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

Daniel Tavares de Souza

Desenvolvimento de um dashboard para visualização de dados conectados de instituições acadêmicas

São Luís - MA
2025

Daniel Tavares de Souza

Desenvolvimento de um dashboard para visualização de dados conectados de instituições acadêmicas

Trabalho de Conclusão de Curso 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

2025

Daniel Tavares de Souza

Desenvolvimento de um dashboard para visualização de dados conectados de instituições acadêmicas

Trabalho de Conclusão de Curso 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. São Luís - MA, 31 de Julho de 2025:

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

Prof. Dr. Alana de Araujo Oliveira Meireles Teixeira
Examinadora Interna
Universidade Federal do Maranhão

Prof. Dr. Vandecia Rejane Monteiro Fernandes
Examinadora Interna
Universidade Federal do Maranhão

São Luís - MA
2025

Aos meus familiares.

Agradecimentos

Agradeço ao bom Deus pela bênção e capacitação para executar tal trabalho. Agradeço a minha noiva, meus familiares e amigos que me apoiaram até aqui e ao meu orientador por toda paciência, preocupação e extrema atenção com o trabalho e seu orientando. A todos, os meus sinceros e profundos agradecimentos.

"Estou em encantador estado de confusão."

(Ada Lovelace)

Resumo

Desde 2016, instituições federais, incluindo universidades e institutos de ensino, têm criado e mantido portais de dados abertos. Uma das grandes motivações para o uso de dados abertos está na possibilidade de usar algoritmos para acessar estes dados e apresentá-los da melhor maneira. Contudo, seria impraticável elaborar um algoritmo que conseguisse integrar os dados das dezenas de portais de dados abertos. Com essa motivação, um projeto denominado DBAcademic, tem proposto um portal de dados conectados de todas as instituições acadêmicas do Brasil. Com isso, é notável que a busca de informações dessas instituições acadêmicas tornou-se mais simples. Entretanto, estes dados só são acessíveis por meio de consultas utilizando algoritmos. Este trabalho propõe, então, o desenvolvimento de um *dashboard* como alternativa para tornar estes dados mais acessíveis. *Dashboard* são interfaces gráficas que apresentam informações de forma visual, permitindo a rápida interpretação de dados complexos. Essas interfaces consolidam dados de diferentes fontes e os apresentam de maneira organizada e fácil de entender, geralmente por meio de gráficos, tabelas, mapas e outros elementos visuais. Para isso, com base em trabalhos relacionados, projetou-se um *dashboard* com tecnologias suportadas pela linguagem Python, que encontra-se disponível em <https://dbacademic.streamlit.app>. Este *dashboard* também está disponível como software livre através no repositório Github do grupo de pesquisa LambdaGeo: <https://github.com/lambdageo>.

Palavras-chave: dashboard, dados conectados, instituições acadêmicas, streamlit

Abstract

Since 2016, federal institutions—including universities and technical institutes—have been creating and maintaining open-data portals. A major driver behind this movement is the possibility of employing algorithms to access those data sets and present them in the most useful way. Nevertheless, it would be impractical to design a single algorithm capable of integrating data from the dozens of independent portals now in operation. Motivated by this challenge, the DBAcademic project has proposed a connected-data portal that aggregates information from every academic institution in Brazil. As a result, searching for information about these institutions has become considerably easier; however, the underlying data are still accessible only through algorithmic queries. To lower that barrier, this work proposes the development of a dashboard that makes the data more approachable. Dashboards are graphical interfaces that display information visually, enabling rapid interpretation of complex data. They consolidate disparate data sources and present them in an organized, easy to understand manner—typically through charts, tables, maps, and other visual elements. Drawing on previous research, we designed such a dashboard with Python-based technologies; it is publicly available at <https://dbacademic.streamlit.app>. The full source code is released as free software in the LambdaGeo research group's GitHub repository: <https://github.com/lambdageo>.

Keywords: dashboard, linked data, academic institutions, streamlit

Lista de ilustrações

Lista de Códigos

Lista de tabelas

Lista de abreviaturas e siglas

API	Application Programming Interface
CSV	Comma-Separated Values
JSON	JavaScript Object Notation
PDF	Portable Document Format
SPARQL	SPARQL Protocol and RDF Query Language
XML	eXtensible Markup Language
FOAF	Friend-Of-A-Friend Ontology
GQM	Goal–Question–Metric
HCI	Human–Computer Interaction
HTTP	HyperText Transfer Protocol
IHC	Interação Humano-Computador
KPI	Key Performance Indicator
MIT	Massachusetts Institute of Technology
RDF	Resource Description Framework
SAD	Sistema de Apoio à Decisão
SIE	Sistema de Informação Executiva
SQL	Structured Query Language
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
XQuery	XML Query Language
REST	Representational State Transfer
SDK	Software Development Kit
CCSO	Curriculum and Course Semantic Ontology

Sumário

1 Introdução

A Política de Dados Abertos do Poder Executivo Federal, instituída pelo Decreto nº 8.777/2016 (BRASIL, 2016), estabeleceu diretrizes fundamentais para a disponibilização de informações públicas em formatos acessíveis, legíveis por máquina e passíveis de reutilização. Diferentemente de dados públicos apresentados em formatos estáticos, como arquivos PDF, os dados abertos devem ser estruturados em formatos como CSV, XML ou JSON, de modo a facilitar seu uso por sistemas computacionais e permitir sua integração em diferentes aplicações. A implementação dessa política incentivou órgãos da administração pública federal, incluindo universidades e institutos federais, a criarem seus próprios portais de dados abertos, seguindo planos institucionais alinhados às diretrizes nacionais.

Apesar dos avanços decorrentes dessa iniciativa, a estruturação dos dados acadêmicos no Brasil ainda apresenta desafios significativos. Os portais das instituições operam de forma independente, o que dificulta a realização de consultas integradas entre diferentes bases. Além disso, ainda existe a ausência de padronização nos formatos e nos vocabulários utilizados (SANTOS; FREITAS, 2020) que impõe barreiras técnicas para o reaproveitamento dos dados, especialmente por parte de usuários sem formação especializada. Buscando enfrentar essa limitação, Costa et al. (2020) propuseram o repositório DBAcademic, que adota o modelo de dados conectados (Linked Data) para integrar informações de múltiplas instituições federais de ensino superior. Posteriormente, Nahuz et al. (2023) ampliaram essa base utilizando abordagens de engenharia de dados, aumentando a abrangência e a qualidade da integração.

Embora o DBAcademic represente um importante avanço em termos de interoperabilidade semântica, sua utilização ainda requer conhecimentos técnicos, particularmente no uso de linguagens de consulta como SPARQL (DATA.WORLD, 2024b). Dessa forma, observa-se a carência de ferramentas que promovam a visualização acessível e intuitiva desses dados conectados, de modo a ampliar seu uso em contextos educacionais, científicos e de gestão.

Neste cenário, a utilização de *dashboards* interativos se apresenta como uma solução promissora. *Dashboards* são interfaces visuais capazes de sintetizar e apresentar informações de forma clara, permitindo ao usuário final explorar e compreender grandes volumes de dados com maior agilidade e autonomia (FEW, 2006; TURBAN et al., 2009). No entanto, seu uso ainda é incipiente no contexto da educação superior pública brasileira, especialmente no que se refere à visualização de dados interligados por meio da Web Semântica.

Diante desse panorama, este trabalho tem como **objetivo geral** desenvolver um protótipo de *dashboard* interativo para visualização de dados conectados sobre instituições de ensino superior brasileiras, integrando informações do repositório DBAcademic com

dados complementares da DBpedia[1], na qual destaca-se projeto comunitário de grande escala voltado à extração de dados estruturados da Wikipédia em suas versões multilíngues (AUER et al., 2007).

Como **objetivos específicos**, destacam-se:

- Investigar o potencial de integração semântica entre o DBAcademic e a DBpedia por meio de consultas SPARQL locais e federadas;
- Projetar e implementar um sistema de visualização baseado em tecnologias abertas (Python, Streamlit e Plotly);
- Prover funcionalidades interativas que permitam a exploração de cursos, instituições e docentes por diferentes recortes analíticos;

Ao permitir que pesquisadores, gestores e cidadãos explorem visualmente os dados acadêmicos por meio de uma interface acessível, o *dashboard* proposto busca contribuir com o fortalecimento da transparência institucional, o apoio à tomada de decisão e o estímulo a estudos sobre a dinâmica do ensino superior no Brasil.

[1] <https://www.dbpedia.org/>

2 Fundamentação Teórica

2.1 Dados abertos e conectados

O avanço da tecnologia e a dissipação de práticas de transparência de dados têm impulsionado a disponibilização e o uso de dados abertos, principalmente em contextos governamentais e acadêmicos. Neste sentido, nota-se a fundamentalidade em compreender não apenas o conceito de dados abertos, mas também sua conexão com a Web Semântica, o papel de linguagens como SPARQL na integração de diferentes bases de dados e a importância de fontes como a DBpedia para o enriquecimento semântico das informações disponibilizadas abertamente.

2.1.1 Conceito de dados abertos

Segundo a Group (2015), dados abertos são aqueles que podem ser livremente utilizados, reutilizados e redistribuídos por qualquer pessoa, estando sujeitos, no máximo, à exigência de atribuição da fonte original e ao compartilhamento sob as mesmas licenças em que as informações foram apresentadas. Essa definição, difundida pela Foundation (2015) contempla três características essenciais para os dados abertos:

- Disponibilidade e Acesso: os dados devem estar disponíveis como um todo e sob custo não maior que um custo razoável de reprodução, preferencialmente possíveis de serem baixados pela internet. Os dados devem também estar disponíveis de uma forma conveniente e modificável;
- Reutilização e Redistribuição: os dados devem ser fornecidos sob termos que permitam a reutilização e a redistribuição, inclusive a combinação com outros conjuntos de dados;
- Participação Universal: todos devem ser capazes de usar, reutilizar e redistribuir - não deve haver discriminação contra áreas de atuação ou contra pessoas ou grupos. Por exemplo, restrições de uso 'não-comercial' que impediriam o uso 'comercial', ou restrições de uso para certos fins (ex.: somente educativos) excluem determinados dados do conceito de 'abertos'.

Este conceito de dados abertos enfatiza a importância de garantir o acesso amplo e irrestrito à informação, promovendo a transparência, a inovação e a participação social. A partir deste contexto, a abertura dos dados contribui ativamente para o fortalecimento

de práticas colaborativas e para o desenvolvimento de soluções inovadoras em todos os setores, principalmente no acadêmico.

2.1.2 Dados conectados e web semântica

Segundo Yu (2011), a palavra "semântica" está relacionada à palavra "sintaxe", pois, enquanto a sintaxe define como algo é dito, a semântica está relacionada ao significado do que é dito. Assim, a Web Semântica pode ser entendida como "a Web de significados", ou seja, uma extensão da Web atual, onde as informações recebem um significado bem definido, o que possibilita uma cooperação mais eficiente entre pessoas e computadores (YU, 2011).

De acordo com o Berners-Lee, Hendler e Lassila (2001), "a Web Semântica" é uma extensão da Web atual na qual as informações possuem significado bem definido, tornando mais fácil para máquinas e seres humanos trabalharem juntos". Essa visão propõe uma estrutura comum que permite que os dados sejam compartilhados e reutilizados em diferentes aplicações, empresas e comunidades, promovendo maior interoperabilidade e integração entre sistemas diversos.

Com isso, Yu (2011) conclui que a Web Semântica pode ser compreendida como um conjunto de tecnologias e padrões que capacitam as máquinas a interpretar e compreender o significado (semântica) das informações disponíveis na Web. Para o autor, esse conjunto de padrões é fundamental para que sistemas automáticos possam extrair, processar e conectar dados de maneira inteligente, possibilitando o desenvolvimento de soluções inovadoras e eficientes.

Através das tecnologias e de padrões da Web Semântica surge o conceito de dados conectados, também conhecidos como Linked Data. Assim, por meio da interligação e publicação destes dados pode-se construir uma Web de Dados (YU, 2011). Nesse sentido, nota-se a conexão direta entre Web Semântica e dados conectados, essencialmente pela dependência direta que esses dados têm.

Nesse contexto, destaca-se a relevância dos dados conectados, que se fundamentam em dados abertos, já apresentados anteriormente nesta seção. A motivação e relevância para a utilização de dados conectados são notórias, pois, como apontado por Isotani (2015), é importante frisar que a geração de novos dados baseada em dados anteriormente consumidos é inerente à sociedade. Isso mostra a importância que a estruturação e a conexão de dados possuem para simplificar e facilitar a recuperação da informação e a produção de novos conhecimentos.

Segundo Berners-Lee (2009), os dados conectados representam um conjunto de práticas recomendadas para publicação de dados estruturados na Web. Ele destaca quatro regras essenciais para a construção de uma eficiente interconexão de dados:

- Usar URIs (termo genérico para sequências de caracteres que identificam um recurso na internet) como nomes para coisas, abrangendo também a nomeação de relações entre essas entidades. A utilização de URIs do mundo real nos dados é essencial para estabelecer conexões com outros conjuntos de dados;
- Usar HTTP URIs para facilitar a busca desses identificadores pelos usuários;
- Ao procurar uma URI, fornecer informações úteis utilizando padrões como RDF ou SPARQL;
- Incluir links para outras URIs, permitindo que tanto pessoas quanto sistemas computacionais possam explorar informações adicionais.

A relação entre dados abertos e dados conectados se torna mais evidente com a proposta do sistema de avaliação de 5 estrelas, também sugerido por Berners-Lee (2009). Esse sistema avalia, por meio de estrelas, o grau de abertura e conexão dos dados. Quanto mais abertas e conectadas às informações, maior a quantidade de estrelas que os dados recebem, tornando mais fácil seu enriquecimento e integração. As cinco estrelas para dados abertos são: Disponível na Web (em qualquer formato), com licença aberta, para ser considerado dado aberto; Disponível na Web como dado estruturado possível de ser lido por máquina (por exemplo, em um arquivo Excel ao invés de uma imagem escaneada de uma tabela); Como o anterior, mas em formato não proprietário (CSV em vez de Excel); Como no anterior, mas usando os padrões estabelecidos pelo W3C (RDF e SPARQL) para identificar coisas, de modo que as pessoas possam apontar para suas coisas; Todas as anteriores, mais: conectar seus dados a dados de outras pessoas para prover contexto.

2.1.3 SPARQL na web semântica

Segundo Allemang e Hendler (2008), SPARQL (SPARQL Protocol and RDF Query Language) é a linguagem padrão para consulta de dados RDF, permitindo a busca eficiente em grafos que podem ser criados a partir de um único tipo de dado ou da fusão de vários conjuntos distintos. De acordo com os autores, a SPARQL compartilha várias características com outras linguagens de consulta, como SQL e XQuery, mas destaca-se por ser projetada especificamente para explorar conexões e relações entre diferentes fontes de dados na Web Semântica. Logo, a linguagem tem destaque por conseguir acessar, consultar e integrar dados estruturados em diferentes formatos e de bases heterogêneas.

Além disso, os padrões de consulta SPARQL são representados em variantes da linguagem Turtle, facilitando a leitura e o entendimento das consultas por desenvolvedores e pesquisadores. Por meio do SPARQL, torna-se possível integrar dados heterogêneos, provenientes de múltiplas origens, promovendo uma visão unificada e interoperável dos

dados, aspecto essencial para aplicações em ciência de dados, gestão da informação e desenvolvimento de *dashboards* acadêmicos (ALLEMANG; HENDLER, 2008).

Harris e (eds.) (2013) exemplifica como é realizada uma consulta SPARQL, de acordo com o código 2.2. O exemplo a seguir apresenta uma consulta SPARQL simples, cujo objetivo é recuperar o título de um livro a partir de um grafo RDF. Essa consulta é composta por dois elementos principais: a cláusula `SELECT`, que especifica quais variáveis serão retornadas como resultado, e a cláusula `WHERE`, que define o padrão de tripla sujeito-predicado-objeto a ser buscado no grafo. No caso apresentado, o sujeito e o predicado são URIs conhecidas, enquanto o objeto é uma variável representada por `?title`.

