Pós-Graduação em Ciência da Computação

Wilker Cavalcante do Rego Santos

IAGO: Um Sistema Gerenciador de Dados na Web

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

Recife
2019

Wilker Cavalcante do Rego Santos

IAGO: Um Sistema Gerenciador de Dados na Web

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Banco de Dados
Orientador: Profa. Dra. Bernadette Farias Lóscio

Recife
2019

Catalogação na fonte
Bibliotecária Mariana de Souza Alves CRB4-2105

S237i Santos, Wilker Cavalcante do Rego
IAGO: um Sistema Gerenciador de Dados na Web/ Wilker Cavalcante do Rego Santos – 2019.
100 f., fig.

Orientadora: Bernadette Farias Lóscio.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação. Recife, 2019.
Inclui referências e apêndice.

1. Banco de Dados. 2. Publicação de Dados. 3. Boas Práticas. 4. Dados na Web. I. Lóscio, Bernadette Farias (orientadora). II. Título.

025.04 CDD (22. ed.) UFPE-MEI 2019-169

Wilker Cavalcante do Rego Santos

“Iago: Um Sistema Gerenciador de Dados na Web”

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Aprovado em: 27 de agosto de 2019.

BANCA EXAMINADORA

_____

Prof. Dr. Alex Sandro Gomes
Centro de Informática/UFPE

_____

Prof. Dr. Carlos Denner dos Santos Júnior
Departamento de Administração / UnB

_____

Profa. Dra. Bernadette Farias Lóscio
Centro de Informática/UFPE
(**Orientadora)**

Dedico esta dissertação à minha noiva Beatriz, à minha mãe Suzana, à minha avó Valda, a meus familiares, amigos e professores que de alguma forma contribuíram para a sua realização.

"Todos que choram a queda de Jerusalém
têm o privilégio de ver sua alegria"
(TALMUD)

RESUMO

A Web tornou-se uma importante ferramenta de compartilhamento e consumo de dados e informações. Apesar de a ideia de compartilhamento de dados na Web não ser nova, inúmeros desafios ainda podem ser encontrados e estes podem dificultar a comunicação entre quem compartilha os dados e quem os consome, como o gerenciamento dos metadados, atualização dos conjuntos de dados e versionamento dos mesmos. Neste contexto, este trabalho propõe o IAGO, um Sistema Gerenciador de Dados na Web que, além de seguir a maioria das boas práticas para dados na Web propostas pelo W3C, tem a finalidade de automatizar os processos envolvidos. O IAGO se destaca pelos seguintes diferenciais: i) Extração de dados de diversos tipos de fontes de dados; ii) Geração automática de descrição dos Conjuntos de Dados através de metadados, legível por seres humanos e máquinas; iii) Atualização automática dos conjuntos de dados de acordo com frequências de atualização pré-definidas; iv) Gerenciamento de versões dos conjuntos de dados; v) Alta adaptabilidade do sistema;

Palavras-chaves: Publicação de Dados. Boas Práticas. Dados na Web. SGDW. Serviços. Versionamento.

ABSTRACT

The Web has become an important tool for sharing and consuming data and information. Although the idea of data sharing on the Web is not new, numerous challenges can still be encountered and these can make it difficult to communicate between those who share data and those who consume it, such as metadata management, dataset updating, and versioning. . In this context, this paper proposes IAGO, a Data on the Web Management System, which, in addition to following most of the best practices proposed by the W3C, aims to automate the processes involved. IAGO stands out for the following advantages: i) Data extraction from various types of data sources; ii) Automatic generation of datasets description through metadata, readable by humans and machines; iii) Automatic update of datasets according to predefined update frequencies; iv) Dataset version management; v) High adaptability of the system;

Keywords: Data Publishing. Best Practices. Data on the Web. DWMS. Services. Versioning.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo será apresentada a visão geral deste trabalho, bem como o seu contexto. Na Seção 1.1, a motivação para o desenvolvimento desta pesquisa é apresentada, bem como o problema de pesquisa é explanado. Em seguida, os objetivos do trabalho e suas contribuições são mostrados na Seção 1.2. Na Seção 1.3 é apresentada a metodologia de pesquisa utilizada. Por fim, a estrutura do documento é descrita na Seção 1.4.

1.1 MOTIVAÇÃO E CARACTERIZAÇÃO DO PROBLEMA

A Web tornou-se uma importante ferramenta de compartilhamento e consumo de dados e informações. Com sua popularização, a quantidade de dados gerados por seus usuários cresce a cada dia. Em paralelo, novos paradigmas vêm sendo desenvolvidos ao longo dos últimos anos para encontrar novas formas de como os usuários podem fazer uso do que a Web pode oferecer, fazendo dessa uma área de grande importância para o meio científico. A literatura explora diversas áreas relacionadas aos dados na Web tais como recomendação de melhores práticas para dados na Web (LóSCIO; BURLE; CALEGARI, 2017), catalogação de dados (STONEBRAKER et al., 2013), integração de dados (NASSAR, 2018), visualização de dados (CARDNO et al., 2018) e análise de dados (WANG; KUNG; BYRD, 2018).

Apesar de a ideia de compartilhamento de dados na Web não ser nova (BERNERS-LEE; CONNOLLY; SWICK, 1999), inúmeros desafios ainda podem ser encontrados, dificultando a comunicação entre quem compartilha os dados e quem os consome (ZUIDERWIJK et al., 2012). Desta forma, grupos de empresas, universidades e instituições tais como o W3C[1] (World Wide Web Consortium) vem trabalhando constantemente na elaboração de soluções para padronizar a publicação e o consumo de dados. Dentre os frutos dos trabalhos do W3C, encontra-se uma recomendação de 35 boas práticas de dados na Web (Data on the Web Best Practices - DWBP) (LóSCIO; BURLE; CALEGARI, 2017).

Sabe-se que a publicação de dados na Web requer um investimento, seja ele de tempo ou de finanças e que nem sempre se tem um retorno positivo (ZUIDERWIJK et al., 2012). Por isso, atualmente existem inúmeras ferramentas disponíveis para a catalogação e publicação de dados na Web que visam atacar os desafios existentes. Essas soluções buscam cada vez mais facilitar a extração e o compartilhamento de dados, idealizando sempre a automatização dos processos envolvidos. Porém, nem sempre essas ferramentas se caracterizam como ideais, pois deixam lacunas em sua eficiência (OLIVEIRA; OLIVEIRA; LÓSCIO, 2017), ferindo fatores de suma importância para os dados como sua confiabilidade, reúso e compreensão (LóSCIO; BURLE; CALEGARI, 2017). A falta de atualização dos dados de acordo com uma frequência preestabelecida e falta de formatos legíveis por seres humanos

[1] https://www.w3.org/Consortium/

e máquinas são exemplos de problemas muito comuns na maioria das soluções disponíveis. Além disso, as tecnologias são carentes de mecanismos para o gerenciamento de versões dos conjuntos de dados publicados na Web. De acordo com o as boas práticas propostas pelo W3C, é essencial que os conjuntos de dados sejam versionados, a fim de que os usuários que venham a consumi-los possam ter acesso aos inúmeros estados que os dados tiveram ao longo de sua vida. A recomendação do W3C deixa claro porém, que não existe um consenso de quando uma nova versão de um Conjunto de Dados deve ser criada ou que metadados devem ser utilizados para descrevê-la. Com isso, fica clara a necessidade do desenvolvimento de uma abordagem para tratar o versionamento de dados na Web de maneira definitiva.

Neste contexto, esta dissertação é norteada pela seguinte pergunta de pesquisa: *RQ: Como implementar uma solução para o compartilhamento de dados na Web a partir do DWBP, a fim de promover o reúso dos dados?*. Desta maneira, este trabalho tem o objetivo de propor o IAGO, uma ferramenta de gerenciamento de dados na Web que, além de procurar seguir a maioria das recomendações propostas pelo W3C, busca preencher as lacunas em aberto por parte de outras soluções. Como benefícios resultantes desta pesquisa, é esperado: i) promover o reúso dos dados; ii) contribuir para a agregação de valor aos dados; iii) garantir uma maior confiabilidade aos dados. Somado a isso, esse trabalho apresenta uma abordagem para versionamento de conjuntos de dados na Web, em que todos os estados dos dados ao longo de sua vida podem ser recuperados de forma integral. Com isso, busca-se guiar a comunidade científica a uma discussão mais madura sobre como deve ser feito o versionamento de dados na Web e quais de dados e metadados devem ser armazenados de maneira a representar as versões para seus usuários da melhor forma possível.

1.2 OBJETIVOS E CONTRIBUIÇÕES

O principal objetivo deste trabalho consiste no desenvolvimento de um sistema de gerenciamento de dados na Web que permite o versionamento automático dos conjuntos de dados e segue as boas práticas propostas pelo W3C. Para isso, os seguintes objetivos específicos foram seguidos:

- Realização de um levantamento das principais tecnologias de publicação e catalogação de dados na Web, a fim de identificar os principais desafios para o compartilhamento de dados na Web;
- Identificar os requisitos necessários para o desenvolvimento de uma solução adequada para o gerenciamento de dados publicados na Web;
- Especificar uma abordagem para versionamento de conjuntos de dados publicados na Web;

- Especificação e implementação do IAGO, o Sistema de Gerenciamento de Dados na Web, proposto neste trabalho.

1.3 METODOLOGIA

Uma metodologia de pesquisa pode ser definida como um conjunto de princípios organizacionais em torno dos quais dados empíricos são coletados e analisados. A metodologia deve estar diretamente ligada ao problema que será estudado, sendo rigorosamente analisada antes mesmo de sua execução (MARCONI; LAKATOS, 2017).

A construção desta dissertação foi baseada em um paradigma filosófico pragmático. A postura filosófica é caracterizada pela aceitação de diversos conceitos existentes de maneira a apoiar a pesquisa, buscando utilizar todos os meios disponíveis para compreender o problema em questão (CRESWELL, 2010). O pragmatismo valoriza o conhecimento prático sobre o conhecimento abstrato e faz uso dos métodos apropriados disponíveis com o objetivo de obtê-lo (EASTERBROOK et al., 2008). Desta forma, o método filosófico pragmático é utilizado por pesquisadores que acreditam que a compreensão do problema é mais importante e que utilizam os meios disponíveis para construir uma solução.

Assim, esta pesquisa foi desenvolvida a partir das etapas descritas abaixo:

- **Revisão *ad-hoc* da Literatura**: Uma revisão *ad-hoc* foi feita com o objetivo de encontrar trabalhos relacionados que pudessem fornecer um embasamento teórico. Revisões *ad-hoc* caracterizam-se por ser uma busca informal da literatura, visando encontrar trabalhos de uma temática específica;
- **Definição de uma abordagem para versionamento de conjuntos de dados na Web**: Baseando-se nos trabalhos encontrados na etapa anterior, foi definida uma abordagem para versionamento de conjuntos de dados na Web;
- **Definição do IAGO**: Com a fundamentação teórica encontrada na fase de revisão e levando em consideração a abordagem de versionamento para conjuntos de dados na Web proposta, o IAGO foi definido;
- **Implementação e experimentos**: Por fim, o IAGO foi implementado e experimentos foram realizados com o objetivo de valida-lo.

1.4 ORGANIZAÇÃO DO TRABALHO

Neste capítulo, a introdução ao trabalho foi mostrada. A partir daqui, este texto está organizado da seguinte maneira: no Capítulo 2, a fundamentação teórica para a compreensão do trabalho é mostrada. Em seguida, no Capítulo 3, as principais tecnologias para publicação de dados na Web são descritas e analisadas levando em consideração as boas práticas de dados na Web. No Capítulo 4, é apresentada a proposta de versionamento

para conjuntos de dados na Web. No Capítulo 5, o IAGO é apresentado completamente. No Capítulo 6, são dados os detalhes de implementação do IAGO, juntamente com a validação do sistema. Por fim, a conclusão está disponível no Capítulo 7.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, serão apresentadas as definições necessárias para a compreensão total desta dissertação. Na Seção 2.1, os conceitos sobre Dados na Web serão apresentados, incluindo a explanação sobre as boas práticas de dados na Web e seu ciclo de vida. Em seguida, os conceitos básicos sobre versionamento são mostrados na Seção 2.2. Por fim, as considerações finais na Seção 2.4.

2.1 DADOS NA WEB

A Web vem crescendo nos últimos anos. Cada vez mais usuários vêm utilizando seus recursos e consequentemente a quantidade de dados gerados vem crescendo de maneira significativa. Devido a isso, inúmeras novas aplicações que consumem o conteúdo disponível na Web são criadas todos os dias e a sociedade é beneficiada de várias maneiras com isso. Desta forma, Dados na Web podem ser definidos como o termo geral para dados publicados de acordo com os princípios da arquitetura Web[1] (LóSCIO; BURLE; CALEGARI, 2017). Neste cenário, vários esforços têm sido desenvolvidos para facilitar a publicação e o consumo de Dados na Web. Entre as abordagens mais populares, podem ser encontrados dados em formato aberto (KUCERA et al., 2015) e dados conectados (BIZER; HEATH; BERNERS-LEE, 2011).

Os recursos da Web podem ser identificados de forma exclusiva através do Uniform Resource Identifier (URI), pelo qual são representados em distribuições padronizadas como JSON [2], HTML [3], CSV [4] ou RDF [5] (JACOBS et al., 2004). Dessa forma, na Web é possível a troca de informações em formatos padronizados (dados e metadados), permitindo que a mesma seja facilmente compreendida, tanto por seres humanos quanto por máquinas.

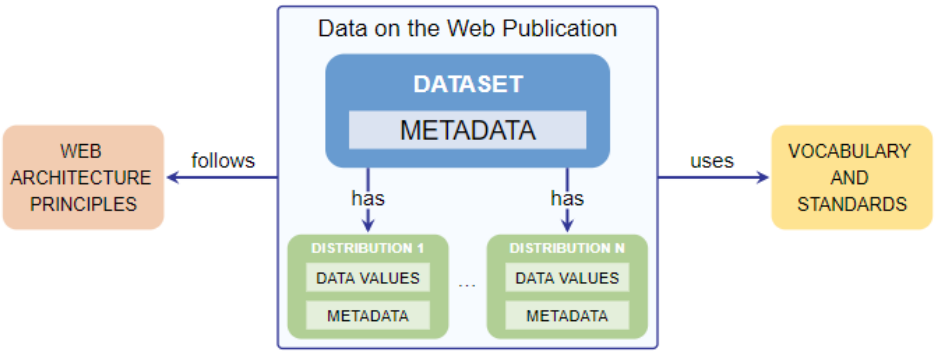
Nesse contexto, a publicação e consumo de dados podem ser executadas por um conjunto de atores. Estes que podem assumir dois papéis diferentes: o de publicador e consumidor de dados, sendo, respectivamente, o primeiro responsável por disponibilizar os dados na Web e o segundo realiza o uso dos dados (LóSCIO; OLIVEIRA; BITTENCOURT, 2015).

Em geral, dados na Web se referem a conteúdos específicos chamados de Conjunto de Dados, podendo estes ser criados e publicados por um único ator e em diversos formatos (CONSORTIUM et al., 2014). Somado a isso, os dados devem estar em um formato legível para máquinas, de forma a diminuir as barreiras tecnológicas a seu consumidor e, con-

[1] https://www.w3.org/TR/webarch/
[2] https://www.json.org/
[3] https://techterms.com/definition/html
[4] https://www.techopedia.com/definition/24364/comma-separated-values-file-csv
[5] https://www.w3.org/RDF/

sequentemente, permitir o processamento automático dos mesmos. A Figura 1 ilustra o contexto considerado neste trabalho.

Figura 1 – Arquitetura de Dados na Web.



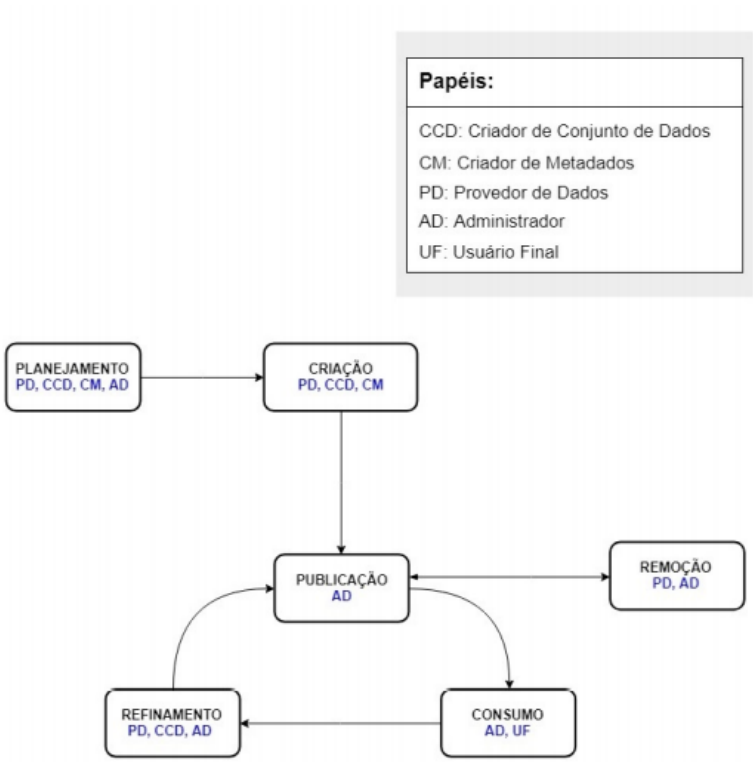
Fonte: (LóSCIO; BURLE; CALEGARI, 2017).

2.1.1 Ciclo de vida de Dados na Web

No contexto de dados na Web, existem atividades que compõem o processo de publicação e consumo de conjuntos de dados, que abrangem desde o planejamento até a sua remoção. Baseando-se no *Abstract Data Lifecycle Model* (LóSCIO; OLIVEIRA; BITTENCOURT, 2015), o Modelo Abstrato de Ciclo de Vida de Dados na Web (*Data on the Web LifeCycle Model - DWLM*) foi proposto (SILVA, 2019). Levando em consideração o DWBP (LóSCIO; BURLE; CALEGARI, 2017), o DWLM tem o objetivo de prover um entendimento comum para as etapas de um conjunto de dados ao longo de sua vida, garantindo também requisitos para que os dados possam ser lidos por humanos e máquinas. Dividido em seis fases, o modelo descreve cuidadosamente todos os papéis, atividades e entradas e saídas de cada fase. Assim como demonstrado na Figura 2, as fases narradas pelo modelo são:

- **Planejamento** - Nesta fase, são coletadas as informações descritivas do conjunto de dados e a solução para a sua publicação é escolhida. As fontes de onde os dados são extraídos também é especificada.
- **Criação** - Aqui, o conjunto de dados é criado e toda a avaliação de qualidade é feita, fazendo uso do *Data Quality Assessment - QA*. (UMBRICH; NEUMAIER; POLLERES, 2015).
- **Publicação** - A disponibilização do Conjunto de Dados na Web o deixado totalmente acessível a seus consumidores está nesta fase.
- **Consumo** - Nesta fase, o conjunto de dados é acessado pelos usuários, o uso dos mesmos é realizado e um feedback é retornado.

Figura 2 – Ciclo de vida de Dados na Web.



Fonte: (SILVA, 2019).

- **Refinamento** - Nesta fase, os conjuntos de dados são corrigidos e enriquecidos. Com alterações realizadas, são finalmente versionados.
- **Remoção** - Aqui, o acesso ao conjunto de dados é removido.

Além disso, o DWLM define um conjunto de papéis para atores que interagem com os dados ao longo do tempo como também pode ser observado na Figura 2. É importante salientar que um ator, dependendo no contexto, pode desempenhar mais de um papel. Os papéis descritos são:

- **Provedor de Dados** - O papel do provedor é dado ao ator que fornecerá os dados, ou seja, o dono dos dados que posteriormente serão publicados. Além disso, é responsabilidade do provedor participar de todas a validações necessárias do conjunto de dados.
- **Criador de conjuntos de dados** - Aqui, encontra-se o papel responsável para elencar e descrever as informações necessárias sobre o conjunto de dados que será criado, bem como participar de todas as atividades relacionadas à criação do mesmo. Em algumas situações, o provedor de dados também poderá assumir o papel de criador.

- **Criador de Metadados** - Assim como o criador de conjunto de dados, o criador de metadados tem o objetivo de elencar e descrever, no caso, porém, os metadados disponibilizados juntamente com os conjuntos de dados. De acordo com o DWLM, nem sempre o clico de vida irá adotar um ator exclusivo para este papel, ficando ,assim, a responsabilidade da criação dos metadados com o ator da criação dos conjuntos de dados.

- **Administrador** - O administrador tem o objetivo de realizar a publicação do conjunto de dados e acompanhar de perto o conjunto de dados em todas as fases seguintes até, finalmente, a sua remoção, garantindo que tudo corra como o planejado. É importante ressaltar que o administrador deve exercer o seu papel sem alterar o formato e significado dos dados e metadados.

- **Usuário Final** - O usuário final representa todos os que irão fazer o consumo dos dados e metadados dos conjuntos de dados. Neste papel, é possível a criação de aplicações baseadas nos dados consumidos com o intuito de oferecer serviços e produtos. Além disso, o usuário final pode enviar um *feedback* a respeito do conjunto de dados a fim de o tornar melhor no futuro.

É importante salientar que, na fase do refinamento, o DWLM indica a importância do versionamento do conjunto de dados, sendo necessário oferecer um identificador de versão juntamente com um histórico de versões. Na próxima seção, serão explanados os conceitos básicos sobre versionamento para um melhor entendimento deste trabalho.

2.1.2 Boas práticas de Dados na Web

A Web vem crescendo constantemente de forma não padronizada e, como consequência disso, seus usuários muitas das vezes são desfrutam de seu total potencial (NATH; DHAR; BASISHTHA, 2014). Todos os dias, devido à flexibilidade da Web, inúmeros desafios são encontrados pelos publicadores e consumidores de dados, tal como representar, descrever e disponibilizar os dados de forma à fácil compreensão. Nesse contexto, visando guiar os publicadores a gerenciar seus dados e estimular a contínua expansão da Web, um guia de recomendação de melhores práticas na Web (LóSCIO; BURLE; CALEGARI, 2017) foi desenvolvido pelo W3C[6] (World Wide Web Consortium).

O DWBP (Data on de Web Best Practices) lista um total de 35 boas práticas que ditam padrões para publicação e consumo de dados, tais como distribuições de dados a utilizar, metadados necessários para descrever um Conjunto de Dados e uso de identificadores para os dados. As características listadas em relação aos conjuntos de dados que utilizam a recomendação incluem: fácil compreensão dos dados, processamento por máquinas, descoberta de sua existência por parte dos consumidores, possibilidade de reúso,

[6] https://www.w3.org/Consortium/

confiabilidade, fácil acesso, uso de dados conectados e interoperabilidade. A lista com todas as boas práticas pode ser visualizado no Quadro 1.

