UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA - CCET
ENGENHARIA DA COMPUTAÇÃO

Breno Baptista Nahuz

Uma abordagem baseada em Engenharia de Dados para extração, transformação e carregamento de dados de instituições acadêmicas

São Luís - MA
2023

Breno Baptista Nahuz

Uma abordagem baseada em Engenharia de Dados para extração, transformação e carregamento de dados de instituições acadêmicas

Trabalho de Conclusão de Curso 2 apresentado ao Curso de Bacharelado em Engenharia da Computação da Universidade Federal do Maranhão como requisito parcial para a obtenção do grau de Bacharel em Engenharia da Computação.

Bacharelado em Engenharia da Computação

Universidade Federal do Maranhão

Orientador: Prof. Dr. Sérgio Souza Costa

São Luís - MA

2023

Breno Baptista Nahuz

Uma abordagem baseada em Engenharia de Dados para extração, transformação e carregamento de dados de instituições acadêmicas

Trabalho de Conclusão de Curso 2 apresentado ao Curso de Bacharelado em Engenharia da Computação da Universidade Federal do Maranhão como requisito parcial para a obtenção do grau de Bacharel em Engenharia da Computação.

Trabalho de conclusão de curso 1. São Luís - MA, 07 de Junho de 2023:

Prof. Dr. Sérgio Souza Costa
Orientador
Universidade Federal do Maranhão

São Luís - MA
2023

"Estou em encantador estado de confusão."

(Ada Lovelace)

Resumo

Um número significativo de instituições de ensino superior públicas já disponibiliza seus dados por meio de portais abertos, sem restrições de acesso. Como resultado, diversos estudos têm explorado a possibilidade de transformar esses dados, que até então refletem apenas o contexto da instituição específica, em dados conectados que possam ser relacionados entre centenas de universidades distintas. O objetivo do trabalho apresentado foi desenvolver uma abordagem baseada em Engenharia de Dados para extração, transformação em dados conectados e carregamento dos dados, em um repositório unificado de maneira automática denominado DBAcademic, seguindo o modelo RDF de dados conectados. Para alcançar tal meta foi desenvolvida uma arquitetura escalável baseada em gerenciadores de fluxo de trabalho, para lidar com as etapas do processo, bancos de dados não relacionais, para agir como uma camada intermediária de armazenamento e portais de acesso para dados conectados com intuito de publicação dos resultados. De modo a garantir um processo prático, foram criadas DAG(*Directed Acyclic Graph*) em tempo de execução a partir de um único arquivo de configuração, tornando o processo modular. Com a aplicação desses processos em 64 instituições públicas de ensino superior foi possível obter um repositório de dados conectados com aproximadamente meio milhão de discentes e sessenta mil docentes. Além disso, também foi possível concluir a falta de disponibilidade dos dados em diversas instituições, somado a impossibilidade de atualização de diversso recursos.

Palavras-chave: Dados abertos, dados conectados, engenharia de dados, ETL, RDF.

Abstract

A significant number of public higher education institutions already make their data available through open portals, with no access restrictions. As a result, several studies have explored the possibility of transforming these data, which until then only reflect the context of the specific institution, into connected data that can be related between hundreds of different universities. The objective of the work presented was to develop an approach based on Data Engineering for extraction, transformation into connected data and data loading, in an automatically unified repository called DBAcademic, following the RDF model of connected data. To achieve this goal, a scalable architecture based on workflow managers was developed to handle the process steps, non-relational databases to act as an intermediate layer of storage and access portals for connected data in order to publish the results. In order to guarantee a practical process, DAG(*Directed Acyclic Graph*) were created at runtime from a single configuration file, making the process modular. With the application of these processes in 64 public institutions of higher education, it was possible to obtain a linked data repository with approximately half a million students and sixty thousand professors. In addition, it was also possible to conclude the lack of data availability in several institutions, in addition to the impossibility of updating several resources.

Keywords: Open Data, linked data, data engineer, ETL, RDF.

Lista de ilustrações

Lista de Códigos

Lista de tabelas

Lista de abreviaturas e siglas

PDA	Portal de Dados Abertos
OGP	Open Government Partnership
OCDE	Organização para Cooperação e Desenvolvimento Econômico
URI	Uniform Resource Identifier
HTTP	Hypertext Transfer Protocol
PDF	Portable Document Format
CSV	Comma-Separated Values
JSON	JavaScript Object Notation
YAML	YAML Ain't Markup Language
UI	User Interface
DAG	Directed Acyclic Graph
ETL	Extract, Transform, Load
ELT	Extract, Load, Transform
TTL	Terse RDF Triple Language
RDF	Resource Description Framework
CCSO	Curriculum Course Syllabus Ontology

Sumário

1 Introdução

O contexto globalizado em que o mundo se encontra hoje não é o mesmo de dez anos atrás e provavelmente não será o mesmo no próximo ano. Essas mudanças vieram principalmente através dos avanços tecnológicos pelos quais a humanidade passou. Contudo, nenhuma dessas mudanças seria capaz sem a quantidade de dados que geramos e que utilizamos diariamente. Estimativas indicam que chegamos ao total de 1000 Exabytes gerados anualmente e com a expectativa de aumentar esse número em 20 vezes nos próximos 5 anos(SOWMYA; SUNEETHA, 2017). Ocorre que ter uma volumetria tão grande não é sempre benéfica, de acordo com Sowmya(2017), esses dados em sua maioria geralmente são pouco estruturados e frequentemente incompletos, devido aos processos em que eles são gerados conterem falhas, ocorre que dados essenciais são frequentemente inacessíveis para os usuários. Logo, precisamos de tecnologias e ferramentas para encontrar, transformar, analisar e visualizar dados para torná-los consumíveis.

Atualmente, estas técnicas e ferramentas são estudadas em uma área de pesquisa denominada de engenharia de dados. Segundo (CRICKARD, 2020), a engenharia de dados é o desenvolvimento, operação e manutenção de infraestrutura de dados, seja no local ou na nuvem (ou híbrida ou multi nuvem), compreendendo bancos de dados e *pipelines* para extrair, transformar e carregar dados.

1.1 Dados Públicos

Na esfera pública existe a responsabilidade de criar normas que estimulem a transparência dos dados que possam ser usados na fiscalização e acompanhamento das ações dos órgãos públicos. No Brasil, mesmo com a previsão estabelecida pela carta magna de 1988 que previa essa garantia, foi promulgada a Lei de Acesso à Informação (Lei n.º 12.527/2011), que trouxe medidas adicionais. Além disso, o Decreto 8.777 de maio de 2016 estabeleceu a obrigatoriedade dos órgãos e entidades da administração pública federal a elaborar e divulgar um Plano de Dados Abertos (PDA). Essas medidas visam proporcionar um ambiente de transparência e facilitar o acesso aos dados públicos. Atualmente diversas instituições possuem o seu Plano de Dados Abertos (PDA), que é o instrumento que institucionaliza uma política de dados abertos dentro de cada instituição. Esse plano tem vigência de dois anos e define as ações que visam a abertura e sustentação de dados abertos nas organizações públicas. Entretanto, alguns autores apontam dificuldades na implantação desta política e também em analisar os dados públicos advindos dessas instituições, seja por não estarem facilmente disponíveis ou por não seguir algum dos princípios de dados abertos (ALBANO; CRAVEIRO; GAFFREE, 2021; MELO et al., 2017; SILVA; MONTEIRO;

REIS, 2020).

1.2 DBAcademic

O principal motivo dessa problemática se dá ao fato de que esses dados têm sido disponibilizados muitas vezes de forma isolada e com formatos e estruturas distintos. Com isso a tarefa de analisar os dados de todas as 108 instituições brasileiras de ensino superior públicas se tornaria praticamente impossível. Tendo isso em vista, (COSTA et al., 2020) demonstraram como a conexão entre esses dados poderiam gerar informações valiosas e muito mais expressivas, favorecendo, consequentemente, a implementação de políticas públicas mais assertivas. Tal objetivo se fez possível graças aos estudos de Berners-Lee (2009), onde o autor demonstrou métodos e boas práticas para a criação de dados conectados. Essa demanda por uma abordagem mais unificada e interoperável é evidente na literatura, onde alguns estudos destacam a necessidade de acesso aos dados dessas instituições (CAROSSI, 2016; GAMA; RODRIGUES, 2016; ZORZAL; RODRIGUES, 2015).

Costa et al. (2020) propuseram um repositório de dados conectados, denominado DBAcademic. Seria possível então, realizar diversas consultas entre todas as instituições federais de ensino, seja relacionar informações de docentes entre todas as universidades federais as quais o mesmo possui algum vínculo ou até mesmo identificar todas as produções que um determinado discente fez com outros estudantes de outras instituições. Na época de sua publicação o estudo de Costa et al. (2020) gerou um repositório de dados conectados contemplando 25 instituições brasileiras de autarquia federal.

Os autores propuseram uma maneira de extrair os dados, transformar em dados conectados e publica-los de forma semiautomática. Logo, caso exista a necessidade de adicionar uma nova instituição o processo seria um pouco trabalhoso, com isso pode-se concluir que a metodologia apresentada contava com algumas limitações:

- Todo o processo era realizado via execução de um *script* em Python através de linha de comando em um terminal.
- Não existe um mecanismo de atualização automática dos dados nem de gerenciamento de execuções.
- O envio para o repositório de dados conectados era feito de forma manual.

1.3 Objetivos

O pressuposto do presente trabalho é aplicar os padrões e boas práticas defendidos por Crickard (2020) em cima da solução proposta por Costa et al. (2020), servindo como

uma extensão ao estudo dos autores, criando a possibilidade de se analisar todas as 108 instituições de ensino superior públicas do país. Deste modo, o objetivo geral deste trabalho é propor uma arquitetura baseada nas ferramentas e técnicas de engenharia de dados para realizar a coleta, a transformação e a população dos dados de todas as instituições federais de ensino em um repositório de dados conectados. Entre os principais desafios, destacam-se:

- Adaptar a metodologia utilizada em (COSTA et al., 2020) para a utilização destas técnicas e ferramentas no projeto DBAcademic.
- Manter a base atualizada, definindo períodos específicos para essa tarefa. No caso de instituições acadêmicas, alguns dados precisam ser atualizados no mínimo semestralmente.
- Reduzir o custo operacional para adicionar mais instituições públicas, dado a totalidade de 108 instituições.
- Atualização dos vocabulários utilizados para descrever os dados conectados.
- Publicar os dados conectados em um repositório unificado.
- Demonstrar uma coleção de novas consultas que passam ser possíveis de ser realizadas dentro deste repositório unificado.

2 Fundamentação

Neste capítulo serão abordados todos os conceitos necessários para validar a necessidade e importância do projeto. Pode encontrar-se também, a motivação do estudo de forma aprofundada, assim como alguns exemplos para a melhor compreensão.

2.1 Dados Abertos

Atualmente, dados, informações digitais que podem ser sensíveis ou não vem se tornando um recurso tão vital quanto qualquer outra commodity ou recurso refinado, uma vez que, em 2011, durante o Fórum Econômico Mundial esse recurso digital foi considerado como o novo petróleo, tal afirmação também é corroborada por Stach (2023): "dados são um tipo de recurso tão importantes para a quarta revolução industrial quanto o petróleo foi para a revolução tecnológica no final do século XIX". Porém, nem sempre esse recurso teve seu acesso garantido de maneira acessível. No final do século XX, em meados de 1995, houve uma das primeiras considerações ao que seriam dados abertos em uma agência científica nos Estados Unidos em documentos que tratavam apenas sobre a divulgação de dados ambientais (CHIGNARD, 2013).

A definição de Dados Abertos, conforme estabelecido pela Open Definition (GROUP, 2015), refere-se a informações que estão disponíveis de forma a permitir sua livre utilização, reutilização e redistribuição, por qualquer indivíduo e para qualquer propósito, sem restrições excessivas. Esses dados devem requerer apenas a devida atribuição às fontes originais ou podem ser compartilhados sob as mesmas licenças. No entanto, conforme definido pela Open Knowledge (FOUNDATION, 2015), devem apresentar três características principais:

- **Disponibilidade e Acesso**: os dados devem estar integralmente disponíveis e com um custo de reprodução razoável, preferencialmente por meio de download na internet. Além disso, os dados devem ser acessíveis de forma conveniente e passíveis de serem modificados.
- **Reutilização e Redistribuição**: os dados devem ser fornecidos em termos que permitam sua reutilização e redistribuição, incluindo a possibilidade de combinação com outros conjuntos de dados.
- **Participação Universal**: todos devem ter a capacidade de usar, reutilizar e redistribuir os dados, sem discriminação em relação a áreas de atuação, pessoas ou

grupos. Restrições "não comerciais"que impediriam o uso "comercial"ou restrições de uso para fins específicos não são permitidas.

Em síntese, a interoperabilidade é a palavra-chave que classifica um conjunto de dados como aberto. Ao garantir que os conjuntos de dados estejam em conformidade com esse princípio, torna-se possível inserir qualquer tipo de informação em diversos sistemas distintos, permitindo a criação de ambientes mais interconectados e, por consequência, promovendo uma interação entre o público e o privado.