O dado RDF utilizado encontra-se estruturado no formato Turtle, conforme exemplificado no Código 2.1:

Código 2.1 – Exemplo de dado RDF em formato Turtle

```
<http://example.org/book/book1>
   <http://purl.org/dc/elements/1.1/title> "SPARQL Tutorial" .
```

A consulta SPARQL correspondente é apresentada no Código 2.2:

Código 2.2 – Consulta SPARQL para obtenção do título do livro

```
SELECT ?title
WHERE {
  <http://example.org/book/book1>
      <http://purl.org/dc/elements/1.1/title> ?title .
}
```

A execução dessa consulta retorna um único resultado, conforme apresentado na Tabela 1:

Tabela 1 – Resultado da consulta SPARQL

title
SPARQL Tutorial

Observe que a consulta utiliza o padrão de tripla sujeito-predicado-objeto, onde o sujeito e predicado são URIs conhecidas, e o objeto é uma variável (`?title`) que será substituída pelo valor encontrado nos dados.

2.1.4 DBpedia e enriquecimento semântico

A **DBpedia** configura-se como um projeto comunitário de grande escala voltado à extração de dados estruturados da Wikipédia em suas versões multilíngues. Conforme Lehmann et al. (2012) e Auer et al. (2007), o projeto processa edições da enciclopédia

em mais de cem idiomas (BIZER et al., 2009) e disponibiliza, na Web, um repositório de conhecimento baseado nas tecnologias da Web Semântica e dos Dados Conectados.

A base de conhecimento multilíngue da DBpedia cobre uma extensa variedade de temas, o que a torna valiosa para diversos contextos, incluindo integração de dados, reconhecimento de entidades nomeadas, análise de tópicos e classificação de documentos (MENDES et al., 2011). Por sua abrangência e estrutura, a DBpedia é amplamente adotada como ambiente de testes pela comunidade acadêmica, servindo de base para o desenvolvimento de múltiplas aplicações, algoritmos e ferramentas. Publicada como Linked Data na Web, a DBpedia estabelece conexões RDF para uma série de fontes externas, o que faz com que muitos provedores de dados optem por criar links apontando para ela em seus próprios conjuntos. Dessa maneira, a DBpedia tornou-se um ponto central de interligação na Web de Dados Conectados, desempenhando papel fundamental para o avanço da iniciativa *Linked Open Data* (LEHMANN et al., 2012).

Ao ser publicada como *Linked Data*, a DBpedia estabelece conexões RDF com diversas fontes externas, funcionando como um nó central na Web de Dados Conectados (HEATH; BIZER, 2011). Essa característica atrai provedores de dados interessados em vincular seus repositórios a entidades da DBpedia, ampliando a interoperabilidade semântica e fortalecendo a iniciativa de *Linked Open Data*.

Para acessar o DBPedia, há duas maneiras (YU, 2011):

1. Usar um navegador da Web para visualizar diferentes gráficos RDF contidos no conjunto de dados do DBpedia ou
2. Usar um endpoint SPARQL para consultar o conjunto de dados do DBpedia com o objetivo de descobrir informações com muito mais facilidade.

Logo, os as duas maneiras de acesso, tanto orientado à inspeção humana quanto à execução automatizada de consultas, complementam-se e tornam a navegação no conhecimento da DBpedia simples e flexível, atendendo tanto a usuários ocasionais quanto a cenários de integração de dados em larga escala.

2.2 Visualização de dados e *dashboards*

A popularização das técnicas de visual analíticas transformou a forma como organizações analisam e comunicam informações. Ao convergir princípios de design visual, percepção humana e computação interativa, os *dashboards* ganharam espaço como artefatos centrais para monitorar indicadores e apoiar decisões em tempo real (FEW, 2006; WARE, 2013). Nesse cenário, compreender a evolução histórica dos Sistemas de Informação Executiva (SIE), os fundamentos de percepção visual e as heurísticas de usabilidade que

regem a interação humano-computador torna-se imprescindível para projetar painéis que sejam, ao mesmo tempo, claros, objetivos e capazes de estimular insights.

2.2.1 Histórico dos Sistemas de Visualização

De acordo com Furlan, Ivo e Amaral (1994), a expressão Sistema de Informações Executivas (SIE) surgiu no final dos anos 1970, originada a partir de estudos conduzidos no Massachusetts Institute of Technology (MIT) por pesquisadores como Rockart e Treacy.

A partir da década de 1980, os computadores passaram a se tornar mais acessíveis para a população de forma proporcional à queda de seus preços e aumento da sua capacidade de processamento. Isso proporcionou aos usuários finais maior autonomia em relação à serviços de tecnologia da informação. Durante esse período e meados da década de 1990, observou-se também a transição dos Sistemas de Apoio à Decisão (SAD) para os Sistemas de Informação Executiva (SIE), que, com sua evolução, passaram a disponibilizar, de forma simplificada, informações essenciais para os gestores por meio de interfaces amigáveis, utilizando recursos visuais como gráficos, dando origem aos *dashboards* (LAUDON; LAUDON, 2007; FURLAN; IVO; AMARAL, 1994; TURBAN et al., 2009; FEW, 2006).

Apesar de facilitarem a compreensão de dados e informações, os *dashboards* só passaram a ganhar destaque a partir de 2001, impulsionados pelas consequências do Caso Enron, no qual caracteriza-se um dos maiores escândalos corporativos da história dos Estados Unidos, envolvendo fraude contábil, manipulação de balanços e práticas ilícitas de gestão financeira (HEALY; PALEPU, 2003). Esse episódio levou muitos a perceberem que os *dashboards* ofereciam benefícios que iam além de uma tecnologia ainda emergente utilizada cotidianamente (FEW, 2006).

O termo *dashboard* abrange um espectro amplo de interpretações, com distintas definições adotadas por diversos especialistas. No entanto, Few (2006) traz uma explicação clara ao descrevê-lo como exibições visuais, que normalmente reúnem elementos textuais e gráficos, sendo que a prioridade recai nos gráficos. Few (2006) também afirma que *dashboards* tem como principal objetivo apresentar informações importantes por meio de uma interface simples na qual possibilita o entendimento de qualquer indivíduo, caracterizando-os como uma ferramenta de grande importância para a compreensão destas informações, que são consolidadas e organizadas em uma tela única para serem absorvidas em um único olhar de seu leitor (TURBAN et al., 2009). Com isso, espera-se que os *dashboards* criem melhores tomadas de decisões, ampliando a cognição e capitalizando as capacidades perceptivas humanas (YIGITBASIOGLU; VELCU, 2012).

2.2.2 Princípios de percepção visual e boas práticas

Mesmo existindo muitas maneiras de projetar e visualizar um *dashboard*, há um acordo entre muitos estudos de caso e escritores de que um *dashboard*: não deve sobrecarregar os usuários com informações excessivas (YIGITBASIOGLU; VELCU, 2012); deve evitar confusão visual, prevalecendo a clareza das informações (FEW, 2006); deve evitar um design visual deficiente e escolher KPIs (*Key Performance Indicators*) cuidadosamente (RAHMAN; ADAMU; HARUN, 2017); deve alinhar-se com os fluxos de trabalho existentes (FAIOLA; SRINIVAS; DUKE, 2015); não deve mostrar muitos dados (JANES; SILLITTI; SUCCI, 2013); deve ter recursos funcionais (ou seja, o que o painel pode fazer) e recursos visuais (ou seja, como as informações são apresentadas) (YIGITBASIOGLU; VELCU, 2012); deve fornecer consistência, recursos de interação e gerenciar a complexidade (SARIKAYA et al., 2019); e deve organizar os gráficos simetricamente, agrupar os gráficos por atributo, separar claramente esses grupos de gráficos e ordenar os gráficos de acordo com o tempo (BERNARD et al., 2019).

Além disso, Janes, Sillitti e Succi (2013) argumentam que para a obtenção de um *dashboard* ideal, dois aspectos precisam ser concentrados: selecionar os dados "certos" e escolher a técnica de visualização "certa".

Para orientar a seleção dos dados "certos" em contextos de avaliação e análise, Janes, Sillitti e Succi (2013) propõem o modelo GQM (Goal-Question-Measurement), que estrutura o processo de medição em três níveis hierárquicos: objetivo, questões e medidas. A Tabela 2 apresenta um resumo descritivo desses níveis.

Tabela 2 – Níveis do modelo GQM (Goal-Question-Measurement)

Nível	**Descrição**
Objetivo (conceitual)	Define o que se deseja estudar e por quê. Inclui o objeto de estudo (produtos, processos, recursos), os motivos da investigação, os pontos de vista considerados e o ambiente do estudo.
Questões (operacional)	Delineiam quais partes do objeto de estudo são relevantes e quais propriedades dessas partes serão observadas para avaliar o objetivo. Representam o "foco" da análise.
Medidas (quantitativo)	Determinam quais dados serão coletados para responder objetivamente às questões levantadas, permitindo análises quantitativas.

Fonte: Adaptado de Janes e Succi (2013).

Esses três níveis são complementados por um modelo de interpretação que define como interpretar os dados coletados para avaliar o objetivo da medição.

Além disso, a escolha da técnica de visualização de dados "certa" depende muito dos requisitos que o *dashboard* deve cumprir. Para isso, existem dois cenários para o

uso do *dashboard*, nos quais são os cenários de "puxar" e "enviar" (JANES; SILLITTI; SUCCI, 2013). No primeiro cenário chamado "puxar", o usuário deseja obter uma informação específica e utiliza o *dashboard* para obtê-la. Para este cenário, devem-se levar em consideração os seguintes pontos (FEW, 2006):

- O *dashboard* deve ajudar o usuário a entender o contexto dos dados: por que foram coletados, como devem ser interpretados, como podem ser utilizados em projetos futuros, etc.
- O *dashboard* deve ajudar o usuário a compreender o significado dos dados. As visualizações devem exigir um esforço mínimo para captar a mensagem transmitida, ser coerentes, permitir ao usuário escolher o nível de detalhe dos dados, etc.

No cenário de "enviar", o painel deve ser projetado para que as informações mais importantes sejam enviadas ao usuário. Assim, deve, além de captar a atenção do usuário, informá-lo. Segundo Few (2006) e Ware (2013), para configurar um *dashboard* que seja usado em um cenário enviar informações para o usuário são importantes as seguintes considerações:

- O usuário deve ser capaz de ver o painel sem nenhum esforço, ou seja, deve conter apenas as informações mais importantes de modo que seja "enxuto".
- O usuário não deve precisar interagir com as visualizações para compreender os dados, pois as informações precisam estar bem claras à primeira vista.
- Organize os dados para minimizar o tempo necessário para consultar o *dashboard*. Isso pode ser feito alocando as mesmas informações sempre num local específico, de modo que o usuário desenvolva um hábito de enxergar o que deseja à primeira vista.
- Sempre orientar a atenção do usuário para informações mais importantes, mas não abusando da maneira que essas informações são visualizadas.
- Fatores estéticos devem ser considerados como pontos importantes para que os usuários tenham seus interesses atraídos pela forma visual do *dashboard*.

Por fim, se um *dashboard* é mais adequado ao cenário "puxar" ou "enviar" depende de quanto esforço o usuário precisa investir para visualizar o *dashboard*. Um *dashboard* projetado para enviar informações ao usuário é mais vantajoso para informá-lo em situações inesperadas e imprevistas sobre problemas e afins. Um *dashboard* projetado para "puxar" informações deve oferecer mais possibilidades de explorar os dados, filtrar e pesquisar, investigar os motivos que causaram os dados, e assim por diante.

2.2.3 Usabilidade e IHC: heurísticas de Nielsen aplicadas a *dashboards*

A Interação Humano-Computador (IHC) destaca-se por ser uma área interdisciplinar que reúne ciência da computação, ciências cognitivas e engenharia de fatores humanos, com surgimento no início dos anos 1980 como uma especialização de ciência da computação para estudar, projetar e melhorar as formas pelas quais as pessoas interagem com sistemas computacionais, pois na época, com o início da computação pessoal, eram evidentes as deficiências dos computadores em relação à usabilidade. Seu principal objetivo é compreender as limitações e oportunidades presentes nessa interação, de modo a criar experiências mais intuitivas, eficientes e satisfatórias para os usuários, no qual converge com as aplicações de *dashboards* (CARROLL, 2014).

Ainda na década de 1980, o termo usabilidade teve ascensão como um substituto para a expressão user-frendly, até os dias atuais (ALMEIDA, 2018). Na mesma época, Jakob Nielsen publicou contribuições fundamentais para a área, primeiro com o arcabouço de usability engineering (NIELSEN, 1993) e, em seguida, com o conjunto das dez heurísticas de usabilidade (NIELSEN, 1994), que, resumidamente, servem como princípios gerais do design de interface do usuário, e, com isso, são usadas para avaliar interfaces de maneira rápida e de baixo custo. Assim, parafraseando Nielsen (1994) e adaptando as heurísticas para o contexto de *dashboards*, tem-se:

1. **Visibilidade do status do sistema**: o sistema deve informar claramente o que está acontecendo no sistema, por exemplo, com filtros ativos e atualização de dados, evitando que o usuário fique sem feedback.

2. **Correspondência entre sistema e mundo real**: utilizar termos, metáforas e representações visuais familiares ao usuário, como mapas para localização e séries temporais para evolução, em vez de jargões internos.

3. **Controle e liberdade do usuário**: oferecer caminhos para desfazer/reativar ações e sair de estados indesejados com facilidade, como por exemplo botões “voltar” e “limpar filtros”.

4. **Consistência e padrões**: manter padrões de cor, tipografia, ícones e comportamento em todas as telas para reduzir a carga cognitiva.

5. **Prevenção de erros**: projetar a interface para evitar entradas inválidas ou combinações sem sentido, como por exemplo, menus restritivos, mensagens preventivas antes de executar consultas custosas, etc.

6. **Reconhecimento em vez de memorização**: preferir opções e rótulos visíveis a comandos que exijam lembrança; legendas, dicas e indicadores inline reduzem a necessidade de memória de curto prazo.

7. **Flexibilidade e eficiência de uso**: permitir tanto rotas rápidas (atalhos, busca textual) para usuários experientes quanto caminhos mais guiados para iniciantes.

8. **Estética e design minimalista**: remover elementos supérfluos que distraiam do foco analítico; priorizar clareza e relevância das informações exibidas.

9. **Ajuda no reconhecimento, diagnóstico e correção de erros**: mensagens de erro devem ser compreensíveis e indicar o que fazer para corrigir o problema, como por exemplo: "nenhum dado retornado, ajuste o filtro X".

10. **Ajuda e documentação**: mesmo interfaces bem projetadas podem exigir suporte pontual; oferecer instruções breves, exemplos e links para documentação quando necessário.

2.3 Ferramentas e tecnologias

A oferta crescente de bibliotecas Python, serviços em nuvem e plataformas de dados abertos criou um ecossistema fértil para a construção rápida de aplicações analíticas. Do ponto de vista de infraestrutura, soluções como o Data.World fornecem catálogos compatíveis com padrões da Web Semântica, enquanto frameworks como Streamlit e Plotly permitem transformar código em *dashboards* interativos com poucas linhas (STREAMLIT, 2024; Plotly Technologies Inc., 2023). Esta seção mapeia as tecnologias adotadas no protótipo, desde a publicação e a consulta a conjuntos RDF até a visualização final no navegador, destacando os motivos da escolha e a forma como cada ferramenta se encaixa no fluxo de dados proposto.

2.3.1 Plataformas para publicação e consulta de dados abertos: o caso da Data.World

A organização e a disseminação de dados abertos exigem a existência de catálogos de dados bem estruturados, que atuem como repositórios centrais e contenham não apenas os dados em si, mas também metadados descritivos sobre sua origem, estrutura, qualidade e semântica. Tais catálogos viabilizam o gerenciamento eficiente dos conjuntos de dados, favorecendo sua descoberta, reutilização e interoperabilidade entre sistemas.