Como pode ser observado, de acordo com a boa prática de número 7, o DWBP sugere que os conjuntos de dados sejam versionados, que tais versões sejam identificadas unicamente e que os identificadores sejam padronizados com um sistema de numeração consistente de forma que consumidor de dados possa reconhecer quando há uma nova versão existente. Além disso, os dados, caso disponíveis através de API, devem ter URIs únicas para cada versão, de maneira a disponibilizar todas as versões anteriores para o acesso dos usuários. Somando a isso, a recomendação de número 8 prevê que os conjuntos de dados contenham um histórico de versões. Segundo o guia, o histórico é necessário para a compreensão das alterações realizadas ao longo do tempo, por isso, é necessário prover para o consumidor de dados um meio para acessar uma lista com todas as alterações realizadas até o momento.

No DWBP, porém, também é dito que não há um consenso de quando uma alteração deve gerar um novo conjunto de dados ou uma versão a partir de um conjunto de dados original.

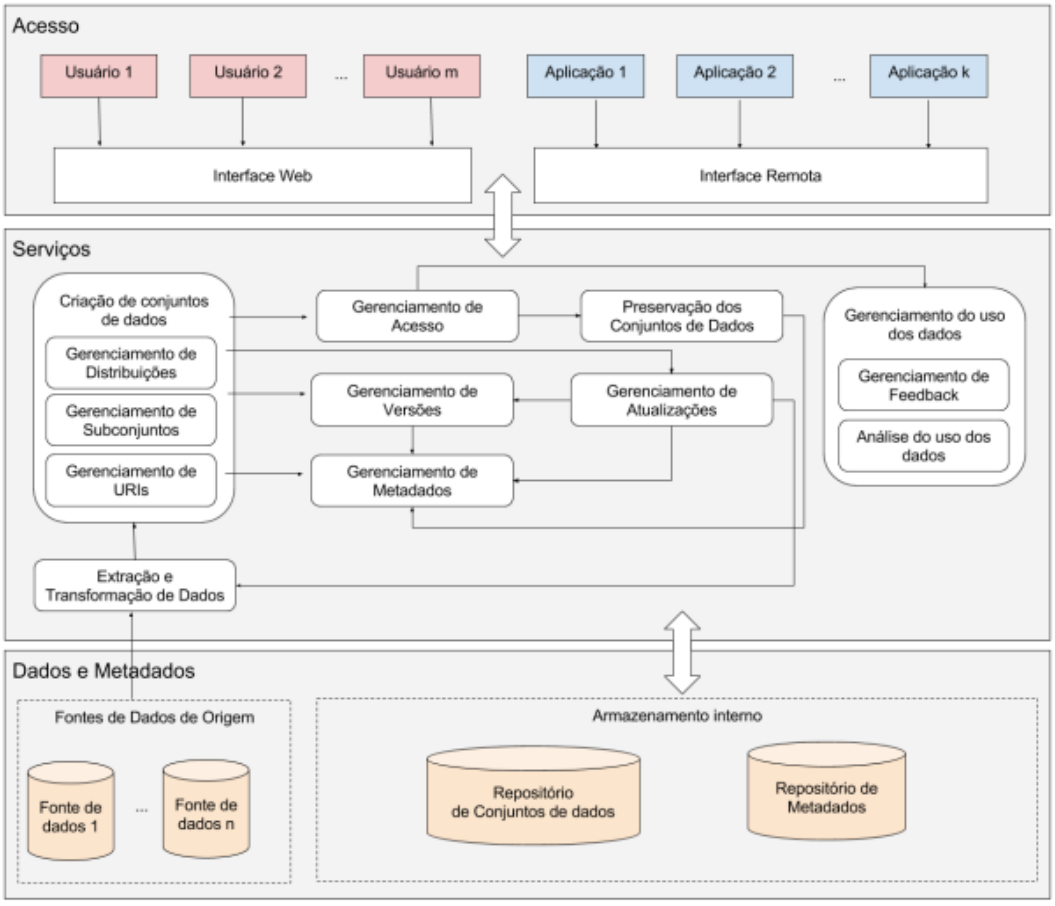
2.1.3 Sistema Gerenciador de Dados na Web

Visando preencher lacunas deixadas por soluções para catalogação e compartilhamento de dados na Web, foi proposto um modelo de arquitetura para um Sistema Gerenciador de Dados na Web (SGDW) (OLIVEIRA, 2017). Um SGDW pode ser caracterizado como um conjunto de serviços para que o usuário possa compartilhar seus dados na Web facilitando a criação, manutenção, manipulação e compartilhamento dos dados de maneira compreensível para seres humanos e máquinas. Levando em consideração o DWBP, um SGDW deve ser construído obedecendo as seguintes requisitos de compartilhamento de dados na Web:

- **R1** - O sistema deve prover mecanismos para a criação de conjuntos de dados autodescritivos;
- **R2** - O sistema deve disponibilizar os dados em mais de um formato;
- **R3** - O sistema deve prover múltiplas formas de acesso aos dados, podendo ser desde download dos dados até consumo de APIs;
- **R4** - Prover mecanismos para criação de canais de comunicação entre os atores do ecossistema de dados na Web;
- **R5** - Realizar a atualização dos dados de acordo com a fonte de origem de maneira automática;

- **R6** - O sistema deve conter mecanismos para extração e transformação dos dados da fonte de origem;
- **R7** - Prover mecanismos para gerenciamento de versões;
- **R8** - Prover mecanismos para gerenciamento de metadados;
- **R9** - Garantir mecanismo para preservação dos conjuntos de dados ao longo do tempo;
- **R10** - Prover identificação única para os Conjuntos de Dados, suas distribuições, suas versões e para cada tupla armazenada nos Conjuntos de Dados.

Figura 3 – Arquitetura para um SGDW.



Fonte: (OLIVEIRA, 2017).

Quadro 1 – Lista das Boas Práticas de Dados na Web.

DWBP	Descrição
1	Disponibilizar metadados
2	Disponibilizar metadados que possibilitem a interpretação da natureza dos dados
3	Disponibilizar metadados que descrevam a estrutura dos dados
4	Disponibilizar o contrato de licença de uso dos dados
5	Disponibilizar informações sobre a proveniência dos dados
6	Disponibilizar informações sobre qualidade dos dados
7	Disponibilizar um indicador de versão ou data de atualização para os dados
8	Disponibilizar um histórico de versões
9	Utilizar URIs persistentes como identificadores dos conjuntos de dados
10	Utilizar URIs persistentes com identificadores dentro dos conjuntos de dados
11	Utilizar URIs individuais para cada versão do Conjunto de Dados
12	Disponibilizar os dados em um formato padronizado e legível por máquinas
13	Utilizar representações neutras de localidade e cultura
14	Disponibilizar os dados em mais de um formato
15	Reutilizar vocabulários para codificar dados e metadados
16	Optar por um nível de semântica formal
17	Prover um serviço de download em massa dos dados
18	Prover um serviço de download de subconjuntos de dados
19	Utilizar negociação de conteúdo para permitir a escolha do formato de download
20	Quando necessário, disponibilizar os dados em tempo real
21	Prover os dados atualizados conforme uma frequência que reflita o mundo real
22	Quando os dados estão indisponíveis, disponibilizar uma explicação
23	Disponibilizar os dados através de uma API
24	Utilizar padrões da Web nas APIs dos dados
25	Disponibilizar a documentação completa para as APIs
26	Evitar alterações nas APIs dos dados
27	Preservar o identificador quando o Conjunto de Dados é arquivado
28	Avaliar a cobertura dos dados antes do arquivamento
29	Prover um meio fácil para que os usuários possam realizar o *feedback*
30	Tornar o *feedback* publicamente disponível
31	Enriquecer os dados
32	Disponibilizar visualizações complementares para os dados
33	Prover o *feedback* para o produtor dos dados
34	Utilizar os requisitos de licença original dos dados quando republicados
35	Disponibilizar publicação original em seus metadados

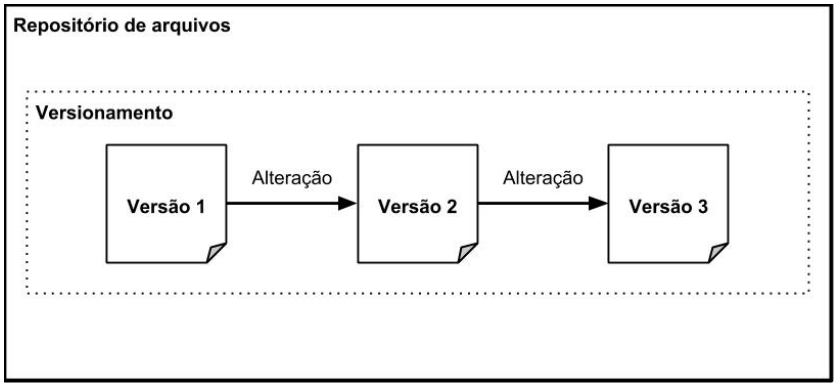
Fonte: (LóSCIO; BURLE; CALEGARI, 2017).

Como pode ser observado na arquitetura ilustrada na Figura 3, o SGDW foi proposto utilizando um modelo de arquitetura em camadas, sendo elas a de Acesso, Serviços e de Dados e Metadados. Primeiramente, a camada de Acesso provê uma interface Web e Remota para serem acessadas por usuários. As interfaces providas servem como pontos de acesso dos usuários para os serviços fornecidos pelo SGDW nas camadas inferiores. Em seguida, a camada de Serviços é responsável por toda a manipulação dos dados e implementação de todas as regras de negócio do SGDW. Nesta camada, são encontrados serviços como gerenciamento dos dados, de suas versões, de sua preservação e de *feedback*. Por fim, a camada de dados e metadados fica responsável por cuidar de toda parte de persistência de dados, seja ela de extração da fonte de dados de origem, ou armazenamento em um banco de dados do sistema.

2.2 VERSIONAMENTO

Uma versão pode ser definida como um estado em particular de um determinado objeto, ou um *snapshot* do mesmo (ZHU, 2003). O versionamento trata de uma técnica para armazenar os múltiplos *snapshots* de um objeto de maneira independente e sem perda de dados. Na Figura 4 pode ser observado um arquivo que foi alterado mais de uma vez. Para cada uma das alterações, uma nova versão é criada. Estando todas as versões devidamente armazenadas e sua recuperação possível, tem-se um versionamento.

Figura 4 – Definição de Versionamento.



Fonte: O autor (2019).

A grande maioria das técnica disponíveis para versionamento são encontradas na área de engenharia de software, as quais são utilizadas para rastrear as diversas alterações feitas pelos desenvolvedores ao longo do tempo de um desenvolvimento de um programa de computador (PHILLIPS; SILLITO; WALKER, 2011). Além de garantir um melhor acompanhamento pelos líderes técnicos através de relatórios em tempo real, tecnologias de controle de versão como o Git [7] permitem o acesso e utilização a todas as versões anteriores do software. Técnicas de versionamento também podem ser encontradas quando se

[7] https://git-scm.com/

referindo a dados, sejam disponíveis em formato de arquivo ou em meios mais sofisticados como em banco de dados (FRANCONI; GRANDI; MANDREOLI, 2000; SOULES et al., 2002).

Para um sistema suportar um versionamento, é preciso que seja possível salvar, representar e extrair versões (ZHU, 2003). Salvar, significa guardar o acesso ao *snapshot* de maneira a o deixa-lo completamente acessível futuramente, representar significa manter todas as versões autosuficientes, independente de qualquer outro objeto armazenado no repositório e extrair significa que o versionamento deve permitir que todas as versões sejam consultadas de maneira ágil e fácil.

É importante ressaltar que este trabalho apresenta uma abordagem de versionamento, que diferentemente de sistemas de controle de versão, visa guiar os usuários a conceitos comuns de como tratar o versionamento em conjuntos de dados na web, tal como quais metadados devem ser utilizados para descrever uma versão e qual deve ser o formato de um identificador de versão. Sistemas de controle de versão têm o objetivo de cuidar de problemas como armazenamento de versões e segurança dos dados. Algumas propostas de sistemas de controle de versão para conjuntos de dados podem ser encontradas na literatura (BHATTACHERJEE et al., 2015; BHARDWAJ et al., 2014).

2.3 VERSIONAMENTO SEMÂNTICO

Introduzido por criadores do Sistema de Controle de Versão Git[8], o Versionamento Semântico foi criado com o objetivo de propor uma abordagem para a criação de identificadores de versão e diminuir o chamado inferno das dependências ("dependency hell") no versionamento em engenharia de software, ou seja, complicações no ao lidar com múltiplos pacotes dependentes em um software (PRESTON-WERNER, 2010). Segundo os autores, o número identificador de versão que dita como as versões evoluem pode vir a melhorar o gerenciamento das mesmas ao longo do tempo e impedir que problemas venham a congelar a geração de novas versões.

O número de versão proposto utiliza-se do padrão $X.Y.Z$ em que os valores iniciam sempre com zero, e com as alterações especificas sendo realizadas no objeto versionado, as casas têm seus valores somados em um. As casas podem ser incrementadas da seguinte maneira:

- **X** - Chamado de versão maior (MAJOR), o valor deve ser somado sempre quando uma alteração for realizada a ponto de mudar a compatibilidade do objeto versionado de alguma forma, por exemplo, a mudança de uma tecnologia utilizada em um determinado software.
- **Y** - Chamado de versão menor (MINOR), o valor deve ser somado sempre que uma alteração for feita e não mudar a compatibilidade do objeto versionado. A versão menor pode representar novas funcionalidades em um sistema.

[8] https://git-scm.com/

- **Z** - Chamado de versão de correção (PATCH), o valor sempre deve ser somado quando uma alteração relacionada a uma correção for realizada no objeto versionado, sem alterações na compatibilidade do mesmo.

Além das casas descritas acima, o versionamento semântico também permite a adição de novas casas, de maneira que o desenvolvedor das versões tenha liberdade para adicionar novas informações ao identificador quando necessário. Para isso, o versionamento semântico propõe o uso de hífen ou o sinal de adição após a casa de correções. Além disso, as novas casas devem ser caracteres alfanuméricos separados por pontos.

2.4 CONSIDERAÇÕES FINAIS

Foram apresentados neste capítulo os principais conceitos para o entendimento deste trabalho. Na Seção 2.1 foi dada uma breve explanação sobre dados na Web, bem como o ciclo de vida dos dados. Também foi mostrada a definição de um Sistema Gerenciador de Dados na Web e seus devidos requisitos. Por fim, foram mostrados os conceitos básicos sobre versionamento na Seção 2.2.

3 SOLUÇÕES PARA PUBLICAÇÃO DE DADOS NA WEB

Neste capítulo será apresentada uma análise de algumas das tecnologias para catalogação e publicação de dados na Web. Desta forma, uma comparação entre as soluções, incluindo o IAGO, será feita de maneira mais profunda. A partir daqui, este capítulo está organizado na seguinte maneira: Na Seção 3.1 será apresentada uma visão geral da análise comparativa feita e em seguida, na Seção 3.2, a comparação é detalhada. Na Seção 3.3 são dadas as considerações finais.

3.1 VISÃO GERAL DA COMPARAÇÃO

Ao longo do tempo, o interesse em se desenvolver soluções para a publicação e consumo de dados na Web vem crescendo (LóSCIO; OLIVEIRA; BITTENCOURT, 2015), e novas propostas relevantes surgem a cada dia, solucionando os desafios encontrados na área(GOLDACRE; GRAY, 2016; ZHAI; LIANG; HUANG, 2016). Ainda se sabe, porém, que muitas questões estão em aberto (CONRADIE; CHOENNI, 2014). Das soluções propostas disponíveis para uso, as mais populares se destacam por seu propósito genérico que permitem, no fim, uma publicação de dados na Web.

Embora as tecnologias de publicação de dados na Web disponíveis resultem em frutos como inúmeros portais de dados na Web muito utilizados, elas ainda apresentam características bastante heterogêneas. Devido ao alto número de soluções, a escolha de qual se encaixa no cenário em questão pode ser uma atividade de bastante custo para o usuário. Por isso, um critério de avaliação bastante relevante para esta questão é a análise de quais boas práticas propostas pelo DWBP as tecnologias estão a fazer uso (OLIVEIRA; OLIVEIRA; LÓSCIO, 2017). Na próxima Seção, uma análise comparativa é feita, levando em consideração cada uma das boas práticas e se as soluções listadas as atendem.

Nesta análise, foram selecionadas tecnologias levadas em consideração em outras análises comparativas como em (OLIVEIRA; OLIVEIRA; LÓSCIO, 2017), (STRÅLE; LINDÉN, 2014) e (LISOWSKA, 2016). Ao final, as seguintes soluções foram consideradas: CKAN[1], DKAN[2], Junar[3], OpenDataSoft[4], Knoema[5] e Socrata[6]. Para a validação de cada boa prática levada em consideração, foram analisadas as documentações e demonstrações gratuitas fornecidas pelas soluções, verificando se estas atendem às especificações descritas no documento do DWBP. Para ilustrar melhor, a Figura 5 mostra uma das descrições fornecidas pelo DWBP de como validar se uma boa prática está sendo atendida.

[1] https://ckan.org/
[2] https://getdkan.org/
[3] http://www.junar.com/
[4] https://www.opendatasoft.com/
[5] https://pt.knoema.com/
[6] https://www.tylertech.com/products/socrata

Quadro 2 – Análise comparativa de tecnologias para publicação de dados na Web.

DWBP	IAGO	CKAN	DKAN	Junar	OpenDataSoft	Knoema	Socrata
BP1	A	A	A	A	A	A	A
BP2	A	A	A	A	A	A	A
BP3	A	AP	N	AP	A	AP	A
BP4	A	A	A	A	A	N	A
BP5	A	A	A	A	A	A	A
BP6	AP	N	N	N	A	N	A
BP7	A	A	A	A	A	A	A
BP8	A	AP	AP	N	N	N	AP
BP9	A	A	A	A	A	A	A
BP11	A	N	N	N	N	N	N
BP12	A	A	A	A	A	A	A
BP13	A	AP	AP	AP	AP	A	AP
BP14	A	A	A	A	A	A	A
BP15	A	A	A	A	A	AP	A
BP17	A	A	A	A	A	A	A
BP18	A	A	N	A	A	A	A
BP19	A	N	N	A	A	N	A
BP20	N	N	N	N	A	N	N
BP21	A	N	AP	A	N	AP	A
BP22	A	AP	N	N	N	N	N
BP23	A	A	A	A	A	A	A
BP24	A	A	A	A	A	A	A
BP25	A	A	A	A	A	A	A
BP26	A	AP	AP	AP	AP	AP	AP
BP27	A	A	N	N	N	N	N
BP28	AP	N	N	N	AP	N	N
BP29	A	A	N	N	A	A	A
BP30	A	A	N	N	N	N	A
BP31	AP	N	N	AP	N	N	AP
BP32	A	A	A	A	A	A	A
BP33	AP	AP	N	N	AP	AP	A

Fonte: O autor (2019).

Figura 5 – Exemplo de como validar uma boa prática de dados na Web.

Fonte: (LóSCIO; BURLE; CALEGARI, 2017).

3.2 ANÁLISE COMPARATIVA

Para uma melhor visualização dos pontos levados em consideração na comparação, o Quadro 3.1 foi construído. A cada linha do Quadro, uma boa prática é analisada juntamente a cada uma das tecnologias selecionadas. Para representar a adoção total de uma BP por uma determinada tecnologia, foi utilizada a letra A. A adoção parcial é representada por AP. Finalmente, quando uma BP não é considerada em uma solução específica, então utiliza-se a letra N.

- **BP1** - Primeiramente, o DWBP propõe que os dados devem conter metadados para descrever os conjuntos de dados, que devem ser legíveis por seres humanos e máquinas. Mesmo utilizando metadados diferentes, todas as soluções analisadas cumprem a boa prática.
- **BP2** - A BP2 se refere disponibilizar metadados que descrevam a natureza dos dados e de suas distribuições, podendo ser lido por seres humanos e máquinas. Todas as soluções contempladas nesta análise atendem à boa prática.
- **BP3** - Esta boa prática propões que os conjuntos de dados sejam acompanhados de metadados que descrevam o esquema dos dados e de suas distribuições. A única tecnologia que não faz referências a este tipo de recurso é o DKAN, enquanto o Junar e Knoema apresentam a funcionalidade legível apenas por máquinas, atendendo

apenas parcialmente esta boa prática. No CKAN, é possível para o usuário a personalização de metadados, dando, assim, a possibilidade da inserção de metadados que descrevam a estrutura dos conjuntos de dados, mas não chega a ser necessariamente um recurso do sistema. As soluções restantes como IAGO, OpenDataSoft e Socrata atendem a esta boa prática.

- **BP4** - A BP4 fala a respeito da licença dos dados, propondo que os mesmos estejam vinculados a um link ou cópia da licença de uso. Nesta boa prática, apenas o Knoema deixa de cumprir, não fazendo nenhuma referência ao tipo de licença dos dados em seu sistema e em sua documentação.

- **BP5** - Indo para a BP5, é aconselhado informar a proveniência dos dados, ou seja, informações de onde os dados foram extraídos. Neste ponto, todas as tecnologias analisadas atendem a esta boa prática.

- **BP6** - Esta boa prática sugere que o Conjunto de Dados possua metadados sobre a qualidade dos dados. Das soluções avaliadas, apenas o OpenDataSoft e Socrata possuem esse tipo de informação.

- **BP7** - A BP7 é a primeira das boas práticas que falam a respeito de versionamento. Especificamente, que os conjuntos de dados sejam acompanhados de um identificador de versão ou data. Neste ponto, a maioria das soluções conseguem atender a esta boa prática informando a data que os dados foram disponibilizados. O IAGO, porém, é a única tecnologia que apresenta um identificador de versão numérico que pode ser interpretado para a compreender a versão em questão.

- **BP8** - Seguindo a linha da boa prática anterior, a BP8 aconselha que os conjuntos de dados possuam um histórico de versões e que este consiga descrever as mudanças que ocorreram entre uma versão e outra. De maneira geral, apenas o IAGO segue a boa prática de forma completa, apresentando um histórico de versão que detalha com riqueza as alterações realizadas entre uma versão e outra. Das demais, apenas o CKAN, DKAN e Socrata atendem parcialmente à boa prática, apresentando o histórico, mas sem entrar em detalhes a respeito das mudanças feitas.

- **BP9** - É necessário também, o uso de URIs persistentes para identificar os Conjunto de dados publicados na Web, de acordo com a BP9. Esta boa prática é seguida em todas as soluções em questão.

- **BP11** - A BP11 volta a mencionar o uso de versões e aconselha a utilização de URIs diferentes para cada versão existente dos conjuntos de dados, para que seus usuários possam acessar as mesmas de forma independente. Das soluções analisadas, apenas o IAGO permite que versões possam ser acessadas por diferentes URIs.