2.2 Dados Abertos no contexto governamental

Desde o advento da utilização da tecnologia da informação no setor público no início do século XXI para gerenciamento, controle e principalmente armazenamento de informações públicas, não era um objetivo formal do governo disponibilizar as informações coletadas pelas suas autarquias de forma digital, mesmo com princípio da publicidade(art. 37 da Constituição Federal brasileira de 1988)(BRASIL, 1998), que garante ao cidadão o acesso a informações públicas, o contexto de disponibilização no meio digital ainda era visto de forma nebulosa. Contudo, através da Lei de Acesso à Informação(Lei n.º 12.527/2011) a administração pública passou a ser obrigada, formalmente, a publicar os dados na internet em formatos abertos e processáveis de maneira programática, preceitos definidos por órgãos internacionais, como o Open Government Partnership (OGP) e a Organização para Cooperação e Desenvolvimento Econômico (OCDE) as quais ambos o Brasil faz parte. Com isso, em 2008, diversos Ministros de Ciência e Tecnologia de reuniram em Paris, com o objetivo de desenvolver um guia internacional voltado ao dados advindos de pesquisas (HENNING et al.,2019).

Dessa forma, foi instaurado, por meio Decreto 8.777/2016(BRASIL, 2016), a política de dados abertos do poder executivo federal, a partir deste, todos os órgãos e entidades da administração pública, portanto, também as universidades e institutos federais, deveriam elaborar e publicar um Plano de Dados Abertos(PDA) assim como a criação, manutenção e atualização periódica de inventários e catálogos corporativos de dados, o mesmo definiu que as instituições possuíam 180 dias para o cumprimento do decreto, um prazo caso considerar autarquias que nunca tiveram uma demanda do tipo.

Ao analisar a literatura recente, pode-se notar como é uma demanda latente a necessidade de consulta e utilização dos dados abertos, tanto no ambiente educacional quanto em outros cenários, como citado no Capítulo 1. Na época da resolução, poucos eram os portais de dados que algumas universidades tinham criado por conta própria, atualmente, ainda existe certa dificuldade para atender essa exigência, contudo, à medida que o tempo passa é perceptível a importância cada vez maior que o tema vem recebendo.

2.3 Dados Conectados

Dados conectados revolucionaram a forma como interagimos e exploramos informações na era digital. Berners-Lee (2009) destacou: "dados conectados são sobre vincular estruturas de dados heterogêneas na Web de forma a permitir que máquinas e pessoas explorem toda a informação disponível". Essa abordagem possibilita a interconexão de diferentes conjuntos de dados, promovendo a descoberta de *insights* e o compartilhamento de conhecimento de maneira mais abrangente.

Ao adotar a prática de dados conectados, os benefícios vão além da simples troca de informações. Conforme Berners-Lee (2009) afirmou: "dados conectados abrem caminho para a criação de uma Web de dados, na qual os dados são interligados e podem ser acessados por qualquer pessoa, máquina ou aplicativo". Isso impulsiona a colaboração e a inovação, permitindo que a inteligência coletiva seja aplicada para resolver problemas complexos.

2.3.1 Definição dos dados conectados

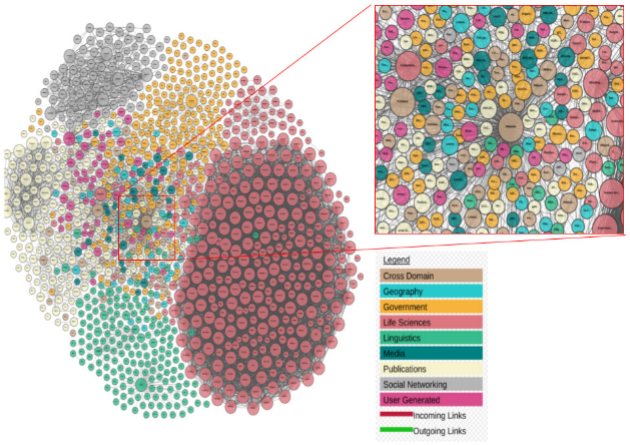
Por definição, dados conectados envolvem a adoção de boas práticas para publicar e interligar conjuntos de dados estruturados na Web, com o objetivo de estabelecer uma Web de Dados(BIZER; HEATH; BERNERS-LEE, 2009). Essa abordagem representa uma evolução do conceito de dados abertos, permitindo que os dados sejam interconectados seguindo algumas normas (BERNERS-LEE, 2010):

- Usar URIs(Uniform Resource Identifier) para identificar catálogos;
- Usar HTTP URIs de forma a possibilitar que as pessoas possam procurar esses recursos na internet;
- Ao realizar uma busca por uma URI, deve-se gerar informações úteis utilizando formatos padrões;
- Sempre incluir conexões para outras URIs para que mais catálogos possam ser acessados.

É necessário frisar que, com os critérios atendidos, é possível cada vez mais iterar e aumentar um banco de dados a nível mundial da internet. Mcrae (2020) demonstra isso na Figura 1, é possível notar cada conexão entre diversas bases de dados pelas linhas que conectam cada nó, que por sua vez representa um recurso de um catálogo.

O centro da imagem se destaca por conter a DBPedia, um projeto que extraiu todos os dados da wikipedia e os deixou disponíveis em formato de dados conectados (CIMIANO et al., 2020). Cimiano et al. (2020) fala sobre como a DBpedia utiliza a Wikipédia como

Figura 1 – Diagrama de uma nuvem LOD em 2017



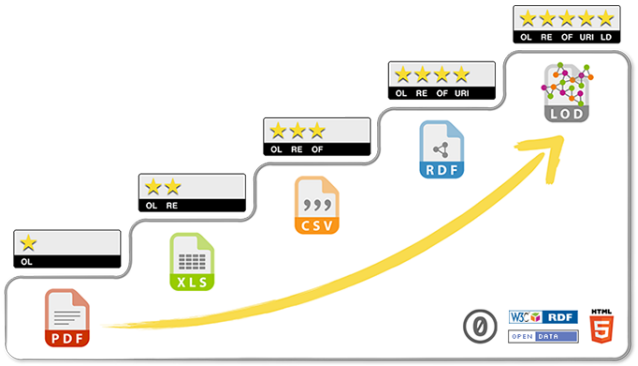
Fonte: (CIMIANO et al., 2020).

um super banco de dados com o objetivo de estruturar e tornar todas as informações contidas ali disponíveis na internet de forma onde possa ser acessado por qualquer outro sistema de consulta.

2.3.2 Classificação dos dados conectados

Seguindo a proposta de Berners-Lee (2010), foi estabelecido um sistema de classificação em cinco estrelas para avaliar o grau de abertura dos dados, sendo que quanto maior o número de estrelas, mais fácil é conectar esses dados como demonstrado na Figura 2.

Figura 2 – Classificação de bases de dados.



Fonte: (BERNERS-LEE, 2010).

Por exemplo, alcançar a primeira estrela requer que os dados estejam disponíveis sob uma licença aberta, mesmo que estejam no formato PDF (Portable Document Format). A segunda estrela pode ser obtida uma vez que os dados estejam disponíveis de modo que sejam estruturados e lidos por máquinas. Uma vez que se enquadrem em formatos não proprietários como CSV(Comma Separated Values) ou JSON(Javascript Object Notation) é possível classificar o conjunto como 3 estrelas. Com todos os fatores anteriores atendidos,

para obter a 4 estrela é necessário que o conjunto de dados use os formatos estabelecidos pela W3C1 e por consequência, a classificação máxima é obtida uma vez que os dados estejam conectados com o maior número de outros dados existentes possível. As siglas que estão abaixo de cada estrela na Figura 2 indicam qual princípio está sendo atendido, no caso do primeiro, OL(*Open License*).

2.3.3 Representação dos dados conectados

Os dados conectados classificam-se como um paradigma a parte para representar informações que se relacionam entre si, são construídos a partir de três princípios (ISOTANI; BITTENCOURT, 2015):

1. Seguir um modelo de dados padrão;
2. Possuir vocabulários de referência;
3. Possibilitar um protocolo padrão de consulta.

Para o modelo de dados padrão e o vocabulário de referência é utilizado, em sua maioria o RDF. O RDF (*Resource Description Framework*) é um modelo de dados simples e padronizado para descrever semanticamente recursos na Web, considerado um dos pilares da Web de dados como comentado na subseção anterior. O mesmo fornece uma estrutura flexível e extensível para descrever recursos, suas propriedades e relacionamentos.

Com este modelo, tem-se a oportunidade de estabelecer a semântica de seus metadados de forma formal, ou seja, de determinar o significado dos elementos de metadados de acordo com suas necessidades específicas de descrição e de maneira que possa ser processada por máquinas.

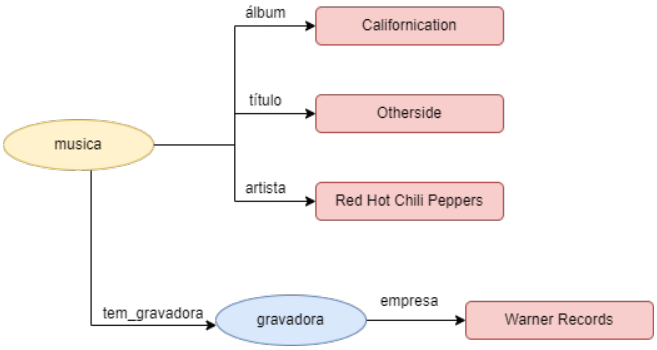
O modelo surgiu de uma inciativa do *World Wide Web Consortium*(W3C) em 1995 como um método para permitir a codificação, o intercâmbio e o reuso de metadados estruturados, com isso é possível obter a interoperabilidade de metadados mediante a concepção de mecanismos que suportem convenções comuns de semântica, sintaxe e de estrutura (MILLER, 2001).

Um documento RDF é estruturado em forma de um grafo orientado, isto é, um conjunto de nós(vértices) que estão ligados por arestas direcionadas(setas). Ambos os elementos são rotulados com identificadores que os distinguem. Não obstante, cabe salientar que esses elementos, juntos, são considerados uma coleção afirmativa. Declaradas por, um sujeito, um predicado e um objeto, também reconhecidas como triplas RDF.

Ao analisar as triplas pode-se definir um dado conectado dependendo do contexto da informação. O sujeito e o objeto representam quais recursos irão se relacionar entre si(os nós do grafo), já o predicado determina qual a natureza daquele relacionamento

direcionado(as arestas do grafo) sempre entre o sujeito ao objeto. Ou seja, o predicado pode-se colocar como sendo uma propriedade intrínseca daquela tripla e um objeto, em algum momento, pode ser um dado literal, como um número, um texto ou até mesmo uma data. Na Figura 3 tem-se uma demonstração exemplificada do exposto.

Figura 3 – Exemplificação de um grafo RDF



Fonte: De autoria própria.

Aqui, percebem-se diversas triplas no grafo, podem ser representadas individualmente seguindo os padrões de uma tripla, por exemplo:

- **Sujeitos:** musica,gravadora.
- **Predicados:** tem_gravadora, álbum, título, artista, empresa.
- **Objetos:** Californication, Otherside, Red Hot Chilli Peppers, Warner Records.

Entretanto, vale ressaltar que o sujeito precisa sempre ser um recurso e estar associado a um identificador único(URI). Na problemática que o presente trabalho se propõe a solucionar os sujeitos são os recursos dos Portais de Dados Abertos, como docentes, discentes e cursos. Suas URIs serão geradas posteriormente com base em informações oriundas dos portais e a instituição em questão.

2.3.4 Formato de Serialização

Após a definição das correspondentes URIs em consonância com a seleção de vocabulários(que serão abordados em detalhes no próximo tópico) é necessário serializar o documento gerado entre diferentes formatos.

Entre os principais tipos de serialização de dados conectados estão:

1. *RDF/XML*(XML).
2. *Terse RDF Triple Language* (TTL),

3. *Notation3*(N3).

4. *N-Triples*(NT).

Logo, como se tem a necessidade de transformar essas informações para dados conectados é necessário uma transformação na serialização daquele documento, por exemplo para o *RDF/XML*, porém este modelo foi o primeiro a ser adotado, logo ele carece de algumas funcionalidades que são vistas em seus concorrentes mais novos, entre eles o *Terse RDF Triple Language*(TTL), o formato ttl consegue ser menos verboso e mais fácil de ser compreendido, melhorando a geração de tuplas. Convém frisar que RDF é um modelo de dados teórico, diferente do TTL que se caracteriza como um formato de serialização. O Código 1 contém um fragmento de documento RDF serializado como TTL.

Código 1 Biblioteca Simpot

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ex: <http://example.org/stuff/1.0/> .