Nesse contexto, destaca-se a plataforma Data.World, concebida como um ambiente colaborativo e baseado em nuvem para hospedagem, catalogação e consulta de dados abertos. Desenvolvida com foco na usabilidade, a plataforma permite que usuários de diferentes perfis — incluindo não especialistas em ciência de dados — possam buscar, explorar e integrar conjuntos de dados de forma rápida e intuitiva (DATA.WORLD, 2024a).

Além de oferecer mecanismos para pesquisa e filtragem, o Data.World possibilita o envio de dados próprios, a criação de projetos compartilhados e a colaboração entre múltiplos usuários. A plataforma conta com milhares de bases de dados públicas e privadas organizadas por temas diversos, funcionando como um ponto central de articulação entre produtores e consumidores de dados (HOYT, 2018). Sua compatibilidade com padrões abertos, como RDF e SPARQL, torna-a particularmente adequada para aplicações que envolvem dados conectados e Web Semântica.

2.3.2 Frameworks Python para construção de *dashboards*

A linguagem Python consolidou-se como uma das principais ferramentas no campo da ciência de dados, destacando-se por sua versatilidade, ampla adoção na comunidade científica e grande ecossistema de bibliotecas voltadas à análise, processamento e visualização de dados. Conforme argumenta McKinney (2018), Python apresenta vantagens competitivas em relação a outras linguagens e ferramentas amplamente utilizadas, como R, MATLAB, SAS e Stata, especialmente devido à evolução de seu suporte a bibliotecas especializadas, como `pandas` e `scikit-learn`. Aliando essas capacidades à sua natureza de linguagem de propósito geral, Python tornou-se a melhor escolha para o desenvolvimento de aplicações de dados.

Utilizando a linguagem Python, o Streamlit segue o seguinte paradigma declarativo: cada execução do script reconstrói a "árvore" de componentes e a reconcilia com o DOM gerado em React. Essa estratégia garante *hot-reloading* e ciclo de desenvolvimento rápido, semelhante aos notebooks, mas com apresentação de página Web em tela cheia. A biblioteca possui um conjunto de "widgets" (entradas, controles deslizantes, caixas de seleção) e blocos de texto que podem ser compostos de forma imperativa; quando um widget muda de estado, o script é reexecutado de cima a baixo, preservando valores em cache conforme necessário (STREAMLIT, 2024; RICHARDS, 2023).

Diferentemente de abordagens *callback-based*, como o Dash (PLOTLY, 2023), Streamlit recomenda o uso de `@st.cache_data` e `@st.cache_resource` para armazenar resultados de funções dispendiosas, impressos na documentação oficial como peça-chave para manter latência baixa (STREAMLIT, 2024).

Em conjunto, Streamlit e Plotly constituem uma pilha moderna para construção de *dashboards* em Python que equilibra (STREAMLIT, 2024; RICHARDS, 2023; Plotly Technologies Inc., 2023):

- **Rapidez de desenvolvimento**: *hot-reload*, sintaxe declarativa e ausência de configuração *front-end*;
- **Interatividade nativa**: gráficos Plotly, widgets reativos e callbacks automáticos;

- **Desempenho controlado**: cache flexível em múltiplas camadas (`st.cache_data`, `pandas` in-memory) e renderização WebGL quando necessária;
- **Escalabilidade e distribuição**: implantação gratuita no Streamlit Community Cloud, com manutenção simplificada via CI/CD;
- **Boas práticas visuais**: suporte a designs minimalistas e responsivos, alinhados à literatura de visualização de dados.

2.3.3 Integração via APIs Python

A integração de APIs é um componente fundamental em *dashboards* acadêmicos modernos, permitindo a obtenção e atualização dinâmica de dados provenientes de fontes externas. No contexto deste trabalho, a integração foi realizada utilizando a biblioteca datadotworld, que facilita o acesso programático a datasets hospedados na plataforma Data.World. Por meio dessa biblioteca, é possível executar consultas SPARQL diretamente em bases de dados abertas, como o projeto DbAcademic, que reúne informações sobre cursos e docentes de universidades brasileiras. Segundo a documentação oficial, "A biblioteca Python do Data.World simplifica o processo de acessar e utilizar dados do data.world. Ela oferece interfaces convenientes para as APIs do Data.World, permitindo criar e atualizar conjuntos de dados, adicionar e modificar arquivos e muito mais. Com isso, é possível até mesmo construir aplicações completas utilizando a plataforma Data.World." (DATA.WORLD, 2024a).

2.3.4 Publicação via Streamlit Community Cloud: acesso público e replicabilidade

O Streamlit Community Cloud é apresentado como uma solução prática e rápida para o deploy de aplicações desenvolvidas em Streamlit, sendo a principal recomendação para publicar esses aplicativos (STREAMLIT, 2024).

O processo de implantação é facilitado: basta enviar o código para um repositório no GitHub e conectar esse repositório ao Streamlit, que cuida automaticamente do restante. Embora seja possível configurar detalhes como armazenamento e memória, na maioria dos casos o serviço gerencia todos esses aspectos, tornando o desenvolvimento e a publicação de aplicações significativamente mais simples para os usuários (RICHARDS, 2023).

3 Metodologia

3.1 Tipo de Pesquisa

A presente investigação caracteriza-se como pesquisa aplicada e exploratória, baseando-se em princípios metodológicos, citados na seção 2, voltando-se à geração de conhecimento prático por meio do desenvolvimento de um protótipo e à exploração de tecnologias emergentes no contexto de dados acadêmicos abertos e semânticos (GIL, 2010; VERGARA, 2016).

O foco principal do projeto está em destrinchar problemas notáveis de visualização de dados acadêmicos, através do desenvolvimento de um protótipo funcional de *dashboards* capaz de integrar, enriquecer e apresentar informações oriundas das fontes abertas Data.World e DBpedia, enfatizando a usabilidade e utilidade para a comunidade acadêmica do país. Com isso, o protótipo desenvolvido segue a lógica de prototipagem incremental descrita na engenharia de software (PRESSMAN; MAXIM, 2014), utilizando Python, `pandas` e o framework Streamlit para implementar, testar e refinar uma solução interativa de visualização de dados (MCKINNEY, 2018; RICHARDS, 2023; STREAMLIT, 2024).

3.2 Pipeline de Dados

A principal fonte de dados é o DbAcademic, disponibilizado na plataforma Data.World, ambiente colaborativo para publicação e consumo de dados abertos em múltiplos domínios (HOYT, 2018; DATA.WORLD, 2024a). O dataset reúne informações estruturadas sobre cursos, universidades e institutos brasileiros em RDF, o que facilita consultas semânticas e integração via Linked Data (HEATH; BIZER, 2011).

A escolha do Data.World se deve à sua compatibilidade com padrões de dados ligados (Linked Data), à facilidade de acesso via APIs e à possibilidade de atualização contínua dos conjuntos de dados, garantindo a atualidade e a relevância das informações apresentadas (DATA.WORLD, 2024a).

A extração e manipulação dos dados foram conduzidas por meio de consultas SPARQL, linguagem padrão recomendada pelo W3C para consulta e manipulação de grafos RDF (HARRIS; (EDS.), 2013). A linguagem SPARQL permite selecionar, filtrar e combinar entidades heterogêneas, viabilizando integrações semânticas entre o DbAcademic e a DBpedia por meio de consultas federadas (ALLEMANG; HENDLER, 2008; HEATH; BIZER, 2011).

O backend do sistema, implementado em Python, utiliza a biblioteca datadotworld

para executar as queries SPARQL e obter os resultados em formato de dataframes pandas. Nessa etapa, são realizadas as seguintes operações:

- Limpeza e normalização: Conversão de tipos, tratamento de valores ausentes e padronização de nomes.
- Enriquecimento semântico: Mapeamento de universidades para regiões brasileiras, categorização de cursos, cálculo de estatísticas agregadas.
- Cache de resultados: Utilização do cache do Streamlit para otimizar o tempo de resposta e reduzir o número de requisições aos endpoints.

Após o processamento, os dados processados são apresentados ao usuário por meio de um *dashboard* interativo desenvolvido com Streamlit e Plotly. As principais características desta etapa incluem:

- Visualizações dinâmicas: Gráficos de barras, histogramas, mapas e tabelas interativas.
- Filtros inteligentes: Permitem ao usuário explorar os dados por região, estado, nome do curso, entre outros.
- Exportação de dados: Possibilidade de baixar os dados filtrados em formato CSV.
- Atualização em tempo real: O usuário pode forçar a atualização dos dados, limpando o cache e executando novamente o pipeline.

3.3 Modelagem de Dados

Para estruturar as entidades acadêmicas no dashboard, o DBAcademic adota a ontologia CCSO (*Curriculum and Course Semantic Ontology*), na qual foi concebida para oferecer uma cobertura "rica e suficiente" de elementos de currículo, desde cursos e sílabos até vínculos institucionais, superando ontologias educacionais anteriores e viabilizando serviços avançados, como comparação semântica de currículos e recomendações automáticas (KATIS et al., 2018). Na Tabelas 3 e 4, são mostradas propriedades de classes utilizadas neste projeto.

Tabela 3 – Propriedades da classe `ccso:ProgramOfStudy` (Curso)

Propriedade	**Descrição**
`ccso:psName`	Nome oficial do curso
`ccso:code`	Identificador único do curso
`ccso:belongsTo`	Associação do curso à instituição responsável

Fonte: Elaborado pelo autor.

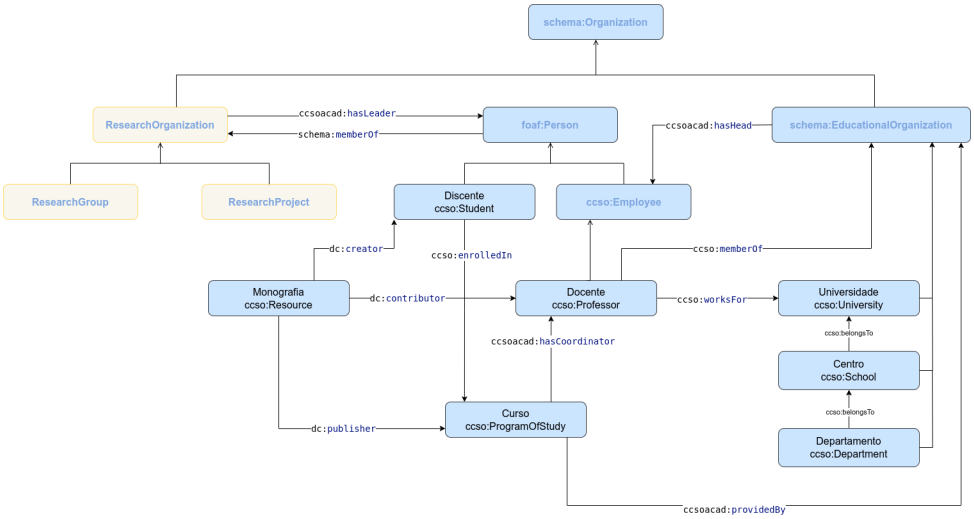
Tabela 4 – Propriedades da classe `ccso:Professor` (Docente)

Propriedade	**Descrição**
`ccso:hasDegree`	Grau acadêmico atribuído ao docente
`ccso:worksFor`	Vínculo institucional (universidade/departamento)
`foaf:name`	Nome do docente
`foaf:personID`	Matrícula ou identificador interno
`foaf:gender`	Gênero do docente

Fonte: Elaborado pelo autor.

A Figura 1 mostra, de forma resumida, como os elementos do ambiente acadêmico se são modelados no DBAcademic. No centro da modelagem está o Curso, no qual pode ser identificado por seu código (`ccso:code`) e nome oficial (`ccso:psName`), que pertence (`ccso:belongsTo`) a um Departamento, Centro ou Universidade. O Docente, descrito por `foaf:name`, `foaf:personID` e `foaf:gender`, trabalha na instituição (`ccso:worksFor`), possui um grau acadêmico (`ccso:hasDegree`) e pode coordenar o curso. O Discente está matriculado no curso, podendo produzir uma monografia com orientação do docente. Acima, surgem organizações de pesquisa (grupos e projetos) às quais pessoas podem pertencer e que podem ter um líder. Todas essas unidades formam, juntas, a organização educacional maior, permitindo consultas como "quantos cursos cada universidade oferece?" ou "onde trabalham determinados docentes?".

Figura 1 – Diagrama de Modelagem de Dados DBAcademic



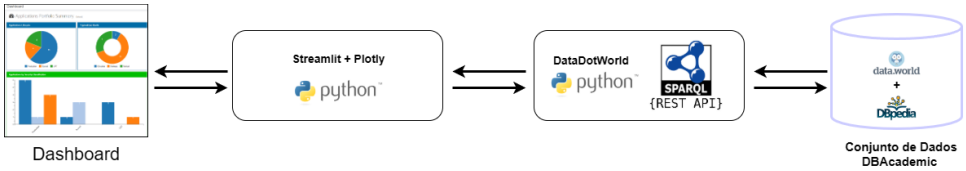
Fonte: Lambdageo, Acesso em: 04 Ago. 2025

3.4 Arquitetura do Sistema

O sistema do projeto foi arquitetado para ser modular, escalável e de fácil manutenção, princípios clássicos de projeto de software nos quais visam separar responsabilidades e

facilitar evolução (BASS; CLEMENTS; KAZMAN, 2003). A base de dados principal é o DbAcademic, hospedado no Data.World, o qual reúne informações estruturadas em RDF sobre cursos, universidades e institutos brasileiros (DATA.WORLD, 2024a). Para agregar contexto e enriquecer os dados, o sistema também integra informações da DBpedia, uma base de conhecimento em larga escala derivada da Wikipedia, amplamente utilizada no projeto como fonte de interligação e enriquecimento semântico na Web de Dados (LEHMANN et al., 2012). Na figura 2 é possível observar de forma ampla a arquitetura adotada para o desenvolvimento do sistema, cujas características são detalhadas nesta seção.

Figura 2 – Arquitetura do projeto



Fonte: Elaborado pelo autor

A integração ocorre por meio de consultas SPARQL tanto locais (restritas ao grafo do DbAcademic) quanto federadas (acessando endpoints externos via cláusula `SERVICE`), explorando a capacidade do SPARQL 1.1 de cruzar múltiplas fontes heterogêneas (HARRIS; (EDS.), 2013).

No *backend*, implementado em Python, as consultas são executadas via `datadotworld`, processadas em `pandas` e enriquecidas semanticamente, por exemplo, com o mapeamento de universidades para regiões, antes de serem repassadas ao *frontend*. Além disso, o uso do cache do Streamlit garante menor latência e evita recomputações desnecessárias, mantendo o sistema responsivo mesmo em cenários de consultas federadas mais custosas (STREAMLIT, 2024).

A interface com o usuário, desenvolvida com Streamlit, oferece uma navegação simples entre diferentes análises dos dados presentes, aplicação de filtros inteligentes e visualizações interativas com Plotly, cujo ecossistema disponibiliza diversos tipos de gráficos, como barras, pizza, dispersão e histogramas (Plotly Technologies Inc., 2023; RICHARDS, 2023). Uma função no frontend da aplicação disponibiliza a exportação de dados filtrados (CSV), permitindo reuso em outros contextos analíticos.

Logo, para o usuário final, o fluxo é transparente: ao interagir com o *dashboard*, as funções Python são acionadas para executar as consultas SPARQL, processar e enriquecer os dados e, finalmente, renderizar os resultados. Todo esse processo pode ser atualizado sob demanda, reforçando o caráter exploratório e iterativo do protótipo.

A estratégia de enriquecimento com a DBpedia destaca-se como diferencial, pois

amplia o contexto semântico e a qualidade das análises ao incorporar nomes oficiais, estados e outras propriedades das instituições. Por fundamentar-se em tecnologias abertas e padronizadas — SPARQL, Python e Streamlit — o sistema mantém portabilidade, escalabilidade e facilidade de manutenção, além de beneficiar-se do modelo de publicação gratuito do Streamlit Community Cloud (STREAMLIT, 2024).

