necessárias

Figura 36 – Tela inicial do Apache Ambari



Fonte – Elaborado pelo autor

na Camada de Ingestão foram concluídas. Passamos então à execução de ações na Camada de Armazenamento: criptografar os dados no HDFS conforme as políticas de segurança aplicáveis, disponibilização dos dados aos interessados permitidos e definição das políticas de acesso e segurança baseadas nos dados.

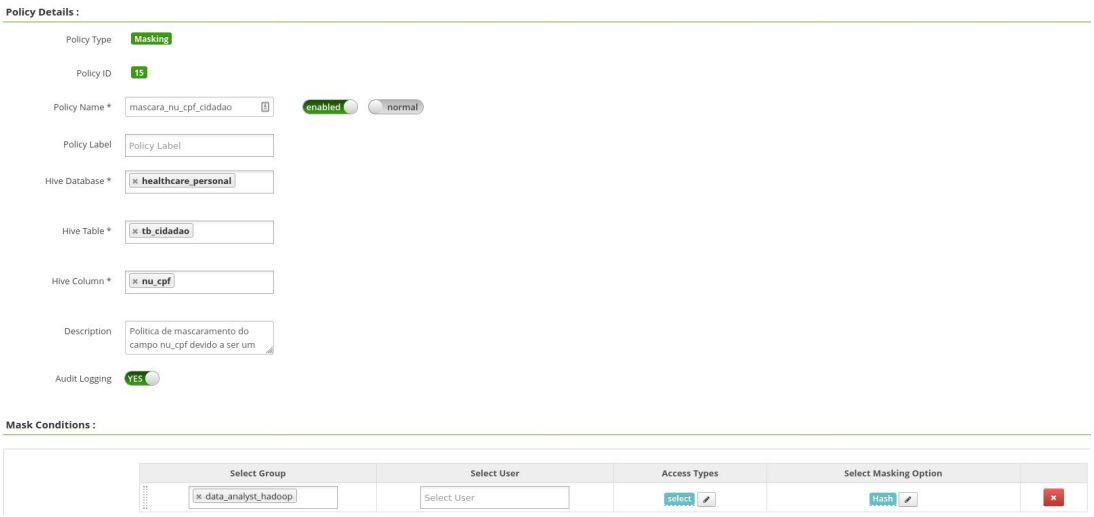
Para fins de exemplificação, consideramos que ambos os tipos de dados (pessoais e sensíveis) deverão ter seus dados criptografados nos diretórios onde estão armazenados. Para alcançar esse objetivo, o *script* descrito no Anexo C gera as chaves de encriptação e cria os diretórios no HDFS. Essas atividades representam o trabalho realizado pela Camada de Armazenamento da arquitetura apresentada na Seção 2.1 e vincula eles às chaves.

Após a realização da criptografia dos dados, é necessário disponibilizar esses dados às partes interessadas e aplicar as respectivas políticas de segurança. Inicialmente criaremos as tabelas no *Hive*, tornando possível realizar consultas utilizando a linguagem SQL. O Anexo D mostra o comando *create* de ambas as tabelas no Hive, com a localização dos dados apontada para os diretórios do HDFS criptografados.

Com relação à políticas de segurança, utilizando o *Ranger*, montamos um cenário com dois grupos. O primeiro grupo é o dos *analistas de dados*, denominado *data_analysts_hadoop*. Esse grupo pode realizar consultas no DL, mas não pode visualizar o conteúdo de colunas pessoais ou sensíveis. O segundo grupo é o dos *titulares dos dados*, denominado *data_holder*. Para exemplificar o controle do grupo de analistas de dados, a Figura 37 mostra a tela de configuração de política de segurança do Ranger. Cria-se uma política de segurança de mascaramento de dados para a coluna *nu_cpf*, da tabela *tb_cidadao*, para o grupo de usuários *data_analysts_hadoop*. O tipo de máscara escolhido foi o *hash*. O tipo de mascaramento *Hash* substitui todos os

valores da coluna referenciada por um *hash* calculado a partir de todos os valores do respectivo registro (DOCS, s.d.).

Figura 37 – Criação de política de segurança no Ranger do tipo mascaramento de dados

