UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
TRABALHO DE GRADUAÇÃO

LUCAS JOB BRITO DE ARAÚJO

Gestão de Conhecimento e suas abordagens para financiamento, tomadas de decisões e métricas de performance

Recife

2025

LUCAS JOB BRITO DE ARAÚJO

Gestão de Conhecimento e suas abordagens para financiamento, tomadas de decisões e métricas de performance

Trabalho de Graduação apresentado ao Centro de Informática da Universidade Federal de Pernambuco, como requisito para obtenção do grau de Bacharel em Engenharia da Computação.

Área de Concentração: Engenharia da Informação

Orientador: Kiev Gama

Recife

2025

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Araújo, Lucas Job Brito de.
Gestão de Conhecimento e suas abordagens para financiamento, tomadas de decisões e métricas de performance / Lucas Job Brito de Araújo. - Recife, 2025.
43 p. : il.

Orientador(a): Kiev Santos da Gama
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Engenharia da Computação - Bacharelado, 2025.
Inclui referências.

1. Gestão de Conhecimento. 2. Transferência de Conhecimento. 3. Extração de dados. 4. Repositórios institucionais. 5. Lattes. 6. Open Science. I. Gama, Kiev Santos da. (Orientação). II. Título.

000 CDD (22.ed.)

LUCAS JOB BRITO DE ARAÚJO

Gestão de Conhecimento e suas abordagens para financiamento, tomadas de decisões e métricas de performance

Trabalho de Conclusão de Curso apresentado ao Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de bacharel em Engenharia da Computação.

Aprovado em: ___/___/_____

BANCA EXAMINADORA

_____

Prof. Dr. Kiev Santos da Gama (Orientador)

Universidade Federal de Pernambuco

_____

Prof. Dr. Ricardo Massa Ferreira Lima (Examinador Interno)

Universidade Federal de Pernambuco

AGRADECIMENTOS

Primeiramente, a meu melhor amigo e Senhor Jesus, a quem tenho o prazer de servir, independentemente do quão indigno eu seja.

Gostaria de agradecer também ao meu orientador e a todos os outros professores por quem tive o prazer de conhecer e ser ensinado, que sempre existe algo novo a ser feito.

Em seguida, a minha família, que permitiu que esta etapa fosse alcançada, e que foram um pilar fundamental para minha formação e desenvolvimento. Aos meus amigos, que sempre estiveram presentes para me exortar com amor e companheirismo, e por fim, a minha namorada, que manteve um firme apoio em momentos cruciais.

RESUMO

A acelerada produção de conhecimento nas organizações voltada à tecnologia resulta em um significativo acúmulo de recursos intelectuais, que impulsionam a inovação e o desenvolvimento. Muito desse conhecimento, no entanto, permanece subutilizado devido à falta de documentação sistemática, publicação e integração em repositórios virtuais. Essa questão é especialmente relevante em centros de inovação sem fins lucrativos, onde a gestão eficaz do conhecimento desempenha um papel importante na obtenção de financiamento através de indicadores Qualis de desempenho. Neste contexto, este trabalho de graduação apresenta o desenvolvimento de um repositório digital integrado à plataforma Lattes, empregando protocolos SOAP para extração e transformação de dados XML em JSON estruturado para armazenamento, análise e indexação. O sistema proposto simplifica a retenção e recuperação de conhecimento, resultando em um ambiente onde contribuições acadêmicas são mais acessíveis, mensuráveis e impactantes. Pela automatização do processo de centralizar *output* acadêmico, a plataforma não só aumenta a visibilidade institucional, como também facilita a tomada de decisão baseada em evidências para a gestão de pesquisa. Além disso, a solução oferece perspectivas de integração futura com ontologias, expandindo seu alcance e usabilidade. Este estudo ressalta a importância de abordagens sistemáticas em engenharia de software para otimizar os processos de gestão do conhecimento, demonstrando como os repositórios digitais podem servir como ferramentas estratégicas para o avanço acadêmico e organizacional.

Palavras-chaves: Lattes. Extração de dados. Repositórios institucionais. Gestão de conhecimento. Acervos digitais. Open Science.

ABSTRACT

The rapid generation of knowledge within technology-oriented organizations often leads to a significant accumulation of intellectual assets. However, much of this knowledge remains underutilized due to the lack of systematic documentation, publication, and integration into research repositories. This issue is particularly relevant in non-profit innovation centers, where effective knowledge management plays a crucial role in securing funding through performance indicators aligned with CAPES and Qualis standards. In this context, this paper presents the development of a digital repository that integrates with the Lattes Platform, employing SOAP protocols for data extraction and transforming XML data into structured JSON for storage, analysis, and indexing. The proposed system streamlines knowledge retention and retrieval, fostering an environment where research contributions are more accessible, measurable, and impactful. By automating the process of aggregating academic outputs, the platform not only enhances institutional visibility but also facilitates informed decision-making in research management. Additionally, the solution offers prospects for future integration with platforms such as DBLP and Google Scholar, expanding its reach and usability. This study underscores the importance of systematic approaches in software engineering to optimize knowledge management processes, demonstrating how digital repositories can serve as strategic tools for both academic and organizational advancement.

Keywords: Lattes. Data Extraction. Institutional Repositories. Knowledge Management. Digital Repositories. Open Science.

SUMÁRIO

1 INTRODUÇÃO

1.1 PROBLEMÁTICA

A produção contínua de conhecimento é um aspecto inerente às atividades profissionais e de pesquisa em centros de tecnologia e inovação. Seja decorrente de requisitos de projetos, inovações impulsionadas por clientes ou do desenvolvimento profissional individual, um volume substancial de produção acadêmica é gerado diariamente. No entanto, uma parte significativa desse conhecimento não se traduz em um ativo tangível para o ecossistema da organização. As principais razões para essa subutilização incluem a ausência de publicações acadêmicas formais, a falta de documentação sistemática em repositórios de pesquisa e as ineficiências inerentes às plataformas existentes, como o Lattes, que apresenta desafios de usabilidade tanto na submissão quanto na recuperação de dados (Ramos et al. 2017; Rosielli e Silva e Marcelo Ferreira e Milton Cinelli e Monique Vandresen 2017), e o SUCUPIRA, conforme apontam outras obras. (C. E. Maciel et al. 2017; Pimentel et al. 2017; C. Maciel, Trierweiller e Ferenhof 2019).

Esse problema é particularmente evidente em instituições de pesquisa sem fins lucrativos, onde a obtenção de financiamento depende da capacidade de demonstrar impacto científico por meio de indicadores de desempenho (*KPIs*) alinhados com *frameworks* de avaliação governamentais, como a CAPES e o Qualis. Ao contrário das empresas com fins lucrativos, que geram receita diretamente a partir de seus serviços e produtos, as sem fins lucrativos dependem fortemente de mecanismos de financiamento externo para sustentar e expandir suas iniciativas de pesquisa. Sem um repositório estruturado e acessível da produção acadêmica, essas organizações enfrentam dificuldades para consolidar e apresentar suas contribuições, limitando, assim, seu potencial para garantir subsídios e fomentar novas iniciativas de pesquisa.

Um estudo de caso relevante desse desafio pode ser observado no Centro de Estudos e Sistemas Avançados do Recife (CESAR), um centro de inovação tecnológica sem fins lucrativos. Atualmente, a gestão do conhecimento dentro da instituição depende de um processo *ad hoc*, no qual funcionários administrativos coletam manualmente as contribuições científicas dos colaboradores e as registram em planilhas. Essa abordagem não apenas é ineficiente, como também desencoraja a participação ativa na disseminação do conhecimento devido à sobrecarga administrativa imposta aos pesquisadores. Além disso, as alternativas existentes, como a Plataforma Lattes, apresentam barreiras de usabilidade, levando a um engajamento

abaixo do ideal dos pesquisadores na documentação formal de suas contribuições.

Para enfrentar esses desafios, este trabalho apresenta o desenvolvimento de um repositório digital projetado para automatizar e otimizar a gestão do conhecimento no CESAR. O sistema proposto integra-se à Plataforma Lattes, utilizando protocolos SOAP para extrair dados e transformá-los em JSON estruturado para armazenamento centralizado, indexação e análise. Ao introduzir mecanismos para agregação automática de dados, classificação e visualização, o repositório aprimora a acessibilidade e usabilidade da produção científica, promovendo uma cultura de compartilhamento de conhecimento e tomada de decisões informadas na gestão da pesquisa.

Além disso, o repositório é concebido como uma ferramenta estratégica para aumentar a visibilidade institucional, tanto internamente — ao fornecer à liderança da organização insights em tempo real sobre a produtividade científica — quanto externamente — ao facilitar a integração com plataformas de indexação acadêmica, como DBLP e Google Scholar. Essa iniciativa não apenas se alinha ao objetivo mais amplo de reforçar a pesquisa como um pilar fundamental das instituições tecnológicas sem fins lucrativos, mas também destaca a importância de abordagens sistemáticas e metodológicas na engenharia de software para a gestão do conhecimento.