- **BP12** - Quando se referindo ao formato dos dados, a BP12 propõe que os dados sejam disponibilizados em uma distribuição legível por máquinas e que seja adequado ao potencial pretendido. Todas as soluções atendem à boa prática.
- **BP13** - A BP13 sugere que os dados não sejam influenciados por aspectos culturais ou linguísticos. Neste ponto, a maioria das tecnologias apresentam lacunas para o não cumprimento dessa boa prática em diversos níveis, apesar de oferecer a opção de representar metadados para descrever atributos específicos, como data e idioma. A únicas soluções que conseguem atender totalmente a esta boa prática são o IAGO e Knoema.
- **BP14** - O DWBP também sugere que os dados sejam disponibilizados em mais de um formato. Neste ponto, todas as soluções atendem a este requisito. É importante ressaltar, porém, que as mesmas não estimulam o uso de mais de uma distribuição, deixando a cargo do usuário a total responsabilidade de realizar a geração de múltiplas versões. No estudo feito, foi possível observar que a maioria dos conjuntos de dados publicados nessas tecnologias acabam apresentando apenas um formato de dados. Apenas o IAGO disponibiliza mais de um formato em todos os conjuntos de dados por fazer uso de processos automáticos de transformação.
- **B15** - A BP15 se refere ao uso de vocabulários que padronizam a codificação dos dados e de seus metadados. Em geral, todas as soluções fazem uso de vocabulários para decodificação de dados e metadados, apenas o Knoema, apesar de suportar o cumprimento da boa prática, deixa deixa a total critério do usuário o uso ou não dos vocabulários. Assim, foi considerado que o Knoema atende apenas parcialmente a esta boa prática.
- **B17** - A BP17 se refere à permissão que os dados possam ser recuperados totalmente por *download* em massa. Por se tratar de um requisito simples, todas as soluções seguem esta boa prática, independente da estratégia que utilizam para prover o *download*.
- **B18** - Já a B18 se assemelha à anterior, porém, trata-se de oferecer *download* dos subconjuntos de dados, ou seja, apenas uma parte dos conjuntos de dados, quando requerido. Apenas o DKAN segue parcialmente esta boa prática, deixando os subconjuntos legível apenas para visualização, não permitindo o *download* dos mesmos. Para todas as outras, a consulta de subconjuntos de dados é possível.
- **BP19** - Ainda se tratando de *download*, a B19 sugere que haja negociação de conteúdo para que o usuário possa definir qual o formato desejado dos dados por parametrização. Apesar de poder oferecer em mais de um formato, as tecnologias CKAN, DKAN e Knoema não oferecem o download das mesmas por negociação de conteúdo. As demais soluções conseguem seguir a recomendação com sucesso.

- **BP20** - Quando existem dados que são produzidos em tempo real, o DWBP propõe que os conjuntos de dados baseados nestes também possam ser fornecidos de igual forma. De todas as ferramentas em questão, apenas a OpenDataSoft apresenta o recurso.

- **BP21** - Em um cenário semelhante, a boa prática BP21 faz a recomendação de estabelecer uma frequência de atualização previamente e que os dados sejam atualizados de maneira a respeitar esta regra. Apenas o IAGO, Socrata e Junar seguem esta recomendação por meio de rotinas periódicas para a extração dos dados. No caso da DKAN e Knoema, existe a provisão de um metadado referente à frequência de atualização, este, porém, por não apresentar nenhum fluxo para a realização da atualização automática, muitas vezes deixa em aberto a atualização dos dados, atendendo, assim, a boa prática de maneira parcial.

- **BP22** - No cenário de dados na Web, é possível que um Conjunto de Dados seja descontinuado e até o acesso ao mesmo seja removido. Por isso, a BP22 sugere que, quando não mais disponíveis, os usuários sejam informados que o Conjunto de Dados existe, mas que não pode mais ser acessado, informando também o motivo. Esta boa prática é seguida apenas pelo IAGO. No CKAN a boa prática é atendida parcialmente, uma vez que informa a remoção do Conjunto de Dados, mas não dá detalhes do motivo e a solução recomenda que o usuário entre em contato com o administrador do sistema.

- **BP23** - O DWBP deixa claro, por meio da BP23, que os dados devem ser fornecidos por intermédio de uma API. De maneira unânime, as ferramentas levadas em consideração nesta análise seguem esta boa prática, além de utilizar as tecnologias mais recentes quando se tratando de criação de APIs.

- **BP24** - A BP24 sugere o uso de padrões da Web como REST[7] ou SOAP[8]. De semelhante modo, todas as soluções também fazem uso de padrões populares da Web em suas APIs.

- **BP25** - Ainda relacionado a APIs, a BP25 propõe que a mesma seja completamente documentada. Todas as tecnologias também apresentam a documentação completa de como utilizar suas APIs.

- **BP26** - Esta boa prática sugere que se devem evitar a alterações críticas nas APIs dos conjuntos de dados, evitando a provável quebra de aplicações que fazem consumos dos dados. Também é aconselhado que, caso necessário, a alteração na API seja comunicada. A maioria das soluções analisadas buscam manter o padrão em

[7] https://pt.wikipedia.org/wiki/REST
[8] https://pt.wikipedia.org/wiki/SOAP

suas APIs, mas não tratam a questão de quando uma mudança ocorre. Apenas o IAGO atende completamente a esta boa prática, criando novas versões das APIs quando necessário (ainda mantendo as versões anteriores disponíveis) e informando a complexidade das alterações realizadas.

- **BP27** - Quando o Conjunto de dados tiver seu acesso impedido, a BP27 sugere que o seu identificador seja preservado para guardar a sua identidade única. Dessa forma, não poderá existir outro Conjunto de Dados criado posteriormente, fazendo uso de um identificador de um outro Conjunto de Dados de acesso removido. Neste cenário, apenas o IAGO e CKAN fazem a preservação do identificador, as outras tecnologias avaliadas não fazem nenhum tipo de menção a este tipo de recurso.

- **BP28** - O DWBP também propõe que antes de arquivar um Conjunto de Dados, seja feita uma avaliação de sua cobertura, ou seja, localizar os recursos que fazem uso dos dados. De todas, apenas o IAGO e OpenDataSoft atendem parcialmente a esta boa prática, pois os mesmos permitem que seus consumidores cadastrem aplicações para informar aos administradores do sistema como os dados estão sendo utilizados.

- **BP29** - Tratando-se de *feedback* para os dados, as boas práticas de dados na Web aconselham que os consumidores possam prover o *feedback* de forma fácil, com o objetivo de relatar ao responsável pelos dados como está a sua experiência de consumo. Apenas o DKAN e Junar não atendem a este requisito.

- **BP30** - A BP30 sugere que o *feedback* fornecido pelos consumidores deve estar disponível publicamente. As soluções em questão, em sua grande maioria, apesar de apresentar o recurso de recolher o *feedback*, não o deixa publico para que qualquer tipo de usuário do sistema possa ter acesso. O IAGO, CKAN e Socrata atentem a esta boa prática.

- **BP31** - O DWBP recomenda que haja uma maneira de os dados serem enriquecidos, de acordo com a BP31. Com mecanismos de atualização automática de dados e com o devido cuidado com o *feedback* de seus consumidores, as soluções IAGO, Junar e Socrata conseguem atender parcialmente a BP31.

- **BP32** - Quando o assunto é visualizar o conteúdo do Conjunto de Dados, o DWBP propõe que existam visualizações complementares de forma a compreender melhor o que pode ser feito com os dados. Estas visualizações podem ser através de aplicativos da Web, tabelas, gráficos e etc. No geral, as ferramentas analisadas atendem bem a esta boa prática.

- **BP33** - Na BP33, é sugerido o provimento de *feedback* para o produtor original dos dados. De maneira completa, apenas o IAGO e Socrata atendem a esta boa prática. Já para as outras, especificamente CKAN, OpenDataSoft e Knoema, fica a critério

do administrador do sistema informar ou não o *feedback* recebido para o produtor de dados, desta maneira, atendendo parcialmente à boa prática.

As boas práticas BP10, BP16, BP34 e BP35 estão relacionadas à forma como os dados serão utilizados e não à catalogação, publicação ou acesso, ou seja, o seu atendimento não fica a cargo de quem publica, mas de quem consome os dados. Dessa maneira, estas boas práticas ficaram fora do escopo da avaliação.

Quadro 3 – Resultado da Análise Comparativa.

Solução	DWPB Atendida	DWPB parcialmente atendida
IAGO	26	4
CKAN	18	6
DKAN	14	4
Junar	17	4
OpenDataSoft	20	4
Knoema	15	5
Socrata	22	4

Fonte: O autor (2019).

O resultado final da análise comparativa feita pode ser observada no Quadro 3. Em questão de boas práticas atendidas, a solução que se sobressai é o IAGO. Pode-se dar destaque também à cobertura por parte dos sistemas Socrata e OpenDataSoft. É importante observar na análise feita que a maioria das soluções apresentam pendências em comum. Uma delas é a ausência de mecanismos que garantam a atualização automática dos dados, necessitando de interferência humana para que possa ser feita. Em alguns casos, mesmo que a tecnologia disponibilize uma frequência de atualização, o Conjunto de Dados não é atualizado conforme o combinado. Outro ponto a se dar destaque é a ausência de mecanismos para disponibilização automática dos dados em múltiplos formatos. Na maioria das tecnologias analisadas, o formato em que o Conjunto de Dados é publicado fica de autoria do trabalho manual do publicador de dados. Com isso, há ocorrências de publicações de dados legíveis apenas por máquinas ou de dados em apenas um formato. Por fim, apenas o IAGO apresentou recursos que fazem o gerenciamento de versões dos conjuntos de dados. As demais tecnologias, apesar de em sua minoria apresentarem um identificador de versão, não permitem que antigas versões dos conjuntos de dados sejam recuperadas e nem informam como os dados sofreram alterações ao longo do tempo.

3.3 CONSIDERAÇÕES FINAIS

Neste Capítulo foi mostrada uma análise comparativa entre tecnologias de publicação e catalogação de dados levando em consideração as boas práticas propostas no DWBP. Na

Seção 3.1 foi descrita uma visão geral da análise. Em seguida, na Seção 3.2, a comparação foi mostrada.

4 UMA ABORDAGEM PARA VERSIONAMENTO DE CONJUNTOS DE DADOS NA WEB

Neste capítulo, será detalhada a abordagem de versionamento de conjuntos de dados publicados na Web, implementada no IAGO. A partir deste ponto, o capítulo está estruturado da seguinte maneira: Na Seção 4.1, será apresentada uma visão geral da abordagem, seguido das principais definições para o entendimento da abordagem, na Seção 4.2. Na Seção 4.3 as operações de atualização dos conjuntos de dados são listadas, enquanto na 4.4, será explicado a abordagem para a construção de identificadores de versão, seguido da listagem dos metadados utilizados para descrever as versões, na Seção 4.5. Por fim, na Seção 4.6, são dadas as considerações finais.

4.1 VISÃO GERAL

Devido a flexibilidade e evolução da Web, dados e suas estruturas nela compartilhados podem sofrer alterações ao longo do tempo. A evolução dos conjuntos de dados, juntamente com a falta de conhecimento prévio de quem faz seu uso, torna o gerenciamento de versões algo de grande importância para o compartilhamento de dados na Web. Apesar da grande relevância do tema em questão, poucos trabalhos na área de dados na Web tratam deste assunto, como a proposta do Datahub (BHARDWAJ et al., 2014). O Datahub é um sistema de controle de versão para conjuntos de dados, cuidando para que as versões possam ser armazenadas e recuperadas posteriormente. O trabalho citado, porém, busca solucionar problemas de processamento de grandes quantidades de dados encontrados em outros sistemas de controle de versão como o Git[1] e SVN[2], deixando em aberto para seus usuários a abordagem que será utilizada para gerar novas versões dos conjuntos de dados. Semelhantemente, podem ser encontradas na literatura propostas para sistemas de controle de versão para dados em RDF[3] (FROMMHOLD et al., 2016) e ontologias na Web semântica (KLEIN; FENSEL, 2001). Por isso, neste trabalho, propomos uma abordagem de versionamento de conjuntos de dados publicados na Web. Baseando-se em trabalhos encontrados na área de engenharia de software (ZHU, 2003; PRESTON-WERNER, 2010), essa abordagem leva em consideração as operações realizadas nos conjuntos de dados, rastreando-as, descrevendo-as e permitindo a sua recuperação.

A abordagem proposta é composta pelos seguintes elementos: i) Um conjunto de operações para a atualização dos conjuntos de dados; ii) Uma abordagem para definição de identificadores de versão dos conjuntos de dados; iii) Um conjunto de metadados para descrever as informações das versões.

[1] https://git-scm.com/
[2] https://subversion.apache.org/
[3] https://www.w3.org/RDF/

4.2 CONCEITOS BÁSICOS

Para uma melhor compreensão da abordagem proposta, considere o conjunto de definições listadas a seguir:

Um Conjunto de Dados ou *dataset* pode ser definido como uma coleção de dados estruturados, semiestruturados ou não estruturados. Dados são valores que possuem um significado implícito, com o objetivo de se obter informação.

- **Definição 1. Conjunto de Dados:** Um Conjunto de Dados C é composto por uma tripla $\{I,\ M,\ V\}$, em que I é um conjunto de instâncias dos dados $\{i_1,\ ...,\ i_x\}$, M um par composto pelos conjuntos de metadados $\{Md, Me\}$ e V um conjunto de versões $\{v_1, ..., v_z\}$;

Uma instância pode ser caracterizada como um objeto que tem o estado e comportamento definidas por uma classe. Embora com diferentes valores armazenados, um grupo de instâncias compartilham atributos em comum. Um atributo pode ser definido como uma característica própria de alguém ou alguma coisa. Em computação, significa um elemento que define a estrutura de uma classe.

- **Definição 2. Atributo**: Um atributo t é formado pela dupla *(nm, vl)* em que *nm* significa o nome do atributo e *vl* o valor nele armazenado.

- **Definição 3. Instância**: Uma instância i é um vetor composto por um conjunto de atributos $\{t_1, ..., t_i\}$.

Para descrever um Conjunto de Dados, é necessário que o mesmo esteja acompanhado de uma série de atributos que ajudarão a compreender o seu conteúdo. As descrições destinadas a um Conjunto de Dados são chamadas de metadados ou metainformação.

- **Definição 4. Metadados**: Os metadados de um conjunto de dados C são descritos pelo par *{Md, Me}*, definidos como se segue:

  - **Metadados Descritivos** - O vetor Md remete a uma instância que possui descrições gerais sobre Conjunto de Dados tais como título, licença, proveniência, autor, cobertura fítica e cobertura temporal. Cada Md é formado pelo vetor $\{amd_1, ..., amd_j\}$ em que cada amd_j é um atributo composto pelo nome do metadado descritivo e o valor nele armazenado.
  - **Metadados Estruturais** - Os metadados armazenados em Me tem o objetivo de descrever as instâncias de dados I contidas dentro do Conjunto de Dados C. Cada Me é uma instância formada pelo vetor $\{tme_1, ..., tme_k\}$ em que cada tme_k representa um atributo composto pelo nome do metadado estrutural e o tipo de dado.

Uma atualização pode ser definida como o ato de realizar alterações necessárias um objeto em questão visando torná-lo mais adaptado para o cenário atual. Uma operação de atualização significa uma inserção, remoção ou alteração em instâncias. Como resultado de uma operação de atualização, tem-se uma mudança em um atributo do Conjunto de Dados.

- **Definição 5. Operação de Atualização**: Dado um Conjunto de Dados C, composto por um vetor de instâncias de dados I e uma dupla de metadados M, uma operação de atualização pode ser representada pela função *Opat(entrada, tipoOperação, nomeAtributo, novoValor)*, em que *entrada* pode ser uma instância de dados ou metadados. A *tipoOperação* significa operação a ser realizada podendo ser uma inserção, remoção ou revisão. *nomeAtributo* se refere ao nome do atributo alvo contido no vetor de entrada na função. Por fim, o *novoValor* representa o novo valor a ser armazenado no atributo referenciado em questão (este parâmetro não deve ser informado em operações de remoção).

- **Definição 6. Atualização de um Conjunto de Dados**: Uma atualização de conjuntos de dados A pode ser representada pelo par $(dtAtualização, \{Opat_1, ..., Opat_h\})$ em que *dtAtualizao* representa um *timestamp* para localizar a atualização em um ponto no tempo enquanto o vetor composto por $Opat_h$ significa o conjunto de operações de atualização aplicadas ao Conjunto de Dados.

Um log pode ser descrito como um conjunto de registros com o objetivo de documentar eventos importantes de um sistema, podendo ser utilizado para desfazer operações anteriores ou para manter o administrador do sistema informado de atividades no passado. Assim, um log de atualização de um Conjunto de Dados é criado para documentar todas as operações de atualização aplicadas por meio de uma atualização.

- **Definição 7. Log de Atualização**: Dado um Conjunto de Dados C que sofre a atualização A, devido à atualização, um novo Log é criado. O Log é composto por um conjunto de registros $\{r_1, ..., r_h\}$. Para cada operação de atualização $Opat_i$ contida em A, é adicionada uma nova linha ao registro r_h ao Log.

Um erro significa uma má compreensão ou análise deficiente a respeito de um fato. Em um Conjunto de Dados, diz respeito a um atributo que já foi inserido anteriormente, seja ele um dado ou metadado, porém que não reflete fielmente a realidade. A correção de um erro é realizada a partir de uma operação de atualização de revisão.

- **Definição 8. Erro**: Um conjunto de erros E em um Conjunto de Dados C é formado por $\{e_1, ..., e_o\}$, em que cada e_o é representado pela dupla {entrada, nomeAtributo}, sendo *entrada* uma instância i, enquanto *nomeAtributo* remete ao nome do atributo em que o erro se encontra.

Como já mencionado anteriormente, uma versão pode ser definida como um *snapshot* de um objeto em questão em determinado ponto no tempo (ZHU, 2003). Para este trabalho, uma versão de um Conjunto de Dados remete ao *snapshot* do mesmo após qualquer atualização realizada, não importando sua complexidade. A possibilidade de rastreamento e recuperação das múltiplas versões de um objeto, de forma que o usuário possa ter acesso a todo o histórico de alterações é chamado de versionamento (ZHU, 2003). Assim, além de considerar a geração de versões dos conjuntos de dados, o armazenamento e a recuperação das versões é de fundamental para o versionamento de conjuntos de dados na Web.

- **Definição 9. Versão de Conjunto de Dados**: Como já citado anteriormente, faz parte de um Conjunto de dados C um vetor de versões V, que pode ser representado por $\{v_1, ..., v_z\}$. Cada versão v_z é composta por $\{Opat(v_{z-1}),\ mv\}$, em que $Opat(v_{z-1})$ representa a aplicação das operações de atualização na versão anterior, sendo $v_1 = C$. Em mv podem ser encontrados os metadados de versão formados pelo conjunto de atributos $\{amv_1, ..., amv_u\}$.

4.3 OPERAÇÕES DE ATUALIZAÇÃO

A atualização de um Conjunto de Dados envolve um conjunto de operações que podem ser classificadas em dois grupos diferentes: limpeza e enriquecimento. Limpeza diz respeito a um processo que visa encontrar dados imprecisos ou incompletos, e corrigi-los, com o intuito de melhorar a qualidade do Conjunto de Dados(CHAPMAN, 2005). As operações de atualização de limpeza são:

- **Remoção de atributo de instância** - Esta operação é feita quando é necessário alterar as instâncias de um Conjunto de Dados, removendo o atributo de uma instância;
- **Alteração de atributo de instância** - Esta operação é feita quando é necessário alterar as instâncias de um Conjunto de Dados, alterando o atributo de uma instância, sem removê-lo;

Enriquecimento diz respeito a processos que visam agregar valor aos dados, seja na adição de novos dados ou de novas descrições (SANTOS, 2018). Neste sentido, as operações de atualização de enriquecimento inclui:

- **Adição de atributo de instância** - Esta operação é feita quando é necessário alterar as instâncias de um Conjunto de Dados, adicionando um novo atributo a uma instância;

4.4 IDENTIFICADOR DE VERSÃO

De acordo com o DWBP, uma versão de um conjunto de dados publicado na Web precisa conter um identificador que possa ser interpretado. Apesar de ser comum o uso de um valor que corresponda à data e à hora como identificador de versão, esta prática não consegue representar a complexidade das alterações que geraram a nova versão, ou se a mesma pode apresentar mudanças que tornam os dados incompatíveis com as estratégias utilizadas pelos consumidores dos dados. Assim sendo, a abordagem de versionamento de dados da Web proposta neste trabalho também sugere uma abordagem para identificadores de versão em conjuntos de dados publicados na Web.

Assumindo que os conjuntos de dados serão disponibilizados por intermédio de APIs, a abordagem proposta foi desenvolvida baseada no Versionamento Semântico (PRESTON-WERNER, 2010), em que o padrão MAJOR.MINOR.PATCH é utilizado. Dado um identificador de versão, os valores neles armazenados remetem a:

1. **MAJOR** - O número de versão maior é incrementado quando a atualização do Conjunto de Dados inclui alterações nos metadados estruturais, mudando, assim, a compatibilidade da API;

2. **MINOR** - O número de versão menor é incrementado quando a atualização do Conjunto de Dados inclui alterações que não comprometem a compatibilidade da API, ou seja, mudanças nas instâncias e metadados descritivos do Conjunto de Dados;

3. **PATCH** - O número de versão de correção é incrementado quando a atualização do Conjunto de Dados inclui correções a erros tanto nas instâncias quanto nos metadados;

Para o incremento do número de versão, é necessário seguir uma série de especificações, detalhadas na lista a seguir. As notações “DEVE”, “NÃO DEVE”, “OBRIGATÓRIO”, “DEVERÁ”, “NÃO DEVERÁ”, “PODEM”, “NÃO PODEM”, “RECOMENDADO”, “PODE” e “OPCIONAL” utilizadas nas especificações seguem o padrão estabelecido em (BRADNER, 1997).

1. O identificador de versão DEVE acompanhar o formato MAJOR.MINOR.PATCH, em que os valores que preenchem as casas são inteiros, não negativos e NÃO DEVE ser acompanhado de zeros à esquerda. Cada casa DEVE aumentar numericamente;

2. Uma vez que uma versão for criada e atribuído a ela um identificador, o seu conteúdo NÃO DEVE ser modificado. Qualquer modificação DEVE ser considerada uma nova versão e precisa receber um novo identificador;

3. A primeira versão do Conjunto de Dados DEVE conter o valor 1 no número de versão maior (MAJOR), enquanto os valores das demais casas DEVE ser 0;

4. A versão de correção (PATCH) DEVE ser incrementada apenas quando houver uma atualização que envolve operações que corrigem erros, podendo ser aplicadas a instâncias e metadados descritivos;

5. A versão menor (MINOR) DEVE ser incrementada em caso de atualizações que alterem as instâncias ou os metadados descritivos do Conjunto de Dados. PODE incluir mudanças a nível de correção (PATCH). A cada incremento da versão menor, a versão de correção DEVE ser definida para 0;

6. A versão maior (MAJOR) DEVE ser incrementada em caso de atualizações que envolvam a alteração em metadados estruturais. PODE incluir mudanças de versões menores (MINOR) e mudanças de correção (PATCH). A cada incremento na versão maior, o valor contido nas versões de correção e menor DEVE ser definido a 0.

Para exemplificar, supondo que exista um Conjunto de Dados em sua primeira versão 1.0.0, têm determinado momento, ele sofre com uma atualização com o objetivo de corrigir um erro. O número de versão passará a ser 1.0.1. Em seguida, novas instâncias de dados são inseridas, dando origem para uma nova versão de identificador 1.1.0. Por fim, os metadados estruturais têm um atributo removido, dando origem para a versão de identificador 2.0.0.