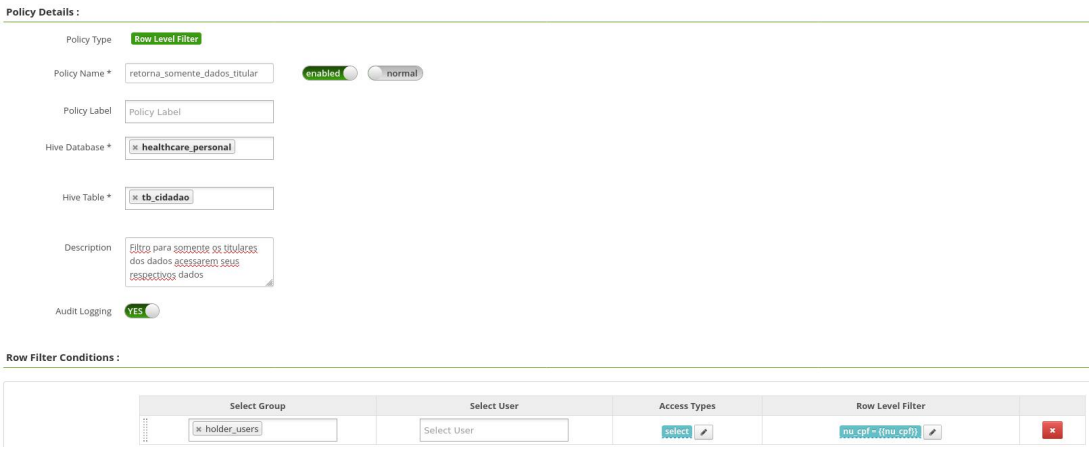


Fonte – Elaborado pelo autor

No segundo grupo temos os titulares dos dados, que podem somente acessar os seus dados. A Figura 38 mostra a tela de configuração de política de segurança do Ranger do tipo filtro de linha. Para tanto, foi criada uma política de segurança *retorna_somente_dados_titular*, que aponta para o BD *healthcare_personal* e tabela *tb_cidadao*, sendo aplicável ao grupo de usuário *holder_users*, que representa o titular do dado, com a expressão filtrando o campo *nu_cpf* igual ao cpf do usuário da sessão que se conecta ao BD (*nu_cpf = nu_cpf*), representado pela coluna *Row Level Filter*. Com essa política de segurança, o Ranger somente retorna a própria informação do titular ao executar a consulta aos dados no Hive.

Na parte dos metadados e rastreamento dos dados no DLMS, temos o Atlas como ferramenta responsável. Sobre os metadados, com as ferramentas configuradas e integradas no DLMS, o Atlas recebe das ferramentas os metadados e a localização dos dados no DLMS conforme descrito no Capítulo 6. Com esta integração, o Atlas guarda qualquer alteração nessas informações, gerando um histórico que possibilita rastrear todo o ciclo dos dados e metadados no DLMS. A Figura 39 mostra um exemplo para a tabela *tb_escolaridade*, uma tabela disponibilizada pelo Hive, que registra informações como propriedade, rastreabilidade (o Atlas se refere ao termo como *lineage*) e relacionamentos, entre outros.

Figura 38 – Criação política de segurança no Ranger do tipo filtro de linha



Fonte – Elaborado pelo autor

Figura 39 – Rastreamento dos dados da tabela *tb_escolaridade no Atlas*

Fonte – Elaborado pelo autor

7.3 CONSIDERAÇÕES FINAIS

Este capítulo valida a proposta arquitetural proposta no Capítulo 4 e instanciada com tecnologias atuais no Capítulo 6 através de um estudo de caso. Neste estudo de caso optamos pelo domínio da saúde pública devido a importância que dados pessoais e sensíveis apresentam nesse contexto. Foi considerada uma aplicação real, no caso o PEC, que é um sistema utilizado pelas UBS de todo o Brasil. Contudo, devido a questões de segurança e especificidades desse tipo de aplicação, não foi possível ter acesso a dados públicos. Logo, conforme descrito nesse capítulo, criamos um *script* que gerou dados nas tabelas principais do sistema para exemplificar a aplicação das políticas de segurança.

Sobre as políticas de segurança aplicadas aos dados do estudo de caso, optamos por não detalhar cada política por entendermos que o mais importante no

momento é demonstrar a aplicação de técnicas de segurança e privacidade, tornando esse estudo de caso, mesmo sendo da área da saúde, de fácil portabilidade a outros cenários. Logo, como forma de exemplificação do estudo de caso, para os dados públicos, optamos por separar eles dos demais para devidas precauções de segurança e melhor gerenciamento de todos os dados no DLMS. No caso dos dados sensíveis, os quais se assemelham na aplicação de políticas de segurança, não detalhamos tal aplicação pois dados sensíveis na área da saúde contêm diversas regras de domínio que não são o foco dessa dissertação.

Com base nas tecnologias selecionadas no Capítulo 6 sobre a proposta arquitetural descrita no Capítulo 4, utilizamos o HDP como implementação do ambiente de teste de um DLMS. Mesmo o HDP disponibilizando muitas ferramentas para diversos propósitos, somente utilizamos as tecnologias necessárias para mostrar o que um DLMS, de forma mínima, precisa para atender os requisitos de gerenciamento de segurança e privacidade. A Camada de Ingestão é formada por tecnologias que importam dados de fontes externas para o DLMS. A Camada de Armazenamento inclui ferramentas que gerenciem os dados, segurança e metadados e, por fim, a Camada de Transformação limpa, processa e realiza transformações nos dados com o objetivo de facilitar o acesso aos interessados.