Ao enfatizar a gestão estruturada de dados, a indexação automatizada e a agregação de pesquisa orientada por desempenho, este estudo contribui para o debate mais amplo sobre o papel dos repositórios digitais na otimização dos fluxos de trabalho acadêmicos. Em última análise, a solução proposta serve como um modelo para organizações similares que buscam aprimorar suas capacidades de gestão do conhecimento e garantir financiamento sustentável por meio de abordagens estratégicas baseadas em dados.

2 FUNDAMENTAÇÃO

Esta seção tem como objetivo apresentar conceitos que embasem a solução proposta, evidenciando o estado da arte na teoria e na prática.

Os fundamentos da solução proposta se resumem em três princípios: simplicidade, correlação e evidências. A simplicidade foi considerada devido a infraestrutura limitada, para garantir que a adoção e manutenção da ferramenta fossem viáveis mesmo em ambientes com recursos técnicos limitados. A correlação se manifesta na forma como os dados extraídos e organizados se mantém consistentes, interpretando a informação de acordo com outras semelhantes. Por fim, o princípio da evidência fundamenta-se na construção de uma base sólida e verificável de dados, que sustenta análises, relatórios e decisões estratégicas sobre a atuação científica da instituição. Esses princípios serviram como norteadores desde a concepção até a implementação da solução, assegurando coerência metodológica e relevância prática.

2.1 TEÓRICA

Os conceitos chave para a solução proposta são três:

2.1.1 Gestão de conhecimento

A geração e compartilhamento de conhecimento exercem papéis fundamentais na promoção do desenvolvimento de organizações sem fins lucrativos e de ambientes acadêmicos. Colaborações bem-sucedidas incluem troca aberta de informações, suporte na captação de recursos, desenvolvimento de políticas e participação conjunta em eventos acadêmicos. No entanto, desafios como escassez de recursos e diferenças culturais entre instituições dificultam a implementação eficaz de *Knowledge Transfer (KT)* (Jansson et al. 2010, tradução nossa).

Isto auxilia para que descobertas científicas sejam aplicadas em desafios reais. Nos centros voltados a estudos sem fins lucrativos, observa-se uma ampla oferta de bolsas de pesquisa com ênfase na aplicação prática. Em outras palavras, essas instituições — que atuam como ponte entre a comunidade e a produção científica — têm papel essencial na redução das distâncias entre teoria e prática. As pesquisas acadêmicas, por sua vez, frequentemente enfrentam dificuldades para se tornarem aplicáveis, em razão de seu foco predominante na construção

teórica, em detrimento da resolução de problemas concretos. Nesse contexto, evidencia-se um problema de natureza sistêmica na própria lógica de elaboração das pesquisas científicas, cujos resultados nem sempre se traduzem em soluções tangíveis. Por esse motivo, reforça-se a relevância dos centros de pesquisa, que podem atuar como mediadores, adaptando os achados acadêmicos às necessidades da comunidade de forma mais eficaz (Ma, Ovalle e Wang 2023; Jansson et al. 2010, tradução nossa).

Nesse contexto, a atuação de *knowledge brokers* tem se mostrado uma estratégia viável para superar tais barreiras, facilitando a aplicação prática de descobertas científicas em contextos reais.

2.1.2 Indexação de dados e captação de *Key Performance Indicators (KPIs)*

A utilização estratégica da indexação de dados e de indicadores-chave de desempenho (KPIs) como métricas tornou-se indispensável para otimizar a performance organizacional, tanto em setores sem fins lucrativos quanto em centros de pesquisa acadêmica. Esse conhecimento, fundamentado em experiências práticas e observações recorrentes, demonstra que essas ferramentas viabilizam a tomada de decisões orientadas por dados, promovem a responsabilização individual e estabelecem estruturas mensuráveis para avaliar o progresso. A aplicação contínua destes métodos em diferentes contextos reforça sua eficácia, consolidando-os como práticas essenciais para a gestão e evolução dessas instituições.

Ademais, as métricas de performance servem como ferramentas de navegação para organizações que operam em ambientes complexos. No setor sem fins lucrativos, *KPIs* tornam missões abstratas em resultados quantificáveis. Já nas instituições acadêmicas, as métricas de indexação — tais como fator de impacto de periódicos, número de citações, índice H e classificações Qualis — são usadas para aferir a qualidade das pesquisas e seu impacto social. O ponto de convergência de ambos os ambientes reside na ênfase em transparência, otimização de recursos, e aprendizado adaptativo. (Kaganski, Eerme e Tungel 2019, tradução nossa)

O último ponto a ser considerado para a captura de *KPIs* é sua qualidade como evidência para financiamento, geralmente concedido pelo CAPES para instituições de acordo com seu engajamento científico. Zarkesh e Beas (2004) relatam, sua capacidade de definir prioridades na comunidade e de prover uma base para revisão periódica, resultados estes que são desejáveis para organizações num geral.

2.1.3 O estado da arte atual de repositórios digitais

Parte do problema decorre do fato de que uma parcela das evidências costumam ser publicadas em plataformas instituídas pelo governo federal, como o Lattes — quando não estão restritas a eventos acadêmicos ou periódicos científicos. O sistema curricular Lattes permanece como um pilar no rastreio acadêmico; No entanto, sua infraestrutura limitada e os desafios relacionados à usabilidade impõem barreiras sistêmicas à gestão eficaz do conhecimento. Repositórios intermediários surgem como uma alternativa promissora para suprir lacunas críticas no ecossistema de pesquisa no Brasil, enquanto alinhados com requisitos do Qualis, usando implementações técnicas de um sistema *proof-of-concept*.

Ramos et al. (2017, p. 153–164) conclui que "os usuários possuem dúvidas quanto à facilidade de uso do sistema, além de alegarem a necessidade de auxílio para executar as tarefas e sentirem que novos usuários enfrentariam dificuldades para aprender como utilizá-lo".

O acordo de 5 (cinco) anos assinado pela CAPES e ACS, em vigor desde 27 de fevereiro de 2024, converte o contrato de 68 (sessenta e oito) periódicos de *read only* para *read and publish* pelo tempo descrito. (CAPES 2024; ACS 2024)

Esse movimento indica que estão sendo realizados esforços concretos de integração com outros sistemas de indexação e acervos. Assim, ainda que a CAPES atualmente não reconheça publicações armazenadas em um repositório interno, a simples existência de tal acervo demonstra o potencial de investimento institucional e o comprometimento da organização em viabilizar, futuramente, a publicação desses artefatos em veículos reconhecidos e elegíveis.

Ademais, o uso de repositórios institucionais como plataforma de publicação acadêmica, bem como os modelos *open access* baseados nestes repositórios, já vem sendo objeto de estudos em diversas instituições (Ranasinghe e Min 2018).

Nesse contexto, um repositório intermediário pode contribuir para a linearização do processo de gestão e rastreamento das produções acadêmicas, de forma a assegurar que sejam devidamente documentadas e acessíveis para fins de avaliação e elaboração de relatórios.

2.2 TÉCNICA

Esta seção tem o objetivo de apresentar conceitos relevantes para o desenvolvimento deste trabalho, abordando as métricas mais relevantes para os propósitos anteriormente destacados, os meios de obtenção dessas métricas em cenários reais, as possibilidades de tratamento automatizado (ou manual) dos dados coletados, as ferramentas utilizadas no desenvolvimento de sistemas de gestão e as metodologias correlatas.

2.2.1 Aquisição de dados

A aquisição de dados é um passo essencial para consolidar a gestão do conhecimento em instituições acadêmicas e centros de pesquisa. Para garantir a qualidade e a usabilidade dos dados coletados, é necessário adotar estratégias que permitam tanto a automação do processo quanto a participação ativa dos colaboradores na curadoria das informações.

Duas abordagens principais podem ser utilizadas para coletar dados acadêmicos e garantir sua integridade:

- **Aquisição automatizada via Lattes Extractor**: possibilita a extração de informações diretamente da Plataforma Lattes, facilitando a obtenção de um histórico acadêmico consolidado.
  - Utiliza uma API privada, cujo acesso depende de uma solicitação formal, sujeita a trâmites burocráticos (empiricamente observado).
  - Permite maior integração com sistemas acadêmicos preexistentes.
  - Reduz a necessidade de intervenção manual, garantindo maior consistência e padronização dos dados.
- **Aquisição manual via API interna**: oferece uma alternativa viável para organizações que não possuem acesso à API oficial do Lattes ou que buscam reduzir custos operacionais.
  - Representa uma solução menos onerosa para a instituição.
  - Possui menor nível de integração com sistemas acadêmicos já estabelecidos, demandando maior esforço para consolidação e normalização dos dados.

– Segue a abordagem adotada por outros sistemas institucionais de gestão acadêmica. (Ranasinghe e Min 2018; CAPES 2024).

2.2.2 Tratamento e correlação de dados

Independentemente do método utilizado, a higienização dos dados coletados é um fator crítico para garantir a confiabilidade da base de conhecimento. Isso inclui processos como deduplicação, normalização e enriquecimento das informações, assegurando que o repositório gerado sirva como um recurso estruturado para análise e futuras integrações.