4 Desenvolvimento do Protótipo

4.1 Implementação

Esta seção apresenta a materialização técnica do protótipo, dividindo-se em três etapas complementares. Primeiro, descreve-se como as consultas SPARQL foram concebidas para extrair, integrar e enriquecer os dados do DBAcademic com informações provenientes da DBpedia; em seguida, detalha-se a camada de conexão Python + API Data.World, responsável por executar as consultas, tratar a autenticação e entregar os resultados em estruturas `pandas`; por fim, apresenta-se a construção do *dashboard* no Streamlit, onde os dados processados ganham forma em visualizações interativas, filtros dinâmicos e métricas resumidas. Cada subseção aprofunda um desses pilares, permitindo acompanhar, passo a passo, desde a recuperação semântica dos dados até a interface final acessível ao usuário.

4.1.1 Desenvolvimento das consultas SPARQL e enriquecimento com DBpedia

A espinha dorsal do protótipo reside no conjunto de consultas SPARQL responsáveis por extrair, consolidar e enriquecer o projeto com dados acadêmicos. Essas consultas partem do grafo RDF do DbAcademic (hospedado no Data.World) e, sempre que necessário, recorrem à DBpedia para complementar informações por meio de consultas federadas com a cláusula `SERVICE`. A regra orientadora foi simples: utilizar a federação quando há dados relacionados à consulta no contexto adicional e manter a consulta local no dataset do DbAcademic, caso este recurso da DBpedia não possua dados relacionados disponíveis, como nome oficial da instituição, estado, etc. Esse equilíbrio reduziu a dependência de endpoints externos, preservou desempenho e simplificou a manutenção.

As consultas locais concentram-se em métricas internas, por exemplo, contagem total de cursos ou frequências de nomes de cursos. Para esses casos, bastam os prefixos do vocabulário `ccso` e as agregações básicas. Um exemplo típico do projeto é a quantificação global de cursos, feita com `COUNT(DISTINCT ?cursos)` sobre os indivíduos tipados como `ccso:ProgramofStudy`, sem qualquer necessidade de bloco `SERVICE`, como mostrado no código 4.1.

Para compreender a distribuição dos cursos por universidade e, ao mesmo tempo, apresentar o nome oficial dessas instituições, tornou-se necessário “sair” do grafo local. Cada universidade possui um identificador equivalente na DBpedia via `owl:sameAs`. A consulta aproveita esse elo semântico para acessar o endpoint público da DBpedia, recuperar o literal de `dbp:name` e, então, contabilizar quantos cursos estão vinculados a cada instituição, conforme pode-se observar no código 4.2.

Código 4.1 – Consultando a contagem total de cursos

```
PREFIX :   <https://dbacademic.linked.data.world/d/dbacademic/>
PREFIX ds-institutos: <https://dbacademic.linked.data.world/d/institutos/>
PREFIX ds-universidades: <https://dbacademic.linked.data.world/d/universidades/>
PREFIX ccso: <https://w3id.org/ccso/ccso#>

SELECT (COUNT(DISTINCT ?cursos) AS ?qtcursos)
WHERE {
  ?cursos a ccso:ProgramofStudy .
}
```

Código 4.2 – Consultando cursos por universidade com nome oficial via DBpedia

```
PREFIX ccso: <https://w3id.org/ccso/ccso#>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT ?Universidade (COUNT (DISTINCT ?s) AS ?Cursos)
WHERE {
  ?s a ccso:ProgramofStudy .
  ?s ccso:belongsTo ?url_pt .
  ?url_pt owl:sameAs ?url_eng .
  SERVICE <http://dbpedia.org/sparql> {
    ?url_eng dbp:name ?Universidade .
  }
}
GROUP BY ?Universidade
ORDER BY DESC (?Cursos)
```

Em outras análises sobre cursos, o foco deslocou-se para a frequência de nomes de cada curso ofertado nacionalmente. Nesse caso, continuou-se no contexto local, agrupando-se os cursos por `ccso:psName` e calculando-se sua incidência, como apresentado no código 4.3.

Código 4.3 – Consultando a frequência de nomes de cursos

```
PREFIX :   <https://dbacademic.linked.data.world/d/dbacademic/>
PREFIX ds-institutos: <https://dbacademic.linked.data.world/d/institutos/>
PREFIX ds-universidades: <https://dbacademic.linked.data.world/d/universidades/>
PREFIX ccso: <https://w3id.org/ccso/ccso#>

SELECT ?name (COUNT(?cursos) AS ?qtd)
WHERE {
  ?cursos a ccso:ProgramofStudy .
  ?cursos ccso:psName ?name .
  ?cursos ccso:belongsTo ?u .
}
GROUP BY ?name
ORDER BY DESC(?qtd)
```

Para recortes temáticos específicos, como Engenharia de Computação, uma simples expressão regular resolve o problema, ainda em ambiente local. O uso de `FILTER regex(...,`

Código 4.4 – Consultando cursos de Engenharia de Computação

```
PREFIX :   <https://dbacademic.linked.data.world/d/dbacademic/>
PREFIX ds-institutos: <https://dbacademic.linked.data.world/d/institutos/>
PREFIX ds-universidades: <https://dbacademic.linked.data.world/d/universidades/>
PREFIX ccso: <https://w3id.org/ccso/ccso#>

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

`"i"`) garante que a busca não seja sensível a maiúsculas e minúsculas, retornando todos os cursos cujo nome contém a expressão desejada. Além disso, o `"D."`no regex garante que sejam buscados os cursos que sejam iguais a Engenharia da Computação ou Engenharia de Computação (código 4.4).

A necessidade de considerar um parâmetro geográfico, por exemplo, retornar apenas "engenharias" de um estado selecionado, introduz novamente a consulta federada. Ao associar o curso à universidade e, por meio de `owl:sameAs`, à sua contraparte na DBpedia, tornou-se possível filtrar os resultados com base no estado (`dbp:state`) dentro do bloco `SERVICE`. Essa consulta aceita a região como parâmetro externo (via interface) e retorna, agrupados, os nomes dos cursos de engenharia e suas quantidades naquele estado, conforme o código 4.5.

Código 4.5 – Consultando engenharias por estado

```
PREFIX :   <https://dbacademic.linked.data.world/d/dbacademic/>
PREFIX ds-institutos: <https://dbacademic.linked.data.world/d/institutos/>
PREFIX ds-universidades: <https://dbacademic.linked.data.world/d/universidades/>
PREFIX ccso: <https://w3id.org/ccso/ccso#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?name (COUNT(?cursos) AS ?qtd)
WHERE {
  ?cursos a ccso:ProgramofStudy .
  ?cursos ccso:psName ?name .
  ?cursos ccso:belongsTo ?u .
  ?u owl:sameAs ?u_dbpedia .
  FILTER regex(?name, "engenharia", "i")
  SERVICE <http://dbpedia.org/sparql> {
    OPTIONAL { ?u_dbpedia dbp:state ?estado . }
    FILTER (lcase(str(?estado)) = "{estado}")
  }
}
GROUP BY ?name
ORDER BY DESC(?qtd)
LIMIT 50
```

Para obter nome do curso, universidade e estado em um único resultado, a consulta volta a combinar padrões locais (curso e vínculo institucional) com o resgate de rótulos e propriedades geográficas na DBpedia. A presença de `OPTIONAL` em torno de `dbo:state` evita a perda de linhas quando o estado não está declarado na base de dados externa, como detalhado no código 4.6.

Código 4.6 – Consultando nome do curso, universidade e estado

```
PREFIX ccso: <https://w3id.org/ccso/ccso#>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT ?NomeCurso ?Universidade ?Estado
WHERE {
  ?curso a ccso:ProgramofStudy .
  ?curso ccso:psName ?NomeCurso .
  ?curso ccso:belongsTo ?url_pt .
  ?url_pt owl:sameAs ?url_eng .
  SERVICE <http://dbpedia.org/sparql> {
    ?url_eng dbp:name ?Universidade .
    OPTIONAL {
      ?url_eng dbo:state ?state .
      ?state dbp:name ?Estado .
    }
  }
}
LIMIT 1000
```

Quando o foco mudou para docentes, manteve-se a mesma estratégia. Para saber quantos professores existem por estado, é preciso primeiro identificar a instituição para a qual trabalham (`CCSO:worksFor`) e então consultar a DBpedia para obter o estado da universidade. É uma consulta típica, usando `COUNT(DISTINCT ?s)` para evitar duplicatas e `GROUP BY` para agregar por estado, conforme o código 4.7.

Código 4.7 – Consulta SPARQL para docentes por estado

```
PREFIX CCSO: <https://w3id.org/ccso/ccso#>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT ?Estado (COUNT (DISTINCT ?s) AS ?Docentes) WHERE {
  ?s a CCSO:Professor .
  ?s CCSO:worksFor ?url_pt .
  ?url_pt owl:sameAs ?url_eng .
  SERVICE <http://dbpedia.org/sparql> {
    ?url_eng dbo:state ?state .
    ?state dbp:name ?Estado .
  }}
GROUP BY ?Estado
ORDER BY DESC (?Docentes)
```

Por outro lado, como demonstra o código 4.8, a contagem de docentes por grau de formação não utilizou atributos externos, ou seja, todo o vocabulário foi resgatado no DbAcademic. Assim, a consulta permaneceu local, agregando os docentes pelo objeto de `CCSO:hasDegree`.

Código 4.8 – Consultando docentes por grau de formação

```
PREFIX CCSO: <https://w3id.org/ccso/ccso#>

SELECT ?GrauFormacao (COUNT(DISTINCT ?s) AS ?Docentes)
WHERE {
  ?s a CCSO:Professor .
  ?s CCSO:hasDegree ?GrauFormacao .
}
GROUP BY ?GrauFormacao
ORDER BY DESC(?Docentes)
```

Para cruzar estado e grau de formação em uma mesma visualização, é preciso somar as duas dimensões: o estado vem da DBpedia e o grau, do grafo local. A consulta agrupa ambos os campos, devolvendo uma matriz de combinações (estado × formação) com suas respectivas contagens, como se vê no código 4.9.

Código 4.9 – Consultando docentes por estado e grau de formação

```
PREFIX CCSO: <https://w3id.org/ccso/ccso#>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT ?Estado ?GrauFormacao (COUNT(DISTINCT ?s) AS ?Docentes)
WHERE {
  ?s a CCSO:Professor .
  ?s CCSO:hasDegree ?GrauFormacao .
  ?s CCSO:worksFor ?url_pt .
  ?url_pt owl:sameAs ?url_eng .
  SERVICE <http://dbpedia.org/sparql> {
    ?url_eng dbo:state ?state .
    ?state dbp:name ?Estado .
  }
}
GROUP BY ?Estado ?GrauFormacao
ORDER BY ?Estado DESC(?Docentes)
```

Finalmente, a distribuição de docentes por gênero segue essa mesma divisão de responsabilidades. O gênero (`foaf:gender`) é um atributo opcional no grafo local, daí o uso de `OPTIONAL` para não descartar registros incompletos, enquanto a identificação do estado novamente depende da DBpedia. O resultado é uma visão detalhada, por estado, da quantidade de docentes por gênero, preservando linhas mesmo quando o dado de gênero está ausente, o que ocorre com frequência, como mostra o código 4.10.

Código 4.10 – Consultando docentes por estado e gênero

```
PREFIX CCSO: <https://w3id.org/ccso/ccso#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT ?Estado ?Sexo (COUNT(DISTINCT ?s) AS ?Docentes)
WHERE {
  ?s a CCSO:Professor .
  ?s CCSO:worksFor ?url_pt .
  ?url_pt owl:sameAs ?url_eng .
  OPTIONAL { ?s foaf:gender ?Sexo . }
  SERVICE <http://dbpedia.org/sparql> {
    ?url_eng dbo:state ?state .
    ?state dbp:name ?Estado .
  }
}
GROUP BY ?Estado ?Sexo
ORDER BY ?Estado DESC(?Docentes)
```

4.1.2 Conexão Python + API data.world

A integração do protótipo com a plataforma data.world foi realizada com a biblioteca oficial em Python (`datadotworld`, importada como `dw`). Essa escolha viabilizou o envio de consultas SPARQL diretamente do código e o recebimento dos resultados em estruturas familiares, como `pandas.DataFrame`.

A autenticação no data.world utiliza um token de acesso, que pode ser definido em variável de ambiente (`DW_AUTH_TOKEN`), configurado em `~/.dw/config` ou passado explicitamente na sessão. Em ambientes Streamlit, o token costuma ficar em `.streamlit/secrets.toml` ou em variáveis de ambiente, evitando expor credenciais no repositório.

O fluxo de comunicação adotado foi direto da seguinte maneira:

- O dataset é carregado com `dw.load_dataset('dbacademic/dbacademic', force_update=True)`;
- As consultas são executadas via `dw.query(..., query_type='sparql')`;
- O retorno é acessado em `results.dataframe`, alimentando o restante do pipeline analítico.

Um exemplo típico de função que encapsula esse processo está no código 4.11.

Código 4.11 – Função Python para consultar cursos por universidade

```python
@st.cache_data(ttl=3600)
def get_cursos_por_universidade():
    ds = dw.load_dataset('dbacademic/dbacademic', force_update=True)
    sparql_query = """
    PREFIX ccso: <https://w3id.org/ccso/ccso#>
    PREFIX dbp: <http://dbpedia.org/property/>
    PREFIX owl: <http://www.w3.org/2002/07/owl#>

    SELECT ?Universidade (COUNT (DISTINCT ?s) AS ?Cursos)
    WHERE {
        ?s a ccso:ProgramofStudy ;
           ccso:belongsTo ?url_pt .
        ?url_pt owl:sameAs ?url_eng .
        SERVICE <http://dbpedia.org/sparql> {
            ?url_eng dbp:name ?Universidade .
        }
    }
    GROUP BY ?Universidade
    ORDER BY DESC (?Cursos)
    """
    results = dw.query('dbacademic/dbacademic', sparql_query, query_type='sparql')
    return results.dataframe, sparql_query
```

Na prática, esse padrão evidencia a identificação do dataset (`'dbacademic/dbacademic'`), a definição da consulta como string multilinha, a execução com `query_type='sparql'` e o retorno direto em `DataFrame`.

Para reduzir latências que podem gerar lentidão e gargalos para o usuário final, especialmente quando utiliza-se consultas federadas à base DBpedia, adotou-se cache em duas camadas: `@st.cache_data` no Streamlit (TTL = 3600 s) e o cache local do SDK (`~/.dw/cache/`).

Quando o cache local gera conflitos ("already exists"), o código trata a exceção, remove o diretório corrompido e orienta o usuário a recarregar a página, como mostra o código 4.12.

Código 4.12 – Tratamento de erro e limpeza do cache local

```python
except Exception as e:
    if "already exists" in str(e):
        cache_path = os.path.expanduser("~/.dw/cache/dbacademic/dbacademic")
        if os.path.exists(cache_path):
            shutil.rmtree(cache_path)
            st.warning("Cache local limpo automaticamente. Recarregue a pgina.")
    st.error(f"Erro ao consultar cursos por universidade: {str(e)}")
    return pd.DataFrame(), ""
```

A interface ainda oferece um botão "Atualizar Base de Dados", que limpa o cache do Streamlit (`st.cache_data.clear()`) e força um `st.rerun()`, garantindo dados atualizados.

As consultas também recebem parâmetros vindos da interface (como o estado

escolhido pelo usuário). A interpolação é feita com `f-strings`, controlando a inserção do valor e normalizando comparações no SPARQL com funções como `lcase(str(?estado))`. O código 4.5 da subseção anterior exemplifica uma consulta SPARQL com estado dinâmico — encapsulada em uma string Python.

Após a execução, os `DataFrames` passam por limpeza, agregações e derivação de campos (rankings, percentuais, categorias) antes de alimentar gráficos Plotly e tabelas interativas no Streamlit.

Alem disso, visando tratar possíveis erros dessa conexão, o algoritmo inclui mensagens amigáveis via `st.error` e `st.warning`, informando indisponibilidade de endpoints externos ou limites de requisição, de forma a orientar o usuário.

Conjuntamente, os elementos lsitados nesta subseção tornam a conexão Python + API data.world totalmente funcional e sustentável, ancorando todas as camadas superiores do protótipo, do enriquecimento semântico à visualização no Streamlit.

4.1.3 Construção do *dashboard* com Streamlit

A construção do *dashboard* foi realizada em três módulos principais (`Home.py`, `Cursos.py` e `Docentes.py`), organizados para separar as responsabilidades de página inicial, análises de cursos e análises de docentes. Cada módulo inicia configurando a página com o comando `st.set_page_config`, definindo título, ícone e layout amplo para melhor aproveitamento horizontal do espaço, conforme o trecho a seguir:

Código 4.13 – Configuração inicial da página no Streamlit