<http://www.w3.org/TR/rdf-syntax-grammar>
  dc:title "RDF/TTL Especificação de Sintaxe (Revisado)" ;
  ex:editor [
    ex:fullname "Breno Nahuz";
    ex:homePage <http://purl.org/net/breno_nahuz/>
  ] .
```

Fonte: De autoria própria.

Como é possível observar, toda a estrutura é definida com base nos prefixos. Estes, por sua vez, são instanciados antes da tripla, uma vez que é necessário serem conhecidos antes. Ainda assim esse documento poderia ser também um arquivo JSON ou XML. A escolha do formato vai estar relacionada ao qual vai ser o caso de uso dessas informações.

2.3.5 Ontologias e Vocabulários

De modo onde existe a criação de um dado sem um padrão de quais regras uma tripla pertença não será possível conecta-lo com outros de seus semelhantes, por conta disso utilizam-se as ontologias e vocabulários para exercer práticas ao se seguir para criação de dados válidos.

Uma ontologia pode ser conceituada como um conjunto de conceitos essenciais e suas inter-relações, que captura a compreensão ou interpretação das pessoas em relação a determinado domínio. Ela viabiliza a representação formal desse entendimento, tornando-o acessível tanto para seres humanos quanto para computadores(KITAMURA; MIZOGUCHI, 2004). Logo, fornece uma estrutura para representar e organizar informações de maneira

coerente e consistente, permitindo a inferência de novos conhecimentos a partir dos dados existentes.

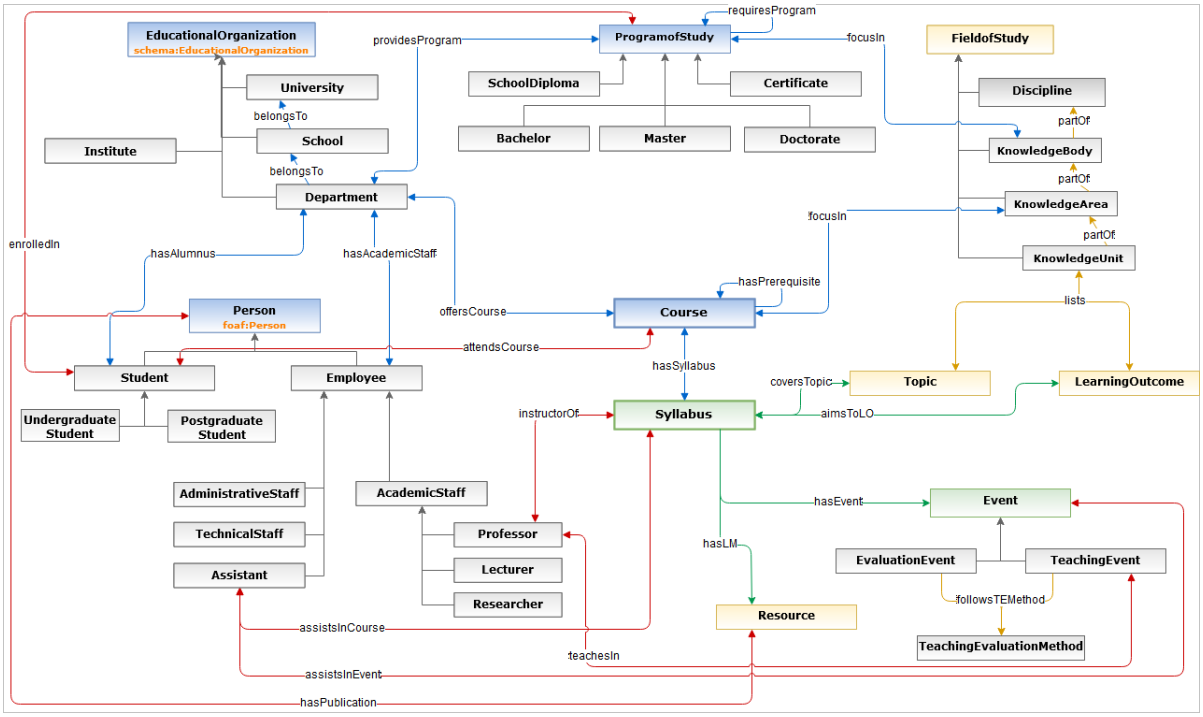
Por outro lado, vocabulários são conjuntos de termos e suas definições, utilizados para rotular e descrever os dados em um domínio específico. Os vocabulários fornecem um conjunto de termos padronizados e reutilizáveis para descrever entidades e propriedades em um domínio específico.

Ao migrar para dados conectados será necessário buscar sempre fazer o reuso de vocabulários, que é uma das melhores práticas para publicação de dados na web descrita em (GREINER et al., 2017).

> Melhor Prática 15: Reuse vocabulários, dando preferência aos padronizados.

No contexto do projeto está sendo selecionado o vocabulário *Curriculum Course Syllabus Ontology* (CCSO) definido por Evengelos Katis(KATIS et al., 2018) e exposto na Figura 4. A escolha deste vocabulário e a modelagem destes dados foram definidas anteriormente, uma vez que estão sendo utilizados também no contexto do projeto DBacademic, enquanto este trabalho desenvolveu a arquitetura para extração, transformação e publicação dos dados.

Figura 4 – Modelo de dados do CCSO



Fonte: (KATIS et al., 2018)

O CCSO se demonstra como um vocabulário muito robusto para a utilização no contexto de instituições de ensino. Os principais sujeitos são relativamente comuns aos dados encontrados nos decorrer do desenvolvimento do projeto, como curso, instituição

e aluno. Essa última em específico vale ser ressaltada, uma vez que demonstra que um vocabulário não esta sujeito a seguir limitações em suas definições. O próprio CCSO utiliza-se de regras encontradas no *Friend of a Friend*(FOAF) que descreve pessoas, suas atividades e suas relações com outras pessoas e objetos.

2.4 Engenharia de Dados

A engenharia de dados é uma disciplina que se concentra na coleta, organização, transformação e análise de dados, com o objetivo de fornecer *insights* valiosos e suportar a tomada de decisões informadas. O termo apareceu pela primeira vez em uma publicação feita por Jim Gray(GRAY; SHENOY, 2000), o autor, apresentava boas práticas sobre o design de sistemas de armazenamentos de dados, contemplava, além do armazenamento também o processamento, os custos e as tendências esperadas de performance levando em consideração o seu custo-benefício. Logo, entende-se que um profissional dessa área deve conseguir construir e manter os sistemas de *pipelines* de dados de uma organização, além de limpar e transformar os dados até que atinjam um estado de utilização favorável (FURBUSH, 2018).

2.4.1 Pipeline de dados

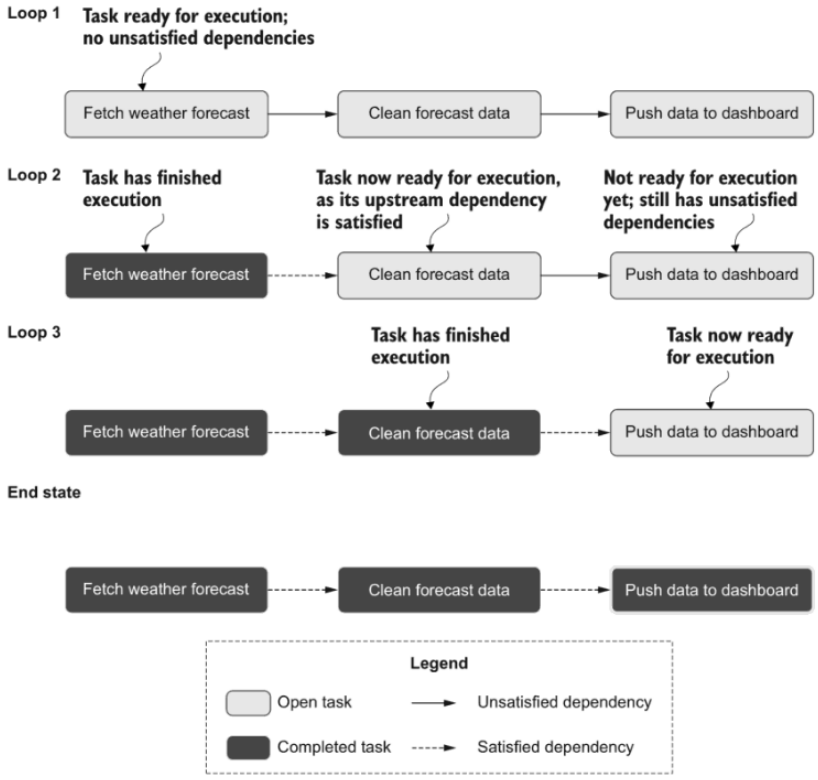
Como é possível notar, as *pipelines* de dados são o centro de todo o contexto em que a engenharia de dados se sustenta. Por definição, uma *pipeline* consiste em diversas tarefas ou ações que precisam ser executadas para atingir um determinado objetivo(HARENSLAK; RUITER, 2021). Tendo em vista que tarefas devem ser executadas em sequência uma forma fácil de lidar com isso, de acordo com com Bas Harenslak é através de grafos, onde a dependência fica explícita entre as linhas que unem tarefas, que são os nós do grafo(HARENSLAK; RUITER, 2021), esse cenário por ser analisado na Figura 5 que é exemplificado uma *pipeline* de dados onde tem-se como objetivo preencher um painel com informações sobre a previsão do tempo, vale frisar a tarefa intermediária de limpeza dos dados antes do carregamento.

É interessante ressaltar que esse grafo é direto, devido a direção das setas que ligam as tarefas mas nada impede de existirem pipelines de dados criados de maneira em que o processo seja acíclico, devido a dependências que podem necessitar de validação, seja cruzada ou não.

2.4.2 Gerenciadores de fluxo de trabalho

Não seria prático e nem confiável executar tarefas de pipelines de dados manualmente, até porque desde os primórdios da computação o problema de executar tarefas sequenciais se mostrou cada vez mais difícil de ser executado com confiabilidade. Através do desenvolvimento

Figura 5 – Pipeline de dados de um sistema de previsão do tempo.



Fonte: (HARENSLAK; RUITER, 2021).

da computação, diversas ferramentas de orquestração de fluxo de trabalho foram surgindo, todas possuem suas diferenças além de seus prós e contras, contudo, geralmente tendem a manter seu foco na execução de tarefas através de grafos.

A Tabela 1 representa algumas das ferramentas de gerenciamento de fluxo de trabalho disponíveis hoje em dia.

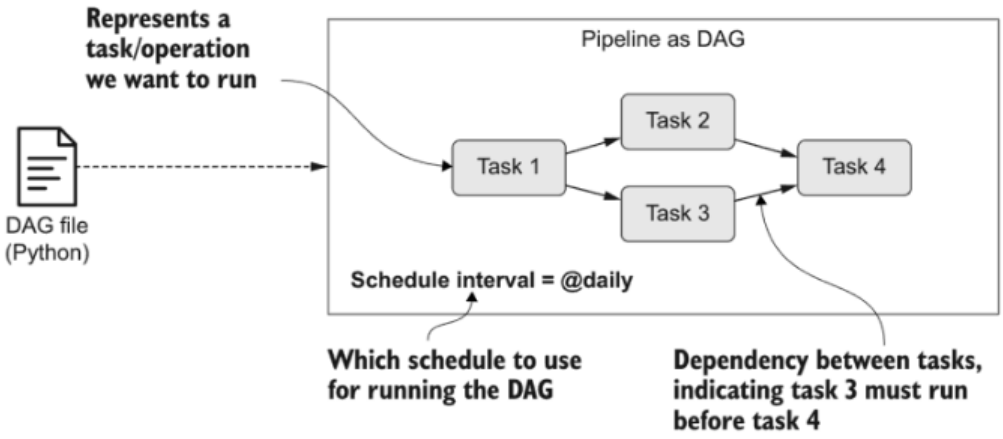
Nome	**Origem**	**Linguagem dos fluxos**	**Escrito em**	**Lançamento**
Airflow	Airbnb	Python	Python	2015
Argo	Applatix	YAML	Go	2022
Azkaban	LinkedIn	YAML	Java	2014
Conductor	Netflix	JSON	Java	2016
Luigi	Spotify	Python	Python	2012
Nifi	NSA	UI	Java	2006

Tabela 1 – Ferramentas de gerenciamento de fluxos de trabalho.

Tendo em vista que atualmente o Airflow é uma das maiores plataformas de orquestração de fluxos de trabalho tanto em *marketshare* quanto ao suporte da comunidade *open-source*, além de que, sua instalação pode ser realizada através de contêineres de aplicação, demonstrando uma variedade enorme dos seus casos de uso em diversos sistemas e sua escalabilidade, ela será a plataforma a ser utilizada para o desenvolvimento deste

trabalho. O Airflow permite toda a definição do fluxo em um código em Python. Logo, pode-se criar todo o fluxo de um Gráfico Direto Acíclico(DAG) a nível de tempo de execução, com isso é possível adicionar elementos dinâmicos na criação de fluxos e até mesmo generalizar sua criação, cabe também salientar que todo DAG pode ser executado conforme um cronograma pré definido, a Figura 6 exemplifica a estrutura de uma DAG dentro do Apache Airflow. Percebe-se, mais uma vez, que a dependência de tarefas é dada pelas setas e as tarefas como nós de um grafo.

Figura 6 – Estrutura de uma DAG no Airflow.



Fonte: (HARENSLAK; RUITER, 2021).

3 Metodologia

O projeto proposto neste estudo tem como objetivo realizar a extração, limpeza, padronização e transformação dos dados abertos de instituições acadêmicas para dados conectados. Por exemplo, dados sobre os discentes, docentes, cursos, projetos de pesquisa, unidades acadêmicas de aproximadamente 80 instituições federais de ensino.Utilizando práticas de engenharia de dados, tais como ELT(Extração, Carregamento, Transformação), gerenciamento de pipelines e processamento distribuído.

Com visto no tópico anterior, será utilizado o Apache Airflow, uma ferramenta de código aberto voltada para a orquestração de *pipelines* (HARENSLAK; RUITER, 2021). O Airflow se concentra na definição de tarefas em estruturas conhecidas como DAGs (Grafos Analíticos Direcionados) criadas com a linguagem de programação Python e exemplificadas na Figura 7. Uma vez com a DAG construída é possível agendar sua execução com base na demanda pretendida, então, à medida que os dados nas instituições forem atualizados, o que ocorre a cada final de semestre, o fluxo será executado de forma automática.

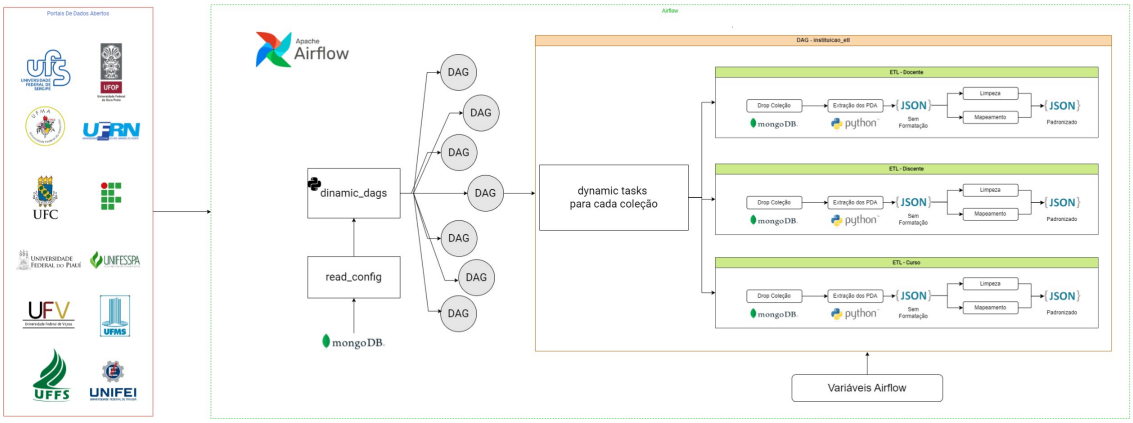
Uma questão importante na modelagem de um pipeline de dados é definir como serão divididas as operações em cada tarefa e estas dentro de cada DAG, um vez que será necessário extrair dados de diversas coleções de inúmeras instituições acadêmicas. Deste modo, a organização proposta é a criação de uma DAG para cada instituição, que irá agregar as operações ELT para cada coleção de dados.

É necessário levar em consideração que cada universidade contará sua própria DAG, uma vez que nem todas as entidades de dados estarão disponíveis para todas as instituições. Entretanto, a manutenção futura desse processo pode ser extremamente morosa, levando em consideração a quantidade de instituições, portanto será desenvolvido um método de geração de DAGs dinâmico, visto que a DAG é criada em tempo de execução, logo, é possível que seja criada com base em parâmetros e os mesmos podem ser armazenados em um local externo, chamaremos esse arquivo de "readconfig.json". A arquitetura sugerida é exposta na Figura 7, como pode ser visto, tanto a geração das DAGs poderá ser dinâmica assim como as tarefas que serão necessárias para cada coleção de cada DAG.

Cabe salientar a necessidade de uma camada de armazenamento para os dados. Optou-se pelo MongoDB Atlas, um banco de dados não relacional com arquitetura baseada em nuvem. Essa escolha foi motivada pelo fato de ser uma solução flexível, capaz de lidar com possíveis alterações na transformação e padronização dos dados brutos, logo também é eficaz para o arquivo readconfig.json.

Cada universidade possui um conjunto de entidades específicas disponíveis para extração. Portanto, faz sentido que cada uma seja tratada como uma coleção dentro do

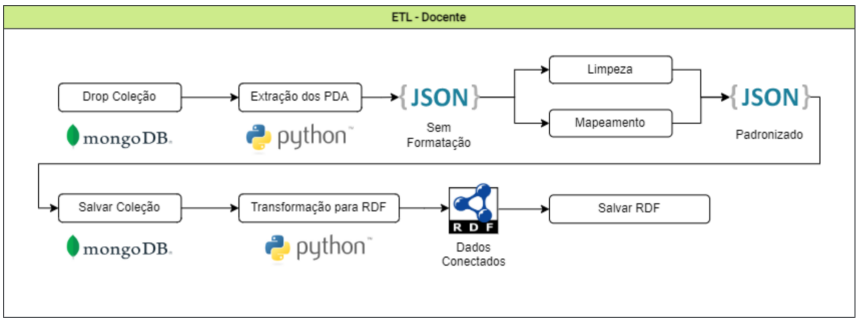
Figura 7 – Arquitetura proposta com o Airflow



Fonte: De autoria própria.

MongoDB, uma coleção é algo próximo do que seria uma tabela em um banco de dados relacional. A Figura 8 detalha como é o fluxo de atualização ou criação de uma coleção dentro do banco de dados respectivo daquela universidade dentro do MongoDB , no caso da Figura 8, para docentes, até a sua disponibilização em plataformas de dados conectados.

Figura 8 – Estrutura de uma DAG para a coleção de docentes



Fonte: De autoria própria.

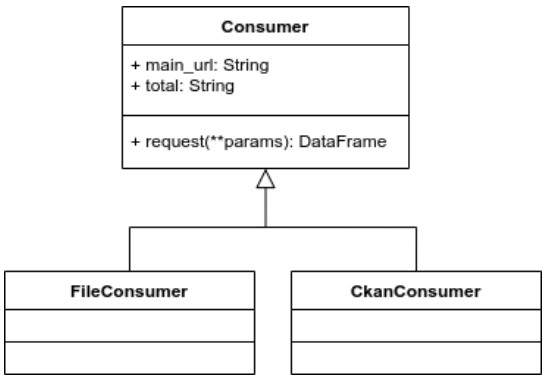
Caso a coleção já exista, ela é apagada, os dados então são extraídos dos portais de dados abertos, gerando um arquivo JSON, o mesmo passa por dois tratamentos de forma simultânea, sua limpeza, onde são desconsiderados datas inválidas, campos vazios e especificidades de acordo com aquela entidade e seu mapeamento, para que todas as coleções de todas as universidades possuam o mesmo padrão estrutural de acordo com a sua entidade. Tendo o arquivo JSON padronizado o mesmo é salvo no MongoDB, em seguida é realizado a transformação para um arquivo TTL, compatível com o padrão RDF de dados conectados. Finalmente, esse arquivo pode tanto ser salvo localmente ou enviado para plataformas de armazenamento em nuvem, bancos de dados para consulta de RDFs ou portais de dados conectados, como o Data World.

3.1 Extraindo os dados

Para todas as insituições consideradas no projeto pode-se notar que existem duas formas principais de acessar os dados disponíveis, por uso de uma API, que significa Interface de Programação de Aplicativos, ou pelo *download* do arquivo com as informações. Uma API funciona como um intermediário que permite que dois programas conversem e troquem informações de maneira organizada e o *download* direto necessita de uma abstração generalista na sua leitura para conseguir exibir os dados. Levando esse cenário em consideração foram construídas duas classes para lidar com a requisição do dados. Ambas possuem a mesma sintaxe para a requisição e são capazes de fazer pequenos ajustes nos dados de acordo com a demanda, por exemplo, caso um conjunto de dados não tenha uma coluna de identificação, as classes de requisição são capazes de gerar uma nova coluna de acordo com o índice do registro, também são capazes de filtrar os dados para não armazenar informações que acabam não sendo relevantes.

Portanto, como existe uma necessidade de expanção futura do projeto, todas as configurações do *Consumer* podem ser resgatadas por qualquer uma das duas classes, respectivamente demonstradas na Figura 9

Figura 9 – Classes implementadas para consumo dos dados



Fonte: De autoria própria.

3.1.1 CKANConsumer

Dentre todas as instituições abordadas neste estudo, a maioria delas implementou seu próprio portal de dados abertos utilizando o CKAN, um sistema de gerenciamento de dados de código aberto. O CKAN disponibiliza APIs que permitem a recuperação programática de informações. No contexto do CKAN, os dados são referenciados como recursos, sendo que a única distinção entre dois endpoints para obtenção de informações sobre as entidades reside no código do recurso. Dessa forma, é possível estabelecer uma generalização para acessar os conjuntos de dados das Instituições de Ensino Superior (IES) de maneira automatizada. Com isso em mente, propõe-se o desenvolvimento de uma

classe com o objetivo de acessar os dados, tendo como parâmetro de instância a URL base da API. Essa classe irá conter um único método denominado *request*, o qual receberá igualmente um único parâmetro obrigatório denominado *endpoint*. A invocação do método *request* resultará na obtenção dos dados no formato JSON.

3.1.2 FileConsumer

Como abordado anteriormente, nem todas as instituições possuem a API do CKAN disponível para consulta, alguns portais até são construidos utilizando a ferramenta, entretanto disponibilizam seus dados apenas através de *download* de arquivos em variados formatos. O único padrão entre insituições que disponibilizam os dados dessa forma é que a URL de *download* faz parte do dóminio em que se encontra a página, assim sendo, cada recurso é referenciado como uma parte da URL de retorno do arquivo. Logo, propõe-se uma classe com o objetivo de adquirir e processar o arquivo, tendo como parâmetro de instância o mesmo do *CKANConsumer*, a URL base do site. Essa classe irá conter, além do único método *request* citado anteriormente com alterações para receber arquivos, também irá possuir um único parâmetro denominado *endpoint*, obrigatório.

3.1.3 Configuração

Como mencionado previamente o arquivo "readconfig.json" será responsável por diversas generalizações e configurações referentes tanto a DAGs, tarefas que as DAGs irão executar e aos recursos que cada instituição irá disponibilizar. O Código 2 representa a estrutura do arquivo JSON para as instituições que utilizam o CkanConsumer e o FileConsumer:

Código 2 Modelo chave-valor para instituições