Projetos que manuseiam um alto volume de dados, geralmente associados a *machine learning* ou inteligência artificial, possuem procedimentos bem conhecidos no campo. Dentre eles, o pré-processamento de dados se mostra particularmente útil, visto que grandes quantidades de informações são submetidas a diversos intermédios onde algo pode estar fora do padrão, portanto precisam ser filtradas. Faz-se uso do famoso princípio *"garbage in, garbage out"* (Seland 2018), aplicável em diversos contextos, como já relatado por Ortiz et al. (2024, tradução nossa): "[...] seu verdadeiro potencial está na habilidade de confiavelmente traduzir *output* cru para dados de alta qualidade [...]".

Correlacionar dados se refere a capacidade do programa de interpretar uma informação e lidar corretamente com ela, relacionando com outras semelhantes (como por exemplo, um artigo e suas referências, ou seus autores), por meio de algoritmos que testem e redirecionem o destino de forma automatizada sem o intermédio humano. Cuidado e atenção nesta parte trazem benefícios a longo prazo, pois os mesmos dados precisarão ser modelados e associados entre tabelas, como é de praxe na construção de bancos de dados.

Portanto, a adoção de uma estratégia híbrida, combinando a extração automatizada com a inserção manual, pode incentivar maior engajamento dos pesquisadores no processo de gestão do conhecimento, promovendo um ecossistema mais colaborativo e sustentável dentro da organização.

O presente trabalho propõe um modelo que não apenas viabiliza essa abordagem, mas também estabelece um *Proof of Concept*, demonstrando sua aplicabilidade e benefícios dentro do contexto institucional.

2.2.3 Ferramentas utilizadas

Para o desenvolvimento do repositório digital, foi adotada uma abordagem fundamentada em boas práticas de engenharia de software, priorizando modularidade, escalabilidade e integração com sistemas acadêmicos. Esta seção demonstra as ferramentas utilizadas, os motivos para tal escolha, e alternativas viáveis.

2.2.3.1 Tecnologias e Ferramentas

A implementação utilizou um conjunto de tecnologias voltadas para a eficiência e interoperabilidade:

2.2.3.1.1 Backend

Para o backend, utilizou-se o FastAPI, um *framework* projetado para o desenvolvimento de APIs com foco em desempenho e escalabilidade. Construído sobre a especificação Asynchronous Server Gateway Interface (ASGI), permite requisições assíncronas e fornece documentação automática via Swagger. Sua integração nativa com o Pydantic também foi determinante, pois oferece validação robusta e tipagem forte — ideal para entrada e saída de dados estruturados, especialmente em interações com APIs externas, como a do Lattes. O framework foi utilizado em conjunto com o SQLAlchemy, integrando uma camada de abstração para as transações com o banco de dados, recurso útil diante da complexidade das relações envolvidas nas sessões.

Outras alternativas foram consideradas, como Django e Flask. Embora viáveis, apresentaram limitações no contexto deste projeto:

- *Features* essenciais, como ORM e validação de formulários, não são nativos ao Flask, exigindo extensões que poderiam aumentar a complexidade e retardar o desenvolvimento.
- Embora Django seja bastante completo, sua curva de aprendizado é ligeiramente maior que FastAPI. Como o contexto do projeto dependia de rápido desenvolvimento, um período curto de rampagem impossibilitou o uso do *framework*. Uma migração futura foi cogitada, visto que a aplicação tem um alto potencial de escalar.

2.2.3.1.2 Frontend

Visando uma interface reativa e ágil prototipagem, condizentes com a proposta de um acervo digital acessível, optou-se pelo desenvolvimento de uma aplicação web utilizando Next.js com TypeScript. Além de oferecer renderização híbrida, - ou seja, via Gerador Estático de Site (SSG) e Renderização *Server-Side* (SSR) -, o *framework* contribui para a Otimização de Motor de Busca (SEO), abstrai processos como *bundling* [1] e é feito sob React, dispondo de suas rotinas ágeis de implementação front-end. Em comparação com outras ferramentas como Vue/Nuxt ou o próprio React puro, o Next.js oferece suporte nativo a SSR e roteamento, além de contar com uma comunidade mais consolidada. (StackOverflow 2024).

2.2.3.1.3 Controle de Acesso

A autenticação foi implementada via "Sign in with Google", estratégia que reduz o atrito de entrada para usuários — especialmente pesquisadores com e-mails institucionais da Google — e elimina a necessidade de gerenciamento de senhas. A autenticação é delegada ao provedor OAuth2, com integração nativa ao FastAPI. A comunicação é baseada no padrão JSON Web Token (JWT), que oferece segurança adequada sem grandes *overheads* adicionais.

2.2.3.1.4 Armazenamento

O banco de dados adotado foi o PostgreSQL, uma solução relacional de código aberto que suporta eficientemente dados estruturados e semi-estruturados, como JSON e JSONB[2], possibilitando consultas otimizadas sobre a produção acadêmica indexada.

Pesquisas indicam que, embora o MongoDB tenha boa performance teórica devido à sua complexidade $O(1)$, o PostgreSQL apresenta desempenho competitivo, com complexidades $O(n)$ e $O(1)$ em diversos contextos. Já soluções como o Neo4j apresentam desempenho inferior, com complexidade $O(n \log n)$. No escopo deste projeto, com até mil registros esperados, o PostgreSQL mostrou-se a opção mais eficiente (Wiseso, Imrona e Alamsyah 2020).

[1] Uma técnica de desenvolvimento *web* que combina múltiplos arquivos em um ou alguns arquivos otimizados, reduzindo o tempo de carregamento.

[2] Segundo sua própria documentação.

2.2.3.1.5 Containerização e Deploy

Utilizou-se Docker para a containerização dos serviços e Docker Compose para orquestração local, permitindo escalabilidade e replicabilidade da aplicação. A integração com GitLab e seu suporte a *pipelines* de Integração Contínua (CI) e Entrega Contínua (CD) foi essencial, considerando as necessidades do projeto.

Embora o Kubernetes seja uma alternativa robusta com foco em escalabilidade, foi considerado excessivo para o estágio atual da aplicação, que ainda não demanda esse nível de complexidade.

2.2.4 Metodologia de Desenvolvimento

A metodologia adotada seguiu princípios ágeis, com entregas incrementais e validação contínua ao longo do projeto.

Foram utilizadas práticas inspiradas no Scrum, adaptadas à realidade do time. Embora os papéis formais e cerimônias não tenham sido seguidos integralmente, os pilares de comunicação constante, entregas iterativas e adaptação foram priorizados.

O ciclo de trabalho foi organizado em ciclos trimestrais, com definição de metas macro para cada período. Dentro desses ciclos, eram realizadas reuniões semanais com a equipe — composta por uma pessoa responsável pelo produto (product owner), uma designer e uma gerente de projeto — nas quais eram apresentados os avanços obtidos, desafios enfrentados e os próximos passos planejados.

O desenvolvimento técnico da arquitetura, implementação e testes foi conduzido individualmente. Como apoio, contaram-se mentorias quinzenais voluntárias com profissionais da área, responsáveis por orientar decisões arquiteturais e validar soluções de maior complexidade.

Essa estrutura híbrida, combinando princípios ágeis com autonomia técnica, garantiu um fluxo contínuo de entregas, permitiu aprendizado iterativo e facilitou ajustes de escopo quando necessário. A validação semanal com *stakeholders* foi fundamental para priorizar funcionalidades e manter o alinhamento com os objetivos do projeto.

2.2.5 Padronização de Arquitetura e Design Patterns

Durante o desenvolvimento do sistema, adotamos padrões de projeto (**Design Patterns**) amplamente reconhecidos para garantir modularidade, reutilização e escalabilidade, além dos inferidos via *frameworks* utilizados. Os padrões utilizados incluem:

- **Adapter e Strategy**, para compilar dados externos e permitir diferentes fontes de aquisição de dados, como uma API externa ou inserção manual.
- **Factory Method e Composite**, amplamente usado em grandes *frameworks front end*, como *React*, por meio de componentes pais e filhos, todos derivados de um bloco fundamental de elemento.
- **Observer Pattern** para propagação de eventos internos ao sistema, como no caso de uma atualização periódica de artigos, garantindo atualização de registros relacionados de forma automatizada.
- **Mediator** para centralizar comunicações e evitar ambiguidades e demais riscos a segurança da aplicação.

A aplicação destes conceitos podem ser melhor compreendidos sob a perspectiva de Whitley (1997 apud Zhang e Budgen 2011, tradução nossa), que “um padrão comum na pesquisa em engenharia de software é o desenvolvimento de técnicas de construção de sistemas, como design orientado a objeto, que são fortemente reforçadas na ausência de evidência". Ou seja, é crucial a compreensão de técnicas de desenvolvimento, pois são fundamentais na implementação de software sustentável, pensando em manutenção a longo prazo e mudanças evolutivas (Venters et al. 2018, p. 185, tradução nossa)

3 TRABALHOS RELACIONADOS

A compreensão do estado da arte é fundamental para a validação e o posicionamento de soluções tecnológicas no contexto acadêmico e profissional. Nesta seção, são apresentados e discutidos trabalhos que abordam a extração de dados de acervos institucionais voltados à gestão interna do conhecimento acadêmico em geral, tanto no contexto nacional — via extração da base Lattes (Alexandre D. Alves, Horacio H. Yanasse e Nei Y. Soma 2011; Mena-Chalco e Junior 2009; Alexandre Donizeti Alves, Horacio Hideki Yanasse e Nei Yoshihiro Soma 2011) — quanto internacional, com abordagens semelhantes aplicadas em diferentes realidades, algumas das quais exploram conceitos de Ontologia e OBDA (Lukman et al. 2022; UTM 2024; Salgueiro et al. 2022; Bienvenu et al. 2015).