```
st.set_page_config(
page_title="Painel Acadmico Brasileiro",
layout="wide"
)
```

O núcleo funcional do *dashboard* baseia-se em consultas SPARQL locais e federadas que integram duas fontes semânticas: DbAcademic e DBpedia. Cada função de aquisição é decorada com `@st.cache_data(ttl=3600)`, garantindo cache por uma hora e reduzindo a latência e a carga sobre o endpoint. Um exemplo é mostrado no código 4.14.

De forma semelhante, o módulo de docentes utiliza consultas federadas para recuperar o estado e atributos adicionais, como mostrado no código 4.7.

Após a recuperação, os *DataFrames* passam por funções de processamento que padronizam tipos numéricos, ordenam e derivam métricas adicionais. No módulo de cursos, por exemplo, a função `process_universidade_data` calcula posição, percentual relativo e categoriza universidades por faixas de quantidade de cursos, como mostrado no código 4.15.

Código 4.14 – Função Python para consultar cursos por universidade

```
@st.cache_data(ttl=3600)
def get_cursos_por_universidade():
ds = dw.load_dataset('dbacademic/dbacademic', force_update=True)
sparql_query = """
prefix ccso: https://w3id.org/ccso/ccso#
PREFIX dbp: http://dbpedia.org/property/
PREFIX dbo: http://dbpedia.org/ontology/
prefix owl: http://www.w3.org/2002/07/owl#
SELECT ?Universidade (count (DISTINCT ?s) as ?Cursos) where {
    ?s a ccso:ProgramofStudy.
    ?s ccso:belongsTo ?url_pt.
    ?url_pt owl:sameAs ?url_eng.
    service <http://dbpedia.org/sparql> { ?url_eng dbp:name ?Universidade. }
}
GROUP BY ?Universidade
ORDER BY DESC (?Cursos)
"""
results = dw.query('dbacademic/dbacademic', sparql_query, query_type='sparql')
return results.dataframe, sparql_query
```

Código 4.15 – Processamento de dados de universidades

```
df['Cursos'] = pd.to_numeric(df['Cursos'], errors='coerce')
df = df.sort_values('Cursos', ascending=False).reset_index(drop=True)
df[' Posio '] = range(1, len(df) + 1)
df['Percentual'] = (df['Cursos'] / df['Cursos'].sum() * 100).round(2)
df['Categoria'] = pd.cut(
    df['Cursos'], bins=[0, 5, 15, 50, float('inf')],
    labels=['Pequena', 'Mdia', 'Grande', 'Muito Grande']
)
```

No módulo de docentes, funções como `process_degree_data` e `process_estado_data` convertem colunas em valores numéricos e calculam percentuais de distribuição por formação ou estado:

Código 4.16 – Processamento de dados de docentes

```
df['Docentes'] = pd.to_numeric(df['Docentes'], errors='coerce')
df['Percentual'] = (df['Docentes'] / df['Docentes'].sum() * 100).round(2)
```

No algoritmo, há, ainda, um enriquecimento semântico por meio do mapeamento de universidades para regiões brasileiras (função `mapear_regiao_brasil`), que reconhece nomes em português e em inglês, para que o reconhecimento de uma universidade pela sua região seja realizado após a extração de dados, pois nas bases de dados não está presente esta informação.

A navegação entre páginas utiliza um controle de rádio na barra lateral para selecionar diferentes perspectivas analíticas, como mostram os códigos 4.17 e 4.18.

Padrão semelhante é aplicado para docentes. A cada seleção, blocos condicionais (`if page == ...`) estruturam cabeçalhos, exibem indicadores de carregamento e validam os dados com `st.stop()` em caso de falha.

Código 4.17 – Navegação por rádio na barra lateral para análises de Cursos

```
page = st.sidebar.radio(
    "Escolha uma anlise:",
    ["Panorama Universitrio", "Ranking de Cursos", "Engenharias por Estado", "Engenharia de Computao"]
)
```

Código 4.18 – Navegação por rádio na barra lateral para análises de Docentes

```
st.sidebar.title(" Navegao ")
page = st.sidebar.radio(
    "Selecione uma anlise:",
    ["Docentes por Estado", "Docentes por Formao", "Anlise Combinada", "Anlise por  Gnero "]
)
```

As visualizações são implementadas com Plotly Express (`px`), explorando gráficos de barras horizontais, histogramas, *pie charts* e comparações agrupadas (Código 4.19).

Código 4.19 – Gráfico de barras horizontal com Plotly

```
fig_universidades = px.bar(
    top_universidades,
    y='Universidade', x='Cursos', orientation='h', color='Regio',
    title='Ranking de Universidades por Nmero de Cursos', text='Cursos'
)
fig_universidades.update_traces(texttemplate='%{text}', textposition='outside')
st.plotly_chart(fig_universidades, use_container_width=True)
```

Para docentes, distribuições por formação utilizam gráficos de barra e pizza, para que o usuário absorva facilmente as informações:

Código 4.20 – Distribuição de docentes por formação

```
fig_degree_bar = px.bar(df_degree, x='GrauFormacao_Formatado', y='Docentes', text='Docentes')
fig_degree_pie = px.pie(df_degree, values='Docentes', names='GrauFormacao_Formatado')
```

Métricas sumarizadas no topo das páginas são exibidas com `st.metric`, como por exemplo na página de Cursos em que apresenta-se totais, médias e máximos, como mostra abaixo o código 4.21.

Código 4.21 – Consultando cursos de Engenharia de Computação

```
st.metric("Total de Cursos", f"{total_cursos:,}",
          help=" Nmero  total de cursos cadastrados na base")
st.metric("Universidades", f"{total_universidades:,}",
          help="Universidades com cursos cadastrados")
st.metric("Mdia por Universidade", f"{media_cursos:.1f}",
          help=" Nmero  mdio  de cursos por universidade")
st.metric("Mximo por Universidade", f"{int(max_cursos):,}",
          help="Maior nmero de cursos em uma universidade")
```

Além disso, filtros interativos permitem refinar análises por região, quantidade, texto livre, atributos demográficos, dentro outros. O código 4.22 presente na página de

cursos ilustra o uso desses filtros com `selectbox` e `slider`.

Código 4.22 – Filtros interativos para cursos

```
filtro_regiao = st.selectbox("Filtrar por Regio:", ['Todas'] + sorted(df_universidade['Regio'].unique()))
top_n = st.slider("Quantidade para exibir:", 5, min(100, len(df_universidade)), 25)
```

O ranking de cursos também combina filtros quantitativos e busca textual:

Código 4.23 – Busca textual e filtro quantitativo

```
min_cursos = st.number_input("Mnimo de ofertas:", min_value=1, value=1)
busca_curso = st.text_input("Buscar curso:", placeholder="Digite parte do nome...")
```

No módulo de docentes, o filtro por gênero, assim como os outros filtros listados, redefine a agregação antes de gerar gráficos comparativos.

Logo, após a demonstração das ferramentas utilizadas, a construção do *dashboard* em Streamlit tornou-se eficaz ao integrar, de forma modular, a camada de obtenção de dados através de consultas SPARQL locais e federadas, o processamento estatístico em `pandas` e a apresentação interativa com Plotly. Após enriquecer semanticamente os dados via algoritmo, ou seja, acrescentando mapa de regiões, normalizações e métricas derivadas, o protótipo entregou visualizações claras e comparáveis, permitindo explorar cursos e docentes sob múltiplas óticas.

4.2 Publicação

A etapa de publicação do protótipo consistiu em transformar o código desenvolvido localmente em um serviço web acessível publicamente, utilizando a infraestrutura gratuita do *Streamlit Community Cloud*. O processo inicia-se com a organização do repositório em um sistema de controle de versão (neste projeto utilizou-se o GitHub), contendo os três módulos principais do aplicativo (`Home.py`, `Cursos.py` e `Docentes.py`), além dos arquivos auxiliares necessários para a reprodução do ambiente. A plataforma do Streamlit monitora o repositório e, a cada alteração na branch configurada, realiza o *build* automático da aplicação e a disponibiliza no endereço gerado.

Para que o ambiente remoto replique fielmente as dependências utilizadas no desenvolvimento, incluiu-se no repositório um arquivo `requirements.txt` com as versões das bibliotecas críticas, garantindo reprodutibilidade e evitando incompatibilidades. Um exemplo simplificado desse arquivo é apresentado a seguir:

Código 4.24 – Exemplo de arquivo `requirements.txt`

```
plotly==5.20.0
pandas==2.2.1
requests==2.31.0
datadotworld==1.8.5
```

A autenticação junto ao Data.World exige a definição de um token privado. Em ambientes públicos, esse segredo não deve ser versionado. No Streamlit Cloud, a prática recomendada consiste em registrar o token na interface de *Secrets* do painel de administração do aplicativo, o que o torna disponível no ambiente como variável de configuração.

Neste projeto, o token não foi armazenado em arquivos de configuração (como `secrets.toml`); ele foi definido manualmente no terminal, exportando a variável de ambiente antes de iniciar a aplicação. Assim, o valor não fica versionado no repositório e permanece fora do código-fonte público. Um exemplo do procedimento adotado está presente no código 4.25 abaixo.

Código 4.25 – Definição do token via terminal

```
export DW_AUTH_TOKEN="SEU_TOKEN_AQUI"
streamlit run Home.py
```

A biblioteca `datadotworld` lê automaticamente a variável `DW_AUTH_TOKEN`. Porém, caso necessário, o código pode apenas garantir sua presença com:

Código 4.26 – Leitura opcional do token no código