8 CONCLUSÃO

Esta dissertação propõe a extensão de uma arquitetura de DL preexistente, no caso de Quix e Hai (2018), com o objetivo de adequá-la aos requisitos regulatórios de privacidade utilizando como referência a GDPR. Para atingir esse objetivo, realizamos uma pesquisa bibliográfica sobre os temas DL, segurança da Informação, GDPR e anonimização de dados. Na sequência, uma revisão sistemática foi executada visando levantar o estado da arte relacionada à proposta. Posteriormente, baseado na literatura sobre GDPR, organizamos os requisitos regulatórios em uma lista de requisitos funcionais e descrevemos a arquitetura conceitual do DLMS com suporte à gerência de privacidade de dados. Descrevemos também um esquema de metadados para atender dados pessoais e sensíveis, bem como o rastreamento do ciclo de vida dos dados no DL. Por fim, sugerimos um conjunto de tecnologias para o desenvolvimento do DLMS e validamos a proposta com um estudo de caso da área da saúde com foco em dados de UBS, o qual contém dados pessoais e sensíveis.

Consideramos todos os pontos elencados no parágrafo anterior contribuições desta dissertação dada a carência de uma literatura específica casando a tecnologia de DL com segurança e privacidade de dados. Esse trabalho de mestrado posibilitou uma publicação relacionada ao tema no evento (qualis B1) *International Conference on Information Integration and Web-based Applications & Services (iiWas)* no ano de 2020. O artigo, intitulado *An Analysis of Confidentiality Issues in Data Lakes* apresenta uma revisão sistemática sobre questões de confidencialidade de dados, com foco na autenticação e autorização, sobre tecnologias utilizadas em DLs. O artigo propõe também uma classificação a respeito de componentes de segurança em um DLMS e recomendações sobre este onde esses componentes podem ser posicionados em uma arquitetura de referência para um DLMS. Esta classificação foi detalhada na Seção 4.2. Um segundo artigo com a proposta e a validação da arquitetura foi submetido para um outro evento qualis B2 (*International Conference on Enterprise Information Systems - ICEIS 2023*) e estamos aguardando o resultado da revisão.

Sobre as limitações desta dissertação, na escolha das tecnologias para a arquitetura proposta (Capítulo 6) optamos por selecionar somente aquelas necessárias para posterior validação. Por exemplo, no caso da Camada de Ingestão, somente optamos pelo Sqoop para representar a importação de dados de BD relacionais do tipo em lote (*batch*). O motivo dessa limitação é a grande disponibilidade de ferramentas úteis para essa camada. Outra limitação é no estudo de caso, onde optamos por somente utilizar um *software* na área da saúde pública e selecionar alguns dados que esse *software* gerencia. A razão para essa tomada de decisão é que a área da saúde pública possui diversas regras de domínio que não são relevantes ao tema desta dissertação, sendo nosso foco a privacidade dos dados dentro do DLMS. Logo, optamos por selecionar

somente exemplos de dados públicos, pessoais e sensíveis da aplicação e demonstrar que com a proposta arquitetural é possível montar um DLMS adequado à privacidade e segurança de dados, ou seja, a mesma é viável. O projeto, desenvolvimento e avaliação de um DLMS com suporte à segurança e privacidade (ou mesmo a extensão de um DLMS existente) é um objetivo futuro imediato, porém, dada a sua complexidade, não seria possível de ser concretizado em nível de um trabalho de mestrado.

Diversos outros trabalhos futuros podem ser considerados para esta dissertação. Por exemplo, na Camada de Ingestão, percebemos a carência de trabalhos que abordam ingestão de dados em fluxo (*streaming*) com atenção à privacidade de dados. Já na Camada de Armazenamento também percebemos que nenhum trabalho aplica o modelo de privacidade diferencial (ver Seção 2.5.3) em um DLMS. Na parte de rastreabilidade, observamos que o Apache Atlas somente funciona dentro da plataforma Hadoop, sendo interessante aqui a proposta de um modelo conceitual unificado que seja útil para qualquer tecnologia de DLMS.

REFERÊNCIAS

ANDRESS, Jason. **The Basics of Information Security: Understanding the Fundamentals of InfoSec in Theory and Practice**. 2nd. [*S.l.*]: Syngress Publishing, 2014. ISBN 9780128008126.

ATLAS, Apache. **Apache Atlas**. Disponível em: https://atlas.apache.org/1.0.0/index.html. (accessed: 09.07.2022).

BARANOVIC, S. Tovernic; V. Banovic; Z. Hrastic; K. Plantic; A. Sandic; M. Solution for detecting sensitive data inside a data lake. **41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)**, p. 1284–1288, 2018.

BATINI, Carol; CERI, Stefano; NAVATHE, Shamkant B. **Conceptual Database Design: An Entity-Relationship Approach**. 1st. [*S.l.*]: Benjamin-Cummings Publishing, 1991. ISBN 9780805302448.

BRITO, Felipe; MACHADO, Javam. Preservação de Privacidade de Dados: Fundamentos, Técnicas e Aplicações. *In*: [*S.l.*: *s.n.*], jul. 2017. P. 40. ISBN 978-85-7669-374-1.