```
"instituicoes": {
       "ufrn": {
           "consumer": "CkanConsumer",
           "main_url": "https://dados.ufrn.br",
           "colecoes": {
               "docentes" : {"resource_id": "6a8e5461-e748-45c6-aac6-432188d88dde"},
               "discentes": {"resource_id": "a55aef81-e094-4267-8643-f283524e3dd7"},
               "cursos"   : {"resource_id": "a10bc434-9a2d-491a-ae8c-41cf643c35bc"},
           },
       },
       "uff": {
           "consumer": "FileConsumer",
           "main_url": "https://dados.uff.br/",
           "colecoes": {
               "docentes": {
                   "resource"  : "dataset/download/rh.csv",
                   "data_type" : "csv",
                   "query"     : "desc_cargo.str.contains('PROF')",
               },
               "cursos": {
                   "resource"  : "dataset/download/grad-cursos.csv",
                   "data_type" : "csv",
                   "encoding"  : "ISO-8859-1"
               }
           }
       }
```

Fonte: De autoria própria.

Entretando, vale citar que entre as universidades que adotam a opção de disponibilização dos dados por API nem sempre as mesmas coleções estão com todos os parâmetros disponíveis para a criação da Tripla RDF. Também é necessário pontuar as diversas formas das quais os arquivos podem ser lidos pelo *FileConsumer*. Com essa problemática apresentada, as duas classes foram criadas de modo espelhado, onde o que fosse construída em uma pudesse ser reutilizada na outra.

A estrutura apresentada pelo Código 2 será utilizada para todas as IEs contempladas no projeto, uma vez definida a instituição como chave, o seu valor correspondente se dará por meio de um dicionário com os seguintes campos:

- `consumer` : Indicação de qual `consumer` aquela universidade usará para fazer a extração de dados do seu PDA, caso outros padrões existam, como API própria, será referenciado neste campo.
- `mainurl`: URL padrão da API ou ferramenta de disponibilização dos dados abertos daquela instituição.

- `colecoes`: Dicionário com a estrutura de chave-valor referente a entidade e qual seu código ou endpoint específico.

Para cada coleção será instanciado o `Consumer` específico com os parâmetros necessários para extrair aquela coleção, como também é possível notar no Código 2, o *FileConsumer* tem alguns campos além dos que foram apresentados no *CkanConsumer*. Esses campos, e outros, opcionais do dicionário de colecoes, são os:

- `data_type`: Uma string designando o tipo do arquivo(.csv,.xlsx,.json), este que por sua vez, se não for atribuido o tipo irá ser considerado um arquivo .json, exclusivo do FileConsumer.
- `encoding`: Uma string que devera indicar qual padrão de codificação usado ao executar a leitura de caracteres do arquivo, caso não especificado será considerado o padrão UTF-8, exclusivo do FileConsumer.
- `query`: String composta de um padrão de consulta onde podem ser filtrados os resultados de uma extração.
- `index_col`: Nomeclatura dada a coluna que será criada como índice baseado na posição do registro, caso esse parâmetro seja adicionado.

Cabe salientar que, tanto o parâmetro *datatype* quanto o parâmetro *enconding* estão disponíveis apenas para o *FileConsumer* por conta do arquivo advindo do *CkanConsumer* sempre ser do tipo JSON.

3.2 Padronizando

Considerando o cenário no qual o projeto está inserido, caracterizado por diversas fontes de dados provenientes de uma mesma plataforma, observa-se que nem todas as informações seguem o mesmo padrão de armazenamento nos PDAs. Em alguns casos, o rótulo da coluna referente aos nomes dos cursos em sua respectiva coleção pode ser identificado como "cursos", enquanto em outros casos é denominado como "nomecurso". Portanto, torna-se necessário realizar um mapeamento abrangente de todas as possibilidades de nomenclaturas para cada rótulo de cada recurso. Embora esse processo possa demandar um tempo considerável, é essencial dentro de um cenário no qual a informação desempenha um papel fundamental. Além disso, algumas fontes de dados apresentam informações incoerentes, como datas incorretas, exemplificadas por casos como "21/43/3120"ou em formatos diferentes, como "2022-01-23"ou "2016-06-01T14:41:36-03:00". Essas discrepâncias devem ser levadas em consideração e, consequentemente, é necessário filtrar todos os campos de datas para realizar os devidos ajustes e outros casos sejam necessários.

Os referidos processos serão executados de duas formas. No que se refere à formatação, todas as possibilidades relativas aos campos associados a cada entidade serão reduzidas ao mínimo necessário para a geração do arquivo TTL, com o intuito de diminuir a quantidade de variações. Foi proposto, por tanto, uma função que seria executada logo após a obtenção dos dados pelo *Consumer*. Denominada como *normalizer*, essa função recebe de entrada o conjuto de dados da ingestão e os normaliza. Faz-se por meio de substituição os espaços em branco por um sinal de sobrescritro, retirando as acentuações e caracteres especiais do nome da coluna caso houver e garantindo que todas as colunas estivesem com letras minúsculas em seus rótulos.

Para as datas, técnicas similares são utilizadas como a formatação para o padrao de data ISO. Por meio de funções já existentes na biblioteca de tratamento. Esses mapeamentos estarão disponíveis no arquivo "readconfig.json", permitindo a adição de futuras possibilidades.

Neste exemplo abaixo, pode-se notar que para a coleção discentes, entre todas as instituições, todas as possibilidades de nome de colunas estão apresentadas devido a normalização.

Código 3 Trecho de código mostrando o mapeamento