4.5 METADADOS DE VERSÃO

Sabendo da importância de metadados para descrição dos conjuntos de dados, é necessário também que cada versão do mesmo seja descrita tão bem quanto. Dessa forma, um conjunto de metadados foi estabelecido e listado no Quadro 4. Primeiramente, os metadados contêm o identificador de versão, este já explicado anteriormente. Em seguida, a data de criação é um valor correspondente ao momento em que a criação da nova versão foi feita. A Identificação do autor também é um metadado importante para apontar o responsável pela criação da versão. Um atributo para descrição livre é proposto, permitindo que o autor da versão disserte a respeito da mesma, caso necessário. Com o objetivo de descrever de maneira fácil a complexidade da atualização sofrida pelo Conjunto de Dados, metadados com valores quantitativos são propostos. Como pode ser observado, referindo-se a instâncias, existem contadores que documentam a quantidade de novas instâncias inseridas na nova versão, a quantidade de instâncias removidas e a quantidade de instâncias veteranas, que sofreram algum tipo de alteração, mesmo mantendo seu posto no Conjunto de Dados. Por fim, se tem o Log de atualização.

Fazendo parte dos metadados de versão, o Log de atualização busca detalhar todas as operações de atualização realizadas no Conjunto de Dados para que uma nova versão

Quadro 4 – Metadados de versão de um Conjunto de Dados

Atributo	Descrição
Identificador	Identificador único de versão
Data de criação	Data e Hora da criação da versão
Autor	Autor responsável pela criação da versão
Log de atualização	Log de atualização do Conjunto de Dados
Instâncias de dados adicionadas	Quantidade total de novas instâncias adicionadas
Instâncias de dados removidas	Quantidade total de instâncias removidas
Instâncias de dados atualizadas	Quantidade total de instâncias já existentes atualizadas
Metadados adicionados	Quantidade total de novos metadados adicionados
Metadados removidos	Quantidade total de metadados removidos
Metadados atualizados	Quantidade total de metadados já existentes atualizados

Fonte: O autor (2019).

possa ser gerada. Tendo seu desenvolvimento baseado em logs de transação utilizados em sistemas gerenciadores de bancos de dados relacionais (ELMASRI; NAVATHE, 2011), o log de versionamento da abordagem proposta registra cada uma das operações de atualização realizadas. Para isso, os atributos listados no Quadro 5 foram definidos. O primeiro registro no log é a identificação de onde a operação foi feita, podendo ser em uma instância, metadados descritivos ou metadados estruturais. Em caso de instância, é registrado identificador, caso contrário, não é necessário. O nome do atributo alvo da alteração também é registrado. Em seguida, o tipo de operação é informado, podendo ser uma limpeza ou enriquecimento e suas devidas classificações. Antigo e novo valor são campos opcionais. Em casos de registros de log de adição, apenas o novo valor é informado, em casos de remoção, apenas o valor antigo, enquanto, em renovação, os dois valores são listados.

Para exemplificar o log, supõe-se um Conjunto de Dados composto por instâncias formadas por dados de pessoas como *nome e idade*, sendo o atributo *nome*, o identificador da instância. Em determinado momento, uma pessoa teve sua *idade* alterada. O registro gerado para essa operação de atualização pode ser conferido no Quadro 6. Em um outro momento, o hipotético Conjunto de Dados recebeu um novo atributo nas instâncias dos dados. Utilizando a instância do exemplo anterior, o registro gerado pode ser observado na Quadro 7.

Quadro 5 – Atributos do *log* de atualização

Nome	Descrição
Nível da alteração	Indicador de onde a alteração foi realizada
Identificador de instância	Valor do atributo identificador de uma instância
Identificador de atributo	Nome do atributo em que a alteração foi feita
Tipo de operação	Campo de classificação do tipo de alteração realizada
Antigo valor	Antigo valor de campo, caso existente
Novo valor	Novo valor de campo, caso existente

Fonte: O autor (2019).

Quadro 6 – Exemplo 1 para *log* de atualização

Nome	Valor
Nível da alteração	Instância de dados
Identificador de instância	nome = Joaquim
Identificador de atributo	idade
Tipo de operação	Revisão
Antigo valor	34
Novo valor	35

Fonte: O autor (2019).

Quadro 7 – Exemplo 2 para *log* de atualização

Nome	Valor
Nível da alteração	Instância de dados
Identificador de instância	nome = Joaquim
Identificador de atributo	altura
Tipo de operação	Inserção
Antigo valor	–
Novo valor	1.73

Fonte: O autor (2019).

4.6 CONSIDERAÇÕES FINAIS

Neste capítulo, foi apresentada uma proposta de versionamento para conjuntos de dados publicados na Web. Na Seção 4.1, foi mostrada uma visão geral da abordagem proposta. Em seguida, na Seção 4.2, os conceitos básicos para uma melhor compreensão da abordagem em questão foram explicados. Na Seção 4.3, foram mostradas as possíveis operações de atualização para os conjuntos de dados. Na Seção 4.4, foi explicado o identificador de

versão usado para abordagem proposta. Por fim, foram apresentados os metadados de versão na Seção 4.5.

5 IAGO

Neste capítulo, será apresentada a nova proposta de Sistema Gerenciador de Dados na Web, o IAGO. Dessa maneira, o resto do capítulo está estruturado como se segue: na Seção 5.1, a visão geral sobre o IAGO é apresentada, tendo como foco seus principais diferenciais. Logo em seguida, na Seção 5.2, são descritos os casos de uso do sistema. A Seção 5.3 mostra a arquitetura do sistema, bem como seus serviços. Em seguida, na Seção 5.4, são mostrados os principais processos do IAGO. Por fim, na seção 5.5, são apresentadas as considerações finais.

5.1 VISÃO GERAL

O IAGO é uma implementação do modelo de referência para Sistemas Gerenciadores de Dados na Web (OLIVEIRA, 2017) e que busca trazer um auxílio para publicadores de dados em seu compartilhamento. O IAGO, além de buscar atender à maioria das boas práticas de dados na Web propostas pelo W3C, tem o propósito de automatizar os processos envolvidos. Para isso, um conjunto de serviços foi criado com o objetivo de fazer extração automática dos dados das fontes de dados diversas e o compartilhamento dos mesmos sem a necessidade de intervenção de seres humanos. Depois de publicado, o Conjunto de Dados disponível no IAGO passa a ser atualizado automaticamente pelo sistema, necessitando da intervenção humana em apenas ocasiões específicas.

É evidente o não atendimento adequado às boas práticas do W3C pelas soluções de publicação e catalogação de dados na Web atualmente disponíveis, como demonstrado no Capítulo 3. Dessa forma, com o intuito de preencher as lacunas em aberto, este trabalho vem propor um Sistema Gerenciador de Dados na Web, que busca atender à maioria das Boas Práticas de Dados na Web e oferece suporte para todas as etapas do ciclo de vida dos dados na Web(SILVA, 2019). Para o IAGO, destacam-se os seguintes diferenciais:

- Extração de dados de diversos tipos de fontes de dados;
- Geração automática de metadados para os conjuntos de dados, legíveis por seres humanos e máquinas;
- Atualização automática dos conjuntos de dados de acordo com frequências de atualização pré-definidas;
- Gerenciamento de versões dos conjuntos de dados;
- Geração automática de APIs para os conjuntos de dados criados;
- Geração automática de múltiplas distribuições para um mesmo conjunto de dados;

- Alta adaptabilidade do sistema.

Nas próximas Seções, serão explanados os diferenciais listados acima, levando em consideração as boas práticas de dados da Web (LóSCIO; BURLE; CALEGARI, 2017) que as mesmas atendem. Ao fim do detalhamento dos diferenciais, um resumo da relação entre os diferenciais e suas respectivas boas práticas serão apresentados.

5.1.1 Extração automática dos dados

Uma das características mais relevantes do IAGO é sua capacidade de extrair dados automaticamente de fontes de dados distintas, sem a presença de intervenção humana, necessitando apenas de um cadastro prévio das fontes de dados. Para isso, é necessário informar ao sistema as fontes de dados a partir das quais os conjuntos de dados serão extraídos. Dessa forma, o IAGO terá acesso aos dados originais para a primeira e futuras extrações. O sistema consegue, além de acessar os dados e realizar novas extrações quando necessário, cumprir a boa prática BP5 que se refere ao armazenamento de informações sobre a proveniência do Conjunto de Dados.

Devido ao grande número de abordagens para armazenamento de dados que utilizam diferentes formas de comunicação e codificação dos dados, a extração automática pode se tornar um desafio. Por isso, o IAGO foi construído para ser flexível ao ponto de permitir a inclusão de fontes de dados de diversos tipos, fazendo dele uma solução multiplataforma. Para isso, o IAGO inclui em seu acervo, bibliotecas que servem como intermediadores entre o sistema e as fontes de dados. Assim que a fonte de dados é identificada, o sistema escolhe a biblioteca compatível e que será responsável por conduzir a comunicação entre as partes. A partir disso, os dados podem ser extraídos, fazendo uso dos padrões tecnológicos reconhecidos pela fonte de dados.

5.1.2 Geração automática de metadados

É fundamental a boa descrição dos conjuntos de dados para uma melhor compreensão dos mesmos. Para isso, primeiramente, deve ser cadastrado um conjunto de metadados a respeito do mesmo. São levados em consideração tanto metadados descritivos que envolvem valores como autor e data de publicação, quanto metadados estruturais como a listagem dos tipos dos atributos que compõem as instâncias armazenadas nos conjuntos de dados. Construído como uma adaptação da proposta utilizada pelo DataCite (STARR; GASTL, 2011), os valores que envolvem os metadados descritivos podem ser observados no Quadro 8. É importante dar destaque primeiramente aos valores obrigatórios de URI, Título, Categoria, Licença e Frequência de Atualização. A URI é um identificador único para os conjuntos de dados. Título representa o nome do Conjunto de Dados no mundo real e, consequentemente, é um atributo de melhor interpretação para humanos. Um Conjunto de Dados cadastrado no IAGO deve estar incluso em uma categoria, e esta deve

ser informada como um metadado. A categoria tem o objetivo de separar os conjuntos de dados pelos seus conteúdos e facilitar o seu consumo. Como uma boa prática, também é de suma importância a definição de uma licença de dados para o Conjunto de Dados. Por fim, a frequência de atualização reflete a periodicidade com que o Conjunto de Dados deve sofrer atualizações. Este último serve tanto para configurar as atualizações automáticas que serão realizadas pelo sistema, quanto para informar aos consumidores quando novas versões serão criadas.

Entrando no campo dos atributos disponíveis, mas não obrigatórios, primeiramente tem-se o atributo de criador do Conjunto de Dados. O criador, como o nome já diz, é o responsável pela criação dos dados. O publicador fica responsável por coletar os dados e compartilha-los na Web. O contato deve conter uma informação para que o consumidor possa se comunicar diretamente com o responsável pelo Conjunto de Dados. Já as palavras-chave, apesar de não serem obrigatórias, são um conjunto de termos para ajudar na classificação do Conjunto de Dados. A língua remete ao domínio linguístico em que os dados se encontram. O período remete ao intervalo de tempo que os dados abrangem e, por fim, a cobertura espacial representa o espaço físico a que os dados fazem referência. Além de todos os metadados descritivos listados, o IAGO oferece a possibilidade de inclusão de novos metadados, uma vez que os diversos cenários da Web podem exigir descrições específicas para regras de negócio em questão.

Além disso, o IAGO também oferece a descrição dos atributos que compõe as instâncias armazenadas nos conjuntos de dados através de metadados estruturais. Os valores que fazem composição dos metadados estruturais podem ser observados no Quadro 9. O nome do atributo se refere ao valor exato da identificação do atributo que fazem composição das instâncias em um Conjunto de Dados. O tipo de dado é a definição do tipo básico do dado armazenado no atributo em questão tal como numeral, booleano ou string. Por fim, a descrição é um campo de texto livre para adição de novas informações que ajudem e compreender melhor o atributo, caso necessário.

É importante ressaltar que todos os metadados descritos acima, tanto descritivos quanto estruturais, são disponibilizados em formatos legíveis por seres humanos e por máquinas.

5.1.3 Atualização Automática dos conjuntos de dados

Um dos principais desafios na publicação e compartilhamento de dados na Web consiste em manter os conjuntos de dados atualizados. Por isso, O IAGO conta com a funcionalidade de manter os conjuntos de dados sempre renovados, fazendo com que os dados reflitam ao máximo a vida real por meio de módulos que permitem a atualização automática dos mesmos, atendendo de maneira direta à boa prática relacionada ao assunto (BP21). Para isso, uma verificação periódica é realizada, em que os conjuntos de dados cadastrados no sistema são verificados, a fim de identificar quais conjuntos de dados devem ser atualizados.

Quadro 8 – Metadados Descritivos no IAGO

Nome	Obrigatoriedade	Descrição
URI	Sim	Identificador único de um Conjunto de dados
Título	Sim	Nome do Conjunto de Dados no mundo real
Categoria	Sim	Classificação do Conjunto de dados entre as categorias
Descrição	Não	Descrição aberta
Criador	Não	Nome do criador do Conjunto de Dados
Publicador	Não	Nome do publicador do Conjunto de Dados
Contato	Não	Contato do responsável pelo Conjunto de Dados
Palavras-chave	Não	Palavras-chave que descrevem o Conjunto de Dados
Língua	Não	Língua predominante do Conjunto de Dados
Licença	Sim	Link para a licença do Conjunto de Dados
Período	Não	Período temporal que os dados representam
Cobertura espacial	Não	Espaço físico que os dados representam
Frequência de atualização	Sim	Frequência em que novas versões são geradas

Fonte: O autor (2019).

Quadro 9 – Metadados Estruturais no IAGO

Nome	Obrigatoriedade	Descrição
Nome do atributo	Sim	Nome correspondente a um atributo que compõe uma instância
Tipo de dado	Sim	Tipo do dado armazenado no atributo
Descrição	Não	Descrição geral sobre atributo

Fonte: O autor (2019).

Quando necessário, o sistema faz uso da extração automática para receber os novos dados e finalizar o processo de atualização. É importante ressaltar que, em seu cadastro, o Conjunto de Dados recebe um atributo que estabelece a sua frequência de atualização. No IAGO, as frequências de atualização disponíveis são a cada hora, diário, semanal, mensal, semestral e anual.

Por necessidade de novas atualizações, novas versões de um mesmo Conjunto de Dados são criadas de acordo com a frequência de atualização estabelecida previamente. Por isso, o IAGO conta com um gerenciamento de versões, explicado na próxima Seção.

5.1.4 Gerenciamento de versões

Como mencionado pelas boas práticas de dados na Web, é necessário que os dados sejam versionados, e que as versões disponíveis sejam bem descritas. O IAGO se destaca por buscar nunca apagar os dados nele armazenados, sempre gerando novas versões oriundas de atualizações. Assim, o IAGO foi implementado com um recurso de gerenciamento de versões. Desta forma, os conjuntos de dados podem ser acessados pelos consumidores em

todos os seus estados ao longo do tempo, com versões que existem independentemente, ou seja, podendo coexistir sem interferências. Com isso, é possível para o consumidor compreender toda a evolução dos conjuntos de dados consultando as versões desejadas pelo identificador de versão.

Para isso, foram levadas em consideração as recomendações das boas práticas que tratam sobre versionamento. As boas práticas recomendam que exista um histórico de versões, um identificador de versão que tenha um significado explícito e que as versões sejam bem descritas por meio de metadados. Por isso, no IAGO, as versões são descritas de maneira detalhada, para se ter a melhor distinção do conteúdo das mesmas. São levadas em consideração características como metadados estruturais, quantidade de instâncias, horário, data que as alterações foram feitas e autor da versão (quando necessário). Além disso, é de suma importância para o módulo de gerenciamento de versões do IAGO, o registro de todas as alterações realizadas nas operações de atualização, por meio de um log. Somado a isso, as versões disponíveis no IAGO também se caracterizam por conter URIs. Para cada nova versão criada, o sistema irá gerar um novo caminho URI, tendo como base para o discernimento entre as versões, o seu identificador.

O IAGO, além de armazenar as várias versões de um Conjunto de Dados, também o faz em vários formatos, visando também obedecer a uma outra boa prática do DWBP. A geração automática das distribuições é descrita na próxima Seção.

5.1.5 Geração automática de múltiplas distribuições

É sabido, de acordo com as boas práticas, que os conjuntos de dados devem ser disponibilizados em diversos formatos para evitar barreiras tecnológicas no consumo. Porém, este requisito, vem se mostrando um desafio para os publicadores, de acordo com o que foi visto na análise das tecnologias de publicação e consumo de dados na Web, muitas vezes, não contam com recursos disponíveis para o atendimento do mesmo.

A fim de sobrepor essa lacuna, o IAGO foi desenvolvido com um módulo de transformação automática dos dados. Este módulo converte, automaticamente, os dados recebidos pelas fontes de dados, para as distribuições que serão disponibilizadas na Web. Dessa forma, os conjuntos de dados publicados pelo IAGO possuem as mesmas distribuições, facilitando assim, a forma de como o consumo dos dados é feita. É importante destacar que o IAGO aplica as transformações a cada atualização do Conjunto de Dados e, consequentemente, em cada nova versão criada. Assim, as versões dos conjuntos de dados geradas pelo sistema também possuem mais de um formato de dados.

5.1.6 Geração Automática de APIs

Como já descrito anteriormente, existe um grupo de boas práticas que dizem a respeito ao uso de APIs para o compartilhamento de dados na Web. A utilização do padrão de API, além de ser bastante popular para desenvolvedores Web, cria uma camada de segurança

na aplicação, escondendo os detalhes de implementação. Apesar da maioria das soluções disponíveis compartilharem os dados na Web por meio de APIs, nem sempre todas as boas práticas são seguidas. Com sua proposta de processos automáticos, o IAGO gera as APIs para o consumo dos dados sem a interferência de seres humanos. Após a geração, é possível a realização de download dos dados tanto de maneira completa, quanto de seus subconjuntos, por meio de filtros personalizáveis. Todas as APIs do IAGO utilizam o padrão de comunicação REST[1].

Além disso, as boas práticas sugerem que haja negociação de conteúdo na solicitação dos dados. Como falado anteriormente, o IAGO disponibiliza seus conjuntos de dados em vários formatos, devido à sua transformação automática dos dados. Por isso, através de parâmetros em sua URI, o IAGO permite ao consumidor informar qual o formato de dados que melhor atenda ao seu propósito de uso dos dados.

Por fim, toda a documentação referente às APIs é disponibilizada pelo sistema de maneira automática. É importante ressaltar que, em função da tecnologia utilizada, toda a documentação é dinamicamente alterada em caso de mudanças nas APIs. Desta forma, o IAGO evita prováveis conflitos na utilização das APIs por parte dos consumidores.

5.1.7 Alta adaptabilidade do sistema

A Web sofreu muitas evoluções desde a sua criação (HALL; TIROPANIS, 2012) e continuará a receber mais adaptações no futuro. Com o passar do tempo, novas tecnologias surgirão e estas trarão novos recursos ainda não conhecidos. Consequentemente, os padrões e tecnologias utilizados para a publicação e compartilhamento de dados na Web tendem a evoluir. A fim de acompanhar a evolução das tecnologias da Web, o IAGO foi criado com o intuito de ser um sistema flexível a alterações, podendo ser adaptado para ser compatível com novas tecnologias e receber novos formatos de dados e metadados. As alterações no IAGO podem ser realizadas nos seguintes pontos:

- **Fontes de Dados** - O IAGO faz extração de dados de soluções desenvolvidas por terceiros, fazendo o uso de bibliotecas disponibilizadas pelos próprios autores desses sistemas. Com o passar do tempo, além da evolução natural dessas tecnologias, que podem gerar conflitos na comunicação com o IAGO, novas propostas podem surgir. Dessa forma, o IAGO oferece a adaptação das estratégias de extração de fontes de dados, oferecendo a possibilidade de adição, remoção ou atualização das bibliotecas responsáveis na comunicação do IAGO com os softwares que dão origem aos dados.

- **Formatos de Dados** - Semelhantemente aos softwares, novos padrões para representar os dados podem surgir. Além dos formatos já disponibilizados em sua versão atual, o IAGO permite que seus desenvolvedores possam incluir novos formatos de dados no processo de transformação automática dos mesmos. Para isso, o módulo

[1] https://pt.wikipedia.org/wiki/REST

de transformação possui um código fonte de fácil compreensão, que requer poucas alterações para incluir novas distribuições de dados, ou alterar as já existentes.

- **Metadados** - Apesar de o IAGO sugerir um formulário de metadados para descrição dos conjuntos de dados, é possível a inserção de novos metadados/ de forma livre. Isso é possível, pois os dados podem ser geridos por diferentes regras de negócio, exigindo metadados específicos para a sua compreensão. Diferente das outras alterações, esta personalização não exige uma mudança no código fonte do IAGO, permitindo desta maneira, que cada Conjunto de Dados possua uma estrutura de metadados diferente.

5.1.8 Diferenciais do IAGO x Boas Práticas

No Quadro 10, é mostrada uma lista que agrupa os diferenciais do IAGO e as boas práticas que estes ajudam a atender. É importante ressaltar que, além das listadas abaixo, o IAGO atende a um conjunto de outras boas práticas por meio de funcionalidades existentes no sistema, mas que não foram levadas em consideração nesta Seção. Uma análise mais geral sobre o IAGO e quais boas práticas foram atendidas foi feita no Capítulo 3.

Como pode ser observado, a extração automática dos dados atende diretamente a boa prática de número 5. A BP5 diz a respeito à provisão de informações sobre a proveniência dos dados. O IAGO não só guarda informações sobre a proveniência em formato legível por máquinas, como também mantém uma conexão com a mesma a fim de extrair novos dados no futuro. A geração automática de metadados em formato legível por seres humanos e máquinas atende à BP1, que sugere que os dados sejam acompanhados de metadados; à BP2, que diz que devem existir metadados que descrevam a natureza dos conjuntos de dados; à BP3, que sugere que existam metadados que descrevam a natureza estrutural do Conjunto de Dados e à BP4, que diz que um link ou cópia de licença deve acompanhar os dados.

O diferencial de Atualização Automática dos conjuntos de dados atende a uma boa prática que trata exatamente do assunto. A BP21 aconselha que os dados recebam uma frequência de atualização e que os dados sofram com atualizações de acordo com o que foi estabelecido. Já o diferencial de Gerenciamento de Versões atende à boa prática BP7, que diz que os dados devem ser acompanhados de um identificador de versão; à BP8, que aconselha que os as versões devem ser apresentadas através de um histórico; à BP11, que sugere que as versões devem poder ser acessadas separadamente através de URIs únicas e à BP26, que sugere que se evitem alterações na API com o intuito de evitar quebra de aplicações que fazem consumo dos dados. Essa última boa prática citada é atendida pois o IAGO disponibiliza as novas versões dos conjuntos de dados em novas APIs.

A Geração Automática de APIs atende à boa prática BP9, que diz que os conjuntos de dados devem ser disponibilizados através de URIs persistentes, a BP17; que diz que os

deve ser oferecida a opção de download em massa dos dados, à BP18, que sugere a opção de download de subconjunto de dados; à BP19, que propõe que seja possível a negociação de conteúdo para o download dos dados nos diversos formatos; à BP23, que diz que os dados devem ser disponibilizados através de APIs; à BP24, que se refere ao emprego de padrões Web como base das APIs tal como REST e à BP25, que sugere que toda a documentação da API seja fornecida. O diferencial de Geração Automática de Múltiplas Distribuições atende à boa prática BP12, que sugere que os dados sejam disponibilizados em um formato padronizado e legível por máquinas e à BP14, que propõe que os dados sejam fornecidos em múltiplos formatos.