```
import os
os.environ.setdefault("DW_AUTH_TOKEN", os.getenv("DW_AUTH_TOKEN"))
```

Após o envio do repositório, a configuração inicial na plataforma é realizada especificando-se o repositório, a branch e o arquivo Python de entrada (por exemplo, `Home.py`). A própria interface do Streamlit Community Cloud fornece um console de logs que exibe o andamento do *build*, mensagens de instalação de dependências e eventuais erros de importação, permitindo correções rápidas sem a necessidade de acesso ao servidor. O comportamento de cache adotado na aplicação (`@st.cache_data`) continua válido em produção: os resultados das consultas são armazenados em memória no contêiner da sessão até o tempo de expiração definido, reduzindo latência e chamadas redundantes ao endpoint SPARQL. Sempre que um novo deploy é disparado, o contêiner é reiniciado e o cache é naturalmente invalidado.

Durante a operação pública, a aplicação precisa lidar com eventuais indisponibilidades dos serviços externos (Data.World ou DBpedia). Por essa razão, o tratamento de exceções implementado localmente com `try/except`, mensagens via `st.error` e limpeza automática de cache quando necessário, permanece essencial em produção, pois quando uma falha é detectada, a mensagem é exibida ao usuário e um aviso orienta a recarga da página, mitigando a experiência negativa sem expor detalhes sensíveis do backend.

Outra particularidade do *Streamlit Community Cloud* é a política de alocação de recursos: instâncias ociosas podem ser pausadas e reiniciadas sob demanda, conforme a documentação oficial da plataforma (STREAMLIT, 2024). No projeto, essa limitação

não impacta a operação, pois os dados são buscados dinamicamente e tratados apenas em memória, e eventuais exportações realizadas pelo usuário são efetuadas por meio de downloads diretos gerados pelo próprio Streamlit. Além disso, o contexto exige que rotinas computacionalmente custosas sejam cuidadosamente cacheadas e que o aplicativo esteja preparado para inicializações frequentes. O uso das funções de cache (`st.cache_data` e, quando pertinente, `st.cache_resource`), já descritas nesta seção, foi, portanto, decisivo para manter o tempo de resposta em níveis aceitáveis.

Concluída a publicação, o endereço gerado (<https://dbacademic.streamlit.app/>) pode ser adicionado ao trabalho acadêmico como material complementar, permitindo que avaliadores e leitores interajam com os dados em tempo real. Sempre que uma melhoria é incorporada ao repositório, o fluxo de integração contínua da própria plataforma realiza o redeploy automaticamente, garantindo que a versão acessível reflita a última revisão do código. Essa dinâmica reforça o caráter iterativo do protótipo, ao mesmo tempo em que assegura transparência e reprodutibilidade dos resultados apresentados.

5 Resultados

5.1 Página inicial

Ao entrar no sistema, o usuário visualiza uma tela inicial que descreve a proposta analítica da aplicação, destacando a utilização dos conjuntos de dados DBAcademic e DBpedia, essenciais para acessar informações sobre cursos e docentes brasileiros. A partir dessa introdução, o usuário pode explorar as diversas análises por meio do menu lateral.

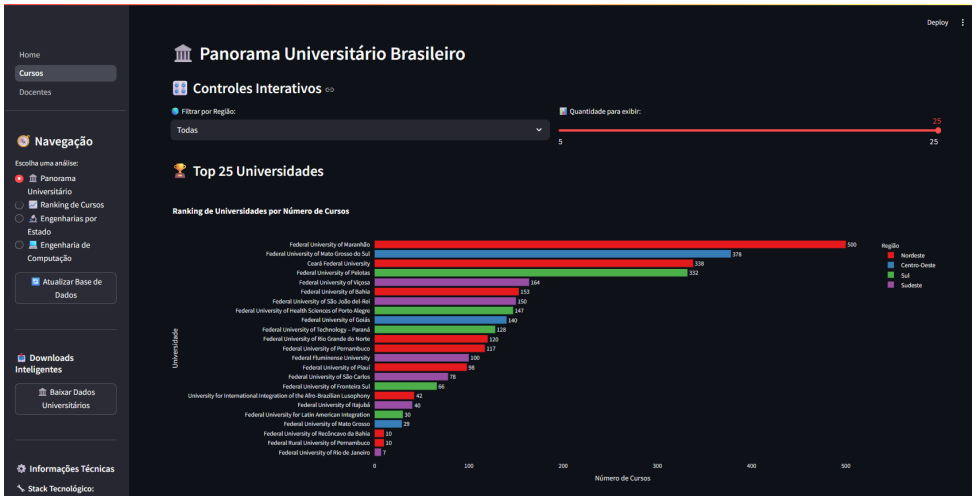
Figura 3 – Página inicial do sistema.

Fonte: Elaborado pelo autor

5.2 Dashboards de cursos

Na página de Cursos, a aplicação oferece várias perspectivas para a exploração dos dados acadêmicos. Em todas as perspectivas, o cabeçalho exibe estatísticas gerais dos dados disponíveis, como total de cursos, número de universidades, média de cursos em cada universidade e maior número de cursos em uma única universidade presente. A primeira análise disponível é o Panorama Universitário, no qual o usuário pode visualizar, por meio de gráficos horizontais interativos, o ranking das universidades brasileiras segundo o número de cursos ofertados. Filtros dinâmicos estão disponíveis para refinar as buscas por região e por limite máximo de universidades exibidas em tela (Figura 4).

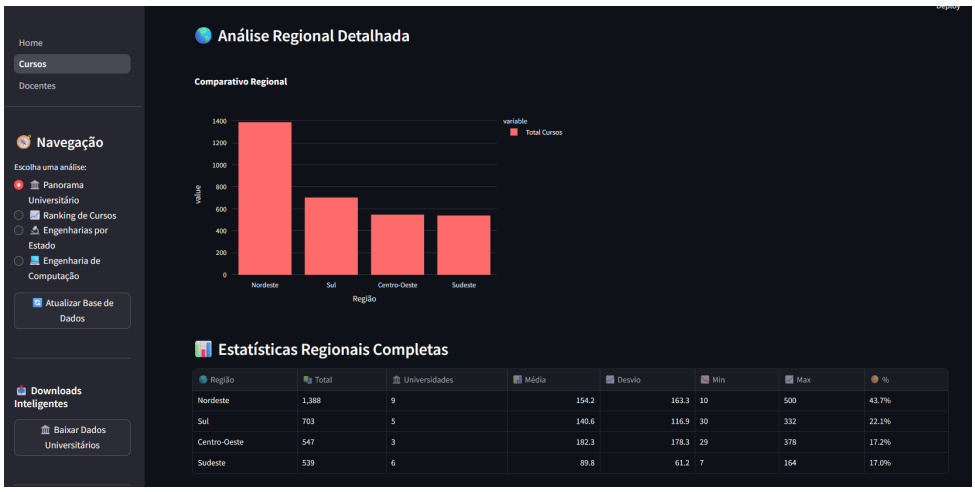
Figura 4 – Panorama Universitário com filtros por região.



Fonte: Elaborado pelo autor

A página de Panorama Universitário em cursos também oferece um *dashboard* que contempla uma análise das estatísticas regionais, apresentando tabelas e gráficos que evidenciam a distribuição de cursos e universidades por região do Brasil. Em tabelas, são exibidos dados como o total de cursos, o número de universidades, médias, desvios, valores mínimos e máximos, além da porcentagem de participação de cada região no cenário nacional. Em gráfico, é exibido apenas o total de cursos por região. Essas informações são apresentadas de forma visualmente simples, utilizando gráficos de barras e tabelas para auxiliar na compreensão do panorama educacional brasileiro (Figura 5).

Figura 5 – Panorama Universitário com análise regional.



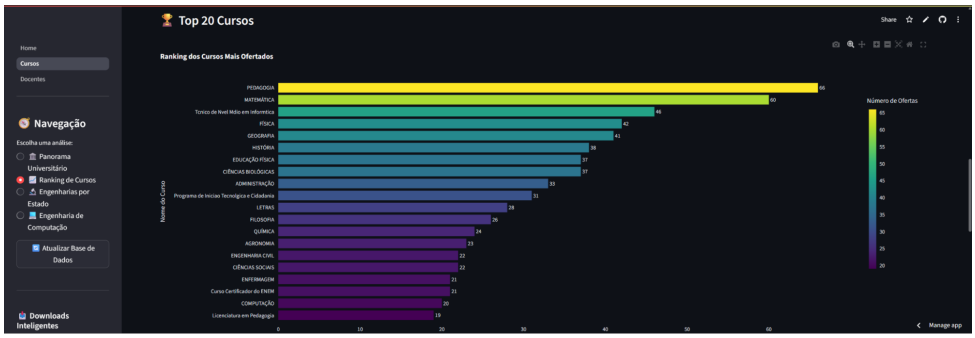
Fonte: Elaborado pelo autor

Ao analisar estas páginas através dos *dashboards*, nota-se que há a ausência de dados abertos de universidades da região norte. Esta ausência segue sem uma justificativa conhecida. Além disso, a região nordeste destaca-se por possuir a maior quantidade de

cursos presentes nas bases de dados, 1.388 cursos, representando 43,7% do total. As regiões sul, centro-oeste e sudeste têm participação relevante, mas possuem números menores registrados.

Ainda na seção de cursos, há o Ranking de Cursos, que permite explorar quais cursos são mais ofertados nacionalmente através de um gráfico interativo. Nesta análise, filtros possibilitam buscas específicas por nome de curso ou pela quantidade mínima de ofertas, acompanhados por tabelas que fornecem informações adicionais sobre frequência e posicionamento dos cursos em um ranking nacional de quantidade de ofertas. Além disso, há um filtro que possibilita ao usuário escolher a quantidade de cursos a serem mostrados em uma única visualização, cujo valor varia entre 5 e 100 cursos. Na Figura 6, é possível identificar que o valor escolhido para a quantidade de cursos a serem mostrados é 20.

Figura 6 – Ranking de cursos configurado para exibir 20 ofertas.



Fonte: Elaborado pelo autor

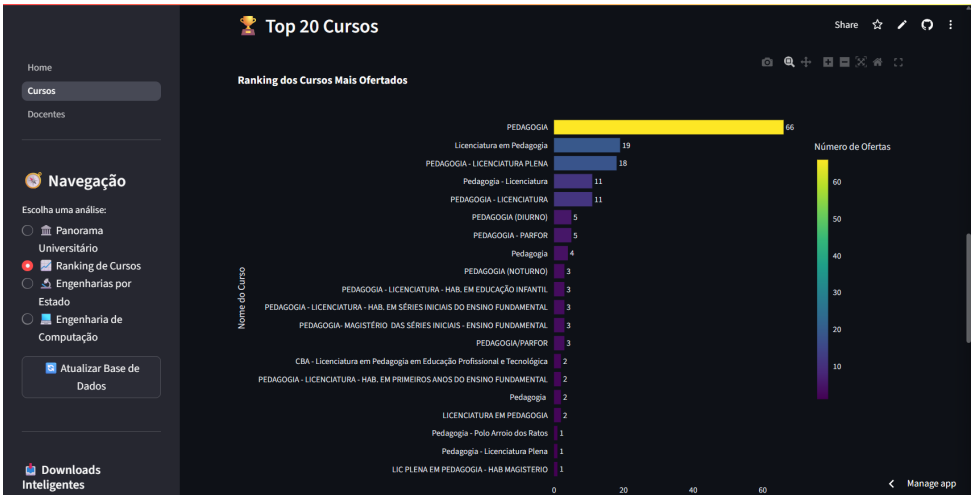
Figura 7 – Ranking de cursos com tabela.

Fonte: Elaborado pelo autor

Nesta análise, percebe-se que o curso de pedagogia tem a maior presença de registros nas bases de dados com 66 ofertas no total, seguido de matemática com 60 ofertas e informática com 46 ofertas. Além disso, se a busca específica por nome de curso for

usada, nota-se que um curso pode apresentar diferentes nomes, como no exemplo do curso de pedagogia nas Figuras 8 e 9.

Figura 8 – Variações nominais do curso de Pedagogia (gráfico).



Fonte: Elaborado pelo autor

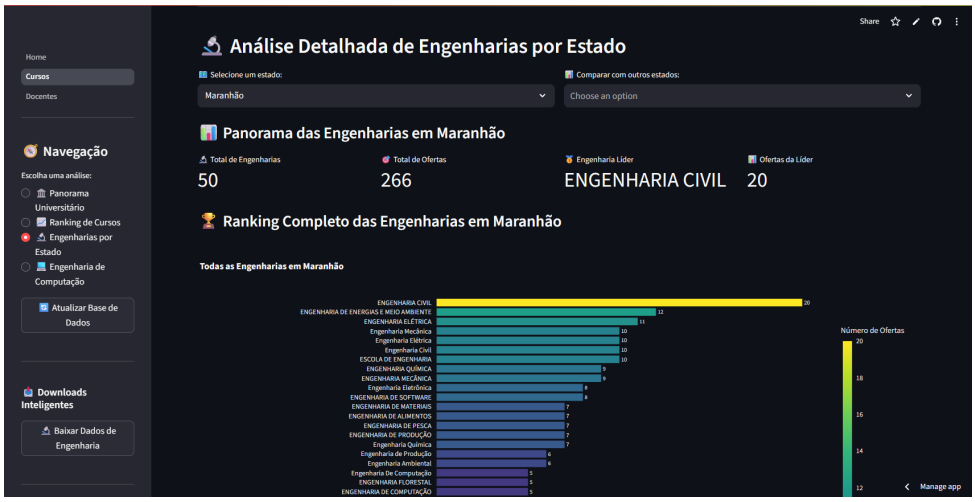
Figura 9 – Variações nominais do curso de Pedagogia (tabela).

Fonte: Elaborado pelo autor

A análise Engenharias por Estado permite ao usuário uma investigação dos cursos de engenharia oferecidos em cada estado selecionado, com gráficos comparativos que facilitam a comparação entre múltiplas localidades. Essa seção também fornece métricas relevantes, como o número total de cursos de engenharia no estado e informações específicas sobre os cursos mais populares e suas respectivas ofertas. Na Figura 10, mostra os filtros previamente ditos e um panorama geral dos dados de cursos de engenharia no estado escolhido, neste caso o Maranhão, que é valor padrão da análise. No panorama geral da análise é possível identificar valores relacionados ao estado escolhido, como o total de cursos de engenharias, total de ofertas para cada curso, engenharia líder de ofertas e a quantidade em que a engenharia líder é ofertada. Além disso, a Figura 11 mostra o

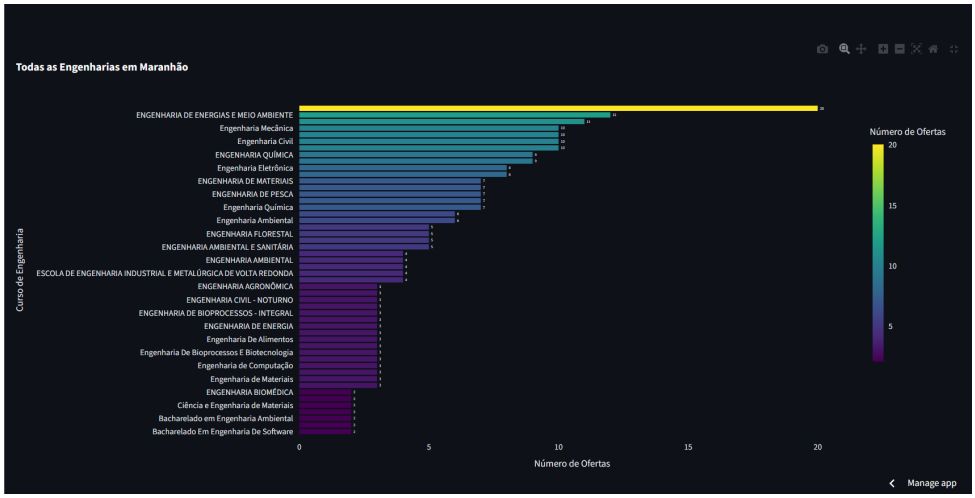
gráfico ampliado para uma melhor visualização neste trabalho, mostrando os 25 cursos de engenharias com mais ofertas do estado. Na Figura 12, pode-se observar uma tabela com os dados do gráfico, mostrando também porcentagens de valores.

Figura 10 – Resumo dos cursos de engenharia no Maranhão.



Fonte: Elaborado pelo autor

Figura 11 – Os 25 cursos de engenharia com maior número de ofertas no estado.



Fonte: Elaborado pelo autor

Figura 12 – Tabela com percentuais por curso de engenharia.

Fonte: Elaborado pelo autor

Nesta análise, realizada com 50 cursos que totalizam 266 ofertas, o *dashboard* aponta que o curso de engenharia civil destaca-se por possuir mais ofertas no estado do Maranhão, totalizando 20 ofertas, dentre os dados presentes na base de dados disponível.

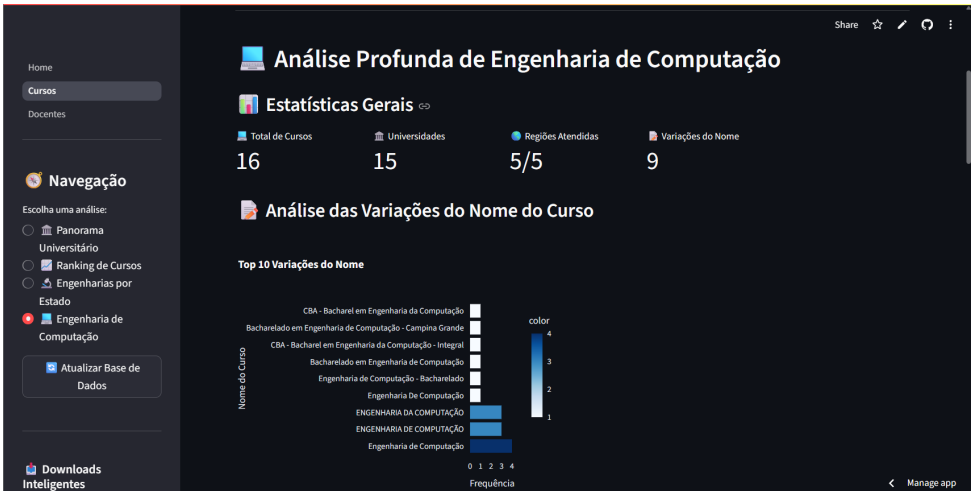
Uma seção exclusiva dedicada à Engenharia de Computação fornece uma listagem dos cursos oferecidos por diferentes universidades, permitindo um olhar diferente sobre essa área específica.

A primeira tela, exibida na Figura 13, traz uma análise dos cursos de Engenharia de Computação, destacando estatísticas gerais como o total de cursos, universidades, regiões atendidas e variações de nome do curso. O gráfico de barras mostra as principais variações nominais encontradas, evidenciando a diversidade de nomenclaturas utilizadas pelas instituições, o que pode ser relevante para estudos sobre padronização e reconhecimento dos cursos.

A segunda tela, mostrada na Figura 14, exibe a Distribuição Geográfica dos cursos, utilizando gráficos interativos para mostrar tanto a proporção de cursos por região do Brasil quanto a concentração regional em relação ao número de universidades. O gráfico de pizza permite identificar quais regiões concentram mais cursos, enquanto o gráfico de bolhas evidencia a relação entre universidades e total de cursos por região, facilitando a comparação entre diferentes contextos regionais.

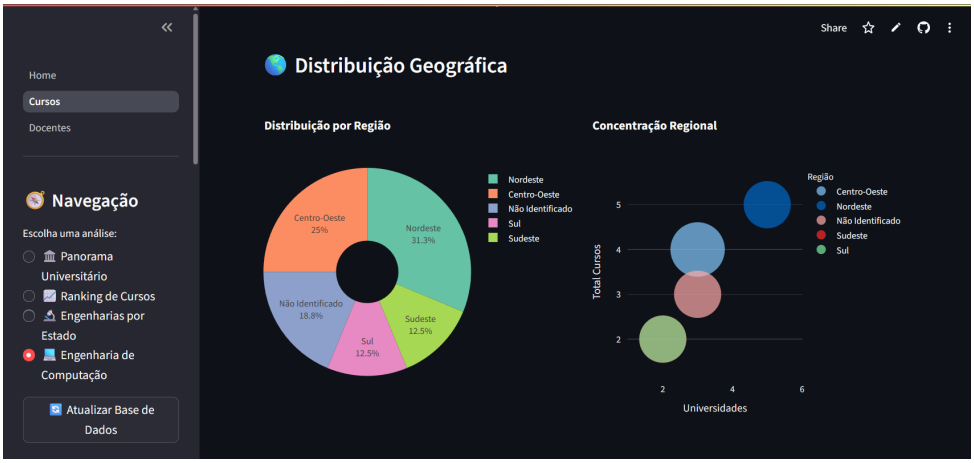