São analisadas abordagens que tratam da extração, indexação e disseminação de produções acadêmicas, bem como estratégias de incentivo à participação colaborativa na construção de repositórios institucionais. O objetivo é identificar pontos de melhoria deixados por outras iniciativas, incorporar práticas consolidadas e fornecer contribuições relevantes que fundamentem futuras linhas de pesquisa.

Dentre os artigos analisados durante a revisão de literatura, destacam-se três ferramentas que buscam extrair dados da base Lattes: o ***ScriptLattes*** (Mena-Chalco e Junior 2009), um script em Python para geração de relatórios; o ***LattesMiner*** (Alexandre D. Alves, Horacio H. Yanasse e Nei Y. Soma 2011), ferramenta vinculada ao SUCUPIRA, sistema disponibilizado pela CAPES; e dois sistemas de busca baseados na web: (1) *Busc@NIMA* e o mais recente (2) ***Quem@PUC*** (Salgueiro et al. 2022), ambos criados na PUC-Rio com propósitos semelhantes.

O *ScriptLattes* é um sistema *open-source* voltado à criação de relatórios acadêmicos de grupos, com base nos currículos da plataforma Lattes. Publicado em 2009 e construído em *GNU*[1], realizava, à época, pesquisas em massa aparentemente via *web scraping* em formato *HTML*. Contudo, com a introdução de *CAPTCHAs* pelo CNPq em 2015, a funcionalidade de automação foi comprometida. Conforme relatam Corrêa et al. (2017), tal medida dificultou processos automatizados de coleta, levando os autores a indicarem o SUCUPIRA como alternativa — embora este apresente limitações quanto à completude e à eficiência da extração de dados.

No que tange ao SUCUPIRA, sua *homepage*[2] evidencia a ausência de jornadas de usuá-

[1] https://www.gnu.org/home.pt-br.html
[2] https://sucupira.capes.gov.br

rio voltadas a discentes e docentes, sendo o sistema majoritariamente orientado à atuação de gestores e coordenadores. A coleta de dados não ocorre automaticamente: exige inserções manuais, não valida automaticamente as informações e possui restrições temporais para alterações, como nos períodos de coleta anual.

O *LattesMiner*, por sua vez, é uma Linguagem Específica de Domínio (DSL)[3], multilíngue e voltada à extração automática de informações e à identificação de redes acadêmicas. É composto por um conjunto de classes em *Java* e, publicado em 2011, utilizava, aparentemente, método semelhante ao do *ScriptLattes* para extração. Por estar associado ao CAPES e ao SUCUPIRA, acredita-se que tenha sido posteriormente integrado a esse sistema.

O sistema *Quem@PUC* faz uso do Lattes Extractor (LE), API oficial da CAPES, para extração em massa de currículos Lattes, permitindo a inserção e posterior edição de informações em um repositório próprio, conforme descrito por Salgueiro et al. Trata-se de um Sistema de Integração de Informações (IIS) orientado a ontologias — especificamente BIBO[4], BIO[5], VIVO[6] e CCSO[7] — que emprega o *Resource Description Framework (RDF)* como modelo de dados e um banco *NoSQL* de esquema flexível para armazenamento. Os *pipelines* de *Extract, Transform and Load* (*ETL*) são construídos com LinkedPipes, convertendo dados em triplas RDF consultáveis via SPARQL, e são complementados por rotinas de *web scraping* em portais institucionais da PUC-Rio.

A aplicação web, desenvolvida com Python/Flask e servida via Apache, utiliza a API AllegroGraph para gerenciamento de grafos RDF e opera em duas instâncias: uma dedicada ao processamento ETL e outra à camada de apresentação de resultados. Além disso, incorpora um módulo de inteligência artificial para análise de 1.200 produções acadêmicas e integra o Google Cloud Translate para tradução automática.

Apesar dos ganhos significativos na consolidação e visualização de dados de pesquisadores, o *Quem@PUC* ainda enfrenta limitações semânticas. Sua busca é predominantemente sintática, não contemplando sinônimos ou interpretações contextuais dos termos. Por exemplo, uma consulta por *"machine learning"* não retornaria registros com as expressões "ML"ou *"deep learning"*. Adicionalmente, o uso da AllegroGraph API em plano gratuito impõe limites ao tamanho do repositório, exigindo consultas mais complexas que aumentam a latência do

[3] Uma linguagem de programação com um nível mais alto de abstração, otimizada para uma classe específica de problemas, segundo JetBrains (s.d., tradução nossa)
[4] http://purl.org/ontology/bibo/
[5] http://purl.org/vocab/bio/0.1/
[6] https://duraspace.org/vivo/
[7] https://w3id.org/ccso/ccso

sistema.

O processo de geração RDF via LinkedPipes tampouco é totalmente automatizado, exigindo intervenção humana em várias etapas:

- Extração inicial de dados via LE, cuja automação não é documentada;
- Criação e ajuste dos *pipelines*, com mapeamento das colunas de entrada às propriedades RDF via componentes SPARQL Update;
- Validação e monitoramento dos resultados, assegurando a conformidade semântica com as ontologias adotadas.

Ademais, a ausência de contas de usuário limita funcionalidades personalizadas como buscas salvas e alertas, o que representa uma oportunidade para futuras melhorias, especialmente na perspectiva de rastrear indicadores-chave de desempenho (KPIs) para fins de financiamento.

Em suma, todas as plataformas analisadas compartilham características esperadas de acervos digitais institucionais, sendo, portanto, integráveis ou já integradas a outros sistemas. As diferenças entre elas refletem o contexto de origem e os problemas específicos que cada uma busca resolver.

4 METODOLOGIA

A metodologia adotada neste trabalho seguiu uma abordagem aplicada ao contexto do Centro de Estudos e Sistemas Avançados do Recife (CESAR), uma organização sem fins lucrativos, durante aproximadamente 18 (dezoito) meses. Com foco na construção de uma solução funcional baseada em requisitos reais observados no contexto organizacional do CESAR, iniciou-se uma investigação empírica sobre as práticas atuais de gestão do conhecimento técnico-científico, com destaque para a análise dos critérios adotados por órgãos como a CAPES na avaliação de financiamento, baseado nos produtos técnico-tecnológicos e publicações em periódicos ou eventos. A partir disso, foram identificadas oportunidades de automatização, padronização e compartilhamento dos dados acadêmicos, o que norteou a escolha das ferramentas, linguagens e padrões arquiteturais empregados.

Segundo o website *gov.br*, a Avaliação do Sistema Nacional de Pós-Graduação (CAPES 2014) é realizada com um de seus objetivos sendo a certificação de qualidade, que também é referência para a distribuição de bolsas e recursos para o fomento da pesquisa para instituições e ademais.

Segundo a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), são considerados para classificação os seguintes elementos técnico-tecnológicos: (CAPES 2025)

- Produto Bibliográfico
- Ativos de Propriedade Intelectual
- Tecnologia Social
- Curso de formação profissional
- Produto de editoração
- Software/Aplicativo (Programa de computador)
- Evento organizado
- Norma ou Marco regulatório
- Base de dados técnico-científica
- Empresa ou Organização social inovadora

Além disso, também são considerados artigos de periódicos e eventos na área de Computação. Neste caso, utiliza-se o modelo QR1, baseado no melhor percentil de um periódico na mesma, em acervos digitais de *abstracts* e citações. Em sua ausência, um modelo de regressão baseado nas métricas *H5* do *Google Scholar* e *Citescore*, da *Scopus*. Também é analisado o alinhamento das dissertações ou teses com as linhas de pesquisas e área(s). A qualidade também é analisada pelas produções resultantes das dissertações ou teses, e prêmios. (CAPES 2023; CAPES 2025)

Tomando este cenário como padrão para embasar decisões responsáveis de modelagem de dados e métodos de extração e tratamento, as seguintes características precisam ser consideradas:

- Qualquer tipo de produto técnico-tecnológico num geral
- Cursos de formação
- Artigos de excelência (Segundo CAPES 2023) ou potencialmente excelentes, para posterior desenvolvimento.
- Eventos
- Produto de editoração

Além de discriminar tais qualidades, prover informações básicas e derivadas do contexto também é importante, como:

- Nome Colaborador
- Projeto pertencente
- Liderança direta
- Cargo

Tais informações serão levadas em consideração na criação do modelo entidade relacionamento.

Em resumo, um acervo digital para os devidos fins deve ser capaz de discernir tais artefatos e interpretar os *outputs* de acervos já-existentes, por boas práticas de integração, visto que plataformas colaborativas são mais eficazes quando atuam como "espaços convergentes"para unir diversas disciplinas e tecnologias. (Winickoff et al. 2021)

5 DESENVOLVIMENTO

Esta seção descreve as etapas envolvidas no desenvolvimento da ferramenta proposta, abordando desde a concepção da arquitetura até sua implementação e integração com fontes de dados externas. São apresentados os principais componentes da aplicação, suas interações, as tecnologias utilizadas e os artefatos manuseados em cada etapa dos *pipelines* - a saber, padrões de design, trechos relevantes de *software*, dentre outros. Além disso, discute-se a estrutura dos dados, estratégias de automação e aspectos de escalabilidade e *OpenAuthorization (OAuth)*. A seção está organizada de forma a refletir o fluxo de construção da solução, facilitando o entendimento de seu funcionamento interno e potencial de evolução.