Por fim, o diferencial da Alta Adaptabilidade do Sistema serve como uma estratégia de reforçar o atendimento às boas práticas. Por meio deste, é possível enriquecer o diferencial de Geração Automática de metadados, adicionando novos metadados aos conjuntos de dados. Também é possível adicionar novos formatos de dados, aumentando, assim, o acervo do sistema de distribuições geradas automaticamente. Por meio de adição de novas formas de extrair dados, o diferencial de Extração Automática dos dados também é beneficiado.

Quadro 10 – Boas Práticas de Dados na Web Atendidas Pelos Diferenciais do IAGO

Diferencial	Boas prática atendidas
Extração automática dos dados	BP5
Geração automática de metadados	BP1, BP2, BP3, BP4
Atualização automática dos conjuntos de dados	BP21
Gerenciamento de Versões	BP7, BP8, BP11, BP26
Geração automática de APIs	BP9, BP17, BP18, BP19, BP23, BP24, BP25
Geração automática de múltiplas distribuições	BP12, BP14
Alta adaptabilidade do sistema	–

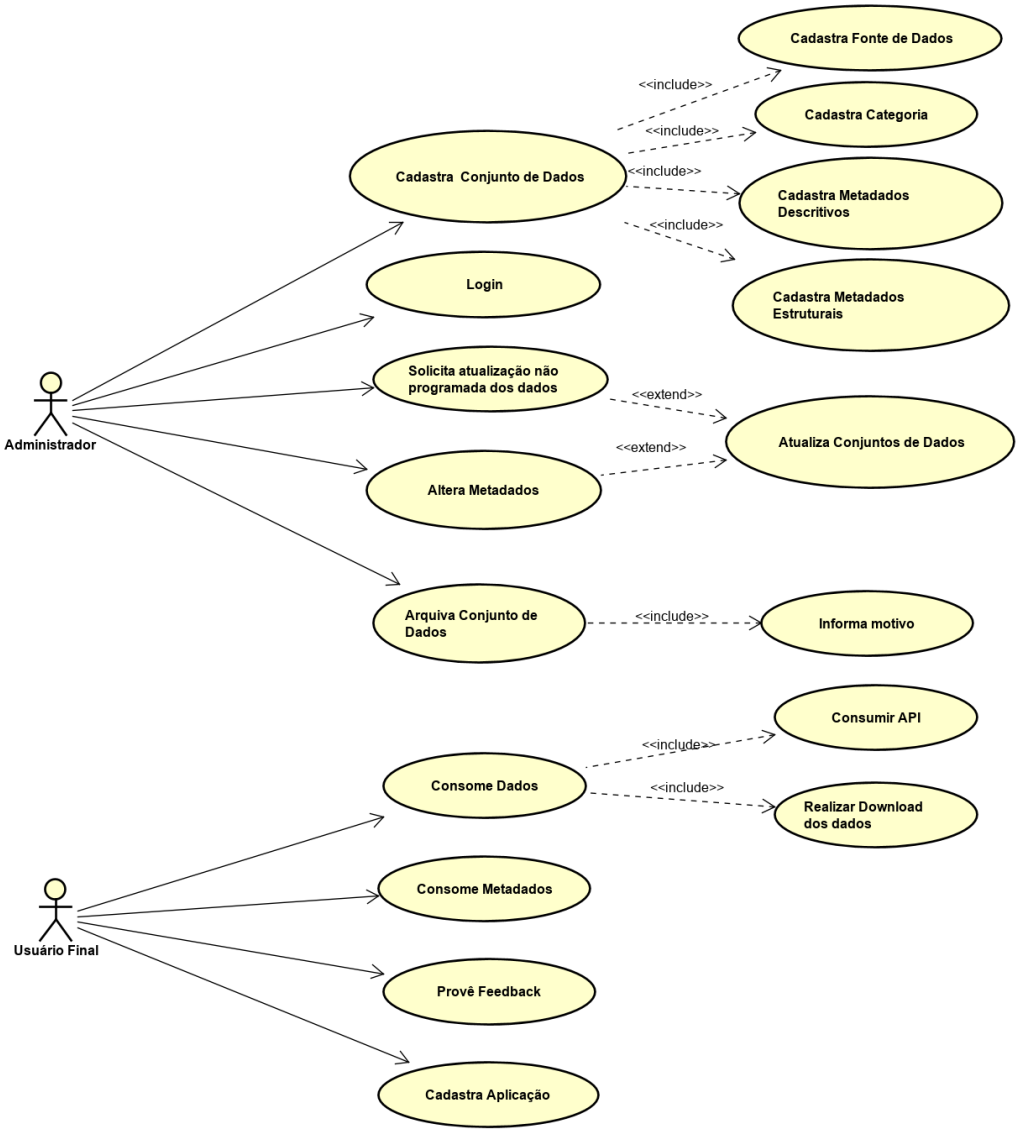
Fonte: O autor (2019).

5.2 PERFIS DE USUÁRIOS DO IAGO

Baseando-se nos papéis propostos no DWLM (SILVA, 2019), no IAGO, os usuários podem assumir dois papéis : Usuário Final e Administrador. É de responsabilidade do Administrador cadastrar conjuntos de dados e refinar os já existentes, caso necessário, por meio de atualizações. Já o Usuário Final é todo aquele que irá fazer o uso dos dados e, de maneira opcional, ajudar na evolução dos mesmos por meio de seu *feedback*. É importante ressaltar que as funcionalidades disponíveis para ambos os papéis podem ser acessadas tanto por seres humanos (interfaces gráficas disponíveis nativamente no sistema) quanto por aplicações (requisições para APIs).

Assim como pode ser observado no diagrama de casos de uso, representado na Figura 6, é possível para o Administrador cadastrar, atualizar e arquivar o conjunto de dados.

Figura 6 – Diagrama de casos de uso do IAGO



Fonte: O autor (2019).

O cadastro inclui a inserção de uma fonte de dados, que indica de onde os dados serão extraídos pelo sistema, o cadastro de uma ou mais categorias, com o intuito de classificar os conjuntos de dados que existirão, a inserção de metadados descritivos do conjunto de dados e o cadastro de metadados estruturais. Além disso, o sistema permite que o Administrador solicite que o Conjunto de Dados seja atualizado, ou seja, o IAGO irá extrair os dados da fonte de dados e irá criar uma nova versão a fim de atender a uma solicitação do Administrador. Por fim, o Administrador pode também remover o acesso ao Conjunto de Dados. Remover o acesso se refere a retirar a permissão para consultar o Conjunto de Dados por parte do Usuário Final, preservando os metadados e devidas identificações que caracterizam a anterior existência de um Conjunto de Dados. Dessa forma, o IAGO não permite que um novo Conjunto de Dados seja criado utilizando as mesmas informações

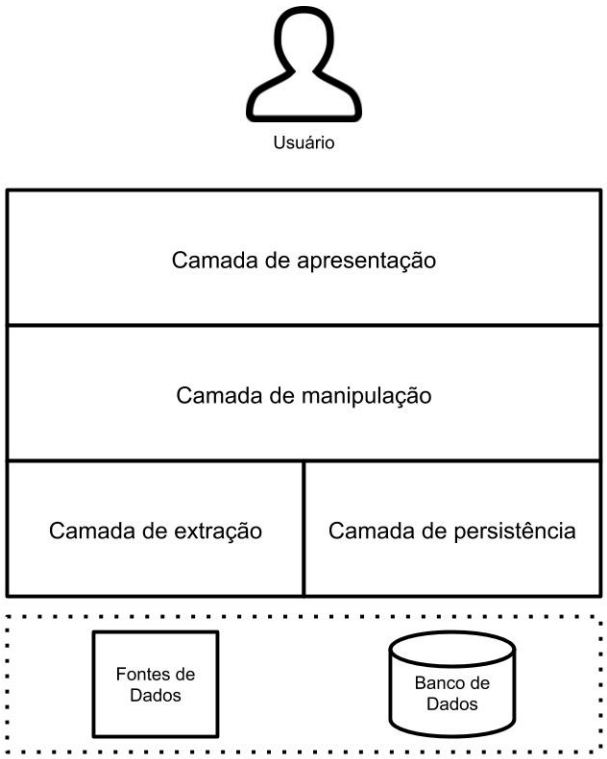
já cadastradas anteriormente em um outro Conjunto de Dados arquivado. Para que todos os casos de uso descritos acima possam ser acessados, é fundamental que o Administrador realize uma autenticação por intermédio de um login e senha.

Já para o Usuário Final, é possível consumir o conteúdo do Conjunto de Dados, sejam dados ou metadados, fornecer o *feedback* e cadastrar aplicações. Quando se referindo ao consumo, o caso de uso inclui tanto o acesso aos dados em suas devidas interfaces quanto o uso dos metadados. Também é possível a ele, enviar o *feedback* detalhando sua experiência de uso com um determinado conjunto de dados, classificando sua qualidade e sugerindo mudanças. Por fim, também é permitido o cadastro de aplicações, com o objetivo de informar ao Administrador que existe uma aplicação fazendo o uso dos dados daquele determinado Conjunto de Dados, e impedindo uma remoção indesejada no futuro.

5.3 ARQUITETURA DO IAGO

O IAGO segue o padrão de arquitetura em camadas, assim como proposto no modelo de arquitetura de SGDWs (OLIVEIRA, 2017), permitindo uma melhor organização do projeto e a fácil manutenibilidade de seu código fonte (SHAW; GARLAN et al., 1996). A Figura 7 ilustra as camadas da arquitetura proposta, e nas próximas seções, serão explanadas as camadas em mais detalhes.

Figura 7 – Camadas da Arquitetura do IAGO



Fonte: O autor (2019).

Como pode ser observado na Figura 8, um diagrama de componentes foi construído

para explanação de como todos os serviços implementados no IAGO funcionam entre si para que o sistema possa atender a todos os requisitos. Como explicado anteriormente, os serviços do IAGO são separados em camadas com responsabilidades distintas e o diagrama foi representado demonstrando a divisão das camadas, semelhantemente à Figura 7. Todos os serviços providos pelo IAGO estão dentro das caixas que são nomeadas com os títulos das respectivas camadas. É importante observar que existem componentes que representam serviços providos por terceiros, estes, sendo mostrados fora das camadas do IAGO.

5.3.1 Camada de Apresentação

A Camada de Apresentação é responsável por oferecer a interação dos usuários, com seus devidos papéis, com o sistema. Nesta camada é que são encontradas a interface Web e onde são geradas as APIs para o acesso aos conjuntos de dados. Também é de responsabilidade da Camada de Apresentação, a validação de todos os valores de entrada no sistema, com o objetivo de evitar o acionamento de erros não esperados. Um exemplo da funcionalidade da Camada de Apresentação é a verificação dos atributos para cadastro de um Conjunto de Dados, ou seja, se estes foram preenchidos corretamente.

A Camada de Apresentação é composta por uma interface Web com a proposta de ser de fácil uso para seres humanos, sejam eles do perfil de Administrador ou de Usuário Final. As interfaces Web são bem divididas em dois grupos de maneira a não misturar as responsabilidades dos perfis de usuário do sistema. Na interface Web voltada ao consumo (Usuário Final), é possível realizar todas as funcionalidades listadas anteriormente nos casos de uso, sem a necessidade de autenticações ou autorizações. Já a interface Web voltada à publicação (Administrador), diferencia-se por conter operações que irão manipular os conjuntos de dados, necessitando de uma autenticação prévia para que todos os seus serviços possam ser utilizados. É importante ressaltar que, fora as interfaces de uso amigáveis, o IAGO também apresenta um guia de utilização das mesmas.

De modo semelhante à interface Web, mas desta vez voltada a desenvolvedores, a interface remota tem o objetivo de oferecer uma maneira fácil de processamento por parte de máquinas. Fazendo o uso de linguagens de comunicação populares e tecnologias atuais, as interfaces remotas geradas pelo IAGO buscam não impor barreiras tecnológicas para o consumo dos dados. Para cada Conjunto de Dados, e cada versão do mesmo, o sistema gera automaticamente URIs únicas. Além disso, toda a documentação da interface remota também é disponibilizada, informando descrições gerais das APIs, os parâmetros que devem ser informados na entrada e o formato esperado na saída. Toda a documentação é gerada automaticamente pelo sistema.

Na Camada de Apresentação, podem ser encontrados dois componentes: a Apresentação do Administrador e o de Apresentação Usuário Final. Estes, proveem interfaces que podem ser acessadas, tanto mediante de páginas Web, quanto por serviços de APIs.

Abaixo, a descrição mais detalhada dos componentes:

- **Apresentação Administrador** - Em geral, a *Apresentação Administrador* possui serviços voltados a operações de manipulação dos conjuntos de dados, tais como criação, remoção, atualização e edição. Além disso, para que o usuário possa fazer uso da interface provida pelo componente de *Apresentação Administrador*, uma autenticação deve ser realizada, em que o usuário informará suas credenciais e o sistema permitirá ou não que se possa fazer uso dos serviços;

- **Apresentação Usuário Final** - Em contrapartida, a *Apresentação Usuário Final* possui operações relacionadas com o acesso aos dados, o sistema oferece várias maneiras de consultar os dados, podendo usar filtros com parâmetros tais como formato, versão, quantidade de registros desejados, entre outros.

5.3.2 Camada de Manipulação

A Camada de Manipulação é responsável pelo gerenciamento dos dados e metadados, bem como pela implementação dos requisitos funcionais do sistema. Esta camada é a mais importante, pois ela age de maneira central em relação ao sistema e interage com todas as outras camadas, acionando outros serviços, requisitando dados para as camadas de persistência de dados e interagindo diretamente com as requisições vindas da Camada de Apresentação.

Os componentes de serviços encontrados nesta camada são:

- **Serviço de Acesso** - Quando o Usuário Final requisita os dados, e a requisição é validada pela Camada de Apresentação, o *Serviço de Acesso* fica responsável por interpretar a requisição e retornar os dados nos termos desejados. Esse serviço leva em consideração a versão, o formato de dados e se o usuário deseja retornar o Conjunto de Dados por completo ou não (retornar um subconjunto de dados). Assim que a requisição é completamente processada, uma consulta aos dados é enviada para a Camada de Persistência, que retorna o resultado do pedido. Por fim, quando já estão prontos para serem retornados ao usuário que os requisitou, os dados são enviados novamente para a Camada de Apresentação, que finalizará o processo;

- **Serviço de Usuário** - Responsável por toda a autenticação e autorização do sistema, podendo bloquear acesso a outros serviços. Sendo utilizado exclusivamente pela *Apresentação do Administrador*, este serviço também é responsável pelo registro de todas as ações realizadas por Administradores, para que caso necessário, sejam acessados futuramente;

- **Serviço de Publicação** - Este serviço se destaca por ser o que mais gera dependências no sistema, fazendo dele uma peça central para o funcionamento do IAGO.

É neste serviço que ocorre o gerenciamento dos metadados e são realizadas as configurações do Conjunto de Dados no sistema, como a frequência de atualização dos dados. Além disso, é no *Serviço de Publicação* que os dados vindos da Camada de Extração são transformados para os formatos de dados que serão disponibilizados para o consumo;

- **Serviço de Atualização Automática** - Como dito anteriormente, o IAGO pode realizar a atualização automática dos conjuntos de dados, extraindo-os novamente da fonte de dados e os inserindo como uma nova versão. Para isso, é utilizado o *Serviço de Atualização Automática*, que executa seu código periodicamente independentemente de todos os outros componentes do sistema. Como mencionado anteriormente, no cadastro do Conjunto de Dados é estabelecida uma frequência da atualização, então, o serviço em questão faz uso desse valor para validar se o Conjunto de Dados precisa ser atualizado. Quando necessário, o *Serviço de Atualização Automática* faz utilização do *Serviço de Publicação* para que os novos dados possam ser extraídos, transformados,versionados e armazenados;

- **Serviço de Preservação** - Quando o acesso a um Conjunto de Dados precisa ser removido, a função é de responsabilidade do *Serviço de Preservação*. Sendo uma requisição vinda por um Administrador do Sistema, é possível remover o acesso ao Conjunto de Dados (incluindo todas as versões já criadas) preservando seus identificadores e metadados, de maneira a seguir a recomendação do DWBP. Esse serviço é consultado pelo *Serviço de Acesso* com o objetivo de validar as requisições vindas de Usuários Finais e autorizar ou não o acesso a determinado Conjunto de Dados requerido;

- **Serviço de Feedback** - O *Serviço de Feedback* é responsável por gerenciar as informações de feedback fornecidas pelos Usuários Finais para que se possa ter uma avaliação externa se os conjuntos de dados estão sendo utilizados. O serviço organiza as informações de feedbacks, relaciona-os aos conjuntos de dados requeridos e solicita o armazenamento para a camada de persistência. Além disso, o *Serviço de Feedback* também disponibiliza de forma pública os feedbaks já armazenados anteriormente, de maneira que outros usuários do sistema possam visualiza-los livremente. Por fim, também fica a cargo deste serviço a possibilidade de realizar o cadastro de aplicações que consomem os conjuntos de dados, a fim de coletar a informação de como os dados estão sendo utilizados por softwares de terceiros;

- **Serviço de Versionamento** - O *Serviço de Versionamento* é utilizado pelo *Serviço de Publicação*, sendo responsável por executar todas as rotinas do IAGO relacionadas ao versionamento dos conjuntos de dados. É nesse serviço que o IAGO identifica as alterações realizadas nos conjuntos de dados, registra-os através de logs de alte-

rações e os descreve por meio de metadados de versão. Somado a isso, o *Serviço de Versionamento* é responsável por criar e incrementar os identificadores de versão.

5.3.3 Camada de extração

A Camada de Extração é responsável pela comunicação do IAGO com os demais sistemas a partir dos quais os dados serão extraídos. Devido às tecnologias utilizadas, o sistema consegue recuperar os dados em um formato abstrato (sem um esquema pré-definido) e enviá-los para a camada de manipulação. Com a proposta de se ter um sistema com alta adaptabilidade, a Camada de Extração foi implementada juntamente com um conjunto de bibliotecas, proporcionando a compatibilidade com várias interfaces de acesso de dados, como Banco de Dados Relacionais. Além disso, a camada foi construída de maneira a se permitir a inclusão de novas estratégias para extração de fontes de dados.

Os componentes encontrados nesta camada incluem:

- **Serviço de Extração de Dados** - Este se refere a um conjunto de procedimentos que estabelecem a comunicação entre o IAGO e a Fonte de Dados. Para que isso seja possível, o IAGO executa seus algoritmos de extração juntamente com bibliotecas desenvolvidas por responsáveis pelas fontes de dados, criando uma comunicação sem ruídos entre as partes. Dessa maneira, este serviço trabalha com o objetivo de que os dados sejam recebidos pelo IAGO e enviados para a camada de manipulação em um formato abstrato e otimizado de forma que possam ser facilmente manipulados pelos outros serviços;

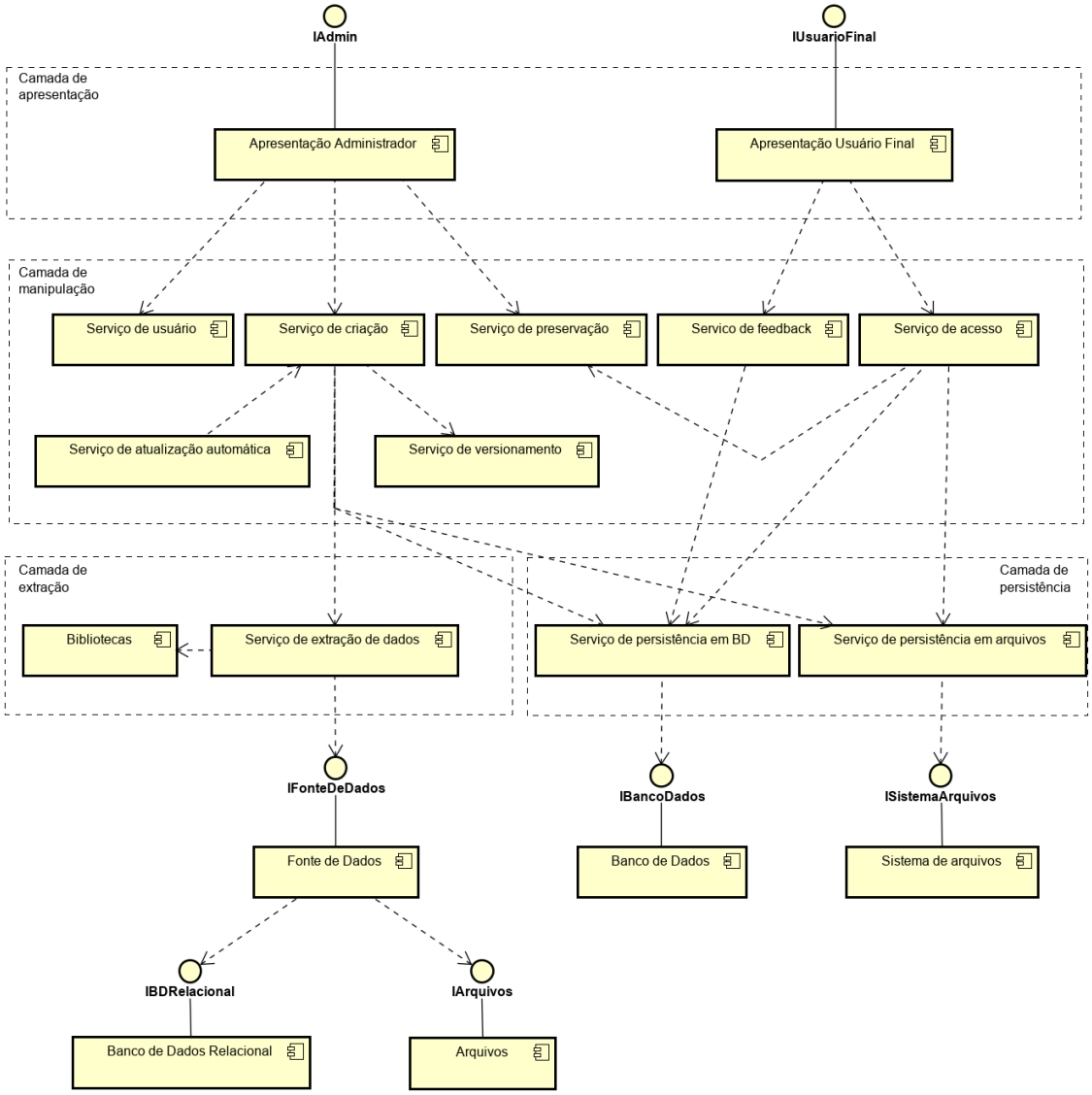
- **Bibliotecas** - Como citado anteriormente, para que o *Serviço de Extração* possa funcionar corretamente, algumas bibliotecas precisam ser compiladas juntamente com as solicitações às fontes de dados para que o IAGO possa encontrar uma linguagem de comunicação em comum com as mesmas. As bibliotecas incluídas no IAGO podem ser encontradas em repositórios online, disponibilizadas pelos próprios criadores. Por isso, o componente de *Bibliotecas* inclui todos os arquivos de terceiros que são necessários para a integração do IAGO com das tecnologias de fontes de dados mais variadas. Dando destaque para o diferencial de adaptabilidade do sistema, é possível a adição de novas bibliotecas, aumentando o acervo do sistema.