Por fim, a terceira tela, mostrada na Figura 15, apresenta a Base Completa de Dados, onde é possível buscar e visualizar todas as universidades que oferecem Engenharia de Computação, juntamente com a respectiva região. Essa tabela dinâmica permite ao usuário filtrar por universidade, facilitando análises específicas e a identificação de padrões regionais ou institucionais.

Figura 13 – Variações nominais em Engenharia de Computação.



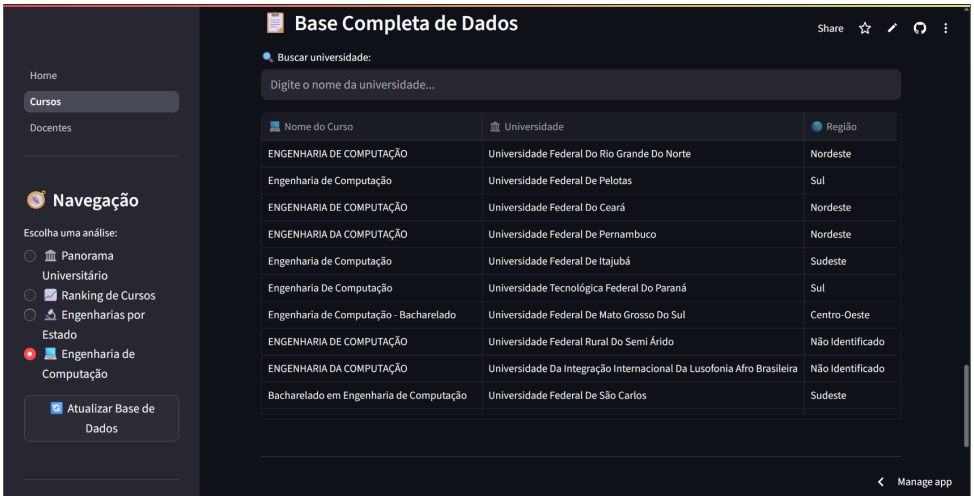
Fonte: Elaborado pelo autor

Figura 14 – Distribuição geográfica dos cursos de Engenharia de Computação.



Fonte: Elaborado pelo autor

Figura 15 – Universidades que oferecem Engenharia de Computação e suas regiões.



Fonte: Elaborado pelo autor

Nesta análise, realizada com 15 universidades, nota-se uma variação com 9 nomenclaturas para o mesmo curso, desmontando que essa diversidade pode dificultar a padronização e a busca por informações consolidadas sobre o curso. Além disso, os cursos de Engenharia de Computação possuem dados de todas as regiões do Brasil, menos da região norte, cujos dados não foram fornecidos, mas há uma concentração maior em algumas regiões, como Nordeste e Centro-Oeste, conforme evidenciado pelo gráfico de pizza e pela análise de concentração regional.

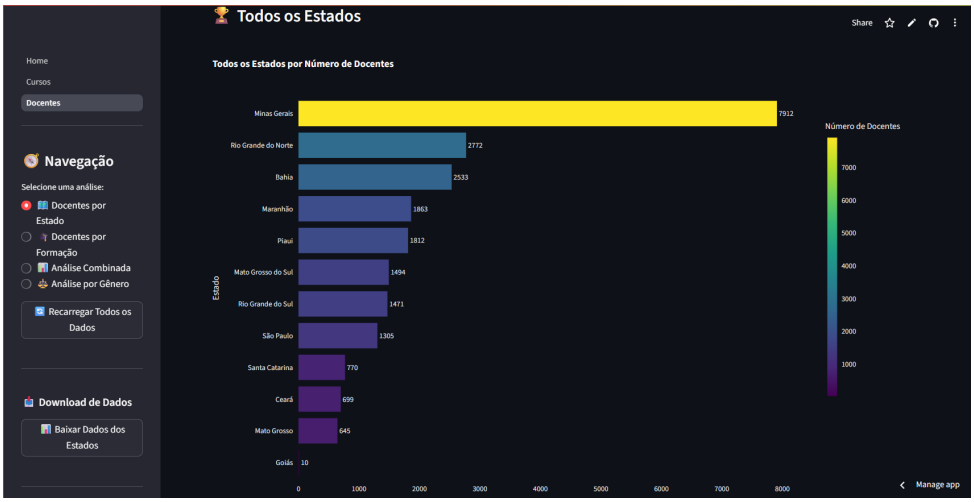
5.3 Dashboards de docentes

A página de *dashboards* de Docentes também apresenta abordagens analíticas. A análise Docentes por Estado mostra, através de gráficos de barras, a distribuição dos docentes por estado com a opção de filtros adicionais por gênero. As métricas principais desta análise oferecem estatísticas básicas sobre o total de docentes e estados com maior concentração acadêmica (Figura 16).

Figura 16 – Métricas principais na análise de docentes por estado.

Fonte: Elaborado pelo autor

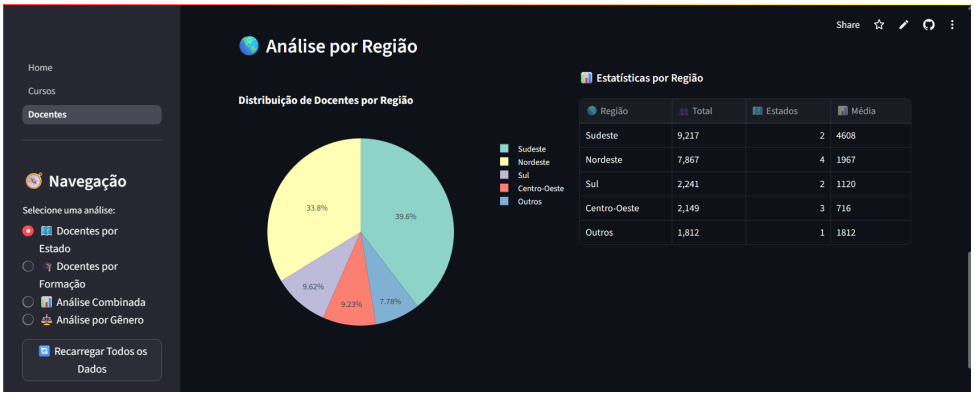
Figura 17 – Distribuição de docentes por estado.



Fonte: Elaborado pelo autor

Além disso, ainda é possível ter uma análise mais precisa com porcentagens exatas da quantidade de docentes por região, como mostra a Figura 18. Com isso, destaca-se 1812 de docentes com região não mapeada. Observando a Figura 19, nota-se que o estado com região não identificada é o Piauí, pois, ainda que nos algoritmos o mesmo esteja apresentando-se como região nordeste quando seu nome é retornado com e sem acento, o estado ainda não é reordenado.

Figura 18 – Distribuição percentual de docentes por estado.



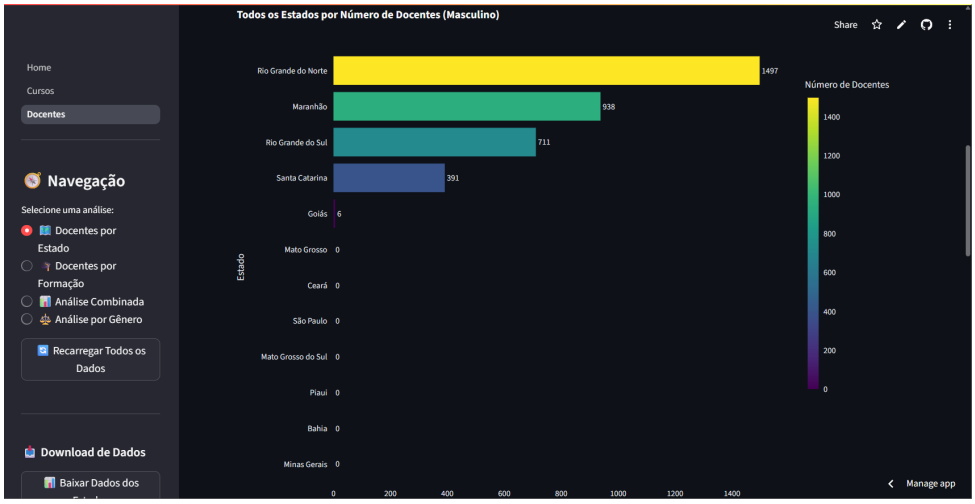
Fonte: Elaborado pelo autor

Figura 19 – Distribuição de docentes por estado e gênero.

Fonte: Elaborado pelo autor

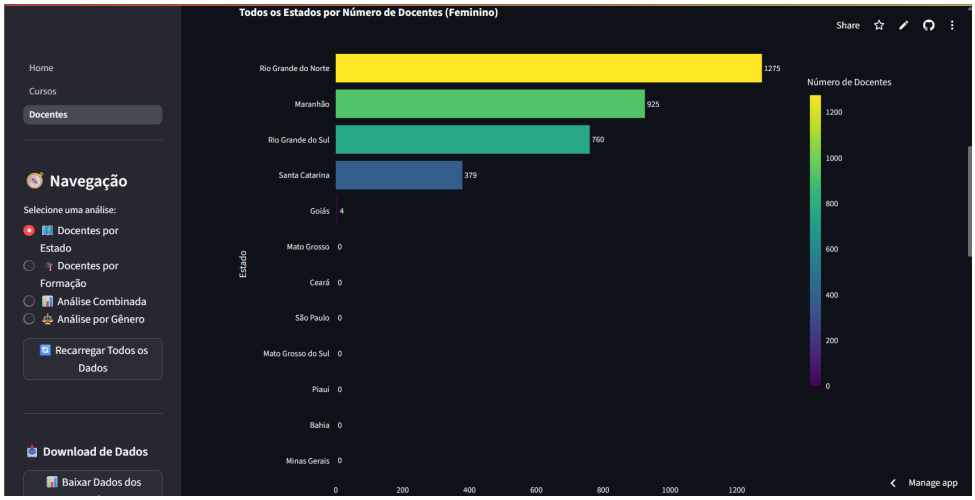
A análise geral com 23.286 docentes de apenas 12 estados mostra como a quantidade mais relevante centraliza-se em Minas Gerais, possuindo 33,98% dos dados disponíveis na base de dados. Porém, ao filtrar os docentes por gênero, como mostrado nas Figuras 20 e 21, nota-se que apenas 5 estados possuem dados que podem utilizar este filtro, que são o Rio Grande do Norte, Maranhão, Rio Grande do Sul, Santa Catarina e Goiás. Com isso, mais uma vez evidência-se como a ausência de dados específicos pode acarretar em uma análise mais rasa sobre o cenário acadêmico brasileiro.

Figura 20 – Docentes do gênero masculino por estado.



Fonte: Elaborado pelo autor

Figura 21 – Docentes do gênero feminino por estado.



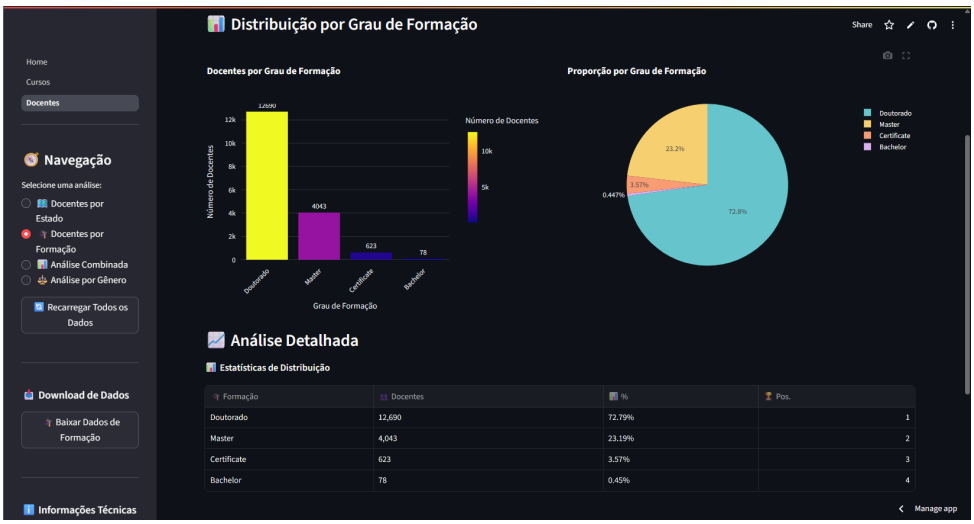
Fonte: Elaborado pelo autor

A análise do perfil dos docentes por formação acadêmica, como mostram as Figuras 22 e 23, também está contemplada no sistema. O usuário pode visualizar o total de docentes, identificar a formação predominante (como Doutorado) e analisar a distribuição por grau de formação por meio de gráficos de barras e pizza. Essa funcionalidade permite avaliar o nível de qualificação do corpo docente de forma clara e objetiva. Nesta análise, realizada com dados de 17.434 docentes, nota-se a quantidade predominante de docentes que possuem doutorado dentro da base de dados, mais especificamente 72,8% do total.

Figura 22 – Métricas gerais de docentes por formação.

Fonte: Elaborado pelo autor

Figura 23 – Distribuição de docentes por grau de formação.



Fonte: Elaborado pelo autor

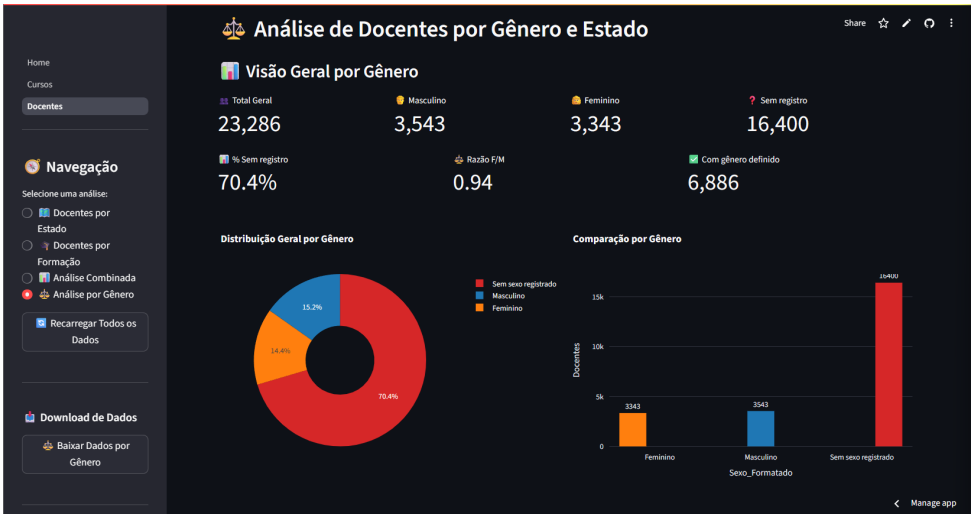
Já a Análise Combinada fornece visualizações cruzadas de estado e grau acadêmico, permitindo ao usuário identificar padrões específicos através de tabelas interativas que possuem filtros por estado, formação e mínimo de docentes, facilitando análises específicas e a identificação de padrões ou lacunas regionais e de formação (Figura 24).

Figura 24 – Análise combinada: estado *versus* formação.

Fonte: Elaborado pelo autor

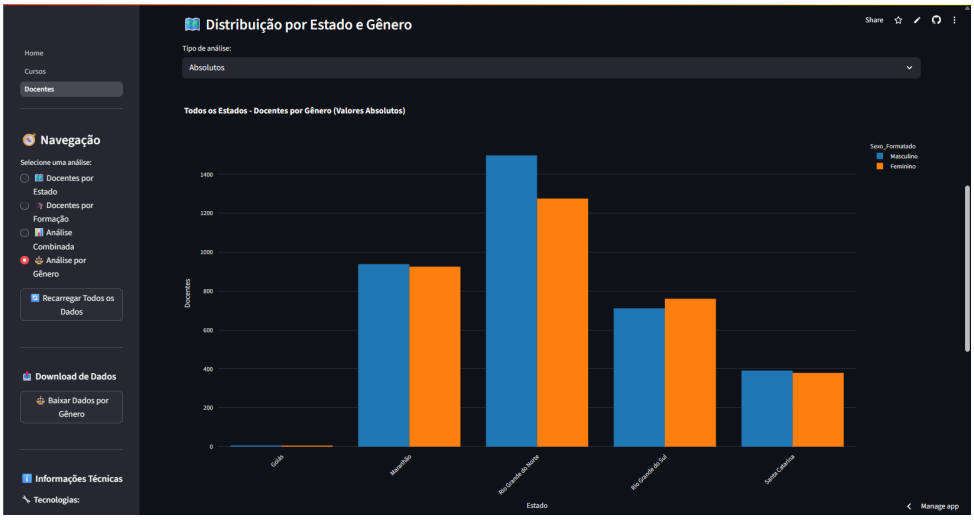
A Análise por Gênero aborda a distribuição de docentes por gênero em cada estado, oferecendo estatísticas básicas sobre questões de equidade de gênero no ambiente acadêmico por meio de gráficos informativos (Figura 25). Ainda na mesma análise, na Figura 26 é possível observar um gráfico de comparação de gênero de docentes por estado, em 5 estados que possuem esses dados, além de uma tabela com mais detalhes sobre todos os estados que fornecem dados sobre os docentes (Figura 27).

Figura 25 – Visão geral da distribuição de docentes por gênero.



Fonte: Elaborado pelo autor

Figura 26 – Gráfico comparativo de gênero em cinco estados.



Fonte: Elaborado pelo autor

Figura 27 – Tabela de docentes por gênero e estado.

Fonte: Elaborado pelo autor

Na visão geral, observa-se que a maior parte dos docentes não possui o gênero registrado (70,4%), o que evidencia uma limitação importante na base de dados e reforça a necessidade de aprimoramento nos processos de coleta e integração dessas informações. Entre os docentes com gênero definido, há um equilíbrio relativo entre homens (3.543) e mulheres (3.343), com uma razão F/M de 0,94, indicando leve predominância masculina, mas sem grande disparidade dos dados disponíveis, como mostra a Figura 25. Além disso, os gráficos de pizza e barras reforçam essa distribuição, mostrando visualmente a grande fatia de docentes sem registro de gênero e a proximidade entre os números de homens e mulheres.

O gráfico e a tabela por estado permitem identificar onde estão as maiores lacunas de registro e onde há maior equilíbrio ou predominância de um dos gêneros. Por exemplo,

apenas na Figura 25, estados como Minas Gerais, Piauí, Mato Grosso do Sul, São Paulo e Bahia apresentam 100% dos docentes sem registro de gênero, enquanto outros, como Rio Grande do Norte, Maranhão, Rio Grande do Sul e Santa Catarina, possuem dados mais completos e mostram percentuais próximos entre homens e mulheres.

Todas as telas do *dashboard* compartilham funcionalidades importantes para melhorar a experiência do usuário, como o cache, implementado via Streamlit, que otimiza o desempenho ao armazenar consultas frequentes e permitir atualizações sob demanda. Além disso, os usuários podem exportar facilmente os dados visualizados em formatos como CSV para uso posterior em relatórios e outras análises. As visualizações interativas e os filtros em tempo real garantem uma experiência dinâmica, facilitando análises personalizados sobre os dados acadêmicos brasileiros.

6 Considerações Finais

O desenvolvimento deste trabalho resultou na criação de um protótipo funcional de *dashboard* interativo voltado à visualização de dados conectados de instituições acadêmicas brasileiras. A aplicação, construída com tecnologias modernas e baseando-se em princípios da Web Semântica, permitiu explorar de maneira acessível e visual os dados disponibilizados pelo repositório DBAcademic, enriquecidos com informações complementares da DBpedia.

A escolha pelo framework Streamlit mostrou-se adequada às necessidades do projeto, oferecendo agilidade na prototipação, integração direta com o ecossistema Python e uma interface declarativa de fácil implementação. A publicação do protótipo na plataforma Streamlit Community Cloud viabilizou o acesso público à aplicação, promovendo sua replicabilidade e democratizando o acesso às informações educacionais.

A utilização da linguagem de consulta SPARQL foi essencial para a extração automatizada e integrada de dados do DBAcademic, garantindo que os painéis estejam sincronizados com as informações mais recentes. As consultas possibilitaram a combinação de múltiplas fontes e o enriquecimento semântico dos dados, ampliando as possibilidades de análise.

Durante o desenvolvimento, no entanto, foram enfrentados alguns desafios relevantes. Inicialmente, foi planejado o uso da DBpedia em português para obtenção de informações complementares sobre universidades e institutos federais. Contudo, ao longo do projeto, tornou-se necessário migrar para a DBpedia em inglês, devido à indisponibilidade do endpoint em português. Essa mudança acarretou em uma redução significativa na quantidade e qualidade das informações disponíveis, uma vez que muitas universidades federais brasileiras, e especialmente os institutos federais, não possuem páginas representativas na Wikipédia em inglês. Em alguns casos, os institutos sequer estão presentes na base, impossibilitando a vinculação semântica.

Outro desafio importante esteve relacionado à própria base DBAcademic. Embora represente um esforço relevante de estruturação de dados conectados no contexto educacional brasileiro, a base ainda sofre com limitações derivadas da escassez de dados abertos publicados por diversas universidades. Essa carência impacta diretamente na profundidade das análises possíveis e restringe o potencial das visualizações no *dashboard*, uma vez que muitas instituições não possuem dados suficientes para compor gráficos mais abrangentes.