```
{
   "discentes": {
            "nome"          : ["nome", "nome_discente","nome_do_aluno","nome_aluno"],
            "matricula"     : ["ra", "matricula","cod_matricula","_id","id"],
            "sexo"          : ["sexo","sg_sexo","genero"],
            "data_ingresso": ["data_inicio","dt_data_inicio"],
            "codigo_curso" : ["id_curso","co_curso"],
            "nome_curso"    : ["curso","nm_curso","nome_curso"],
        }
}
```

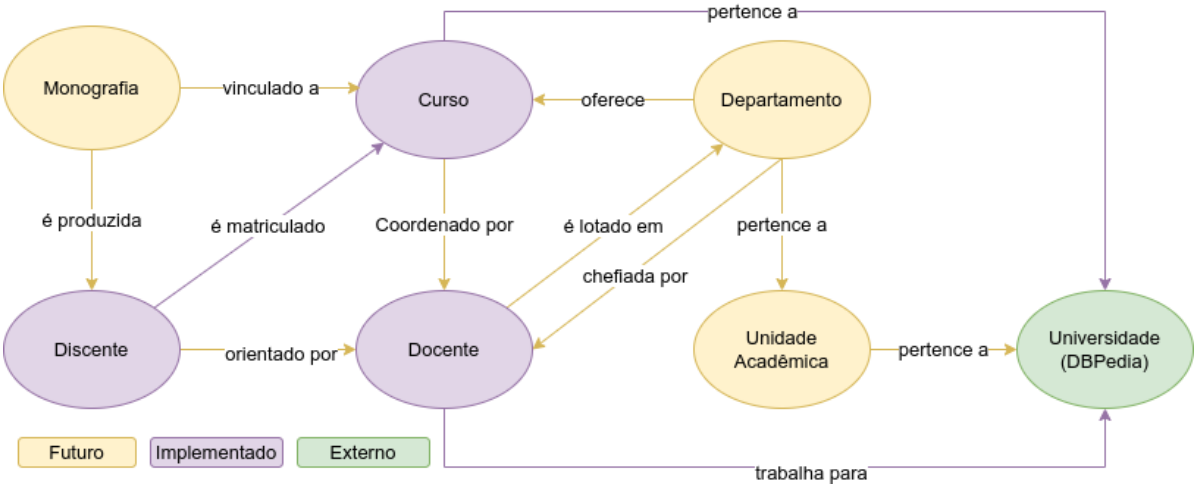
Fonte: De autoria própria.

3.3 Modelagem dos dados

O potencial dos dados conectados, como esperado, está nas suas conexões. Em geral, nas universidades brasileiras os Docentes são lotadas em Departamentos vinculados a Unidades acadêmicas de uma Universidade. Estes departamentos têm um Docente como chefe e ofertam um ou mais Cursos. Nestes Cursos, temos um Docente como coordenador e diversos Discentes matriculados. De modo simplificado, esta descrição é representada pelas conexões destacadas na Figura 10.

Incluir todas as entidades e as ligações previstas no modelo para os dados de todas as instituições demandaria ainda diversos desafios a serem ultrapassados. Os dados

Figura 10 – Modelo de dados do DBAcademic



Fonte: De autoria própria.

encontrados não possuem muitas destas informações, ou demandaria diversos ajustes para encontrar estes dados quando possível.

Seria necessária uma etapa similar a normalização realizada em banco de dados relacionais. Por exemplo, ao invés dos dados terem o nome, modalidade, grau e município do curso, ele deveria ter apenas um código que ligue esta monografia a um dado curso, e as demais informações deveriam estar nos dados do curso. Para conseguir encontrar estes códigos seria necessário processar os dados, dado que geralmente tem apenas um nome. Por exemplo, nos dados de um docente pode ter o nome do departamento. A partir deste nome seria necessário ainda processar todos os dados dos departamentos para encontrar um código a partir do nome.

3.4 Transformação para dados conectados

Antes de transformar para os dados conectados é necessário mapear os atributos para predicados conforme o modelo RDF. Para estes predicados é necessário definir e/ou reusar temos já existentes. Neste trabalho utilizou-se principalmente o vocabulário CCSO (*Curriculum Course Syllabus Ontology*) que é uma ontologia educacional que atua como um modelo de dados para conceitos e entidades em um ambiente acadêmico e foi descrito brevemente na Seção 2.3.5. Esta onotologia reutiliza outras mais genéricas, como:

- *Friend Of A Friend*(FOAF) que é usado para representar o vínculo de pessoas, sejam por informações formais como documentos físicos e digitais ou por relacionamentos não formais.
- *Schema Voabulary*(SCHEMA) que é um vocabulário desenvolvida de modo colaborativo pela comunidade incluindo grandes empresas da web para criar, manter e promover

esquemas para dados estruturados na Internet.

Escolhido os vocabulários a próxima etapa consiste em mapear as entidades e seus atributos para termos de algum vocabulário. Tomando como exemplo a entidade Docente, primeiro temos para qual classe, que neste trabalho ela está sendo mapeada para (**ccso**:**Professor**).

No modelo RDF cada entidade irá se transformar em um recurso e precisará de uma URI única, que no contexto do projeto DBAcademic está sendo usado o seguinte esquema. Todo recurso estará localizado em uma URL persistente com a seguinte base: <https://purl.org/dbacademic/resource#>. Essa URL base será seguida por um código hash que usará algumas informações, como o nome da universidade, da coleção e um identificador daquele recurso. Deste modo será garantido que cada recurso terá uma URL única.

O mapeamento dos atributos para a entidade Docente pode ser apresentado como na Tabela 2. Mapeamento similar foi realizado para as outras duas entidades implementadas neste trabalho.

Tabela 2 – Mapeando atributos dos docentes para termos de vocabulários

Atributos	**Vocabulario**	**Descrição**
Nome	foaf:name	Nome do docente
Matricula	ccso:personID	Matrícula do docente
Sexo	foaf:gender	Gênero do docente
Formação	ccso:hasDegree	Qual formação ou titularidade, como Doutorado, Mestrado, Especialização ou Graduação
Instituição	ccso:worksFor	A URL da instituição que o docente está lotado
E-mail	foaf:mbox	O e-mail do docente

Fonte: De autoria própria.

Com estas informações, é possível transformar para os dados conectados, para isso será utilizada a biblioteca Simple Object-tripe Mapping (SIMPOT) definida e proposta com (COSTA et al., 2020). A vantagem desta biblioteca é ter apenas um ponto em que de modo declarativo é descrito o mapeamento entre os arquivos estruturados como JSON para triplas no modelo RDF, de modo similar a um framework de mapeamento objeto-relacional.

O Código 4 apresenta um exemplo de mapeamento com a biblioteca.

Fonte: De autoria própria.

A execução deste código irá gerar um dado em um formato de serialização, como no Código 5.

Fonte: acervo do autor.

Código 4 Biblioteca Simpot

```
class Docente:
    nome = FOAF.name
    sexo = FOAF.gender
    formacao = CCSO.hasDegree
    instituicao = CCSO.worksFor
    email = FOAF.mbox

    @RdfsClass(CCSO.Professor, "https://purl.org/dbacademic/resource#")
    @BNamespace('ccso', CCSO)
    @BNamespace('foaf', FOAF)
    def __init__ (self, id: str, nome: str, sexo: str, email: str, instituicao: str):
        self.id = id
        self.nome = Literal (nome)
        self.sexo = Literal(sexo)
        self.email = URIRef(email)
        self.instituicao = URIRef(instituicao)

docente = Docente ('1234', 'Sergio Souza Costa', 'M', 'sergio.costa@ufma.br',
↪  'http://pt.dbpedia.org/resource/Universidade_Federal_do_Maranh%C3%A3o')
print (graph(docente).serialize())
```

Código 5 Biblioteca Simpot

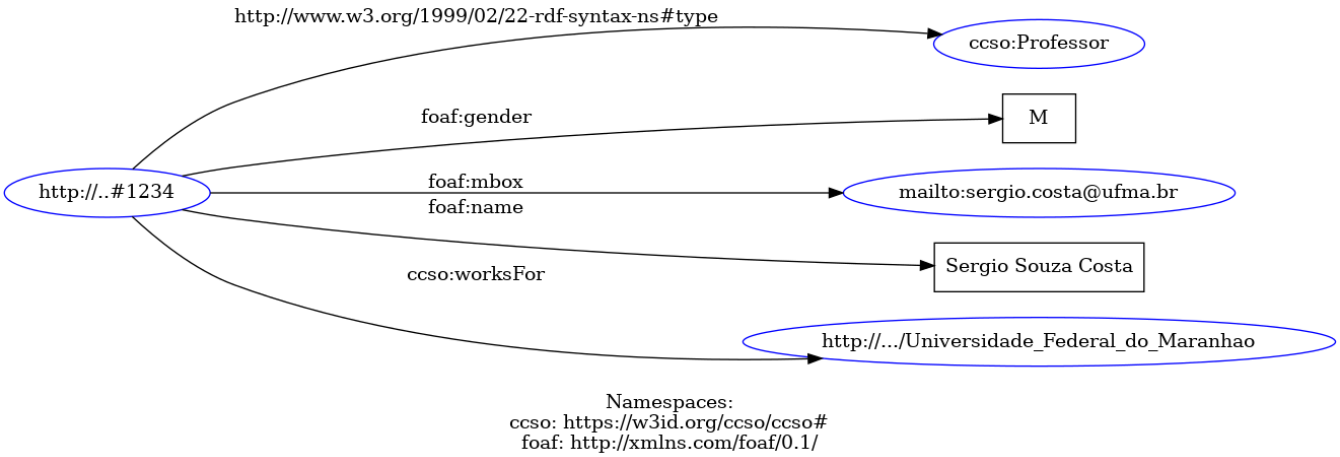
```
@prefix ccso: <https://w3id.org/ccso/ccso#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

<https://purl.org/dbacademic/resource#1234> a ccso:Professor ;
    foaf:gender "M" ;
    foaf:mbox <mailto:sergio.costa@ufma.br>  ;
    foaf:name "Sergio Souza Costa" ;
    ccso:worksFor
    ↪  <http://pt.dbpedia.org/resource/Universidade_Federal_do_Maranh%C3%A3o> .

```

Esse mesmo dado, poderia ser visualizado no formato de grafo como o apresentado na Figura 11.

3.5 Armazenamento e publicação dos dados conectados

Neste trabalho o Apache Airflow está sendo executado localmente, então os dados gerados podem ser salvos localmente ou enviados para um servidor de dados conectados, Google Drive ou portal DataWorld. A ultima opção foi utilizado dado que este portal permite a publicação e acesso aos dados através da linguagem de SPARQL. Essas opções são realizadas por variáveis definidas no Apache Airflow. Para armazenar e publicar os dados no DataWorld, será necessário definir a variável `dw_save` indicando que será salvo do DataWorld. E a variável `dw_settings` específica as informações necessárias para a

Figura 11 – Representação em formato de grafo do dado do Código 5



Fonte: Gerado em <https://www.ldf.fi/service/rdf-grapher>.

publicação dos dados incluindo as credencias, como no Código 6.

Código 6 Exemplo valor da variável dw_setting