CÁTEDRA, Instituto de Desenvolvimento Profissional e Pós-Graduação. **GDPR: o que é e qual a diferença em relação à LGPD?** Disponível em: https://idcatedra.com.br/2021/08/gdpr-o-que-e-e-qual-a-diferenca-em-relacao-a-lgpd/. (accessed: 22.09.2022).

CAVANILLAS, Jose; CURRY, Edward; WAHLSTER, Wolfgang. **New Horizons for a Data-Driven Economy: A Roadmap for Usage and Exploitation of Big Data in Europe**. [*S.l.*: *s.n.*], jan. 2015.

CHAUHAN, Preeti; SOOD, Mohit. Big Data: Present and Future. **Computer**, v. 54, n. 4, p. 59–65, 2021.

CHEN, Yi-Hua; CHEN, Hsin-Hsin; HUANG, Po-Chun. Enhancing the data privacy for public data lakes. *In*: 2018 IEEE International Conference on Applied System Invention (ICASI). [*S.l.*: *s.n.*], 2018. P. 1065–1068.

COUTO, Julia; BORGES, Olimar; RUIZ, Duncan; MARCZAK, Sabrina; PRIKLADNICKI, Rafael. A mapping study about data lakes: An improved definition and possible architectures. **Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE**, 2019-July, p. 453–458, 2019. ISSN 23259086.

DA SILVA, Edna; MENEZES, Estera. **Metodologia da Pesquisa e Elaboração de Dissertação**. [*S.l.*: *s.n.*], jan. 2005. P. 138.

DIXON, James. **Pentaho, Hadoop, and Data Lakes**. 2010. Disponível em: `http://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes`. Acesso em: 29 dez. 2019.

DOCS, Hortonworks. **Providing Authorization with Apache Ranger: Dynamic Resource-Based Column Masking in Hive with Ranger Policies**. Disponível em: `https://docs.cloudera.com/HDPDocuments/HDP3/HDP-3.1.5/authorization-ranger/content/dynamic_resource_based_column_masking_in_hive_with_ranger_policies.html`. (accessed: 13.10.2022).

DWORK, Cynthia. Differential Privacy. *In*: BUGLIESI, Michele; PRENEEL, Bart; SASSONE, Vladimiro; WEGENER, Ingo (Ed.). **Automata, Languages and Programming**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. P. 1–12.

EL EMAM, Khaled; DANKAR, Fida Kamal. Protecting privacy using k-anonymity. eng. **Journal of the American Medical Informatics Association : JAMIA**, American Medical Informatics Association, v. 15, n. 5, p. 627–637, 2008. M2716[PII]. ISSN 1527-974X.

FANG, Huang. Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem. **IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems, IEEE-CYBER 2015**, IEEE, p. 820–824, 2015.

FRANÇA, Tiago C.; NOGUEIRA, José L. T.; ANTUNES, João F. **Minicursos da ERSI-RJ 2019 - VI Escola Regional de Sistemas de Informação do Rio de Janeiro**. 1st. Rio de Janeiro: SBC, 2019. ISBN 978-85-7669-488-5.

FUNG, Benjamin C.M.; WANG, Ke; FU, Ada Wai-Chee; YU, Philip S. **Introduction to Privacy-Preserving Data Publishing: Concepts and Techniques**. 1st. [*S.l.*]: Chapman & Hall/CRC, 2010. ISBN 1420091484.

GDPR. **General Data Protection Regulation**. [*S.l.*: *s.n.*], 2019. `https://gdpr.eu/`. Acesso em: 20 mai. 2019.

GIEBLER, Corinna; GRÖGER, Christoph; HOOS, Eva; EICHLER, Rebecca; SCHWARZ, Holger; MITSCHANG, Bernhard. The Data Lake Architecture Framework: A Foundation for Building a Comprehensive Data Lake Architecture. *In*:

GIEBLER, Corinna; GRÖGER, Christoph; HOOS, Eva; SCHWARZ, Holger; MITSCHANG, Bernhard. Leveraging the Data Lake: Current State and Challenges. **International Conference on Big Data Analytics and Knowledge Discovery**, v. 1, p. 179–188, 2019.

GORELIK, Alex. **The Enterprise Big Data Lake: Delivering the Promise of Big Data and Data Science**. [*S.l.*]: O'Reilly Media, 2019. P. 219. ISBN 9781491931554.

HADOOP, Apache. **HDFS Architecture**. Disponível em: `https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html`. (accessed: 06.07.2022).

HADOOP, Apache. **Transparent Encryption in HDFS**. Disponível em: `https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/TransparentEncryption.html`. (accessed: 06.07.2022).

HIVE, Apache. **Apache Hive**. Disponível em: `https://cwiki.apache.org/confluence/display/Hive/Home`. (accessed: 09.07.2022).