Apesar dessas limitações, a solução apresentada neste trabalho cumpre seu objetivo de viabilizar uma interface interativa, acessível e informativa para pesquisadores, gestores e cidadãos interessados nos dados das instituições de ensino superior do país. O *dashboard*, disponível em <https://dbacademic.streamlit.app/>, facilita a exploração visual de dados

conectados, mesmo por usuários sem conhecimento técnico avançado, contribuindo para a transparência e para a geração de insights sobre o cenário acadêmico brasileiro.

Como trabalhos futuros, sugere-se a expansão do sistema com novos tipos de visualizações (como mapas interativos e séries temporais), integração com outras bases de dados abertas e melhorias em usabilidade e acessibilidade. Essas evoluções poderão tornar a ferramenta ainda mais robusta como apoio à análise, à gestão educacional e à pesquisa científica.

Referências

ALLEMANG, D.; HENDLER, J. *Semantic web for the working ontologist - modeling in RDF, RDFS and OWL*. [S.l.]: Elsevier, 2008. ISBN 978-0-12-373556-0. Citado 3 vezes nas páginas 18, 19 e 28.

ALMEIDA, M. I. A. d. *Metodologias de user research em avaliação user centered design: aplicação em contexto empresarial na Altice Labs*. Dissertação (Dissertação de Mestrado em Comunicação Multimédia) — Universidade de Aveiro, Aveiro, Portugal, 2018. Disponível em: <http://hdl.handle.net/10773/24464>. Citado na página 24.

AUER, S.; BIZER, C.; KOBILAROV, G.; LEHMANN, J.; CYGANIAK, R.; IVES, Z. Dbpedia: A nucleus for a web of open data. In: . [S.l.: s.n.], 2007. v. 6, p. 722–735. ISBN 978-3-540-76297-3. Citado 2 vezes nas páginas 15 e 19.

BASS, L.; CLEMENTS, P.; KAZMAN, R. Software architecture in practice. In: _____. [S.l.: s.n.], 2003. ISBN 978-0321154958. Citado na página 31.

BERNARD, J.; SESSLER, D.; KOHLHAMMER, J.; RUDDLE, R. A. Using dashboard networks to visualize multiple patient histories: A design study on post-operative prostate cancer. *IEEE Transactions on Visualization and Computer Graphics*, v. 25, n. 3, p. 1615–1628, 2019. Citado na página 22.

BERNERS-LEE, T. *Linked data*. 2009. Disponível em: <https://www.w3.org/DesignIssues/LinkedData.html>. Citado 2 vezes nas páginas 17 e 18.

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *ScientificAmerican.com*, 05 2001. Citado na página 17.

BIZER, C.; LEHMANN, J.; KOBILAROV, G.; AUER, S.; BECKER, C.; CYGANIAK, R.; HELLMANN, S. Dbpedia - a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, v. 7, p. 154–165, 09 2009. Citado na página 20.

BRASIL. *Decreto nº 8.777*. 2016. <https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2016/decreto/d8777.htm>. Acesso em: 01 Dez. 2023. Citado na página 14.

CARROLL, J. M. *Human Computer Interaction - brief intro*. 2014. Interaction Design Foundation - IxDF. Accessed: 2025-07-11. Disponível em: <https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/human-computer-interaction-brief-intro>. Citado na página 24.

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 na página 14.

DATA.WORLD. *An Introduction to data catalog*. 2024. Acesso em: 11 jul. 2025. Disponível em: <https://docs.data.world/en/312653-an-introduction-to-data-catalog.html#UUID-2a2f0e4f-ebda-298e-3413-29516ce43db6>. Citado 4 vezes nas páginas 25, 27, 28 e 31.

DATA.WORLD. *SPARQL basics*. 2024. <https://docs.data.world/tutorials/sparql-basics>. Acesso em: 25 mar. 2025. Citado na página 14.

FAIOLA, A.; SRINIVAS, P.; DUKE, J. Supporting clinical cognition: A human-centered approach to a novel icu information visualization dashboard. v. 2015, p. 560–569, 2015. Citado na página 22.

FEW, S. *Information Dashboard Design: The Effective Visual Communication of Data*. [S.l.]: O'Reilly, 2006. Citado 5 vezes nas páginas 14, 20, 21, 22 e 23.

FOUNDATION, O. K. open knowledge foundation, 2015. Disponível em: <http://opendatahandbook.org/guide/en/what-is-open-data/>. Citado na página 16.

FURLAN, J. D.; IVO, I. da M.; AMARAL, F. P. *Sistema de Informação Executiva = EIS-Executive Information System: como integrar os executivos ao sistema informacional das empresas, fornecendo informações úteis e objetivas para suas necessidades estratégicas e operacionais*. São Paulo: Makron Books, 1994. Citado na página 21.

GIL, A. C. *Como elaborar projetos de pesquisa*. 5. ed. São Paulo: Atlas, 2010. Citado na página 28.

GROUP, O. K. O. D. *Open definition*. 2015. Disponível em: <https://opendefinition.org/od/2.1/en/>. Citado na página 16.

HARRIS, S.; (EDS.), A. S. *SPARQL 1.1 Query Language*. 2013. World Wide Web Consortium (W3C) Recommendation. Accessed: 2025-07-11. Disponível em: <https://www.w3.org/TR/sparql11-query/>. Citado 3 vezes nas páginas 19, 28 e 31.

HEALY, P. M.; PALEPU, K. G. The fall of enron. *Journal of Economic Perspectives*, v. 17, n. 2, p. 3–26, 2003. Citado na página 21.

HEATH, T.; BIZER, C. *Linked Data: Evolving the Web into a Global Data Space*. [S.l.: s.n.], 2011. v. 11. Citado 2 vezes nas páginas 20 e 28.

HOYT, R. *data.world: Platform for Data Science Collaboration*: Medium. 2018. Acesso em: 11 jul. 2025. Disponível em: <https://rehoyt.medium.com/data-world-platform-for-data-science-collaboration-3daf78b17aef>. Citado 2 vezes nas páginas 26 e 28.

ISOTANI, I. I. B. S. *Em busca da Web do Conhecimento*. [S.l.]: Novatec Editora, 2015. Citado na página 17.

JANES, A.; SILLITTI, A.; SUCCI, G. Effective dashboard design. *Cutter IT Journal*, v. 26, p. 17–24, 01 2013. Citado 2 vezes nas páginas 22 e 23.

KATIS, E.; KONDYLAKIS, H.; AGATHANGELOS, G.; VASSILAKIS, K. Developing an ontology for curriculum & syllabus. In: *Proceedings of the 15th Extended Semantic Web Conference (ESWC 2018) – Posters & Demos*. [s.n.], 2018. Disponível em: <https://2018.eswc-conferences.org/files/posters-demos/paper_246.pdf>. Citado na página 29.

LAUDON, K. C.; LAUDON, J. P. *Sistemas de Informação Gerenciais*. 8. ed. São Paulo: Pearson Prentice Hall, 2007. Capítulo 6: "Infraestrutura de Tecnologia da Informação". Citado na página 21.

LEHMANN, J.; ISELE, R.; JAKOB, M.; JENTZSCH, A.; KONTOKOSTAS, D.; MENDES, P. N.; HELLMANN, S.; MORSEY, M.; KLEEF, P. van; AUER, S.; BIZER, C. Dbpedia: A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, v. 1, n. 1-2, p. 1–25, 2012. Citado 3 vezes nas páginas 19, 20 e 31.

MCKINNEY, W. *Python for Data Analysis*. 2. ed. Sebastopol: O'Reilly Media, 2018. Citado 2 vezes nas páginas 26 e 28.

MENDES, P.; JAKOB, M.; GARCíA-SILVA, A.; BIZER, C. Dbpedia spotlight: Shedding light on the web of documents. In: . [S.l.: s.n.], 2011. p. 1–8. Citado na página 20.

NAHUZ, B. B.; LIMA, A. M.; SILVA, C. D. dos S.; JUNIOR, N.; FILHO, J. da S. P.; COSTA, S. S. Uma abordagem baseada em engenharia de dados para extraÇÃo, transformaÇÃo e carregamento de dados de instituiÇÕes acadÊmicas. In: *Anais do III Simpósio REACT sobre Descarbonização: economia, energia e ambiente*. São Luís (MA), Brasil: Even3, 2023. p. Artigo completo. ISBN 978-85-5722-947-1. Acesso em: 23 jul. 2025. Disponível em: <https://www.even3.com.br/anais/simposioreact2023/678111-uma-abordagem-baseada-em-engenharia-de-dados-para-extracao-transformacao-e-carregamento-de-dados-de-instituicoes>. Citado na página 14.

NIELSEN, J. *Usability Engineering*. 1993. Página oficial do livro na Nielsen Norman Group. Disponível em: <https://www.nngroup.com/books/usability-engineering/>. Citado na página 24.

NIELSEN, J. *10 Usability Heuristics for User Interface Design*. 1994. Updated 2024. Disponível em: <https://www.nngroup.com/articles/ten-usability-heuristics/>. Citado na página 24.

PLOTLY. *Dash Python User Guide*. 2023. <https://dash.plotly.com>. Acesso em: 10 jul. 2025. Citado na página 26.

Plotly Technologies Inc. *Plotly Python Open Source Graphing Library*. 2023. Acesso em: 11 jul. 2025. Disponível em: <https://plotly.com/python/>. Citado 3 vezes nas páginas 25, 26 e 31.

PRESSMAN, R.; MAXIM, B. *Software Engineering: A Practitioner's Approach, 8th Ed.* [S.l.: s.n.], 2014. ISBN 978-0-07-802212-8. Citado na página 28.

RAHMAN, A. A.; ADAMU, Y. B.; HARUN, P. Review on dashboard application from managerial perspective. In: *2017 International Conference on Research and Innovation in Information Systems (ICRIIS)*. [S.l.: s.n.], 2017. p. 1–5. Citado na página 22.

RICHARDS, T. *Streamlit for Data Science*. 2. ed. Birmingham: Packt Publishing, 2023. Citado 4 vezes nas páginas 26, 27, 28 e 31.

SANTOS, A. C. G.; FREITAS, J. A. G. d. Dados abertos e ciência aberta: como as universidades federais brasileiras se apresentam nesse horizonte. *Biblios*, v. 78, p. 1–19, 2020. Disponível em: <https://dialnet.unirioja.es/descarga/articulo/8031014.pdf>. Citado na página 14.

SARIKAYA, A.; CORRELL, M.; BARTRAM, L.; TORY, M.; FISHER, D. What do we talk about when we talk about dashboards? *IEEE Transactions on Visualization and Computer Graphics*, v. 25, n. 1, p. 682–692, 2019. Citado na página 22.

STREAMLIT. *Streamlit documentation*. 2024. Acesso em: 11 jul. 2025. Disponível em: <https://docs.streamlit.io>. Citado 7 vezes nas páginas 25, 26, 27, 28, 31, 32 e 44.

TURBAN, E.; SHARDA, R.; KING, D.; ARONSON, J. E. *Business Intelligence: Um enfoque gerencial para a inteligência do negócio*. [S.l.]: Bookman, 2009. Citado 2 vezes nas páginas 14 e 21.

VERGARA, S. C. *Projetos e relatórios de pesquisa em administração*. 16. ed. São Paulo: Atlas, 2016. Citado na página 28.

WARE, C. *Information Visualization: Perception for Design*. Elsevier Science, 2013. (Information Visualization: Perception for Design). ISBN 9780123814647. Disponível em: <https://books.google.com.br/books?id=qFmS95vf6H8C>. Citado 2 vezes nas páginas 20 e 23.

YIGITBASIOGLU, O.; VELCU, O. A review of dashboards in performance management: Implications for design and research. *International Journal of Accounting Information Systems*, v. 13, p. 41–59, 03 2012. Citado 2 vezes nas páginas 21 e 22.

YU, L. *A Developer's Guide to the Semantic Web*. [S.l.]: Springer, 2011. ISBN 978-3-642-15969-5. Citado 2 vezes nas páginas 17 e 20.