5.1 ARQUITETURA DA SOLUÇÃO

A figura 1 abaixo ilustra o diagrama da rede de *contêineres*, identificando cada um, algumas das tecnologias relevantes, bem como as comunicações entre si.

Podemos dividir o sistema em 4 (quatro) componentes, posteriormente referidos como *Frontend*, *Backend*, *Extractor* e *Database*. Para compartimentalizar e separar os mesmos, foi utilizado Docker e Docker compose, ferramentas de encapsulamento que empacotam aplicações e suas dependências, para que funcionem consistentemente em diferentes ambientes e instâncias. Essa decisão foi fundamentada nos seguintes fatores:

- Modularidade: Cada serviço executa em seu próprio container, garantindo independência operacional.
- Escalabilidade: Facilidade em aumentar ou reduzir instâncias conforme a demanda.
- Portabilidade: Implementação consistente entre diferentes ambientes (desenvolvimento, homologação e produção).

A respeito dos ambientes, foram criados 3 (três) para o repositório do acervo, bem como cada um de seus serviços: Desenvolvimento, Homologação e Produção. Esta medida permite que diferentes setores do projeto possam trabalhar de forma conjunta e simultânea. O sistema

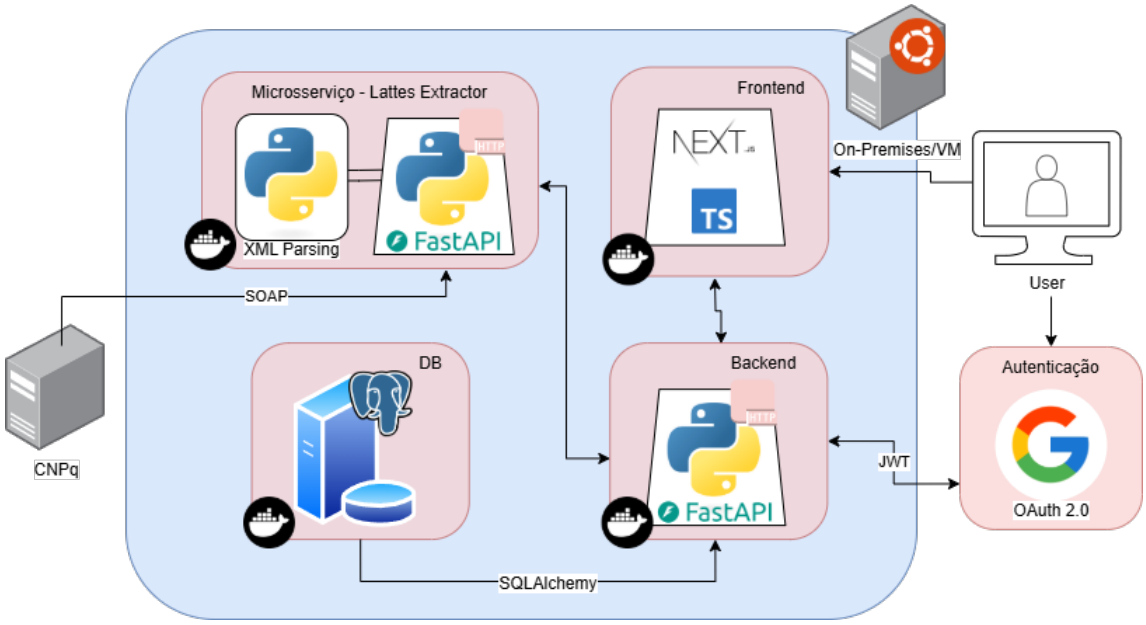

Figura 1 – Diagrama do Acervo Digital

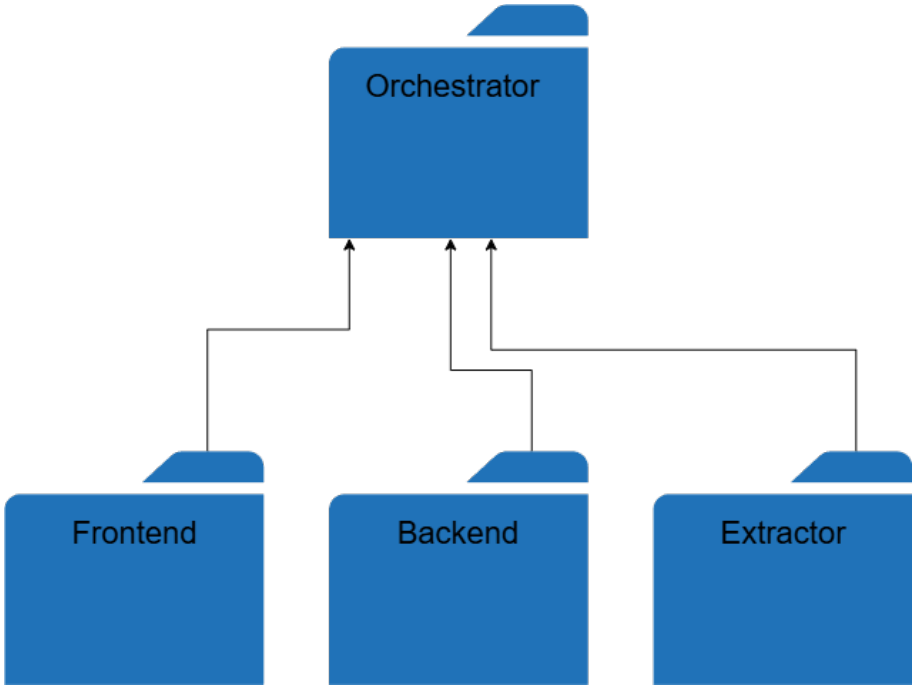

Figura 2 – Estrutura de repositórios git

de versionamento e repositório utilizado foi Git, estruturado em módulos e submódulos, como descreve a Figura 2.

Desta forma, a hierarquia de pastas reflete a disposição dos contêineres, respeitando princípios de isolamento e escalabilidade. Cada serviço pode ser executado, interrompido e modificado sem que prejudique o restante dos mesmos.

A segurança também foi um fator decisivo para a gestão de variáveis de ambiente e segredos. Para isso, as práticas de encapsulamento de variáveis de ambiente e o uso de *gitignore* e *dockerignore* foram consideradas. Assim, evitou-se a exposição de informações sensíveis nos repositórios de versionamento e imagens, ignorando tais variáveis e outras informações particulares (respectivo a cada ambiente ou pessoa) ou privadas (referente ao projeto a qual fazem parte). Essas medidas contribuem para que credenciais, chaves de API e outros dados sensíveis permaneçam protegidos, reduzindo riscos de comprometimento da segurança.

A necessidade de manter a aplicação atualizada e funcional enquanto era ativamente desenvolvida exigiu a automatização tanto do fluxo de entrega quanto de desenvolvimento. Para isso, foram implementados 2 (dois) *pipelines* no sistema de repositórios remotos GitLab.

O fluxo de desenvolvimento segue um princípio simples, mas incrementável, de verificação de erros via *build* no modo estrito de Typescript. Assim, *branches* específicas, como *main* e *development*, estão protegidas contra integrações indevidas.

O fluxo de entrega é descrito pela seguinte lista:

1. Um *push* ou *tag* no repositório Git aciona um gatilho no *pipeline*.
2. O *pipeline* executa a reconstrução da imagem Docker.
3. A nova imagem é publicada no registro privado.
4. O Watchtower, em execução na máquina que hospeda os containers, detecta a atualização e reinicia os serviços com a nova imagem.

Este processo permite que as atualizações sejam propagadas de forma rápida e confiável, reduzindo a necessidade de intervenção manual e minimizando o tempo de indisponibilidade da aplicação.

5.2 FLUXO DE DADOS

A Imagem 3 exibe o fluxo de dados da proposta solução. Nela, percebe-se como o acesso do usuário ao sistema é inteiramente pelo *frontend*, embora a autenticação seja mediada pelo Google, através de sua API pública.

Considerando que apenas IPs cadastrados podem acessar a plataforma Lattes via LE, e esta é sua forma de autenticação, a solução dispõe do módulo Extractor, responsável por manter a comunicação via protocolo SOAP[1] com o CNPq, converter dados adquiridos para XML e JSON, realizar o *parsing* de informações relevantes e servir via FastAPI os dados obtidos. Através das configurações desta e da compartimentalização de processos via Docker, que são capazes de designar comunicações entre serviços, limita-se o acesso a este apenas pelo Backend via portas e metadados de *Allowed Origins*.

Vale mencionar que a norma internacional para formatação de caracteres não foi fornecida, portanto a conversão de uma *string* em *base64* para um arquivo zipado valeu-se de um algoritmo simples de "força-bruta"para encontrar um formato válido. A formatação para *base64* possibilita o envio de arquivos via texto.[2].