HIVE, Apache. **Apache Hive Design**. Disponível em: `https://cwiki.apache.org/confluence/display/Hive/Design`. (accessed: 11.07.2022).

INMON, Bill. **Data Lake Architecture: Designing the Data Lake and Avoiding the Garbage Dump**. 1st. Denville, NJ, USA: Technics Publications, LLC, 2016. ISBN 1634621174.

JOAQUIM, João Luiz Monteiro; SANTOS MELLO, Ronaldo dos. An Analysis of Confidentiality Issues in Data Lakes. *In*: PROCEEDINGS of the 22nd International Conference on Information Integration and Web-Based Applications & Services. New York, NY, USA: Association for Computing Machinery, 2020. P. 168–177. ISBN 9781450389228.

KITCHENHAM, Barbara; CHARTERS, Stuart. **Guidelines for Performing Systematic Literature Reviews in Software Engineering**. [*S.l.*], 2007.

LANEY, Doug. 3-D Data Management: Controlling Data Volume, Velocity, and Variety. **META Group Res Note 6**, v. 6, jan. 2001.

MADERA, Cedrine; LAURENT, Anne. The next information architecture evolution: The data lake wave. **8th International Conference on Management of Digital EcoSystems, MEDES 2016**, p. 174–180, 2016.

NIST, Big Data Public Working Group. **NIST Big Data Interoperability Framework: volume 1, definitions, version 2**. [*S.l.*], jun. 2018. Disponível em: https://doi.org/10.6028/nist.sp.1500-1r1.

NUNES, Raphaela P. **Processamento Analítico Distribuído de Grandes Volumes e Dados Multidimencionais: Uma abordagem baseada em grafos RDF**. 2016. Tese (Doutorado) – Rio de Janeiro, Brasil.

PANWAR, Arvind; BHATNAGAR, Vishal. Data Lake Architecture: A New Repository for Data Engineer. en. **International Journal of Organizational and Collective Intelligence (IJOCI)**, v. 10, n. 1, p. 63–75, 2020. ISSN 1947-9344.

QUIX, Christoph; HAI, Rihan. Data Lake. **Encyclopedia of Big Data Technologies**, p. 552–559, 2018.

RANGER, Apache. **Apache Ranger**. Disponível em: https://ranger.apache.org. (accessed: 09.07.2022).

RANGER, Apache. **Configuring Apache HDFS Encryption**. Disponível em: https://docs.cloudera.com/HDPDocuments/HDP3/HDP-3.1.4/configuring-hdfs-encryption/content/ranger_kms_administration.html. (accessed: 09.07.2022).

RAO, P. Ram Mohan.; KRISHNA, S. Murali; KUMAR, A. P. Siva. Privacy preservation techniques in big data analytics: a survey. **Journal of Big Data**, v. 5, n. 1, p. 33, set. 2018. ISSN 2196-1115.

RAVAT, Franck; ZHAO, Yan. Data Lakes: Trends and Perspectives. *In*: INTERNATIONAL Conference on Database and Expert Systems Applications. [*S.l.*: *s.n.*], 2019. P. 304–313.

SAÚDE, Ministério da. **CAPÍTULO 4 - Cidadão**. Disponível em: `https://cgiap-saps.github.io/Manual-eSUS-APS/docs/PEC/PEC_04_cidadao/`. (accessed: 08.08.2022).

SAÚDE, Ministério da. **CAPÍTULO 6 - Atendimentos**. Disponível em: `https://cgiap-saps.github.io/Manual-eSUS-APS/docs/PEC/PEC_06_atendimentos/`. (accessed: 08.08.2022).

SAÚDE, Ministério da. **CAPÍTULO INTRODUTÓRIO - Base Conceitual do Sistema**. Disponível em: `https://cgiap-saps.github.io/Manual-eSUS-APS/docs/PEC/PEC_00_base_conceitual/`. (accessed: 08.08.2022).

SAÚDE, Ministério da. **Quem é Quem**. Disponível em: `https://www.gov.br/saude/pt-br/composicao/quem-e-quem`. (accessed: 08.08.2022).

SAÚDE, Ministério da. **Sistema Único de Saúde (SUS): estrutura, princípios e como funciona**. Disponível em: `https://www.gov.br/saude/pt-br/assuntos/saude-de-a-a-z/s/sus-estrutura-principios-e-como-funciona`. (accessed: 08.08.2022).

SOLOVE, Daniel J. **Understanding privacy**. [*S.l.*]: Harvard University Press, 2010. ISBN 0674035070.

SOMMERVILLE, Ian. **Software Engineering**. 10. ed. Harlow, England: Addison-Wesley, 2015. ISBN 978-1-292-09613-1.

SQOOP, Apache. **Sqoop User Guide**. Disponível em: `https://sqoop.apache.org/docs/1.4.7/SqoopUserGuide.html`. (accessed: 06.07.2022).

TORRA, Vicenç. **Data Privacy: Foundations, New Developments and the Big Data Challenge**. [*S.l.*]: Springer International Publishing, jan. 2017. v. 28.