5.3.4 Camada de persistência

Por fim, a camada de persistência cumpre o papel de armazenar e recuperar os dados (estes já transformados e vindos da camada de manipulação) e metadados em um repositório, além de guardar todas as configurações gerais do sistema. Antes de entrar em detalhes sobre os serviços, é necessário compreender que o IAGO armazena os dados em dois repositórios diferentes, sempre inserindo os conjuntos de dados em ambos:

- **Repositório em Banco da Dados** - Este repositório consiste em um banco de dados não relacional, caracterizando uma natureza semiestruturada dos dados. Devido à tecnologia utilizada no repositório, é possível o uso de operações mais complexas nos dados tal como a busca por subconjuntos de dados.
- **Repositório em Arquivos** - Aqui os dados são armazenados em arquivos. Por sua natureza não complexa de armazenamento, as operações de retorno dos dados é de alta velocidade. Além disso, por se tratarem de arquivos de texto, os dados são salvos em mais de um formato.

Figura 8 – Diagrama de Componentes do IAGO.



Fonte: O autor (2019).

Como descrito, o IAGO utiliza uma abordagem híbrida para armazenar os seus dados. O primeiro repositório, apesar de apresentar um tempo de resposta significantemente

positivo e permitir operações complexas, este não suporta o uso de mais de um formato de dados no armazenamento. O segundo, por se tratar de uma estratégia mais simplificada de armazenamento, consegue preencher a lacuna deixada pelo primeiro repositório. Este, porém, não permite operações mais complexas com os dados. Dessa forma, quando o sistema recebe uma requisição para retornar dados, visando o custo-benefício, o IAGO tem duas opções em que a primeira consegue realizar operações mais complexas e a segunda retornar uma maior quantidade de dados em uma velocidade maior.

Nesta camada são encontrados dois componentes:

- **Serviço de Persistência em Banco de Dados** - Este serviço é responsável por acessar o repositório em banco de dados. Com ele, é possível manipular, organizar e acessar os dados de maneira rápida e fácil.
- **Serviço de Persistência em Arquivos** - De modo semelhante, o Serviço de Persistência em Arquivos fica a cargo de mediar as requisições ao repositório de arquivos do IAGO, podendo inserir novos arquivos e removê-los.

5.4 DIAGRAMAS DE ATIVIDADES DO IAGO

Nesta Seção, apresentamos dois diagramas de atividades que descrevem dois dos principais processos realizados pelo IAGO: o processo de publicação de um conjunto de dados e o processo de versionamento de um conjunto de dados. O diagrama da Figura 10 apresenta o fluxo de atividades do processo de publicação enquanto que o diagrama da Figura 9 apresenta o fluxo de atividades do processo de versionamento. Cada um desses diagramas será detalhado a seguir.

O processo de publicação de um Conjunto de Dados no IAGO ilustrado na Figura 10 contempla as seguintes atividades:

1. Primeiramente, uma categoria deve ser cadastrada, caso não existente. Caso uma categoria já existente seja selecionada, o processo poderá continuar.
2. Em seguida, é necessário informar ao sistema de onde os dados serão extraídos, para isso, deve ser cadastrada uma fonte de dados. Caso a fonte de dados já tenha sido cadastrada previamente, o processo continua, caso contrário, o IAGO validará a conexão com a nova fonte de dados para certificar que a extração de dados é possível.
3. Assim que a validação da fonte de dados é finalizada, ocorre o cadastro dos metadados que descreverão o Conjunto de Dados.
4. O próximo passo é inserir os dados de consulta que serão utilizados como parâmetro na fonte de dados selecionada a fim de que os dados desejados sejam retornados. É

importante destacar que os dados de consulta podem variar de acordo com o tipo de fonte de dados cadastrada, por exemplo, se os dados forem extraídos de um banco de dados relacional, uma consulta SQL deve ser informada. Em outro cenário, no qual os dados serão extraídos de um arquivo de texto, o caminho para o arquivo deve ser parametrizado.

5. Assim, os metadados da primeira versão são gerados, incluindo o primeiro registro no histórico de versões e o primeiro identificador de versão.

6. Quando a parte cadastral é finalizada, todos os metadados que descrevem o Conjunto de Dados, bem como suas informações para extração na fonte de dados são finalmente armazenados no repositório de dados do IAGO. A partir daqui, todo o processo é feito de maneira totalmente automática.

7. O primeiro passo para a extração de dados é verificar se a fonte de dados está disponível no momento. Caso afirmativo, o processo pode continuar, caso contrário, o sistema espera mais uma hora até que a verificação possa ser feita novamente. O tempo de uma hora representa o período em que as rotinas automáticas do IAGO são executadas para verificar quais conjuntos de dados devem ser atualizados.

8. Com a verificação bem sucedida, os dados são finalmente solicitados e retornados da fonte de dados. Os dados são retornados em um formato pré-processado de maneira a facilitar o próximo passo, que transforma os dados para o formato das distribuições disponíveis no IAGO e, consequentemente, armazenados definitivamente no sistema.

9. Por fim, com todo o processo finalizado, as APIs para acesso aos dados são finalmente criadas e abertas para o uso dos consumidores de dados.

De modo semelhante, o processo de atualização automática dos conjuntos de dados ilustrado na Figura 9 é contemplado pelos seguintes passos:

1. Primeiramente, o IAGO verifica cada um dos conjuntos de dados cadastrados no em sua base de dados. Cada Conjunto possui em sua configuração uma frequência de atualização preestabelecida e, caso seja o momento para que uma nova versão seja gerada, o sistema continua o processo, caso contrário, nada acontece.

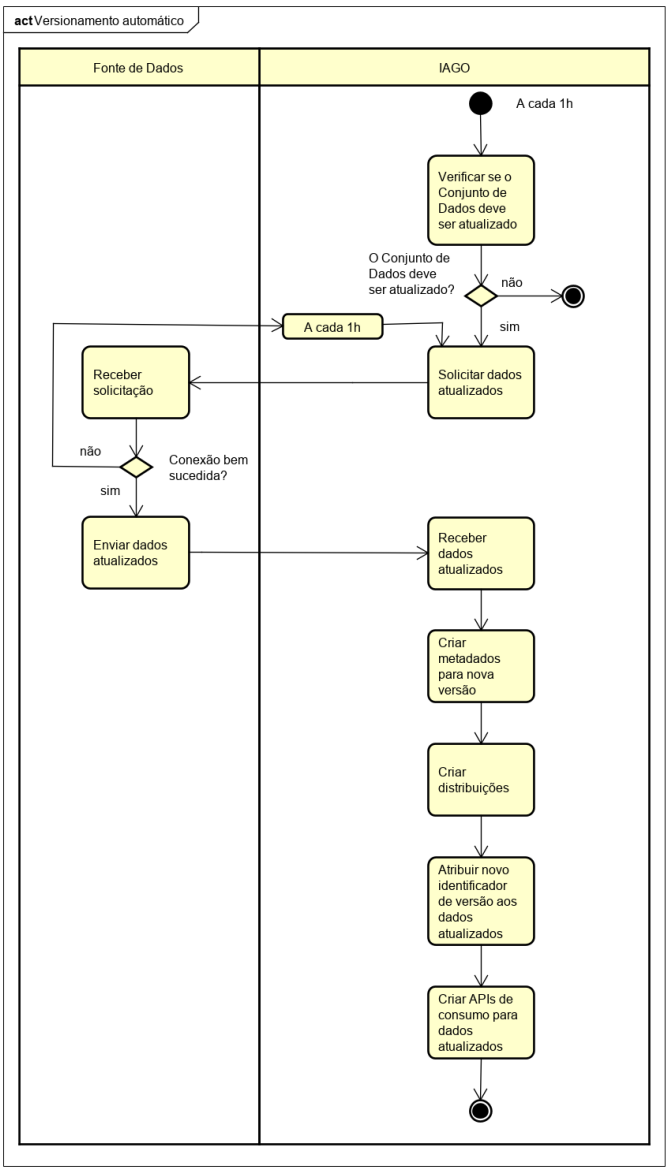
2. Semelhante ao primeiro processo apresentado, os dados são solicitados novamente para a fonte de dados. Se ocorrer alguma falha na solicitação, o IAGO deverá esperar a próxima rotina de atualização, ou seja, em uma hora.

3. Com os novos dados recebidos, o sistema gera automaticamente os metadados de versão, sem apagar ou substituir os outros metadados de versão já disponíveis anteriormente.

4. Em seguida, a partir dos novos dados recebidos, são criadas as devidas distribuições que serão disponibilizadas;

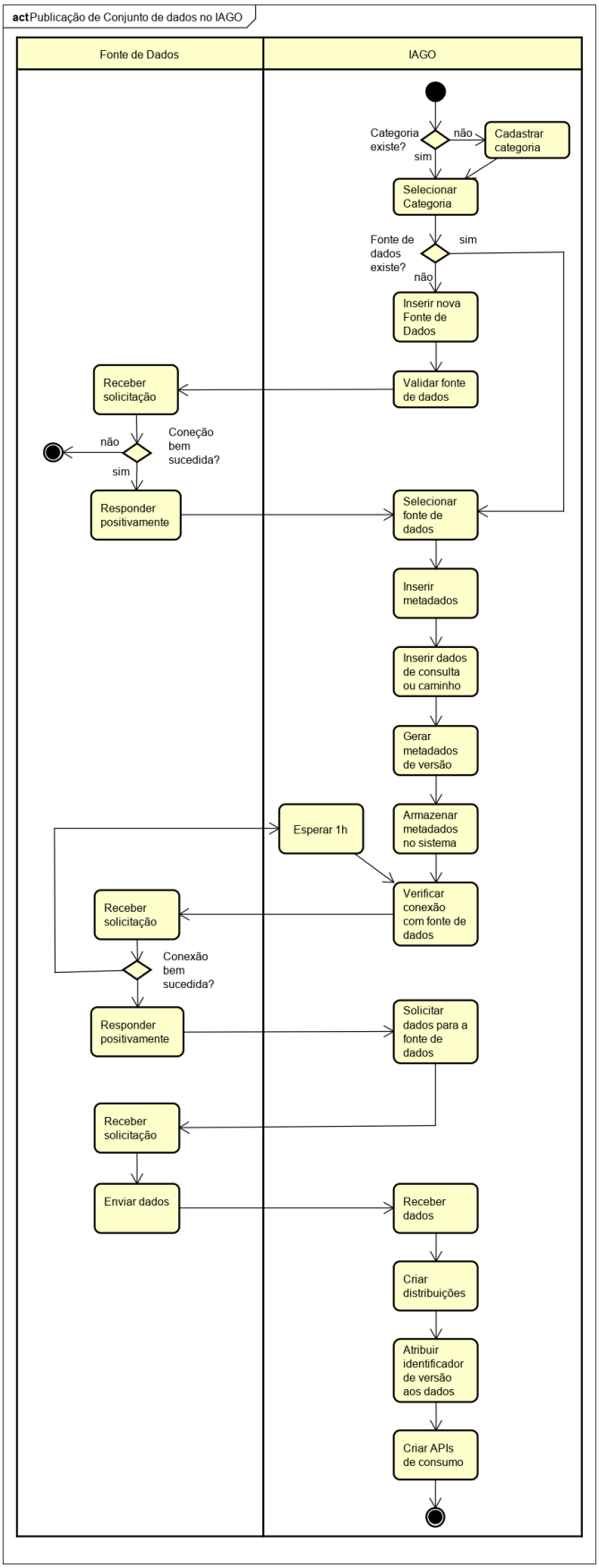
5. Por fim, a API da nova versão é disponibilizada. É importante ressaltar que todas as versões anteriores, com suas devidas APIs ficam disponíveis independentemente de quantas novas versões forem criadas.

Figura 9 – Diagrama de Atividades Sobre o Versionamento Automático no IAGO.



Fonte: O autor (2019).

Figura 10 – Diagrama de Atividades Sobre a Publicação de Conjuntos de Dados no IAGO.



Fonte: O autor (2019).

5.5 CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentadas as funcionalidades presentes no IAGO. Na seção 5.1, a visão geral do sistema foi apresentada dando destaque a seus diferenciais. Em seguida, através de um diagrama de casos de uso, a Seção 5.2 detalha os perfis de usuários disponíveis no IAGO. Em seguida, a arquitetura do IAGO e suas respectivas camadas e serviços foram apresentados na Seção 5.3. Por fim, na Seção 5.4 foram mostrados os processos de publicação e atualização automática dos conjuntos de dados.

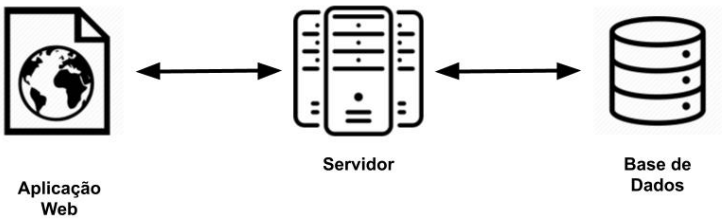
6 IMPLEMENTAÇÃO E EXPERIMENTOS

Neste Capítulo serão apresentados detalhes de implementação do IAGO juntamente com as tecnologias utilizadas no processo. A visão geral das tecnologias utilizadas no desenvolvimento das camadas e serviços é apresentada na Seção 6.1, seguida da validação do sistema na Seção 6.2. Por fim, as considerações finais ficam na Seção 6.3.

6.1 VISÃO GERAL DA IMPLEMENTAÇÃO

Implementando as condições apresentadas no Capítulo anterior, o IAGO foi desenvolvido e seu código fonte está disponibilizado em <https://github.com/wilkersantos/iagopublico>.

Figura 11 – Divisão das tecnologias do IAGO.



Fonte: O autor (2019).

Como mencionado anteriormente, o IAGO foi construído utilizando uma arquitetura para aplicações Web em camadas, permitindo a separação entre os componentes responsáveis pela interação do cliente, o processamento das aplicações e o gerenciamento dos dados, como pode ser observado na Figura 11. Dessa forma, as tecnologias utilizadas se dividem em três: aplicação Web, servidor e base de dados. As bibliotecas utilizadas na aplicação Web do IAGO têm como fruto a geração das páginas HTML, que facilitarão a interação dos usuários com o sistema e ajudarão na visualização dos dados. Além disso, a aplicação Web implementa os formulários que serão enviados ao servidor, validando-os e dando suporte ao usuário, quando necessário. Por sua vez, o servidor representa o núcleo do IAGO e onde todas as funcionalidades são executadas. O servidor é acessado pela aplicação Web do IAGO ou qualquer outra aplicação que consuma as suas APIs. Por fim, a base de dados armazena todos os dados que serão processados pelo sistema.

6.1.1 Tecnologias da aplicação Web

As páginas Web do IAGO foram construídas com o objetivo se apresentarem de fácil utilização para consumidores e publicadores de dados. Assim sendo, a interface Web se caracteriza por suas telas amigáveis e intuitivas. Um dos destaques da aplicação web do IAGO é que as páginas foram desenvolvidas podendo ser personalizadas, permitindo

que o sistema tenha uma temática mais compatível com o que o Administrador deseja. Somadas a isso, as tecnologias utilizadas no desenvolvimento são populares na comunidade de engenharia de software, tornando assim, fácil para que desenvolvedores possam alterar a estrutura da aplicação Web da maneira que achar melhor.

Para isso, o primeiro ponto a se dar destaque é a existência do *framework* de aplicações Web baseado na linguagem Typescript[1], o Angular[2]. O Angular se destaca por separar bem as responsabilidades dos serviços no código fonte entre suas camadas, além de portar com uma ótima abordagem de injeção de dependências. Somada a isso, uma das principais propostas da tecnologia é aumentar a produtividade do desenvolvimento, permitindo com que o desenvolvedor de software possa escrever menos linhas de código e se ter um maior número de funcionalidades criadas. O Angular também conta com uma comunidade de desenvolvimento bastante ativa, sempre somando ao número de bibliotecas compartilhadas em repositórios online que podem ser reutilizadas em projetos que fazem uso da tecnologia. Além disso, o Angular é um *framework* que ainda se mantém líder quando se trata de compatibilidade com as mais recentes novidades na área de desenvolvimento Web.

Sendo utilizado juntamente com o Angular, com o objetivo de se ter uma aparência robusta e bonita, foi utilizado o MaterializeCss[3]. O MaterializeCss é um *framework* que traz um conjunto de componentes visuais já construídos como botões, janelas, campos de textos, que podem ser incluídos em uma página Web em desenvolvimento. As páginas criadas por essa tecnologia são totalmente responsivas, ou seja, se adaptam totalmente a qualquer resolução do dispositivo que as acessam, permitindo, assim, que dispositivos móveis como smartphones e tablets possam encontrar uma visualização agradável e sem problemas na quebra de componentes. Além disso, o MaterializeCss é compatível com a grande maioria das bibliotecas disponíveis junto ao Angular, facilitando, assim, o desenvolvimento de componentes visuais mais complexos.

Para que todas essas principais tecnologias descritas acima pudessem ser compiladas e organizadas na aplicação, foi utilizado o gerenciador de pacotes NPM[4]. Com ele, é possível fazer o gerenciamento de todas as bibliotecas necessárias para a execução do código fonte do projeto, através do download das mesmas através de um repositório online. Somado a isso, foi utilizado o Angular CLI[5]. Esta ferramenta ajuda a seus desenvolvedores na organização do projeto Angular e automatizar a criação e remoção de seus componentes através de linhas de comando. O Angular CLI consegue agilizar toda a fase inicial de desenvolvimento, e ajuda seus usuários a se ter um projeto Angular totalmente padronizado e de manutenção mais ágil.

Com o objetivo de melhor listar as tecnologias utilizadas na aplicação Web do IAGO,

[1] https://www.typescriptlang.org/
[2] https://angular.io/
[3] https://materializecss.com/
[4] https://www.npmjs.com/
[5] https://cli.angular.io/

o Quadro 11 foi construído.

Quadro 11 – Tecnologias utilizadas na aplicação Web do IAGO.

Nome	Descrição
NPM	Ferramenta de gerenciamento automático de bibliotecas
Angular	*Framework* para desenvolvimento Web na linguagem Typescript
MaterializeCSS	*Framework* de componentes visuais
Angular CLI	Ferramenta para organização da estrura de projeto Angular de maneira automárica

Fonte: O autor (2019).

6.1.2 Tecnologias do servidor

Como o componente central do sistema, é no servidor onde se encontram a maioria das tecnologias e algoritmos de alta complexidade. Utilizando a arquitetura apresentada no Capítulo anterior, o servidor utiliza padrões de desenvolvimento comuns na engenharia de software com o objetivo de facilitar a evolução e a manutenção do sistema. Para isso, foram escolhidas bibliotecas populares na comunidade de desenvolvimento de software e de suporte contínuo por parte de seus criadores. A primeira tecnologia a dar destaque é a linguagem de programação Java[6] que, além de ser uma das linguagens de programação mais utilizadas atualmente, possui as vantagens de ser multiplataforma e retrocompatível com as mais diversas versões da linguagem. Além disso, o Java possui uma comunidade de desenvolvimento bastante robusta, que todos os dias procuram enriquecer os recursos oferecidos pela linguagem de programação.

Para a implementação e divisão das camadas e padrões de projeto de programação, o framework para Java chamado Spring[7] foi utilizado. Este se caracteriza por utilizar bem os padrões de projeto como inversão de controle e injeção de dependências, ajudando, assim, a divisão das responsabilidades entre as camadas do IAGO. No IAGO, o Spring também se destaca por ser responsável pela criação dinâmica das APIs dos conjuntos de dados e por desencadear a verificação periódica para atualização automática dos dados. Somado a isso, o Spring possui a possibilidade de adicionar novos recursos ao framework a partir de downloads que podem ser feitos em repositórios online, aumentando, assim, a quantidade de funcionalidades do mesmo. Para o IAGO, foi adicionado Spring Security[8] com o objetivo de gerenciar as contas dos Administradores do sistema.

Para a persistência dos dados vindo da fonte de dados, dá-se destaque ao framework Hibernate[9], que, através de suas bibliotecas, consegue fazer consultas através de *scripts* SQL, permitindo que o IAGO consiga extrair dados de Bancos de Dados Relacionais. De

[6] https://www.java.com
[7] https://spring.io/
[8] https://spring.io/projects/spring-security
[9] https://hibernate.org/

semelhante modo, para que o IAGO consiga extrair dados de arquivos, as bibliotecas do Apache Commons IO[10] foram utilizadas. Quando os dados precisam ser persistidos no Banco de Dados do IAGO, a biblioteca Mongo Java Driver é utilizada. Este, consegue estabelecer uma comunicação comum entre o servidor e o Banco de Dados onde os conjuntos de dados são armazenados.

Por fim, de maneira a se ter todas as tecnologias citadas acima, o IAGO faz uso do Gradle[11]. Através dele, é possível fazer configuração automática do projeto, indicando todas as bibliotecas e suas devidas versões em um único arquivo de configuração. Assim, o Gradle realiza o download de todas as dependências do sistema automaticamente de seus repositórios originais, sem maiores preocupações por parte dos desenvolvedores. Além disso, o Gradle também cuida de executar testes no código fonte, validando a qualidade do mesmo e conferindo se as principais rotinas do sistema estão sendo desempenhadas sem erros.

As principais tecnologias utilizadas no servidor do IAGO podem ser conferidas no Quadro 12.

Quadro 12 – Tecnologias Utilizadas no Servidor do IAGO.

Nome	Descrição	Camada
Java	Linguagem de programação utilizada no servidor do IAGO	Todas
Spring	Framework para padrões de projeto de desenvolvimento	Todas
Spring Security	Framework para autorização e autenticação no servidor	Apresentação
Gradle	Framework para configuração automática de projeto Java	Todas
Junit	Bibliotecas para execução de testes unitários	Manipulação
Hibernate	Framework para persistência em Bancos de Dados relacionais	Extração
Apache Commons IO	Biblioteca para manipulação de arquivos de texto	Todas
Jcraft	Framework para cominicação via SSH[12]	Persistência
Gson	Biblioteca para manipulação de JSON	Manipulação
Jdom	Biblioteca para manipulação de XML	Manipulação
Mongo Java Driver	Biblioteca para comunicação com MongoDB	Persistência

Fonte: O autor (2019).

6.1.3 Tecnologias de armazenamento

Quando se trata de armazenamento dos dados dos conjuntos de dados e as configurações do sistema, o IAGO faz uso do Banco de Dados Não-Relacional MongoDB[13]. Por ser um banco de dados não convencional, a principal proposta do MongoDB é oferecer uma alta disponibilidade dos dados juntamente com uma alta flexibilidade de suas estruturas. Proposta esta que se adapta facilmente inúmeros cenários que o IAGO deve atender. Devido

[10] https://commons.apache.org/proper/commons-io/
[11] https://gradle.org/
[13] https://www.mongodb.com/

à sua abordagem de versionamento de Conjunto de Dados, o IAGO pode por exemplo, necessitar a mudança na estrutura de um Conjunto de Dados em uma atualização. Este tipo de operação é possível ao MongoDB sem que as aplicações necessitem ter suas execuções interrompidas. Além disso, o MongoDB também permite operações de consulta nos dados, permitindo, por exemplo, a aplicação de filtros nos dados.