Para o tempo designado no caso de estudo, o sistema limitou-se ao Lattes como única fonte de aquisição de produções. Posteriores melhorias (conforme mencionadas na Seção 6.3) incluem a criação de rotas de inserção e edição de produções individuais e em lote, em conjunto com reajustes no modelo de dados para suportar diferentes fontes.

[1] **Simple Object Access Protocol**, segundo a Wikipedia
[2] Recomenda-se, inclusive, a descrição do formato pela mesma

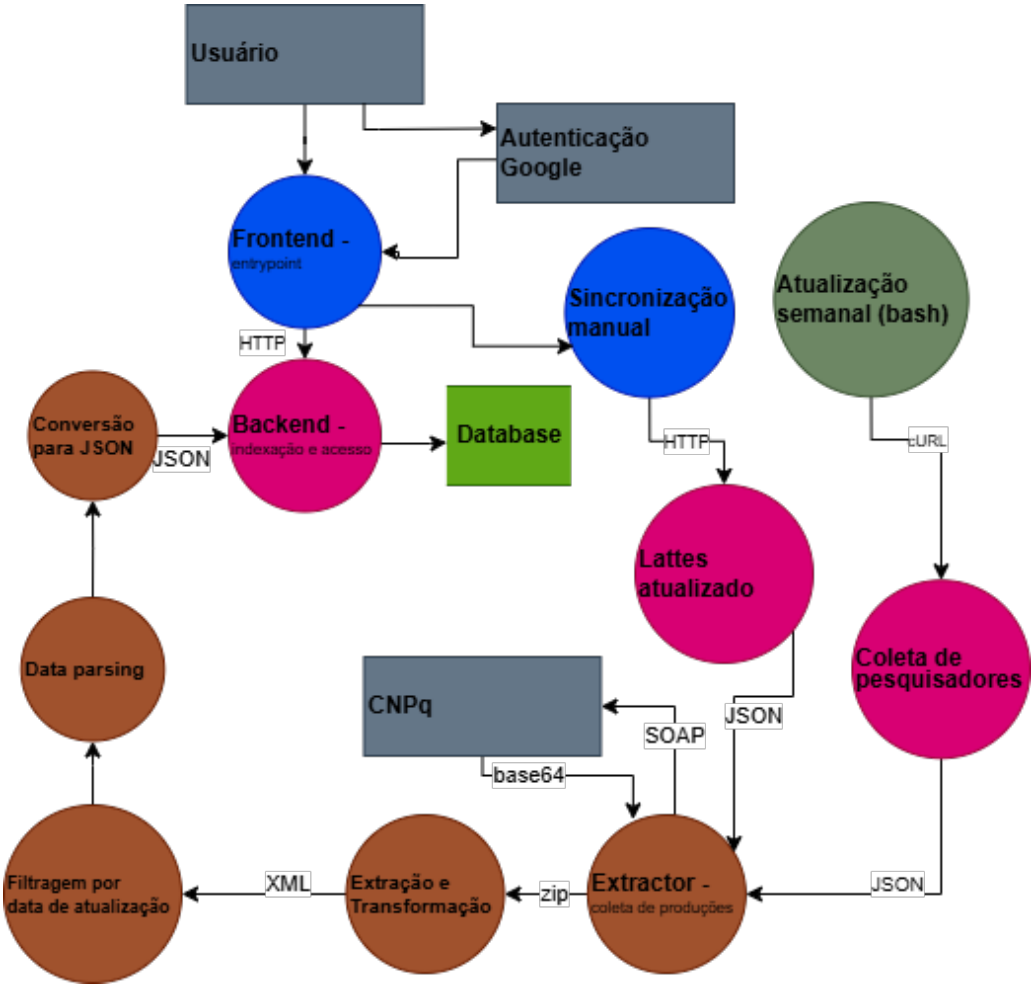

Figura 3 – Fluxo de dados da plataforma

Semanalmente, o serviço de backend realiza uma busca no Lattes de todos os pesquisadores registrados na plataforma por novas produções. Uma rota do LE é capaz de retornar a data de última atualização por LattesID (identificador único na plataforma) pesquisado, que pode ser comparada a última atualização registrada no banco de dados. É realizada uma solicitação via bash ao Extractor para cada indivíduo no banco, que retorna as produções dos indivíduos que tinham novos dados. Como o Lattes não fornece apenas as atualizações, o backend se responsabiliza por indexar e discernir dados. Através de verificações de duplicidade, realizadas por checagem de campos identificadores e chaves estrangeiras, produções novas são inseridas, *linkadas* aos respectivos autores, palavras-chave e áreas, e armazenadas de forma segura no banco de dados.

O acesso do usuário se dá pelo frontend, mediante autenticação institucional via *sign-in with Google*. Após autenticado, solicita-se opcionalmente o LattesID do pesquisador para futuras consultas ao Lattes e a plataforma é disponibilizada, possibilitando consultas simples

por diferentes critérios e atributos e visualização de artigos.

5.3 MICROSSERVIÇO EXTRACTOR

O **Extractor** foi desenvolvido como um microsserviço independente utilizando FastAPI, visando rapidez na entrega e facilidade de manutenção. A escolha dessa tecnologia se deu pela sua eficiência na construção de APIs assíncronas e pelo suporte nativo a validações e documentação automática, o que acelerou o desenvolvimento inicial.

5.3.1 Comunicação via SOAP e Parsing do XML

Uma das principais dificuldades enfrentadas foi a interação com a API SOAP do Lattes, que retorna dados brutos em XML. Esse formato exigiu um processo de estudo e adequação para garantir que a extração fosse precisa e compatível com o restante do sistema.

5.3.2 Uso do Design Pattern Adapter (Wrapper)

Para desacoplar a lógica de comunicação via SOAP da API FastAPI, foi utilizado o padrão de projeto Adapter (ou Wrapper). Essa abordagem permitiu:

- **Abstração das chamadas SOAP**, tornando a API mais modular e facilitando futuras substituições por outras fontes de dados.
- **Padronização da estrutura de saída**, convertendo o XML bruto para um formato JSON compreensível pelo backend.
- **Extensibilidade da API**, possibilitando ajustes futuros sem impactar diretamente os consumidores do serviço.

Essa implementação garantiu que a complexidade da comunicação SOAP ficasse encapsulada dentro do microsserviço, permitindo que o backend trabalhasse com dados estruturados sem precisar lidar diretamente com as particularidades do XML.

5.4 BACKEND

O backend foi desenvolvido utilizando FastAPI e SQLAlchemy, garantindo entregas rápidas e flexibilidade no manuseio dos dados. Durante a implementação, algumas dificuldades surgiram, especialmente relacionadas à modelagem das relações entre objetos no banco de dados, ao uso de *schemas* para validação e às inferências de tipo conforme o protocolo HTTP.

5.4.1 Modelagem de Dados e SQLAlchemy

A estrutura do banco de dados foi modelada utilizando SQLAlchemy ORM, o que permitiu a definição de relações complexas entre entidades. Os principais desafios encontrados foram:

- **Relacionamentos entre tabelas**: Para garantir integridade referencial, foi necessário configurar relações do tipo *one-to-many* e *many-to-many*, considerando a melhor gestão de recursos.
- **Gerenciamento de sessões**: A manipulação da sessão do banco (SessionLocal) exigiu atenção para evitar problemas de commit implícito e estado inconsistente dos objetos. Foi adotado um contexto gerenciado via dependency injection do FastAPI para garantir o escopo correto das transações.

5.4.2 Pydantic para Schemas e Validação

Os schemas foram estruturados com Pydantic, permitindo a conversão automática entre objetos do banco e respostas HTTP. As principais dificuldades incluíram:

- **Mapeamento entre SQLAlchemy e Pydantic**: Como SQLAlchemy não trabalha nativamente com Pydantic, foi necessário criar schemas de entrada e saída separadamente, evitando expor objetos do banco diretamente na API.
- **Validação avançada**: Algumas regras de negócios exigiram validações customizadas, implementadas por meio de validadores específicos do Pydantic.

5.5 FRONTEND

A implementação em **Next.js** com **TypeScript** permitiu o desenvolvimento após pouco tempo de rampagem. O uso de TypeScript permitiu uma **melhor verificação estática de tipos**, reduzindo erros em tempo de execução e aumentando a confiabilidade do código. Além disso, a adoção do **Next.js** proporcionou benefícios como renderização otimizada e suporte nativo a rotas dinâmicas, facilitando o desenvolvimento e melhorando o desempenho da aplicação. Inúmeras técnicas e comodidades implementadas pelo *framework* garantiram boas práticas organizacionais e de escrita de códigos.

5.5.1 Uso de Context API e Providers

Para gerenciar alguns estados globais da aplicação de forma eficiente (como a exibição de certos *dialogs*), foi empregada a Context API do React, combinada com ContextWrapper e ContextProvider. Tais recursos foram essenciais para manter a separação de responsabilidades e evitar a propagação desnecessária de propriedades entre componentes, garantindo uma estrutura mais modular e escalável.

A implementação seguiu as diretrizes do *design pattern* Provider, onde o estado global é centralizado e disponibilizado apenas para os componentes que realmente necessitam dele. Isso reduziu a complexidade da aplicação e melhorou a performance, pois evitou renderizações desnecessárias.