WHITE, Tom. **Hadoop: The Definitive Guide**. 4. ed. Beijing: O'Reilly, 2015. ISBN 978-1-4919-0163-2.

Anexos

ANEXO A – *SCRIPT* DE GERAÇÃO DADOS PARA O BD DO PEC

```
# Versao do python utilizada: 3.10.5
# Libs utilizadas: Faker==14.1.0, psycopg[binary]==3.0.16
import random
from faker import Faker
import psycopg

def persist_data(data, table, cur, conn):
    columns, values = [],[]
    for property in data:
        columns.append(property)
        values.append(str(data[property]) if isinstance(data[property], int) else "'"+str(data[property])+"'")
    cur.execute("INSERT INTO %s ( %s ) VALUES ( %s )" % (table, ', '.join(columns), ', '.join(values)))
    conn.commit()

with psycopg.connect("dbname=esus user=postgres password=esus host=localhost port=5433") as conn:
    fake, qtd = Faker('pt_BR'), 100
    with conn.cursor() as cur:
            for index in range(qtd):

                bairro = fake.bairro().replace("'", "")
                no_cidadao = fake.name()
                no_mae = fake.name_female()
                dt_nasc = fake.date_of_birth(minimum_age=18, maximum_age=30).strftime('%Y-%m-%d')
                dt_nasc_resp = fake.date_of_birth(minimum_age=30, maximum_age=65).strftime('%Y-%m-%d')
                dt_nasc_cuid = fake.date_of_birth(minimum_age=30, maximum_age=65).strftime('%Y-%m-%d')
                nu_cpf = fake.cpf().replace(".","").replace("-","")
                nu_cpf_cuidador = fake.cpf().replace(".","").replace("-","")
                nu_cpf_responsavel = fake.cpf().replace(".","").replace("-","")
                ds_cep = fake.postcode().replace("-","")
                ds_logradouro = fake.street_address().replace("'", "")

                data = {
                    "co_seq_cidadao": index,
                    "st_desconhece_nome_mae": 0,
                    "co_localidade": random.randint(19, 10889),
                    "no_responsavel": fake.name(),
                    "dt_nascimento_responsavel": dt_nasc_resp,
```

```
38                      "no_cuidador": fake.name(),
39                      "dt_nascimento_cuidador": dt_nasc_cuid,
40                      "tp_cds_cuidador": random.randint(1, 8),
41                      "co_unico_cidadao": index,
42                      "co_nacionalidade": random.randint(1, 3),
43                      "co_pais_nascimento": random.randint(1, 243),
44                      "tp_orientacao_sexual": random.choice(["Homossexual"
    , "Bissexual", "Heterosexual"]),
45                      "tp_identidade_genero": random.choice(["Homem", "
    Mulher"]),
46                      "st_ativo": 1,
47                      "nu_cpf": nu_cpf,
48                      "no_cidadao": no_cidadao,
49                      "no_cidadao_filtro": no_cidadao,
50                      "co_escolaridade": random.randint(1, 15),
51                      "co_raca_cor": random.randint(1, 6),
52                      "co_etnia": random.randint(1, 405),
53                      "co_estado_civil": random.randint(1, 5),
54                      "co_cbo": random.randint(1, 2612),
55                      "dt_nascimento": dt_nasc,
56                      "no_mae": no_mae,
57                      "no_mae_filtro": no_mae,
58                      "no_pai": fake.name_male(),
59                      "ds_cep": ds_cep,
60                      "ds_logradouro": ds_logradouro,
61                      "co_uf": random.randint(1, 27),
62                      "co_localidade_endereco": random.randint(19, 10889),
63                      "no_bairro": bairro,
64                      "no_bairro_filtro": bairro,
65                      "tp_logradouro": random.randint(1, 313),
66                      "nu_telefone_celular": fake.cellphone_number(),
67                      "ds_email": fake.ascii_free_email(),
68                      "nu_cpf_cuidador": nu_cpf_cuidador,
69                      "nu_cpf_responsavel": nu_cpf_responsavel,
70                      "no_tipo_sanguineo": random.choice(["O-", "O+", "A+"
    , "A-", "AB+", "AB-", "B+", "B-"]),
71                      "no_sexo": random.choice(["Masculino", "Feminino"]),
72                  }
73                  persist_data(data, "tb_cidadao", cur, conn)
74
75                  data = {
76                      "co_seq_cds_prof": index,
77                      "co_unico_cds_prof": index,
78                  }
79                  persist_data(data, "tb_cds_prof", cur, conn)
80
81                  data = {
```

```
82                      "co_seq_cds_ficha_atend_indivdl": index,
83                      "st_ficha": 0,
84                      "tp_cds_origem":  random.randint(0, 6),
85                      "co_unico_ficha":  index,
86                      "co_cds_prof":    index,
87                      "co_localidade_origem": random.randint(19, 10889)
88                  }
89                  persist_data(data, "tb_cds_ficha_atend_individual", cur,
     conn)
90
91                  dt_nasc = fake.date_of_birth(minimum_age=18, maximum_age
    =30).strftime('%Y-%m-%d'),
92                  nu_cpf_cidadao = fake.cpf().replace(".","").replace("-",
    "")
93
94                  data = {
95                      "co_seq_cds_atend_individual": index,
96                      "co_cds_ficha_atend_individual": index,
97                      "dt_nascimento": dt_nasc,
98                      "co_local_atend": random.randint(1, 15),
99                      "co_cds_aleitamento_materno": random.randint(1, 4),
100                     "co_cds_pic": random.randint(1, 8),
101                     "co_cid10": random.randint(1, 14236),
102                     "co_ad_modalidade": random.randint(1, 5),
103                     "co_sexo": random.randint(0, 4),
104                     "tp_atend": random.randint(1, 9),
105                     "co_cds_turno": random.randint(1, 3),
106                     "st_gravidez_planejada":0,
107                     "qt_gestacao_previa": 0,
108                     "qt_parto": 0,
109                     "co_racionalidade_saude": random.randint(1, 6),
110                     "co_cid10_2": random.randint(1, 14236),
111                     "nu_cpf_cidadao": nu_cpf_cidadao,
112                 }
113                 persist_data(data, "tb_cds_atend_individual", cur, conn)
```