Apesar de o MongoDB preencher quase todos os requisitos necessários para que o IAGO possa funcionar, por ser um Banco de Dados que armazena seus dados em formato JSON, uma lacuna fica em aberto quando os outros formatos de dados são levados em consideração. Por questões de velocidade de resposta, fica inviável a transformação dos dados em tempo real. Por isso, além de um Banco de Dados, o IAGO faz utilização de um sistema de arquivos que servirão para armazenar as outras distribuições já totalmente processadas, sendo elas, até o determinado momento, CSV[14] e XML[15]. Quando um usuário requer um formato específico, por exemplo o XML, o IAGO acessa o arquivo em uma pasta no sistema operacional e o retorna ao seu requisitante. Para cada Conjunto de Dados, o IAGO cria uma pasta e nela são armazenados os arquivos de distribuições e suas respectivas versões.

6.2 VALIDAÇÃO DO IAGO

Para avaliar o IAGO, foram desenvolvidos casos de teste que buscam verificar se o sistema atende aos requisitos propostos para a implementação de um SGDW. Os casos de testes levados em consideração são detalhados na Seção 6.2.1. Por fim, na Seção 6.2.2 é mostrado um estudo de caso em que o IAGO é utilizado para a publicação de conjuntos de dados no Portal de Dados Abertos da Universidade Federal de Pernambuco.

6.2.1 Casos de teste

Os requisitos listados para a implementação de um SGDW e utilizados como guia para o desenvolvimento do IAGO e de seus respectivos casos de teste incluem (OLIVEIRA, 2017):

- **Requisito 1** - O sistema deve prover mecanismos para criação de conjuntos de dados autodescritivos;
- **Requisito 2** - O sistema deve possibilitar a criação de múltiplas distribuições para um mesmo conjunto de dados, ou seja, a disponibilização dos dados em diferentes formatos;
- **Requisito 3** - O sistema deve prover múltiplos meios de acesso aos conjuntos de dados, que podem ser desde o download de arquivos até o acesso por meio de APIs;

[14] https://www.w3.org/TR/tabular-data-primer/
[15] https://www.w3schools.com/xml/

- **Requisito 4** - O sistema deve prover mecanismos para a criação de canais de comunicação entre os atores do ecossistema de dados na Web;
- **Requisito 5** - O sistema deve garantir o acesso a dados atualizados de acordo com a fonte de origem;
- **Requisito 6** - O sistema deve prover mecanismos para a extração e transformação dos dados de origem em dados na Web;
- **Requisito 7** - O sistema deve prover mecanismos para o gerenciamento de versões de conjuntos de dados;
- **Requisito 8** - O sistema deve prover mecanismos para o gerenciamento de metadados (curadoria de metadados);
- **Requisito 9** - O sistema deve garantir a preservação dos conjuntos de dados ao longo do tempo;
- **Requisito 10** -O sistema deve garantir o uso de identificação única, por meio de URIs, para os conjuntos de dados, distribuições, versões e, preferencialmente, para os itens de cada conjunto de dados.

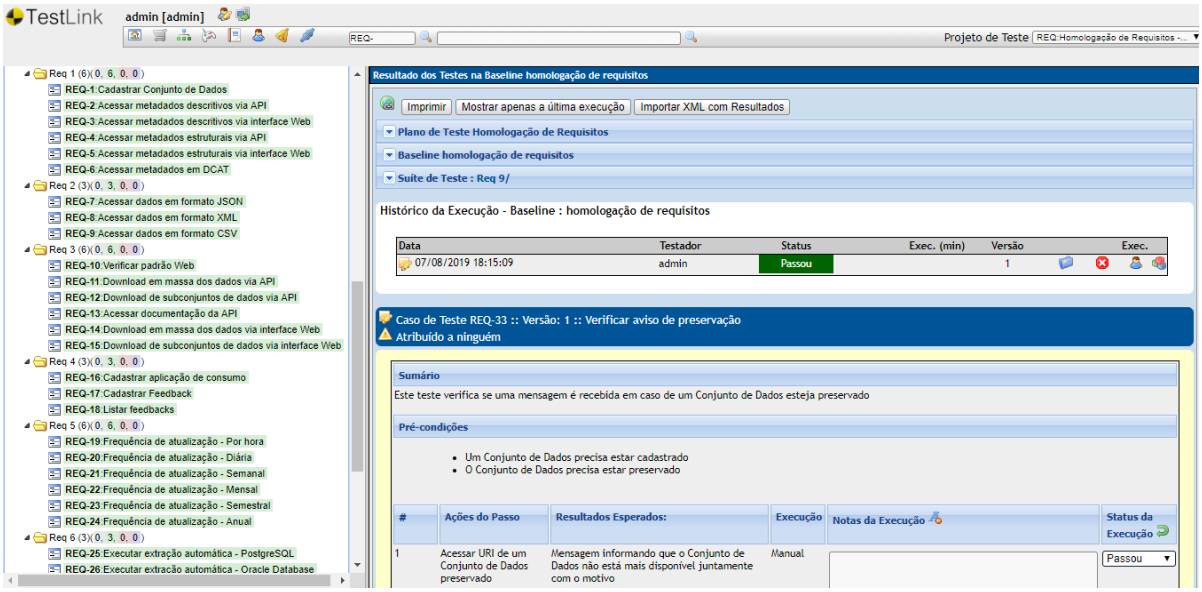
Quadro 13 – Exemplo de Estrutura de Caso de Teste

Identificador	Caso de Teste REQ - XX
Requisito referente	Requisito que o caso de teste busca verificar
Tipo de Execução	Automática ou Manual
Objetivo	Descrição do objetivo do teste
Pré-condições	• Condição um • Condição dois
Passos necessários	1. Passo um 2. Passo dois
Resultado esperado	Descrição do resultado esperado

Fonte: O autor (2019).

A partir deste ponto, os casos de teste seguem o formato que pode ser observado no Quadro 13. Primeiramente, o caso de teste é caracterizado por seu campo de identificação seguido da referência ao requisito que o caso de teste busca verificar. O campo de

Figura 12 – Casos de Teste Executados no Testlink.



Fonte: O autor (2019).

tipo de execução significa a abordagem que o teste deve ser executado, manualmente ou automaticamente. Em seguida, é dada uma descrição do teste juntamente com o passo a passo para a execução do mesmo. Por fim, é mostrado o resultado esperado, caso o recurso testado não apresente falhas. Os casos de teste podem ser conferidos no Apêndice A.

Para a execução de testes classificados como manuais, foram seguidos os passos descritos nos casos de teste, fazendo uso da interface Web do IAGO. Ao fim da execução dos testes manuais, os valores retornados pelo sistema são conferidos. Para execução dos testes de natureza automática, foram utilizadas duas abordagens diferentes. A primeira, voltada para casos de testes que envolvem o consumo de APIs, foi utilizada a ferramenta Postman[16]. Por meio do Postman, é possível personalizar a requisição para a API e receber em detalhes a resposta vinda do servidor. Assim, foi possível monitorar com precisão todo o processo de requisição e resposta, identificando com mais facilidade possíveis falhas. A segunda abordagem de testes de natureza automática, voltados para casos de testes que envolvem processos do sistema que não englobam interferência humana, foi utilizada o JUnit[17]. O JUnuti consiste em uma biblioteca Java que permite a criação de testes a nível unitário. A partir dele, foi possível construir algoritmos que simulam situações específicas no IAGO e monitorar como o sistema se comporta.

A ferramenta TestLink[18] foi usada como solução para o gerenciamento de testes. O uso do Testlink visa facilitar a documentação e execução dos testes, além de assegurar a qualidade do software. Após a execução dos casos de testes listados anteriormente por meio do Testlink, os requisitos que caracterizam o IAGO como um SGDW foram verificados.

[16] https://www.getpostman.com/
[17] https://junit.org/junit5/
[18] http://testlink.org/

A Figura 12 mostra a execução dos testes no Testlink. O Quadro 14 detalha os resultados obtidos, apresentando o identificador do caso de teste, o resultado obtido do mesmo e se a sua execução ocorreu conforme o esperado.

Quadro 14 – Resultado da Execução dos Testes.

Caso de teste	Resultado obtido	Status
REQ-01	Conjunto de dados cadastrado	Passou
REQ-02	Receber metadados descritivos via API	Passou
REQ-03	Receber metadados descritivos via Web	Passou
REQ-04	Receber metadados estruturais via API	Passou
REQ-05	Receber metadados estruturais via Web	Passou
REQ-06	Receber metadados em formato DCAT	Passou
REQ-07	Receber dados em formato JSON	Passou
REQ-08	Receber dados em formato XML	Passou
REQ-09	Receber dados em formato CSV	Passou
REQ-10	Uso de padrão API REST	Passou
REQ-11	Download em massa dos dados via API	Passou
REQ-12	Download de subconjuntos de dados via API	Passou
REQ-13	Documentação da API acessível	Passou
REQ-14	Download em massa dos dados via Web	Passou
REQ-15	Download de subconjunto de dados via Web	Passou
REQ-16	Cadastro de aplicação realizado	Passou
REQ-17	Feedback cadastrado	Passou
REQ-18	Visualizar feedbacks disponíveis	Passou
REQ-19	Atualização dos dados por hora	Passou
REQ-20	Atualização dos dados diariamente	Passou
REQ-21	Atualização dos dados semanalmente	Passou
REQ-22	Atualização dos dados mensalmente	Passou
REQ-23	Atualização dos dados semestralmente	Passou
REQ-24	Atualização dos dados anualmente	Passou
REQ-25	Extração de PostgreSQL feita	Passou
REQ-26	Extração de Oracle feita	Passou
REQ-27	Extração de repositório de arquivos feita	Passou
REQ-28	conjuntos de dados versionados	Passou
REQ-29	Histórico de versões retornado via API	Passou
REQ-30	Histórico de versões retornado via Web	Passou
REQ-31	Metadados alterados via Web	Passou
REQ-32	Conjunto de Dados preservado via Web	Passou
REQ-33	Mensagem de preservação retornada	Passou
REQ-34	URIs únicas	Passou
REQ-35	Identificadores de versão únicos	Passou

Fonte: O autor (2019).

6.2.2 Estudo de caso

O IAGO foi implantado como solução para a publicação de dados abertos em um portal da Universidade Federal de Pernambuco (UFPE), e pode ser acessado em <http://dados.ufpe.br>. A implantação do sistema na UFPE deu-se nas seguintes etapas:

1. Levantamento de requisitos adicionais para implantação do sistema no contexto da UFPE;
2. Desenvolvimento de novas funcionalidades;
3. Execução de testes de software;
4. Correção de erros encontrados;
5. *Release* da primeira versão do Portal de Dados Abertos da UFPE.

Na primeira fase, foram realizadas reuniões juntamente com a equipe do Núcleo e Tecnologia de Informação (NTI) da UFPE. Nas discussões foram acordadas as tecnologias a serem utilizadas nos servidores que viriam a hospedar o IAGO. Como conclusão das negociações, foram disponibilizados dois servidores de sistema operacional Ubuntu[19], o primeiro destinado a um ambiente de testes e o segundo para o ambiente de produção. Ambos os servidores possuíam a tecnologia Java 8[20] instalada juntamente com o banco de dados MongoDB 3.6.13. O Apache Tomcat 8[21] foi utilizado como servidor de aplicação Java para Web. Além disso, foram sugeridas adaptações que deveriam ser realizadas no sistema com o intuito de que o mesmo pudesse atender ao contexto da UFPE. As adaptações sugeridas incluíam:

- Personalização das páginas Web, para que pudessem se encaixar na temática da universidade;
- Inclusão da possibilidade de extrair dados a partir de repositórios de arquivos. Até então, o IAGO realizava extrações apenas de bancos de dados relacionais;
- Pequenas melhorias que viriam a somar à segurança dos dados.

Logo em seguida, a etapa de desenvolvimento adaptou o sistema às demandas vindas do NTI. As páginas de voltadas ao consumo foram personalizadas com a cor, logo e descrições textuais da UFPE e o IAGO passou a extrair dados a partir de repositórios de arquivos via túnel SSH[22]. Com a segunda etapa concluída, o sistema teve sua primeira versão

[19] https://ubuntu.com/
[20] https://www.java.com/pt_BR/download/faq/java8.xml
[21] http://tomcat.apache.org/
[22] https://pt.wikipedia.org/wiki/Secure_Shell

incluída no servidor de testes. Através da ferramenta Sonarqube[23], uma análise profunda de possíveis *bugs*, vulnerabilidades e duplicações no código fonte foi realizada. Além disso, foi feito um levantamento da cobertura do sistema por parte de testes unitários. Como resultado da análise, os principais pontos a serem corrigidos no sistema foram:

- 12 *bugs* encontrados;
- 38 vulnerabilidades;
- Aproximadamente 2 mil *code smells*, ou seja, más práticas de programação que poderiam acarretar em problemas futuros;
- Nenhuma cobertura de testes;
- Problema com processamento de grandes quantidades de dados.

Com os problemas do sistema identificados, a fase de correção foi iniciada. Como resultado dos esforços da equipe de desenvolvimento, os números de *bugs* e vulnerabilidades foram reduzidos a zero. O número de *code smells* também foi reduzido para aproximadamente 50. Por fim, foram desenvolvidos 37 testes unitários que equivalem a 85% de cobertura de testes de sistema, estes que são executados a cada alteração no IAGO com o objetivo de impedir o surgimento de novos erros. Por fim, o problema de processamento de grandes quantidades de dados foi solucionado com a utilização de técnicas de paginação.

Por fim, com todas as outras etapas concluídas, o sistema teve sua primeira versão finalmente disponibilizada no servidor de produção, e sua página inicial pode ser visualizada na Figura 13. A página que lista os conjuntos de dados disponíveis e permite a busca pelos mesmos pode ser conferida na Figura 14. A tela com os detalhes de um Conjunto de Dados pode ser vista na Figura 15, podendo nela ser vistos os metadados descritivos, metadados estruturais e histórico de versões, além de ser possível a realização do download dos dados nos formatos JSON, XML ou CSV.

Com a finalização do processo de implantação do IAGO como principal tecnologia do Portal de Dados Abertos da UFPE, foi possível chegar às seguintes conclusões:

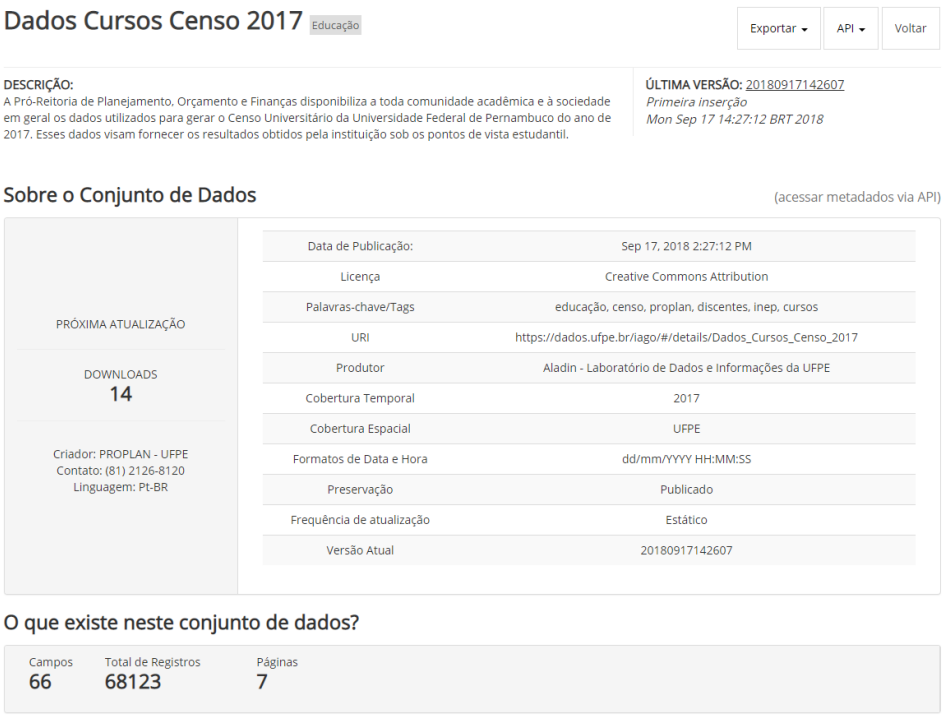
- A alta adaptabilidade do sistema pôde ser comprovada a partir do baixo custo de desenvolvimento para atender às demandas de personalização requeridas pelo NTI;
- Os conjuntos de dados publicados no IAGO atendem a um bom número de boas práticas de dados na Web;
- Os componentes automatizados do sistema tal como a extração automática diminuem o trabalho humano para publicação de dados na Web.

[23] https://www.sonarqube.org/

Figura 13 – Página inicial do Portal de Dados Abertos da UFPE.

Fonte: O autor (2019).

Figura 15 – Tela de apresentação de um Conjunto de Dados do Portal de Dados Abertos da UFPE.



Fonte: O autor (2019).

Figura 14 – Listagem de conjuntos de dados do Portal de Dados Abertos da UFPE.

Fonte: O autor (2019).

6.3 CONSIDERAÇÕES FINAIS

Neste Capítulo foram listadas na Seção 6.1 as principais tecnologias que foram utilizadas no desenvolvimento do IAGO. Em seguida, foi feita uma validação do sistema através de casos de teste que foram construídos a partir dos requisitos propostos no modelo de SGDW na Seção 6.2. Somada a isso, foi mostrada a implantação do IAGO como tecnologia para a publicação de dados em formato aberto para a Universidade Federal de Pernambuco.

7 CONCLUSÃO

7.1 CONSIDERAÇÕES FINAIS

Este trabalho apresentou o IAGO, uma implementação de um Sistema Gerenciador de Dados na Web. O sistema foi desenvolvido levando em consideração os requisitos propostos para um SGDW e as boas práticas de dados na Web do W3C, o DWBP. Dessa maneira, foi construída uma solução capaz de prover mecanismos para um gerenciamento adequado para conjuntos de dados publicados na Web a fim de promover o reúso dos dados.

Uma visão geral da fundamentação teórica foi apresentada, levantando os principais pontos sobre dados na Web para uma melhor compreensão deste trabalho. Foram explanados os conceitos fundamentais sobre o SGDW e do ciclo de vida dos dados na Web. Somadas a isso, as boas práticas de dados na Web foram abordadas. Além disso, uma parte conceitual sobre versionamento foi levada em consideração.

Em seguida, foi feito um levantamento das principais tecnologias disponíveis para compartilhamento de dados na Web. As soluções foram analisadas levando em consideração as boas práticas. Dessa maneira, foi possível identificar os principais desafios ainda não solucionados pelos sistemas analisados.

Neste trabalho também foi proposta uma abordagem para versionamento de conjuntos de dados publicados na Web. A orientação sobre versionamento apresentada visa nortear os seus usuários a quando gerar novas versões dos conjuntos de dados e como descrevê-las. Assim, a abordagem proposta buscou definir os principais metadados de versão, que incluem um *log* de alterações e um identificador de versão.

O IAGO foi apresentado em seguida. Primeiramente, foi dissertado a respeito dos principais diferenciais que o sistema apresenta. Em seguida, sua arquitetura foi detalhada juntamente com os serviços que fazem a composição do sistema. Também foram mostradas as tecnologias utilizadas para o desenvolvimento do IAGO e como as mesmas interagem entre si. Para validar o sistema, um conjunto de casos de testes foram desenvolvidos com intuito de verificar se o IAGO atende a todos os requisitos de um SGDW. Por fim, foi detalhada a implantação do IAGO como tecnologia para a publicação dos dados abertos da Universidade Federal de Pernambuco, abordando os desafios encontrados no processo e como o sistema evoluiu para que o objetivo pudesse ser concluído. Assim, conclui-se que este trabalho conseguiu atingir os seus objetivos, e um SGDW foi desenvolvido.

7.2 TRABALHOS FUTUROS

Como proposta para o direcionamento de futuras pesquisas, foram levados em consideração alguns pontos que ainda serão explorados em outros trabalhos como inclusão de novas funcionalidades ao IAGO de maneira a aumentar o número de boas práticas a se-

rem seguidas. Também é de suma importância que, no futuro, ocorra uma avaliação do IAGO juntamente com potenciais usuários com o intuito de verificar se consegue atender às expectativas dos mesmos. Além disso, é pretendida a inclusão de uma abordagem de refinamento colaborativo para conjuntos de dados no futuro (SANTOS, 2018).

Em questão de versionamento, a abordagem proposta ainda não abrange versões que são geradas de forma incremental, ou seja, que para a compreensão dos dados é necessário consultar as versões em um contexto histórico. Além disso, não foi levado em consideração o custo de armazenamento das múltiplas versões que serão geradas nos conjuntos de dados. Sabe-se, porém, que existem soluções de sistemas de controle de versão para conjuntos de dados que buscam solucionar a questão do armazenamento e que deixa a possibilidade de futuras integrações.

Também é preciso estudar novas abordagens para aumentar a escalabilidade no IAGO. Uma das possíveis soluções é de integração de dois ou mais SGDWs em servidores distribuídos. Também cogita-se o uso de bancos de dados em memória com o intuito de aumentar a velocidade de resposta quando os dados são solicitados. Também será necessário o estudo de abordagens para dados que precisam ser publicados em tempo real, com o objetivo de seguir uma das recomendações do DWBP.

REFERÊNCIAS

BERNERS-LEE, T.; CONNOLLY, D.; SWICK, R. R. Web architecture: Describing and exchanging data. *W3C Note, June*, 1999.

BHARDWAJ, A.; BHATTACHERJEE, S.; CHAVAN, A.; DESHPANDE, A.; ELMORE, A. J.; MADDEN, S.; PARAMESWARAN, A. G. Datahub: Collaborative data science & dataset version management at scale. *arXiv preprint arXiv:1409.0798*, 2014.

BHATTACHERJEE, S.; CHAVAN, A.; HUANG, S.; DESHPANDE, A.; PARAMESWARAN, A. Principles of dataset versioning: Exploring the recreation/storage tradeoff. *Proceedings of the VLDB Endowment*, VLDB Endowment, v. 8, n. 12, p. 1346–1357, 2015.

BIZER, C.; HEATH, T.; BERNERS-LEE, T. Linked data: The story so far. In: *Semantic services, interoperability and web applications: emerging concepts*. [S.l.]: IGI Global, 2011. p. 205–227.

BRADNER, S. Key words for use in rfcs to indicate requirement levels. Havard University, 1997.

CARDNO, A. J.; INGHAM, P. S.; LEWIN, B. A.; SINGH, A. K. *Methods, apparatus and systems for data visualization and related applications*. [S.l.]: Google Patents, 2018.

CHAPMAN, A. D. *Principles and methods of data cleaning*. [S.l.]: GBIF, 2005.

CONRADIE, P.; CHOENNI, S. On the barriers for local government releasing open data. *Government Information Quarterly*, Elsevier, v. 31, p. S10–S17, 2014.

CONSORTIUM, W. W. W. et al. Data catalog vocabulary (dcat). World Wide Web Consortium, 2014.

CRESWELL, J. W. Projeto de pesquisa métodos qualitativo, quantitativo e misto. In: *Projeto de pesquisa métodos qualitativo, quantitativo e misto*. [S.l.: s.n.], 2010.