- **ContextWrapper**: encapsulou os diferentes contextos da aplicação em um único provedor, permitindo uma configuração centralizada e simplificada.
- **ContextProvider**: forneceu estados e métodos compartilhados para os componentes filhos, garantindo que os dados fluíssem de maneira controlada e eficiente.

Essa abordagem permitiu uma melhor organização do código e reforçou a manutenção e extensibilidade da aplicação, uma vez que novos contextos podem ser adicionados sem impactar a estrutura principal.

5.6 CONSIDERAÇÕES SOBRE O DESENVOLVIMENTO

O desenvolvimento do sistema foi orientado pela adoção de metodologias ágeis e padrões de design que garantiram modularidade, escalabilidade e manutenibilidade. A integração do backend, construído com FastAPI e SQLAlchemy, demonstrou como a modelagem cuidadosa das relações entre objetos e a gestão de sessões podem facilitar a manipulação de dados complexos, enquanto a utilização do Pydantic assegurou que as inferências de tipo e a conformidade com o protocolo HTTP fossem rigorosamente observadas.

No microsserviço Extractor, a escolha do FastAPI possibilitou uma API de rápida entrega, essencial para atender aos requisitos de desempenho do sistema. O desafio de interpretar e padronizar os dados XML recebidos via SOAP foi superado por meio do uso do Design Pattern Adapter, que encapsulou a complexidade da conversão para JSON, tornando a integração com o backend mais transparente e robusta.

A principal qualidade do frontend reside em seu conjunto de tecnologias utilizada, que visam prototipagem rápida e entregas agilizadas. A atenção ao padrões de design *Composite*, *Factory* e *Observer* permitiram que o desenvolvimento seguisse a estrutura intencionada pelos autores do *framework*, maximizando a velocidade de produção.

Por fim, a implementação da infraestrutura e das práticas de DevOps, utilizando Docker, Docker compose e pipelines automatizados, assegurou que o sistema fosse distribuído, seguro e capaz de evoluir conforme as demandas. O uso de variáveis de ambiente encapsuladas e a integração de um registro privado com Watchtower demonstraram como a automação e a governança de recursos são fundamentais para operações contínuas e seguras.

Em resumo, a combinação dessas abordagens – backend robusto, microsserviço extractor, frontend moderno e infraestrutura automatizada – não só atende aos requisitos técnicos do projeto, como também se alinha às melhores práticas de engenharia de software e DevOps, estabelecendo um modelo sustentável para futuras integrações e expansões. Este desenvolvimento, fundamentado em princípios teóricos e aplicados de forma prática, comprova a eficácia de uma abordagem integrada e orientada a padrões para a gestão e disseminação de conhecimento em ambientes acadêmicos e tecnológicos.

6 CONCLUSÃO

6.1 RESULTADOS E DISCUSSÃO

6.1.1 Comparações com Soluções Existentes

A solução desenvolvida neste trabalho apresenta semelhanças relevantes com o sistema Quem@PUC, especialmente no que diz respeito ao propósito de consolidar e exibir a produção acadêmica de pesquisadores com base na Plataforma Lattes. Ambos os sistemas partem de uma premissa comum: facilitar o acesso a informações científicas e acadêmicas de maneira estruturada e acessível.

Contudo, as abordagens tecnológicas adotadas divergem significativamente. O Quem@PUC faz uso de *scripts* XSLT para transformar documentos XML do Lattes para o formato RDF e possibilitando consultas semânticas via SPARQL, integrado via *pipelines* LinkedPipes. Além de utilizar ontologias como FOAF, VIVO e BIBO.

O processo de coleta de dados a partir da Plataforma Lattes, no contexto da solução da PUC-Rio, é realizado manualmente por meio de *scripts* em shell. Isso significa que não há integração contínua ou pipelines automatizados de Extração, Transformação e Carga (ETL), o que limita a escalabilidade e a atualização dinâmica do acervo. Em sistemas modernos, essa etapa é frequentemente automatizada com ferramentas como Apache NiFi, Airflow ou mesmo *scripts* Python integrados a cronjobs ou contêineres orquestrados. A ausência dessa automatização acarreta um custo operacional maior e propensão a erros humanos na manutenção dos dados.

A camada de apresentação do sistema Quem@PUC é construída utilizando apenas JavaScript puro, sem o suporte de *frameworks* modernos como React, Vue ou Next.js. Isso implica em maior esforço manual para o gerenciamento de estado, roteamento e reatividade da interface. Além disso, dificulta a manutenção e evolução da aplicação ao longo do tempo, uma vez que padrões e boas práticas consolidadas nos *frameworks* modernos não são aplicados. Em contrapartida, soluções baseadas em Next.js, por exemplo, oferecem renderização híbrida (*Server-Side Rendering e Static Site Generation*), melhor desempenho e suporte nativo a *Search Engine Optimization*, proporcionando uma experiência mais robusta tanto para usuários quanto para desenvolvedores.

O uso do servidor Apache HTTP evidencia uma infraestrutura tradicionalmente robusta

e consolidada, adequada para ambientes corporativos e de alto tráfego. No entanto, sua integração com Flask via mod_wsgi limita o potencial da arquitetura, já que o padrão Web Server Gateway Interface (WSGI) não oferece suporte a operações assíncronas nem conexões em tempo real. Além disso, apesar da presença do ecossistema Apache, tecnologias voltadas para o processamento paralelo de grandes volumes de dados, como Apache Spark (para processamento distribuído) ou Kafka (para comunicação entre serviços via mensagens), não são exploradas. Isso indica que o sistema, embora confiável, está centrado em um paradigma mais linear e síncrono, o que limita sua escalabilidade frente a demandas complexas ou em tempo real.

De acordo com a pesquisa anual realizada pelo Stack Overflow em 2024, FastAPI foi destacada como uma das ferramentas de desenvolvimento mais desejadas entre os desenvolvedores. (StackOverflow 2024). Essa popularidade se deve à sua abordagem moderna, baseada em tipagem forte com Pydantic, suporte nativo a requisições assíncronas (via Asynchronous Server Gateway Interface, ou ASGI), e documentação automática gerada por padrão através de Swagger. Esses fatores contribuem para um ciclo de desenvolvimento mais ágil, seguro e eficiente. Em comparação com Flask, que exige a adição manual de diversas extensões para alcançar funcionalidades similares, FastAPI se apresenta como uma solução mais enxuta e produtiva, especialmente em projetos que demandam velocidade de entrega e manutenção facilitada.

Ao comparar a solução proposta com a arquitetura utilizada pelo projeto Quem@PUC — que combina Apache e Flask — destacam-se diferenças estruturais relevantes. A solução da PUC-Rio opera sob a WSGI, síncrono por natureza, o que limita a escalabilidade em cenários de alta concorrência e impede suporte nativo a conexões persistentes, como WebSockets. Em contraste, a solução proposta adota FastAPI com Uvicorn, baseados na ASGI, que permite manipulação assíncrona de requisições e comunicação em tempo real. Além disso, FastAPI provê documentação automática via Swagger, simplificando a exposição e o consumo da API. Enquanto Apache exige configuração de módulos adicionais para executar aplicações Python (como mod_wsgi), Uvicorn pode ser executado diretamente ou em containers, reduzindo a complexidade de *deploy* e aumentando a portabilidade da aplicação. Essa abordagem torna a arquitetura mais leve, escalável e aderente às demandas modernas de desenvolvimento web.

6.2 CONSIDERAÇÕES FINAIS

Em síntese, o trabalho proposto oferece um modelo de acervo digital - totalmente compartimentalizado - de extração, transformação e gestão de dados de múltiplas origens - incluindo o Lattes e sistemas internos - através de *pipelines* e infraestrutura conteinerizada, com *parsing* completo de produções bibliográficas, tecno-tecnológicas, dentre outras métricas e informações contextualmente relevantes. O uso de KPIs tem relevância reportada e de comum senso entre outras produções, mantendo-se como artefatos-chave para a tomada de decisões e princípios. O mesmo se baseia em princípios Open Access, reconhece esforços de obras na mesma área diante da iniciativa, e propõe sugestões para o âmbito de integração de dados.

Embora o Quem@PUC ofereça uma solução robusta para indexação e busca de perfis acadêmicos dentro do ecossistema da PUC-Rio, a ferramenta proposta contrasta ao automatizar a extração e higienização. Enquanto o Quem@PUC se ancora sobretudo em RDF e NoSQL para armazenar triplas segundo ontologias como BIBO e VIVO, a arquitetura proposta, modular e contêinerizada, facilita tanto o escalonamento quanto a manutenção contínua, além de gerar indicadores (KPIs) alinhados a critérios CAPES/Qualis.

6.2.1 Limitações da Ferramenta

- **Dependência da API SOAP do Lattes:** Qualquer alteração no serviço ou no formato de resposta (XML/base64) pode interromper o fluxo de extração.
- **Ausência de algoritmo de busca próprio:** Atualmente, a ferramenta utiliza índices simples (SQL/NoSQL) sem motor semântico ou *full-text* avançado, comprometendo relevância contextual e desambiguação de termos.
- **Escalabilidade e desempenho:** *Parsing* de grandes volumes de XML e conversão para JSON podem gerar gargalos de CPU e memória em picos de carga; a infraestrutura de containers e CI/CD requer recursos robustos para múltiplas instâncias.
- **Complexidade de implantação e manutenção:** Configuração de variáveis de ambiente, gestão de secrets e registry privado de Docker eleva a barreira de entrada para equipes sem DevOps dedicado; atualizações automáticas via Watchtower podem introduzir instabilidades sem testes adequados.