ANEXO B – SCRIPT IMPORTAÇÃO DADOS BD DO PEC PARA O DL

```
1
2 personal_data_dir="/user/hive/healthcare_personal/"
3 sensitive_data_dir="/user/hive/healthcare_sensitive/"
4 hive_database_healthcare_public="healthcare_public"
5
6 # Importa todas as tabelas auxiliares da tb_cidadao e cria elas no Hive
7 tabelas_auxiliares=("tb_cds_tipo_cuidador" "tb_nacionalidade" "
    tb_escolaridade" "tb_raca_cor" "tb_localidade" "tb_etnia" "
    tb_estado_civil" "tb_cbo" "tb_pais" "tb_uf" "tb_tipo_logradouro")
8
9 for EXPORT_TABLE in "${tabelas_auxiliares[@]}"
10 do
11     sqoop import --connect jdbc:postgresql://$IP_ADDRES/$DATABASE --
    username $USERNAME --password $PASSWORD --hive-import --hive-table="
    $hive_database_healthcare_public.$EXPORT_TABLE" --hive-overwrite --
    table="$EXPORT_TABLE" -m 1 --delete-target-dir
12 done
13
14 # Importa a tabela tb_cidadao
15 EXPORT_TABLE="tb_cidadao"
16 sqoop import --connect jdbc:postgresql://$IP_ADDRES/$DATABASE --username
     $USERNAME --password $PASSWORD --table="$EXPORT_TABLE" -m 1 --target
    -dir $personal_data_dir --delete-target-dir --as-parquetfile --map-
    column-java dt_atualizado=String,dt_nascimento_responsavel=String,
    dt_nascimento_cuidador=String,dt_ultima_ficha=String,
    dt_atualizado_cadsus=String,dt_naturalizacao=String,dt_entrada_brasil
    =String,dt_nascimento=String,dt_obito=String
17
18 # importacao todas as tabelas auxiliares da tb_cds_atend_individual e
    cria elas no Hive
19 tabelas_auxiliares=("tb_prontuario" "tb_local_atend" "
    tb_cds_aleitamento_materno" "tb_cds_pic" "tb_ad_modalidade" "tb_sexo"
     "tb_tipo_atend" "tb_cid10" "tb_cds_turno" "tb_racionalidade_saude" "
    tb_cds_prof" "tb_cds_ficha_atend_individual")
20
21 for EXPORT_TABLE in "${tabelas_auxiliares[@]}"
22 do
23     sqoop import --connect jdbc:postgresql://$IP_ADDRES/$DATABASE --
    username $USERNAME --password $PASSWORD --hive-import --hive-table="
    $hive_database_healthcare_public.$EXPORT_TABLE" --hive-overwrite --
    table="$EXPORT_TABLE" -m 1 --delete-target-dir
24 done
25
26 # Importa a tabela tb_cds_atend_individual
27 EXPORT_TABLE="tb_cds_atend_individual"
```

```
28 sqoop import --connect jdbc:postgresql://$IP_ADDRES/$DATABASE --username
     $USERNAME --password $PASSWORD --table="$EXPORT_TABLE" -m 1 --target
    -dir $sensitive_data_dir --delete-target-dir --as-parquetfile --map-
    column-java dt_nascimento=String,dt_ultima_menstruacao=String
```