```
{
   "token"      : "eyJhbGciOi......",
   "owner"     : "dbacademic",
   "dataset"   : "dbacademic",
}
```

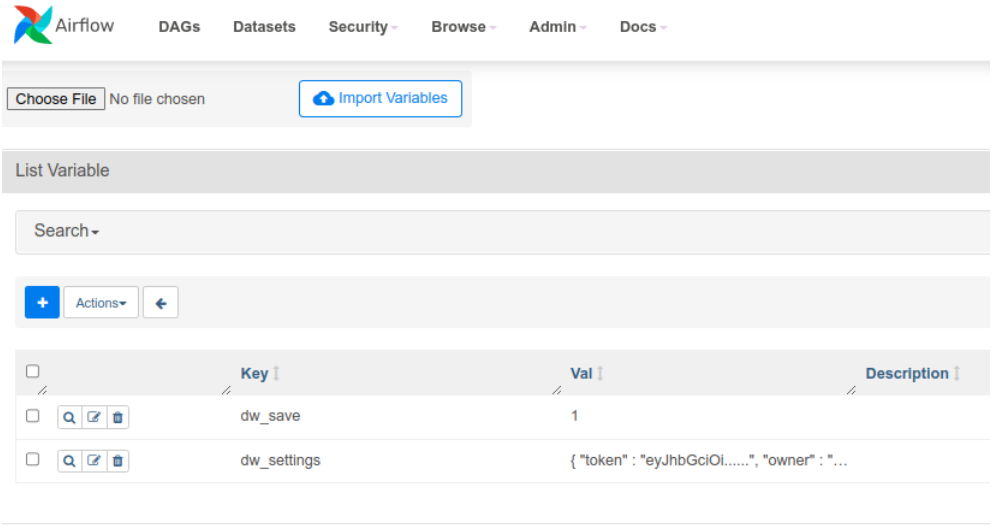
Fonte: De autoria própria.

Essas variáveis podem ser criadas diretamente pela inteface do Apache Airflow Figura 12.

3.6 Execução e Implantação

Para viabilizar a execução do Apache Airflow, é necessário o emprego de uma série de componentes auxiliares. Entre esses componentes, destacam-se o servidor web, essencial para a interação entre humanos e a aplicação; o agendador, um serviço encarregado de agendar as *Directed Acyclic Graphs* (DAGs), que consiste em um processo Python *multithreaded* responsável por determinar quais tarefas devem ser executadas, quando devem ser executadas e onde devem ser executadas; um banco de dados, onde são armazenados os metadados das tarefas e das DAGs, geralmente utilizando o banco de dados relacional PostgreSQL, conforme recomendado pela própria ferramenta; e o executor, um mecanismo que executa as tarefas e pode apresentar diferentes tipos, sendo criado em tempo de execução. O executor é executado no agendador sempre que o Apache Airflow está em operação.

Figura 12 – Variáveis criadas no Apache Airflow



Fonte: De autoria própria.

Diante da variedade de componentes essenciais para o Apache Airflow, sugere-se o desenvolvimento baseado em contêineres Docker, uma tecnologia de virtualização flexível e leve que possibilita a criação e execução de aplicações de maneira isolada em um ambiente compartilhado. Os contêineres encapsulam todos os elementos necessários para o funcionamento de uma aplicação, incluindo código, bibliotecas e dependências, em um pacote portátil. Portanto, ao desenvolver um arquivo Docker Compose, é possível lidar facilmente com as necessidades de escalabilidade do Apache Airflow.

4 Resultados

Este tópico visa apresentar os resultados deste trabalho, com relação à extração, transformação, publicação e consumo e análise dos dados conectados. Além de destacar algumas considerações sobre alguns benefícios das decisões arquiteturais, além dos principais desafios para criar um repositórios dos dados acadêmicos de todas as instituições de ensino federal do Brasil.

4.1 Extração dos dados

Conforme o censo da educação superior, o Brasil possui atualmente 68 universidades federais, 38 Institutos Federais, 02 Centros Federais de Educação Tecnológica (Cefet), a Universidade Tecnológica Federal do Paraná (UTFPR). Totalizando 108 instituições acadêmicas que pertencem à esfera federal e que precisaram se adequar a Política de Dados Abertos do Poder Executivo Federal, instituída pelo Decreto no 8.777/2016. Foram analisadas todas as instituições de ensino superior do Brasil, entretanto, obteve-se o mínimo de informações relevantes de 68 delas (em anexo). Divididas conforme mostra a Tabela 3, as outras 40 instituições não possuíam portais de dados abertos nem estavam catalogadas no portal brasileiro de dados abertos.

Tabela 3 – Ingestões de dados realizadas por tipo de instituição

Dados	**Quantidade**
Universidades	44
Institutos	24
Total de Instituições	68

Fonte: De autoria própria.

Importante destacar que nesse trabalho foi realizada a ingestão e transformação de apenas três coleções de dados: docentes, discentes e cursos. E mesmo assim não foi realizada ainda a extração destas 3 para todas instituições, como apresentado na Tabela 4.Por exemplo, para a coleção de Discentes foram extraídos dados de apenas 22 instituições. Deste modo só foram extraídas 119 coleções ao invés das 192 esperadas.

A decisão primordial em focar os esforços do trabalho apenas nessas coleções se da também por uma validação de prova de conceito. Vale relembrar o que fora citado no Capítulo 2 sobre este estudo ser uma parte estrutural do DBAcademic.

Tabela 4 – Total por coleções

Coleção	**Instituições**
Cursos	48
Discentes	22
Docentes	49
Total	119

Fonte: De autoria própria.

4.2 Publicação dos dados

Conforme relembrado no supracitado, existe o vínculo ao projeto de pesquisa "DBAcademic: Conectando os dados públicos e abertos de instituições públicas de ensino"o referido prevê a publicação destes dados também em um servidor de dados conectados e disponibilizado em um domínio da Universidade do Maranhão. Contudo, atualmente os dados já estão sendo publicados em um repositório público no portal DataWorld[1].

Tabela 5 – Total de triplas por classe

Classe	**Total de Triplas**
`ccso:ProgramofStudy`	10447
`ccso:Student`	481959
`ccso:Professor`	60050
Total	552456

Fonte: De autoria própria.

Após realizar o tratamento de acordo com o que foi citado no Capítulo 3, foi possível enviar os dados ao DataWorld. Plataforma essa que tem por objetivo ser um catálogo de dados conectados. Tendo a possibilidade futura de conectar os dados ali armazenados com outros catálogos. É importante recapitular que também seria possível enviar as triplas para o Google Drive, plataforma de armazenamento de dados em nuvem, ou armazenar localmente os arquivos `.ttl` apenas com mudanças no arquivo de configuração.

4.3 Consumo e Análise dos Dados

Ao finalizar o processo de ingestão e consultar as bases de dados geradas por meio do SPARQL[2] em adição a dados adquiridos e catalogados através de planilhas durante a etapa de desenvolvimento, foi possível tirar algumas informações tanto acerca do processo de ingestão como a exploração dos resultados.

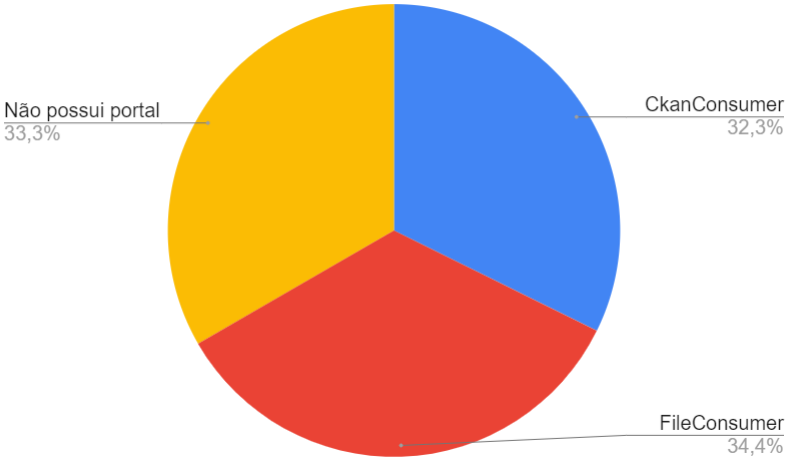
[1] <https://data.world/dbacademic/dbacademic>

[2] No Anexo A são apresentados alguns exemplos de consultas em SPARQL

4.3.1 Consequências da Ingestão

Dado o conjunto de todas as 108 instituições analisadas foi percebida uma distribuição coincidentemente proporcional entre o tipo de `consumer` utilizado, como pode ser analisado na Figura 13 .

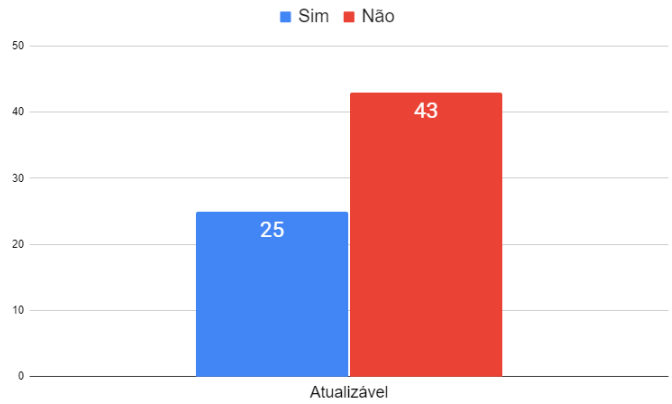
Figura 13 – Gráfico percentual do tipo de consumers por total de instituições



Fonte: De autoria própria.

Um dos pontos importantes a serem levantados no estudo se diz respeito a atualização dos dados. Entende-se que para o projeto ser cada vez mais útil e prático seria conveniente pouca manutenção em origens já catalogadas, ou seja, visto que uma instituição de ensino já foi consolidada no arquivo de configuração não deveria requerer atenção constante. Contudo, dentre todas as universidades abordadas que de fato tiveram seus dados recuperados apenas 25 delas teriam a possibilidade de serem atualizadas.

Figura 14 – Possibilidade de atualização das triplas RDF por instituições



Fonte: De autoria própria.

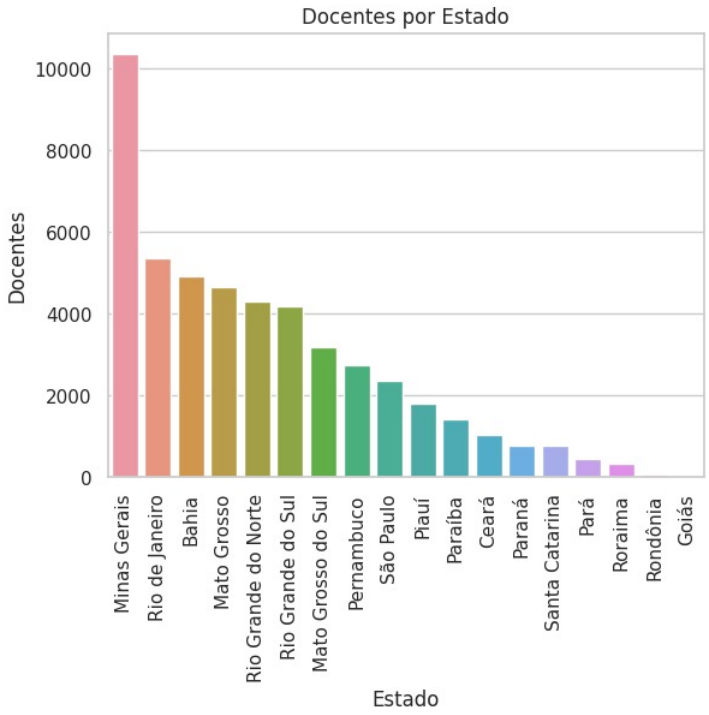
O caso apresentado reporta que 43 das instituições analisadas não possibilitam atualização das suas triplas RDF. Isso ocorre devido a forma com que os portais de dados

abertos foram elaborados, visto que criam diversos recursos para o mesmo tipo de dado. Logo, nesses cenários, todas as vezes em que uma nova leva de dados de determinado tipo entra no portal o mesmo é assimilado como um novo recurso. Por exemplo, a Universidade Federal do Rio Grande do Norte(UFRN) possuí um único código para o recurso de discentes, enquanto a Universidade Federal de São Paulo(UNIFESP) cataloga um recurso diferente de discentes por semestre.

4.3.2 Bases Estratificadas

Conforme as informações obtidas foram sendo analisadas algumas conclusões interessantes acabaram sendo feitas. Ao analisar todos os docentes disponíveis na catalogação foi perceptível a leve vantagem do estado de Minas Gerais(MG) em quantitativo de professores, tal analise pode ser confirmada na Figura 15.

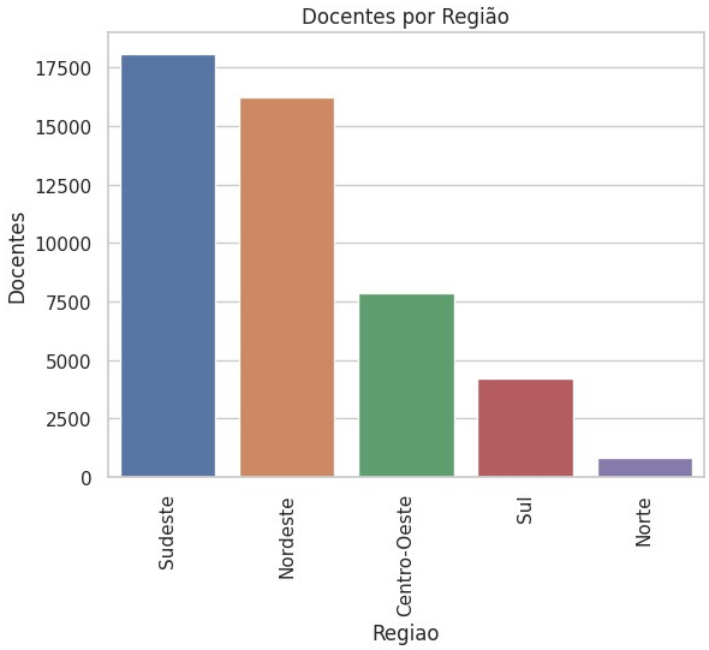
Figura 15 – Docentes por estado



Fonte: De autoria própria.

Entretando, a diferença fica mais acentuada quando se analisado os docentes por região (Figura 16), nesse caso podemos elaborar duas hipóteses, a primeira em que a proporção pode fazer sentido, uma vez que o nordeste é a região que possui a maior quantidade de estados do país, por consequência direta faz sentido possui a maior quantidade de docentes. Porém, a segunda é de que existe a possibilidade de alguma instituição não conter dados apenas de docentes, devido a erros de cadastro dentro do sistema.

Figura 16 – Docentes por região



Fonte: De autoria própria.

No que tange os cursos foi notado um resultado interessante, os 5 cursos com maior oferta nos campus ao redor do país são:

1. Pegadogia: 86
2. Matemática: 82
3. Física: 59
4. Ciências Biológicas: 51
5. Ciências Biológicas: 48

Como pontuação, o Curso de Engenharia da Computação aparece entre os menos ofertados, estando disponível em apenas 11 instituições.

Foi constatado também a fraca quantidade de detalhes quando analisado os dados de discentes. A maioria das instituições não compartilha informações demográficas em seus recursos de discentes, portanto, não há como realizar uma análise aprofundada sobre os discentes coletados, apenas relacionar os mesmos com suas respectivas universidades e, às vezes, com seus cursos.

Vale ressaltar que o trabalho em questão não se propõe em analisar com detalhes as bases coletadas, todos os recursos podem ser encontrados na plataforma DataWorld.

4.4 Execução e Implantação

A execução do projeto se fez por meio do que fora proposto no Capítulo 3. Assim sendo, foi criado um repositório no GitHub[3], onde reside atualmente a versão atual do projeto.

Toda a execução das DAGs e envio de resultados para o DataWorld foi realizada localmente, as variáveis de ambiente do Apache Airflow foram definidas na instância do contêiner.

Uma vez com o ambiente preparado pela plataforma Docker, foi possível executar todas as DAGs referentes as instituições estudadas.

4.5 Considerações sobre a arquitetura e os principais desafios

Notou-se que a arquitetura proposta se mostrou robusta para lidar com escalonamentos de instituições, coleções e mapeamentos distintos. Portanto, pode-se inferir que o que fora construído inicialmente como um simples gerenciamento de processos de ETL para extração de dados tornou-se uma ferramenta poderosa e flexível para serialização de RDFs advindos dos portais de dados abertos de instituições públicas de ensino superior.

Mesmo com o que fora ressaltado sobre a impossibilidade de atualizações da base no que tange algumas instituições, o projeto se mostrou acessível para esse ajuste, uma vez que é necessário apenas trocar o recurso específico daquela coleção.

Em contra partida, entende-se que a arquitetura ainda possa evoluir, tanto no que diz respeito a boas práticas de desenvolvimento contínuo e entrega contínua, tais como abertura do projeto para se tornar um *framework* de código livre e criação dos processos supracitados, como validações de boas práticas de escrita de código, de urls e de informações.

4.5.1 Impacto das decisões arquiteturais na implementação

Uma das melhores escolhas em termos de arquitetura foi possibilitar que o projeto crescesse de forma orgânica com a implantação da criação dinâmica de recursos do Apache Aiflow.

As funções `dynamic_dags` e `dynamic_tasks` aliadas as técnicas de metaprogamação possibilitam diminuir consideravelmente a quantidade de linhas de código necessárias para geração de todas as mais de 60 DAGs. A metaprogamação foi usada principalmente na leitura dos arquivos `.json` e em sua posterior serialização para `.ttl`, sem esse recurso disponível a padronização das tuplas RDF se tornaria um processo extremamente moroso

[3] <https://github.com/LambdaGeo/dbacademic-etl>

para o desenvolvimento e possivelmente até afetaria a performance, visto que, o processo que leva mais tempo para ser executado é o retorno de dados de api do *CkanConsumer*.

Ainda ressaltando a quantidade de linhas de código desenvolvida, a Tabela 6 demonstra o estado atual do projeto, reforçando o que fora afirmado no parágrafo anterior.

Tabela 6 – Estatísticas do Projeto

	Arquivos	**Espaços em branco**	**Comentários**	**Código**
JSON	1	15	0	890
Python	7	186	138	442
YAML	1	10	48	168
Dockerfile	1	0	0	5
Markdown	1	3	0	5
Total	11	214	186	1510

4.5.2 Principais desafios

Diante do exposto sobre os resultados encontrados em adição aos percalços superados, alguns pontos podem ser mais enaltecidos, entre eles:

- **Alta quantidade de rótulos distintos:** Inúmeras possibilidades de nomes de colunas foram encontradas para cada coleção, isso tornou o processo lento, visto que era necessário analisar e testar cada recurso de cada instituição para garantir que o mapeamento estava correto e a biblioteca SIMPOT estava gerando as triplas RDF da forma esperada.

- **Dados que não podem ser atualizados:** Anteriormente essa problemática já tinha sido citada, mas voltar a aparecer como um desafio pelo fato de ser é um caso onde foge do ideal, consequentemente não reproduz com maestria o que uma *pipeline* de dados se propõe a executar, garantir a atualização e preenchimento das bases geradas com base em eventos de tempo.

- **Informações cruciais ausentes:** Diversas vezes foram encontrados casos onde informações essenciais não estavam disponíveis. Entre elas, por exemplo o nome do docente estava ausente ou anonimizado, caso em que não é necessário, pois, como é de conhecimento geral, o nome de todos os funcionários públicos encontra-se no portal da transparência do seu respectivo órgão.

- **Dados semi-disponíveis:** Algumas universidades, entre elas a UFRGS e a UFPR por exemplo, possuem plataformas de visualização de dados própria, contudo, estes não estão disponíveis para *download*, impossibilitando o acesso a essas informações de maneira automática e em alguns casos até não havendo possibilidades para o retorno da informação.

Não obstante, o que fora citado anteriormente reflete apenas a situação atual da base consumida e os pontos elencados não podem ser considerados como problemas perenes do projeto.

5 Conclusões

Conforme o que fora apresentado ao longo do projeto pode-se avaliar a complexidade, diversidade de componentes, configurações e tecnologias envolvidas no desenvolvimento e implementação de um sistema de gerenciamento de fluxo de trabalho baseado no Apache Airflow para realizar a coleta, transformação e armazenamento de dados conectados.

As técnicas de engenharia de dados como a utilização de processos de ETL em união as generalizações de código para a criação de DAGs e tarefas dinâmicas se mostraram processos robustos e complementares na elaboração da arquitetura. Simultaneamente as técnicas abordadas por Costa et al. (2020) conseguiram ser translocadas do seu estudo para este projeto, com isso foi possível aumentar a área de atuação do DBAcademic, permitindo que a cobertura de instituições seja expansível. Com a publicação dos dados conectados em repositórios abertos é possível realizar consultas para extrair resultados pertinentes sobre o cenário de instituições de ensino superior públicas do país.

A adoção de contêineres Docker surge como uma solução eficiente para lidar com a variedade de componentes necessários para o funcionamento do Apache Airflow, permitindo um ambiente isolado e de fácil escalabilidade. A estrutura do arquivo .json para generalizações demonstra uma forma organizada e padronizada de configurar informações relacionadas a diferentes instituições.

Alguns dados interessantes já estão disponíveis nesse trabalho, como exemplo da maioria dos docentes estar localizada na região nordeste e também que existe uma proporção quase que igual entre as instituições que disponibilizam seus dados via API, via arquivo e de instituições que não possuem dados disponíveis ou não tem portais de dados abertos.

Existiram progressos latentes no que diz respeito ao projeto de pesquisa DBAcademic. Com as melhorias implementadas neste estudo foi possível garantir segurança e escalabilidade para o projeto de pesquisa visto que que a atualização dos vocabulários utilizados na ontologia do Curriculum *Course Syllabus Ontology* (CCSO), proporciona uma modelagem precisa devido a criação de recursos com a informações disponíveis no contexto acadêmico.

Por infortúnio, uma das principais metas não pode ser atingida, a garantia da atualização da triplas RDF para todas as instituições a cada semestre, de modo automático. Caso alguns padrões de engenharia de dados tivessem sido seguidos nas elaborações dos Planos de Dados Abertos(PDAs), talvez poderiam ser obtidos resultados mais fidedignos ao estado atual das instituições.

Por fim, o trabalho apresentado ao longo desse texto possui melhorias significativas

a serem implementadas, sejam por projetos herdeiros do DBAcademic ou por futuros pesquisadores que venham a se aventurar em utilizar novas formas de processos de extração de dados utilizando gerenciadores de fluxo de trabalho. É possível sanar a dificuldade que foi encontrada de não obter as atualizações de certas coleções, entre as possibilidades estão um módulo de validação e varredura nos portais de dados verificando os arquivos mais recentes ou então se munir com técnicas de mineração de dados para retirar o *link* do recurso conforme um novo for adicionado.

Referências

ALBANO, C. S.; CRAVEIRO, G. da S.; GAFFREE, J. R. de L. Oferta de dados abertos em universidades federais brasileiras: um estudo dos planos de dados abertos. *Acervo*, v. 34, n. 3, p. 1–18, ago. 2021. Disponível em: <https://revista.an.gov.br/index.php/revistaacervo/article/view/1748>. Citado 2 vezes nas páginas 9 e 10.

BERNERS-LEE, T. *Linked data*. 2009. Disponível em: <https://www.w3.org/DesignIssues/LinkedData.html>. Citado 2 vezes nas páginas 10 e 14.

BERNERS-LEE, T. *Is your Linked Open Data 5 Star?* 2010. Disponível em: <https://www.w3.org/DesignIssues/LinkedData.html>. Citado 2 vezes nas páginas 14 e 15.

BIZER, C.; HEATH, T.; BERNERS-LEE, T. *Linked datathe story so far. Semantic services, interoperability and web applications: emerging concepts (2009), 205–227*. 2009. Citado na página 14.

BRASIL. *Constituição da república federativa do Brasil, 1988*. [S.l.]: Senado Federal, 1998. Citado na página 13.

BRASIL. *Decreto nº 8.777*. 2016. <https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2016/decreto/d8777.htm>. Acesso em: 06 jun. 2023. Citado na página 13.

CAROSSI, D. F. *Ri Ufpe: Dados Abertos: Categorias e Temas prioritários a serem disponibilizados pelas Instituições Federais de Ensino Superior (IFES) AOS cidadãos*. Universidade Federal de Pernambuco, 2016. Disponível em: <https://repositorio.ufpe.br/handle/123456789/20413>. Citado na página 10.

CHIGNARD, S. Paris Tech Review, 2013. Disponível em: <https://www.paristechreview.com/2013/03/29/brief-history-open-data/>. Citado na página 12.

CIMIANO, P.; CHIARCOS, C.; MCCRAE, J. P.; GRACIA, J.; CIMIANO, P.; CHIARCOS, C.; MCCRAE, J. P.; GRACIA, J. Linguistic linked open data cloud. *Linguistic Linked Data: Representation, Generation and Applications*, Springer, p. 29–41, 2020. Citado 2 vezes nas páginas 14 e 15.

COSTA, S. S.; SOUSA, M. V. D.; SILVA, M. L. da; OLIVEIRA, E. C. de; GUIMARÃES, J. V. M. Dbacademic: Conectando os dados abertos das instituições de ensino do brasil. *Ciência da Informação*, v. 49, n. 3, 2020. Citado 4 vezes nas páginas 10, 11, 31 e 43.

CRICKARD, P. *Data Engineering with python: Work with massive datasets to design data models and automate data pipelines using Python*. [S.l.]: Packt Publishing, 2020. Citado 2 vezes nas páginas 9 e 10.

FOUNDATION, O. K. open knowledge foundation, 2015. Disponível em: <http://opendatahandbook.org/guide/en/what-is-open-data/>. Citado na página 12.

FURBUSH, J. *Data engineering: A quick and simple definition*. [S.l.]: O'Reilly Media, 2018. Citado na página 20.

GAMA, J. R.; RODRIGUES, G. M. Transparência e acesso à informação: um estudo da demanda por informações contábeis nas universidades federais brasileiras. *TransInformação*, SciELO Brasil, v. 28, p. 47–58, 2016. Citado na página 10.

GRAY, J.; SHENOY, P. Rules of thumb in data engineering. In: IEEE. *Proceedings of 16th International Conference on Data Engineering (Cat. No. 00CB37073)*. [S.l.], 2000. p. 3–10. Citado na página 20.

GREINER, A.; ISAAC, A.; IGLESIAS, C.; LAUFER, C.; GUéRET, C.; LEE, D.; SCHEPERS, D.; STEPHAN, E. G.; KAUZ, E.; ATEMEZING, G. A.; BEEMAN, H.; BITTENCOURT, I. I.; ALMEIDA, J. P.; DEKKERS, M.; WINSTANLEY, P.; ARCHER, P.; ALBERTONI, R.; PUROHIT, S.; CóRDOVA, Y. *Data on the Web Best Practices (W3C Recommendation)*. 2017. Online. Disponível em: <https://www.w3.org/TR/dwbp/>. Citado na página 19.

GROUP, O. K. O. D. *Open definition*. 2015. Disponível em: <https://opendefinition.org/od/2.1/en/>. Citado na página 12.

HARENSLAK, B. P.; RUITER, J. de. *Data Pipelines with Apache Airflow*. [S.l.]: Simon and Schuster, 2021. Citado 4 vezes nas páginas 20, 21, 22 e 23.

ISOTANI, S.; BITTENCOURT, I. I. *Dados abertos conectados: em busca da web do conhecimento*. [S.l.]: Novatec Editora, 2015. Citado na página 16.

KATIS, E.; KONDYLAKIS, H.; AGATHANGELOS, G.; VASSILAKIS, K. Developing an ontology for curriculum and syllabus. In: *Extended Semantic Web Conference*. [S.l.: s.n.], 2018. Citado na página 19.

KITAMURA, Y.; MIZOGUCHI, R. Ontology-based systematization of functional knowledge. *Journal of Engineering Design*, Taylor Francis, v. 15, n. 4, p. 327–351, 2004. Disponível em: <https://doi.org/10.1080/09544820410001697163>. Citado na página 18.

MELO, K. d. et al. Um estudo voltado à transparência pública através da análise de dados abertos brasileiros. 2017. Citado 2 vezes nas páginas 9 e 10.

MILLER, E. J. An introduction to the resource description framework. *Journal of Library Administration*, Routledge, v. 34, n. 3-4, p. 245–255, 2001. Disponível em: <https://doi.org/10.1300/J111v34n03_04>. Citado na página 16.

SILVA, A. d. A. P.; MONTEIRO, D. A. A.; REIS, A. de O. Qualidade da informação dos dados governamentais abertos: análise do portal de dados abertos brasileiro. *Revista Gestão em Análise*, v. 9, n. 1, p. 31–47, 2020. Citado 2 vezes nas páginas 9 e 10.

SOWMYA, R.; SUNEETHA, K. Data mining with big data. In: IEEE. *2017 11th International Conference on Intelligent Systems and Control (ISCO)*. [S.l.], 2017. p. 246–250. Citado na página 9.

STACH, C. Data is the new oil–sort of: A view on why this comparison is misleading and its implications for modern data administration. *Future Internet*, MDPI, v. 15, n. 2, p. 71, 2023. Citado na página 12.

ZORZAL, L.; RODRIGUES, G. M. Disclosure e transparência no setor público: uma análise da convergência dos princípios de governança. *Informação & Informação*, v. 20, n. 3, p. 113–146, 2015. Citado na página 10.

Anexos

ANEXO A – Exemplo de consultas

Com o uso de expressões regulares (*regex*), a consulta do Código 7 irá trazer tanto os cursos que tenham Engenharias 'De' quanto os 'Da' Computação no nome.

Código 7 Consultando todos os cursos de Engenharia de Computação