EASTERBROOK, S.; SINGER, J.; STOREY, M.-A.; DAMIAN, D. Selecting empirical methods for software engineering research. In: *Guide to advanced empirical software engineering*. [S.l.]: Springer, 2008. p. 285–311.

ELMASRI, R.; NAVATHE, S. B. *Database systems*. [S.l.]: Pearson Education Boston, MA, 2011. v. 9.

FRANCONI, E.; GRANDI, F.; MANDREOLI, F. A semantic approach for schema evolution and versioning in object-oriented databases. In: *Computational Logic—CL 2000*. [S.l.]: Springer, 2000. p. 1048–1062.

FROMMHOLD, M.; PIRIS, R. N.; ARNDT, N.; TRAMP, S.; PETERSEN, N.; MARTIN, M. Towards versioning of arbitrary rdf data. In: ACM. *Proceedings of the 12th International Conference on Semantic Systems*. [S.l.], 2016. p. 33–40.

GOLDACRE, B.; GRAY, J. Opentrials: towards a collaborative open database of all available information on all clinical trials. *Trials*, BioMed Central, v. 17, n. 1, p. 164, 2016.

HALL, W.; TIROPANIS, T. Web evolution and web science. *Computer Networks*, Elsevier, v. 56, n. 18, p. 3859–3865, 2012.

JACOBS, B. S.; BODEN-ALBALA, B.; LIN, I.-f.; SACCO, R. L. Architecture of the world wide web. Citeseer, 2004.

KLEIN, M. C.; FENSEL, D. Ontology versioning on the semantic web. In: *SWWS*. [S.l.: s.n.], 2001. p. 75–91.

KUCERA, J.; CHLAPEK, D.; KLÍMEK, J.; NECASKỲ, M. Methodologies and best practices for open data publication. In: *DATESO*. [S.l.: s.n.], 2015. p. 52–64.

LISOWSKA, B. *Metadata for the open data portals*. 2016. <http://juds.joinedupdata.org/wp-content/uploads/2016/12/JUDS-DP6-Metadata-for-the-open-data-portals.pdf>. Accessed: 2019-05-25.

LóSCIO, B. F.; BURLE, C.; CALEGARI, N. *Data on the Web Best Practices*. 2017. Https://www.w3.org/TR/dwbp/. Accessed: 2018-11-13.

LóSCIO, B. F.; OLIVEIRA, M. I. S.; BITTENCOURT, I. I. Publicação e consumo de dados na web: Conceitos e desafios. *Tópicos em Gerenciamento de Dados e Informações (Minicurso - SBBD 2015)*, p. 39–69, 2015.

MARCONI, M. d. A.; LAKATOS, E. M. *Metodologia do trabalho científico: projetos de pesquisa/pesquisa bibliográfica/teses de doutorado, dissertações de mestrado, trabalhos de conclusão de curso*. [S.l.]: São Paulo: Atlas, 2017.

NASSAR, A. A. *Data integration tool*. [S.l.]: Google Patents, 2018. US Patent 9,984,152.

NATH, K.; DHAR, S.; BASISHTHA, S. Web 1.0 to web 3.0-evolution of the web and its various challenges. In: IEEE. *2014 International Conference on Reliability Optimization and Information Technology (ICROIT)*. [S.l.], 2014. p. 86–89.

OLIVEIRA, L. E. R. d. A. *Um modelo de arquitetura para sistemas gerenciadores de dados na Web*. Dissertação (Mestrado) — Universidade Federal de Pernambuco, 2017.

OLIVEIRA, L. E. R. de A.; OLIVEIRA, M. I. S.; LÓSCIO, B. F. Um survey sobre soluções para publicação de dados na web sob a perspectiva das boas práticas do w3c. In: *SBBD*. [S.l.: s.n.], 2017. p. 148–159.

PHILLIPS, S.; SILLITO, J.; WALKER, R. Branching and merging: an investigation into current version control practices. In: ACM. *Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering*. [S.l.], 2011. p. 9–15.

PRESTON-WERNER, T. *Semantic Versioning 2.0.0*. 2010. <https://semver.org/#semantic-versioning-200>.

SANTOS, H. D. A. d. *Uma estratégia para o refinamento colaborativo de dados na web baseada em social coding*. Dissertação (Mestrado) — Universidade Federal de Pernambuco, 2018.

SHAW, M.; GARLAN, D. et al. *Software architecture*. [S.l.]: prentice Hall Englewood Cliffs, 1996. v. 101.

SILVA, K. M. d. *Um modelo de ciclo de vida de dados na web*. Dissertação (Mestrado) — Universidade Federal de Pernambuco, 2019.

SOULES, C. A.; GOODSON, G. R.; STRUNK, J. D.; GANGER, G. R. Metadata efficiency in a comprehensive versioning file system. Carnegie-Mellon University Pittsburg, 2002.

STARR, J.; GASTL, A. Iscitedby: A metadata scheme for datacite. 2011. Disponível em: <https://schema.datacite.org/>.

STONEBRAKER, M.; BRUCKNER, D.; ILYAS, I. F.; BESKALES, G.; CHERNIACK, M.; ZDONIK, S. B.; PAGAN, A.; XU, S. Data curation at scale: The data tamer system. In: *CIDR*. [S.l.: s.n.], 2013.

STRÅLE, J.; LINDÉN, H. *An evaluation of platforms for open government data*. 2014. Disponível em: <http://kth.diva-portal.org/smash/get/diva2:723341/FULLTEXT01.pdf>.

UMBRICH, J.; NEUMAIER, S.; POLLERES, A. Quality assessment and evolution of open data portals. In: IEEE. *2015 3rd International Conference on Future Internet of Things and Cloud*. [S.l.], 2015. p. 404–411.

WANG, Y.; KUNG, L.; BYRD, T. A. Big data analytics: Understanding its capabilities and potential benefits for healthcare organizations. *Technological Forecasting and Social Change*, Elsevier, v. 126, p. 3–13, 2018.

ZHAI, J.; LIANG, Y.; HUANG, L. Linkage discovery of linked data based on ontology reasoning. In: ACM. *Proceedings of the 17th International Digital Government Research Conference on Digital Government Research*. [S.l.], 2016. p. 522–524.

ZHU, N. *Data Versioning Systems*. [S.l.], 2003. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.3651>.

ZUIDERWIJK, A.; JANSSEN, M.; CHOENNI, S.; MEIJER, R.; ALIBAKS, R. S. Socio-technical impediments of open data. *Electronic Journal of e-Government*, v. 10, 2012.

APÊNDICE A – CASOS DE TESTE

Identificador	Caso de Teste REQ - 01
Requisito referente	Requisito 1
Tipo de Execução	Manual
Objetivo	Este caso de teste verifica se o cadastro de um Conjunto de Dados pode ser realizado sem erros
Pré-condições	• Usuário autenticado
Passos necessários	1. Acessar página de administrador do sistema 2. Cadastrar categoria 3. Cadastrar fonte de dados 4. Cadastrar Conjunto de Dados
Resultado esperado	Conjunto de dados cadastrado, valores armazenados corretamente, dados extraídos, dados transformados e APIs criadas

Identificador	Caso de Teste REQ - 02
Requisito referente	Requisito 1
Tipo de Execução	Automática
Objetivo	Este teste verifica se a funcionalidade de acessar os metadados descritivos dos Conjuntos de Dados é possível via API.
Pré-condições	• Pelo menos um Conjunto de Dados deve estar cadastrado
Passos necessários	1. Consumir API referente à recuperação de metadados descritivos de um Conjunto de Dados
Resultado esperado	Receber metadados descritivos em um formato legível por máquinas

Identificador	Caso de Teste REQ - 03
Requisito referente	Requisito 1
Tipo de Execução	Manual
Objetivo	Este teste verifica se a funcionalidade de acessar os metadados descritivos dos Conjuntos de Dados é possível via interface Web.
Pré-condições	• Pelo menos um Conjunto de Dados deve estar cadastrado
Passos necessários	1. Acessar aba de Conjuntos de Dados no sistema 2. Selecionar um COnjunto de Dados e entrar na página de descrições
Resultado esperado	Receber metadados descritivos em um formato legível por seres humanos

Identificador	Caso de Teste REQ - 04
Requisito referente	Requisito 1
Tipo de Execução	Manual
Objetivo	Este teste verifica se a funcionalidade de acessar os metadados estruturais dos Conjuntos de Dados é possível via API.
Pré-condições	• Pelo menos um Conjunto de Dados deve estar cadastrado
Passos necessários	1. Consumir API referente à recuperação de metadados estruturais de um Conjunto de Dados
Resultado esperado	Receber metadados estruturais em um formato legível por máquinas

Identificador	Caso de Teste REQ - 05
Requisito referente	Requisito 1
Tipo de Execução	Manual
Objetivo	Este teste verifica se a funcionalidade de acessar os metadados estruturais dos Conjuntos de Dados é possível via interface Web.
Pré-condições	• Pelo menos um Conjunto de Dados deve estar cadastrado
Passos necessários	1. Acessar aba de Conjuntos de Dados no sistema 2. Selecionar um Conjunto de Dados e entrar na página de descrições
Resultado esperado	Receber metadados estruturais em um formato legível por seres humanos

Identificador	Caso de Teste REQ - 06
Requisito referente	Requisito 1
Tipo de Execução	Automático
Objetivo	Este teste verifica se é possível a recuperação dos metadados em formato DCAT via API.
Pré-condições	• Pelo menos um Conjunto de Dados deve estar cadastrado
Passos necessários	1. Consumir API referente à recuperação de metadados em formato DCAT
Resultado esperado	Receber metadados de um Conjunto de Dados em formato DCAT válido e legível por máquinas

Identificador	Caso de Teste REQ - 07
Requisito referente	Requisito 2
Tipo de Execução	Automática
Objetivo	Este teste verifica se é possível a recuperação dos dados em formato JSON.
Pré-condições	• Pelo menos um Conjunto de Dados deve estar cadastrado
Passos necessários	1. Consumir API referente à recuperação de dados de um Conjunto de Dados em formato JSON
Resultado esperado	Receber dados de um Conjunto de Dados em formato JSON válido

Identificador	Caso de Teste REQ - 08
Requisito referente	Requisito 2
Tipo de Execução	Automática
Objetivo	Este teste verifica se é possível a recuperação dos dados em formato XML.
Pré-condições	• Pelo menos um Conjunto de Dados deve estar cadastrado
Passos necessários	1. Consumir API referente à recuperação de dados de um Conjunto de Dados em formato XML
Resultado esperado	Receber dados de um Conjunto de Dados em formato XML válido

Identificador	Caso de Teste REQ - 09
Requisito referente	Requisito 2
Tipo de Execução	Automática
Objetivo	Este teste verifica se é possível a recuperação dos dados em formato CSV.
Pré-condições	• Pelo menos um Conjunto de Dados deve estar cadastrado
Passos necessários	1. Consumir API referente à recuperação de dados de um Conjunto de Dados em formato CSV
Resultado esperado	Receber dados de um Conjunto de Dados em formato CSV válido

Identificador	Caso de Teste REQ - 10
Requisito referente	Requisito 3
Tipo de Execução	Automática
Objetivo	Este teste verifica o padrão de comunicação em que os dados são disponibilizados.
Pré-condições	• Nenhuma
Passos necessários	1. Enviar requisição via software para o servidor do sistema, e analisar o padrão de requisição
Resultado esperado	Padrão API REST

Identificador	Caso de Teste REQ - 11
Requisito referente	Requisito 3
Tipo de Execução	Automática
Objetivo	Este teste verifica se é possível o download em massa dos dados de um Conjunto de Dados via API
Pré-condições	• Pelo menos um Conjunto de Dados deve estar cadastrado
Passos necessários	1. Consumir API referente à recuperação de dados em massa de um Conjunto de Dados
Resultado esperado	Receber todos os dados disponíveis em um Conjunto de Dados em uma única requisição, em um formato legível por máquinas

Identificador	Caso de Teste REQ - 12
Requisito referente	Requisito 3
Tipo de Execução	Automática
Objetivo	Este teste verifica se é possível o download em massa de um subconjunto de dados via API
Pré-condições	• Pelo menos um Conjunto de Dados deve estar cadastrado
Passos necessários	1. Consumir API referente à recuperação de subconjunto de dados, informando pelo menos um parâmetro como filtro
Resultado esperado	Receber um subconjunto os dados disponíveis em um Conjunto de Dados em uma única requisição, em um formato legível por máquinas

Identificador	Caso de Teste REQ - 13
Requisito referente	Requisito 3
Tipo de Execução	Manual
Objetivo	Este teste verifica se é possível acessar a documentação da API do sistema via interface Web
Pré-condições	• Nenhuma
Passos necessários	1. Acessar página Web referente à documentação da API do sistema
Resultado esperado	Listagem das descrições das APIs, juntamente com seus parâmetros de entrada e de saída

Identificador	Caso de Teste REQ - 14
Requisito referente	Requisito 3
Tipo de Execução	Manual
Objetivo	Este teste verifica se é possível o download em massa dos dados de um Conjunto de Dados via interface Web
Pré-condições	• Pelo menos um Conjunto de Dados deve estar cadastrado
Passos necessários	1. Acessar aba de Conjuntos de Dados no sistema 2. Selecionar um Conjunto de Dados e entrar na página de descrições 3. Selecionar a opção de exportar dados e escolher um formato
Resultado esperado	Receber todos os dados disponíveis em um Conjunto de Dados em uma única requisição

Identificador	Caso de Teste REQ - 15
Requisito referente	Requisito 3
Tipo de Execução	Manual
Objetivo	Este teste verifica se é possível o download de um subconjuntos dos dados de um Conjunto de Dados via interface Web
Pré-condições	• Pelo menos um Conjunto de Dados deve estar cadastrado
Passos necessários	1. Acessar aba de Conjuntos de Dados no sistema 2. Selecionar um Conjunto de Dados e entrar na página de descrições 3. Selecionar botão de usar API 4. Definir parâmetros e realizar a consulta
Resultado esperado	Receber subconjuntos de dados dos dados de um Conjunto de Dados

Identificador	Caso de Teste REQ - 16
Requisito referente	Requisito 4
Tipo de Execução	Manual
Objetivo	Este teste verifica se é possível cadastrar aplicações que informam que fazem uso dos Conjuntos de Dados
Pré-condições	• Pelo menos um Conjunto de Dados deve estar cadastrado
Passos necessários	1. Acessar aba de Conjuntos de Dados no sistema 2. Selecionar um Conjunto de Dados e entrar na página de descrições e selecionar botão de cadastrar nova aplicação 3. Selecionar botão de cadastrar aplicação 4. Preencher formulário e selecionar o botão de envio
Resultado esperado	Cadastro realizado com todos os dados informados

Identificador	Caso de Teste REQ - 17
Requisito referente	Requisito 4
Tipo de Execução	Manual
Objetivo	Este teste verifica se é possível informar o feedback a respeito de um Conjunto de Dados
Pré-condições	• Pelo menos um Conjunto de Dados deve estar cadastrado
Passos necessários	1. Acessar aba de Conjuntos de Dados no sistema 2. Selecionar um Conjunto de Dados e entrar na página de descrições 3. Selecionar botão de informar feedback 4. Preencher formulário e selecionar o botão de envio
Resultado esperado	Feedback cadastrado conforme solicitado

Identificador	Caso de Teste REQ - 18
Requisito referente	Requisito 4
Tipo de Execução	Manual
Objetivo	Este teste verifica se é possível a visualização do feedback disponível para um conjunto de dados de maneira pública
Pré-condições	• Pelo menos um Conjunto de Dados deve estar cadastrado • O Conjunto de Dados precisa conter pelo menos um feedback cadastrado
Passos necessários	1. Acessar aba de Conjuntos de Dados no sistema 2. Selecionar um Conjunto de Dados e entrar na página de descrições
Resultado esperado	Visualizar os feedbacks disponíveis

Identificador	Caso de Teste REQ - 19
Requisito referente	Requisito 5
Tipo de Execução	Automática
Objetivo	Este teste verifica se é possível a atualização automática a partir da frequência de atualização "por hora"
Pré-condições	• Pelo menos um Conjunto de Dados deve estar cadastrado
Passos necessários	1. Executar programa que simula atualização por hora
Resultado esperado	Conjunto de Dados atualizado

Identificador	Caso de Teste REQ - 20
Requisito referente	Requisito 5
Tipo de Execução	Automática
Objetivo	Este teste verifica se é possível a atualização automática a partir da frequência de atualização "diária"
Pré-condições	• Pelo menos um Conjunto de Dados deve estar cadastrado
Passos necessários	1. Executar programa que simula atualização diária
Resultado esperado	Conjunto de Dados atualizado

Identificador	Caso de Teste REQ - 21
Requisito referente	Requisito 5
Tipo de Execução	Automática
Objetivo	Este teste verifica se é possível a atualização automática a partir da frequência de atualização "semanal"
Pré-condições	• Pelo menos um Conjunto de Dados deve estar cadastrado
Passos necessários	1. Executar programa que simula atualização semanal
Resultado esperado	Conjunto de Dados atualizado

Identificador	Caso de Teste REQ - 22
Requisito referente	Requisito 5
Tipo de Execução	Automática
Objetivo	Este teste verifica se é possível a atualização automática a partir da frequência de atualização "mensal"
Pré-condições	• Pelo menos um Conjunto de Dados deve estar cadastrado
Passos necessários	1. Executar programa que simula atualização mensal
Resultado esperado	Conjunto de Dados atualizado

Identificador	Caso de Teste REQ - 23
Requisito referente	Requisito 5
Tipo de Execução	Automática
Objetivo	Este teste verifica se é possível a atualização automática a partir da frequência de atualização "semestral"
Pré-condições	• Pelo menos um Conjunto de Dados deve estar cadastrado
Passos necessários	1. Executar programa que simula atualização semestral
Resultado esperado	Conjunto de Dados atualizado

Identificador	Caso de Teste REQ - 24
Requisito referente	Requisito 5
Tipo de Execução	Automática
Objetivo	Este teste verifica se é possível a atualização automática a partir da frequência de atualização "anual"
Pré-condições	• Pelo menos um Conjunto de Dados deve estar cadastrado
Passos necessários	1. Executar programa que simula atualização anual
Resultado esperado	Conjunto de Dados atualizado

Identificador	Caso de Teste REQ - 25
Requisito referente	Requisito 6
Tipo de Execução	Automática
Objetivo	Este teste verifica se é possível realizar a extração de dados no banco de dados PostgreSQL
Pré-condições	• Um banco de dados PostgreSQL válido deve existir
Passos necessários	1. Executar programa que simula a extração automática
Resultado esperado	Dados extraídos conforme o solicitado

Identificador	Caso de Teste REQ - 26
Requisito referente	Requisito 6
Tipo de Execução	Automátiva
Objetivo	Este teste verifica se é possível realizar a extração de dados no banco de dados Oracle
Pré-condições	• Um banco de dados Oracle válido deve existir
Passos necessários	1. Executar programa que simula a extração automática
Resultado esperado	Dados extraídos conforme o solicitado

Identificador	Caso de Teste REQ - 27
Requisito referente	Requisito 6
Tipo de Execução	Automática
Objetivo	Este teste verifica se é possível realizar a extração de um repositório de arquivos
Pré-condições	• Um repositório de arquivos válido deve existir
Passos necessários	1. Executar programa que simula a extração automática
Resultado esperado	Dados extraídos conforme o solicitado

Identificador	Caso de Teste REQ - 28
Requisito referente	Requisito 7
Tipo de Execução	Automática
Objetivo	Este teste verifica se os Conjuntos de dados estão sendo versionados corretamente
Pré-condições	• Um Conjunto de Dados deve estar cadastrado • O Conjunto e dados deve ter sofrido pelo menos uma atualização
Passos necessários	1. Verificar todas as versões estão disponíveis para consumo através de requisições na API
Resultado esperado	Dados retornados conforme as versões existentes

Identificador	Caso de Teste REQ - 29
Requisito referente	Requisito 7
Tipo de Execução	Automática
Objetivo	Este teste verifica se é possível a listagem do histórico de versões através via API
Pré-condições	• Um Conjunto de Dados deve estar cadastrado • O Conjunto de Dados deve conter pelo menos duas versões
Passos necessários	1. Consumir API referente à listagem do histórico de versões
Resultado esperado	Listagem de versões existentes

Identificador	Caso de Teste REQ - 30
Requisito referente	Requisito 7
Tipo de Execução	Manual
Objetivo	Este teste verifica se é possível a listagem do histórico de versões através via interface Web
Pré-condições	• Um Conjunto de Dados deve estar cadastrado • O Conjunto de Dados deve conter pelo menos duas versões
Passos necessários	1. Acessar aba de Conjuntos de dados no sistema 2. Selecionar um Conjunto de Dados e entrar na página de descrições
Resultado esperado	Listagem de versões existentes

Identificador	Caso de Teste REQ - 31
Requisito referente	Requisito 8
Tipo de Execução	Automática
Objetivo	Este teste verifica se é possível a alteração de metadados de um Conjunto de Dados
Pré-condições	• Um Conjunto de Dados deve estar cadastrado • O usuário deve estar autenticado
Passos necessários	1. Consumir API que permita alterações nos metadados de um Conjunto de Dados
Resultado esperado	Metadados alterados conforme o solicitado

Identificador	Caso de Teste REQ - 32
Requisito referente	Requisito 9
Tipo de Execução	Automática
Objetivo	Este teste verifica se é possível a preservação de um Conjunto de Dados, mantendo seus identificadores ainda armazenados
Pré-condições	• Um Conjunto de Dados deve estar cadastrado • O usuário precisa estar autenticado
Passos necessários	1. Enviar requisição para API responsável pela preservação de um Conjunto de Dados
Resultado esperado	Conjunto de Dados preservado e seus identificadores ainda armazenados

Identificador	Caso de Teste REQ - 33
Requisito referente	Requisito 9
Tipo de Execução	Manual
Objetivo	Este teste verifica se uma mensagem é recebida em caso de um Conjunto de Dados esteja preservado
Pré-condições	• Um Conjunto de Dados deve estar cadastrado • O Conjunto de Dados precisa estar preservado
Passos necessários	1. Acessar URI de Conjunto de Dados preservado
Resultado esperado	Mensagem informando que o Conjunto de Dados não está mais disponível juntamente com o motivo

Identificador	Caso de Teste REQ - 34
Requisito referente	Requisito 10
Tipo de Execução	Automática
Objetivo	Este teste verifica se os Conjuntos de Dados cadastrados sempre apresentarão URIs únicas
Pré-condições	• Dois ou mais Conjuntos de Dados precisam estar cadastrados
Passos necessários	1. Executar programa que verificará se as URIs dos múltiplos Conjuntos de Dados cadastrados no sistema são únicas
Resultado esperado	Confirmação positiva de URIs únicas

Identificador	Caso de Teste REQ - 35
Requisito referente	Requisito 10
Tipo de Execução	Automática
Objetivo	Este teste verifica se cada versão de um Conjunto de Dados possui um identificador único
Pré-condições	• Um Conjunto de Dados precisa estar cadastrado • Duas ou mais versões devem existir no Conjunto de Dados
Passos necessários	1. Executar programa que verifique se os identificadores de versão possuem valores diferentes
Resultado esperado	Resultado positivo para a existência de diferentes identificadores de versão