ANEXO C – SCRIPT DE CRIPTOGRAFIA DOS DADOS PESSOAIS E SENSÍVEIS NO HDFS

```
#!/bin/bash

personal_data_dir="/user/hive/healthcare_personal/"
sensitive_data_dir="/user/hive/healthcare_sensitive/"

# Cria os diret rios
mkdir personal_data_dir
mkdir sensitive_data_dir

# Cria as chaves de encriptacao
hadoop key create chave_criptografica_dados_pessoais
hadoop key create chave_criptografica_dados_sensiveis

# Antes de inserir os dados nos diretorios e necessario vincular a chave
    as pasta, ou tambem conhecido como zona
hadoop fs -mkdir personal_data_dir
hdfs crypto -createZone -keyName chave_criptografica_dados_pessoais -
    path personal_data_dir
hadoop fs -mkdir sensitive_data_dir
hdfs crypto -createZone -keyName chave_criptografica_dados_sensiveis -
    path sensitive_data_dir
```

ANEXO D – CRIAÇÃO DAS TABELAS PESSOAIS E SENSÍVEIS NO HIVE

```
CREATE DATABASE IF NOT EXISTS healthcare_personal;

CREATE TABLE IF NOT EXISTS healthcare_personal.tb_cidadao
(
    co_seq_cidadao bigint,
    st_desconhece_nome_mae bigint,
    co_localidade bigint,
    nu_area string,
    nu_micro_area string,
    nu_nis_pis_pasep string,
    dt_atualizado string,
    nu_cns_responsavel string,
    no_responsavel string,
    dt_nascimento_responsavel string,
    nu_cns_cuidador string,
    no_cuidador string,
    dt_nascimento_cuidador string,
    tp_cds_cuidador bigint,
    co_unico_cidadao string,
    co_nacionalidade bigint,
    co_pais_nascimento bigint,
    co_unico_ultima_ficha string,
    dt_ultima_ficha string,
    st_registro_cadsus bigint,
    dt_atualizado_cadsus string,
    st_desconhece_nome_pai bigint,
    dt_naturalizacao string,
    dt_entrada_brasil string,
    nu_portaria_naturalizacao string,
    st_fora_area bigint,
    st_infrm_orientacao_sexual bigint,
    tp_orientacao_sexual string,
    st_infrm_identidade_genero bigint,
    tp_identidade_genero string,
    st_compartilhamento_prontuario bigint,
    st_ativo bigint,
    st_nao_possui_cuidador bigint,
    nu_cpf string,
    nu_cns string,
    no_cidadao string,
    no_cidadao_filtro string,
    co_escolaridade bigint,
    co_raca_cor bigint,
    co_etnia bigint,
    co_estado_civil bigint,
```

```
    co_cbo bigint,
    dt_nascimento string,
    dt_obito string,
    no_mae string,
    no_mae_filtro string,
    no_pai string,
    no_social string,
    st_faleceu bigint,
    nu_documento_obito string,
    st_dados_obito_cadsus bigint,
    no_localidade_exterior string,
    co_pais_exterior bigint,
    ds_cep string,
    ds_complemento string,
    ds_ponto_referencia string,
    ds_logradouro string,
    co_uf bigint,
    co_localidade_endereco bigint,
    nu_numero string,
    st_sem_numero bigint,
    no_bairro string,
    no_bairro_filtro string,
    tp_logradouro bigint,
    nu_telefone_residencial string,
    nu_telefone_celular string,
    nu_telefone_contato string,
    ds_email string,
    st_ativo_para_exibicao bigint,
    st_unificado bigint,
    st_territorio_utiliza_cpf bigint,
    nu_cpf_cuidador string,
    nu_cpf_responsavel string,
    no_tipo_sanguineo string,
    no_sexo string
)
STORED AS PARQUET
LOCATION '/user/hive/healthcare_personal/'
TBLPROPERTIES('external'='true');
```

```
CREATE DATABASE IF NOT EXISTS healthcare_sensitive;

CREATE TABLE IF NOT EXISTS healthcare_sensitive.tb_cds_atend_individual
(
    co_seq_cds_atend_individual bigint,
    co_cds_ficha_atend_individual bigint,
    co_prontuario bigint,
    nu_cartao_sus string,
    nu_prontuario string,
```

```
10      dt_nascimento string,
11      co_local_atend bigint,
12      co_cds_aleitamento_materno bigint,
13      dt_ultima_menstruacao string,
14      co_cds_pic bigint,
15      co_cid10 bigint,
16      st_ficou_observacao bigint,
17      st_vacina_em_dia bigint,
18      nu_idade_gestacional bigint,
19      co_ad_modalidade bigint,
20      nu_peso double,
21      nu_altura double,
22      co_sexo bigint,
23      tp_atend bigint,
24      co_cds_turno bigint,
25      st_gravidez_planejada bigint,
26      qt_gestacao_previa bigint,
27      qt_parto bigint,
28      co_racionalidade_saude bigint,
29      nu_perimetro_cefalico double,
30      co_cid10_2 bigint,
31      nu_cpf_cidadao string
32  )
33  STORED AS PARQUET
34  LOCATION '/user/hive/healthcare_sensitive/'
35  TBLPROPERTIES('external'='true');
```