```
1  prefix ccso: <https://w3id.org/ccso/ccso#>
2  SELECT ?name ?u
3  WHERE {
4        ?cursos a ccso:ProgramofStudy.
5        ?cursos  ccso:psName ?name.
6        ?cursos ccso:belongsTo ?u.
7        FILTER regex(?name, "ENGENHARIA D. COMPUTAÇÃO", "i" )
8   }
```

Fonte: De autoria própria.

Essa consulta pode ser executada diretamente na interface do DataWorld, como mostra a Figura 17. Os resultados são apresentados no formato de tabela, onde as variáveis `?name` e `?u` serão apresentadas como colunas.

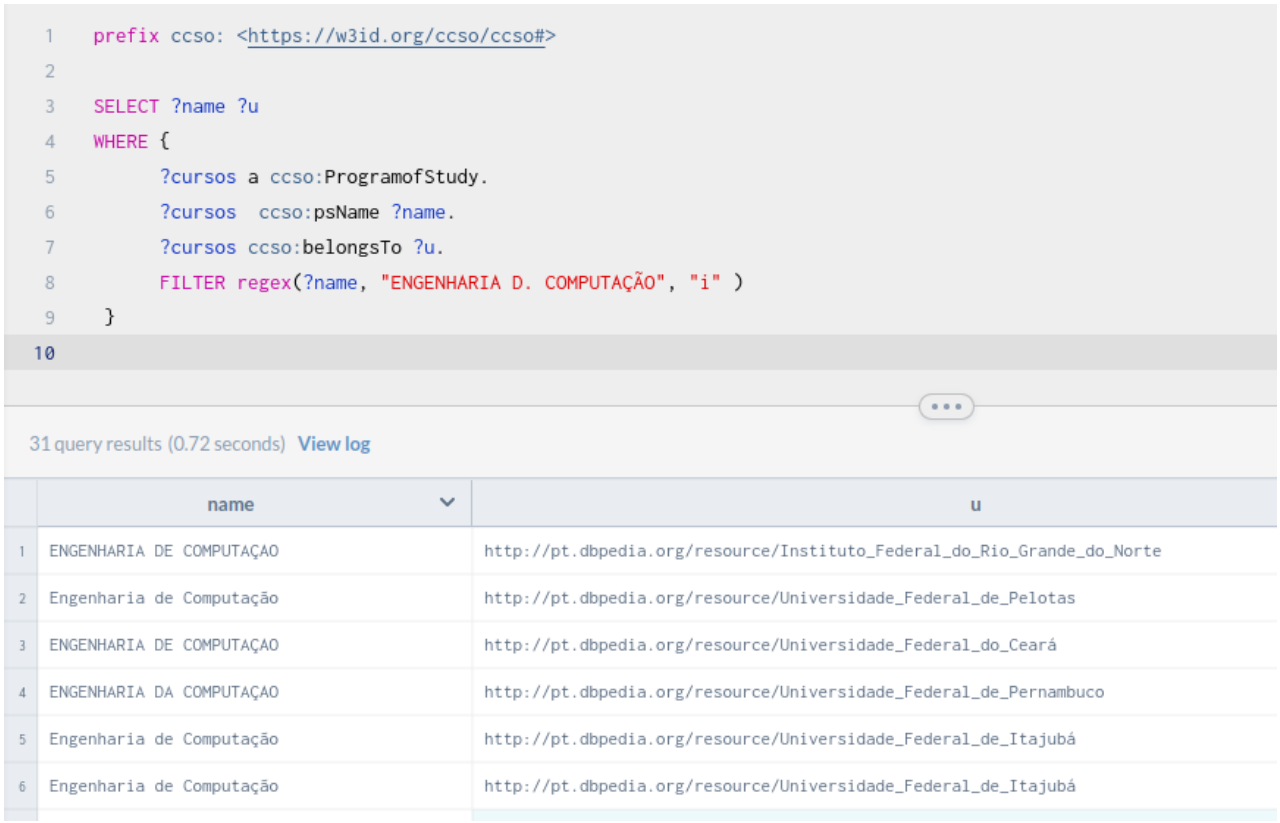
O portal DataWorld provê uma biblioteca em Python denominada datadotworld (<https://github.com/datadotworld/data.world-py>). Esta biblioteca é integrada com a biblioteca para análise de dados Pandas (<https://pandas.pydata.org/>). O resultado da consulta é um Dataframe, podendo assim usar todos os recursos desta biblioteca. A mesma consulta poderia ser executada em Python como no Código 8.

Código 8 Consultando a quantidade de docentes por formação

```
1
2  import datadotworld as dw
3  sparql = '''
4  prefix ccso: <https://w3id.org/ccso/ccso#>
5  SELECT ?name ?u
6  WHERE {
7        ?cursos a ccso:ProgramofStudy.
8        ?cursos  ccso:psName ?name.
9        ?cursos ccso:belongsTo ?u.
10       FILTER regex(?name, "ENGENHARIA D. COMPUTAÇÃO", "i" )
11   }
12 '''
13 results = dw.query('dbacademic/dbacademic', sparql, query_type='sparql')
14 df = results.dataframe
```

Fonte: De autoria própria.

Figura 17 – Executando o Código 7



```
prefix ccso: <https://w3id.org/ccso/ccso#>

SELECT ?name ?u
WHERE {
      ?cursos a ccso:ProgramofStudy.
      ?cursos  ccso:psName ?name.
      ?cursos ccso:belongsTo ?u.
      FILTER regex(?name, "ENGENHARIA D. COMPUTAÇÃO", "i" )
 }

```

31 query results (0.72 seconds) View log

	name	u
1	ENGENHARIA DE COMPUTAÇAO	http://pt.dbpedia.org/resource/Instituto_Federal_do_Rio_Grande_do_Norte
2	Engenharia de Computação	http://pt.dbpedia.org/resource/Universidade_Federal_de_Pelotas
3	ENGENHARIA DE COMPUTAÇAO	http://pt.dbpedia.org/resource/Universidade_Federal_do_Ceará
4	ENGENHARIA DA COMPUTAÇAO	http://pt.dbpedia.org/resource/Universidade_Federal_de_Pernambuco
5	Engenharia de Computação	http://pt.dbpedia.org/resource/Universidade_Federal_de_Itajubá
6	Engenharia de Computação	http://pt.dbpedia.org/resource/Universidade_Federal_de_Itajubá

Fonte: De autoria própria.

No Código 9 é retornado em ordem descendente a quantidade de docentes por formação. Este é um dado interessante, contudo como discutido no Capítulo 4, este dado não está presente em todas as bases.

Código 9 Consultando a quantidade de docentes por formação

```
prefix CCSO: <https://w3id.org/ccso/ccso#>
PREFIX dbp: <http://pt.dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>
prefix foaf: <http://xmlns.com/foaf/0.1/>

SELECT  ?formacao (COUNT(distinct ?s) AS ?qtDocente) where {
    ?s a CCSO:Professor.
    ?s ccso:hasDegree ?formacao.
    ?s foaf:name ?nomep.
}
GROUP BY ?formacao
ORDER BY  DESC (?qtDocente)

```

Fonte: De autoria própria.

No Código 10 é retornado em ordem descendente a quantidade de cursos por instituição acadêmica.

Código 10 Consultando a quantidade de cursos por instituição acadêmica

```
1
2  prefix ccso: <https://w3id.org/ccso/ccso#>
3
4  SELECT ?u (COUNT(distinct ?curso) AS ?qtcursos)
5
6  WHERE {
7        ?curso a ccso:ProgramofStudy.
8        ?curso ccso:belongsTo ?u
9
10  }
11
12 GROUP BY ?u
13 ORDER BY DESC (?qtcursos)
14
```

Fonte: De autoria própria.

No Código 11 é retornado em ordem descendente a quantidade de cursos por nome.

Código 11 Consultando a quantidade de cursos por instituição acadêmica

```
1
2  prefix ccso: <https://w3id.org/ccso/ccso#>
3
4  SELECT ?name (COUNT(distinct ?cursos) AS ?qtcursos)
5
6  WHERE {
7        ?cursos a ccso:ProgramofStudy.
8        ?cursos  ccso:psName ?name.
9
10  }
11
12 GROUP BY ?name
13 ORDER BY DESC (?qtcursos)
14
```

Fonte: De autoria própria.

No Código 12 é um exemplo importante, pois mostra como podemos integrar os dados conectados em diferentes repositórios. No repositório do DBAcademic não tem informações sobre as universidades, como nome, estado, reitor, entre outras. Mas é possível conseguindo estas informações dado que tem a URL de cada universidade que as definem no repositório DBPedia.

Código 12 Buscando a informação do nome do estado no DBPedia

```
1
2  prefix CCSO: <https://w3id.org/ccso/ccso#>
3  PREFIX dbp: <http://pt.dbpedia.org/property/>
4  PREFIX dbo: <http://dbpedia.org/ontology/>
5  prefix foaf: <http://xmlns.com/foaf/0.1/>
6
7  SELECT ?nomep ?nomeuniv ?nomestate where {
8      ?s a CCSO:Professor.
9      ?s CCSO:worksFor ?o.
10     ?s foaf:name ?nomep.
11
12     service <http://pt.dbpedia.org/sparql> {
13         ?o dbp:nome ?nomeuniv.
14         ?o dbo:state ?state.
15         ?state dbp:nome ?nomestate.
16     }
17
18 }
19 limit 100
20
```

Fonte: De autoria própria.