- **Cobertura semântica limitada:** Ainda não há mapeamento completo para ontologias (BIBO, VIVO etc.) nem suporte a inferências OWL/RDFS, o que impede integrações profundas com outras bases de Linked Data.
- **Segurança e governança de dados:** Falta de autenticação de usuários finais e controle de acesso granular aos endpoints da API; ausência de auditoria detalhada sobre acessos e modificações.

Reconhecer essas limitações é fundamental para planejar melhorias futuras, como a implementação de um motor de busca semântico, adoção de autenticação baseada em OAuth2, testes de carga mais rigorosos e expansão do mapeamento ontológico.

6.3 TRABALHOS FUTUROS

Apesar dos resultados, algumas limitações foram identificadas, como a ausência de uma camada semântica para enriquecimento e inferência de dados, em contraste com propostas semelhantes. (Salgueiro et al. 2022). A *Quem@PUC* levanta caminhos e propostas às quais o trabalho pode seguir em esforços futuros. Assim, um dos principais pontos é a adoção de ontologias reconhecidas, como BIBO e VIVO, predispondo adaptações com repositórios internacionais.

Outra vertente relevante envolve a substituição ou complementação da atual estratégia de ETL com ferramentas como o LinkedPipes, além de explorar a aplicação de grafos com bancos como Neo4j para representar relações entre autores, instituições e temas, consolidando um microsserviço próprio de preparação semântica, automatizando desde a ingestão até a publicação de dados Linked Data.

Diversos passos podem ser tomados tanto para melhor se estabelecer como acervo digital — por meio de inserções e edições manuais — quanto para obter mais interoperabilidade e integração — por meio da implementação de ontologias e da conexão com repositórios e indexadores públicos, como *Google Scholar*, *DBLP* e *doi.org*, que podem ser consultados para o enriquecimento de dados.

BIBLIOGRAFIA

Alves, Alexandre D., Horacio H. Yanasse e Nei Y. Soma (2011). "LattesMiner: a multilingual DSL for information extraction from lattes platform". Em: *Proceedings of the Compilation of the Co-Located Workshops on DSM'11, TMC'11, AGERE! 2011, AOOPES'11, NEAT'11, & VMIL'11*. SPLASH '11 Workshops. Portland, Oregon, USA: Association for Computing Machinery, pp. 85–92. ISBN: 9781450311830. DOI: <10.1145/2095050.2095065>. URL: <https://doi.org/10.1145/2095050.2095065>.

Alves, Alexandre Donizeti, Horacio Hideki Yanasse e Nei Yoshihiro Soma (2011). "SUCUPIRA: A system for Information extraction of the Lattes Platform to identify academic social networks". Em: *6th Iberian Conference on Information Systems and Technologies (CISTI 2011)*. IEEE, pp. 1–6. URL: <https://ieeexplore.ieee.org/abstract/document/5974195?casa_token=xh-1_SfycPgAAAAA:b_cqyKagjR8ZfEo2bspbNoAa-SVyLueXi3r6I7ERXefsLc-pak-YGwvMFQhop_q5Od1D1WH4rfpF>.

American Chemical Society (2024). *ACS Expands Read and Publish Access in Latin America With Historic CAPES Consortium Agreement*. URL: <https://axial.acs.org/publishing/acs-expands-read-and-publish-access-in-latin-america-with-historic-capes-consortium-agreement>.

Bienvenu, Meghyn et al. (dez. de 2015). "Ontology-Based Data Access: A Study through Disjunctive Datalog, CSP, and MMSNP". Em: *ACM Trans. Database Syst.* 39.4. ISSN: 0362-5915. DOI: <10.1145/2661643>. URL: <https://doi.org/10.1145/2661643>.

Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (2014). *Sobre a Avaliação*. URL: <https://www.gov.br/capes/pt-br/acesso-a-informacao/acoes-e-programas/avaliacao/sobre-a-avaliacao/avaliacao-o-que-e/sobre-a-avaliacao-conceitos-processos-e-normas/conceito-avaliacao>.

— (2023). *Relatório Qualis Eventos para 2017-2020*. URL: <https://www.gov.br/capes/pt-br/centrais-de-conteudo/documentos/avaliacao/09012022_RELATORIOQUALISEVENTOS20172020
PDF>.

— (2024). *CAPES anuncia parceria para Acesso Aberto*. URL: <https://www.gov.br/capes/pt-br/assuntos/noticias/capes-anuncia-parceria-para-acesso-aberto>.

— (2025). *Orientações quanto ao registro de resultados e produções intelectuais*. URL: <https://www.gov.br/capes/pt-br/acesso-a-informacao/acoes-e-programas/avaliacao/sobre-

a-avaliacao/areas-avaliacao/sobre-as-areas-de-avaliacao/colegio-de-ciencias-exatas-tecnologicas-e-multidisciplinar/ciencias-exatas-e-da-terra/copy2_of_Resultados_e_producoes_intelectuais_2025.pdf>.

Corrêa, Tiago Silva et al. (2017). "O fim do scriptLattes? Uma análise de suas funcionalidades, alternativas para o presente e perspectivas para o futuro". Em: *Revista do EDICC-ISSN 2317-3815* 3. URL: <https://revistas.iel.unicamp.br/index.php/edicc/article/view/5208>.

Jansson, S. Mikael et al. (2010). "In for the Long Haul: Knowledge Translation Between Academic and Nonprofit Organizations". Em: *Qualitative Health Research* 20.1. PMID: 19801416, pp. 131–143. DOI: <10.1177/1049732309349808>. URL: <https://doi.org/10.1177/1049732309349808>.

JetBrains (s.d.). *Domain-Specific Languages*. URL: <https://www.jetbrains.com/mps/concepts/domain-specific-languages/#:~:text=A%20Domain%20Specific%20Language%20is,from%20the%20field%20or%20domain.>.

Kaganski, Sergei, Martin Eerme e Ernst Tungel (2019). "Optimization of enterprise analysis model for KPI selection". Em: *Proceedings of the Estonian Academy of Sciences*. URL: <https://api.semanticscholar.org/CorpusID:214053041>.

Lukman, Nur et al. (2022). "Integration of Repository System in Optimization Data for Graduates' Scientific Paper". Em: *Khizanah al-Hikmah : Jurnal Ilmu Perpustakaan, Informasi, dan Kearsipan*. URL: <https://api.semanticscholar.org/CorpusID:256674439>.

Ma, Ji, Joycelyn Ovalle e Yan Wang (2023). "Institutional factors influencing knowledge production for practice: Evidence from nonprofit studies". Em: *Plos one* 18.10, e0293360. DOI: <10.1371/journal.pone.0293360>. URL: <https://doi.org/10.1371/journal.pone.0293360>.

Maciel, Cássia, Andréa Trierweiller e Helio Ferenhof (nov. de 2019). "Perspectiva do Usuário na Utilização da Plataforma Sucupira como Sistema de Avaliação: Uma Analise Exploratória". Em: *Blucher Design*, pp. 1965–1975. DOI: <10.5151/9cidi-congic-4.0350>. URL: <https://www.proceedings.blucher.com.br/article-details/33777>.

Maciel, Cássia Emidio et al. (2017). "Avaliação da Interface de Interação da Plataforma Sucupira sob a Ótica de Diferentes Usuários". Publicado no repositório digital da instituição. URL: <https://repositorio.ufsc.br/handle/123456789/177607>.

Mena-Chalco, Jesús Pascual e Roberto Marcondes Cesar Junior (2009). "ScriptLattes: an open-source knowledge extraction system from the Lattes platform". Em: *Journal of the Brazilian*

Computer Society 15, pp. 31–39. DOI: <https://doi.org/10.1007/BF03194511>. URL: <https://link.springer.com/article/10.1007/BF03194511>.

Ortiz, Bengie L et al. (set. de 2024). "Data Preprocessing Techniques for AI and Machine Learning Readiness: Scoping Review of Wearable Sensor Data in Cancer Care". Em: *JMIR Mhealth Uhealth* 12, e59587. ISSN: 2291-5222. DOI: <10.2196/59587>. URL: <http://www.ncbi.nlm.nih.gov/pubmed/38626290>.

Pimentel, Bruno de Macêdo Cavalcanti Borges et al. (2017). "A plataforma Sucupira sob a interpretação dos gestores da Pós-Graduação em Educação". Escola de Educação, Tecnologia e Comunicação. Diss. de mestr. Programa Strictu Sensu em Educação. URL: <https://bdtd.ucb.br:8443/jspui/handle/tede/2340>.

Ramos, Christofer et al. (2017). "USABILIDADE DA PLATAFORMA LATTES APRESENTA NÍVEIS INADEQUADOS DE DE DESEMPENHO E SATISFAÇÃO DO USUÁRIO". Em: *Ergodesign & HCI* 5.Especial, pp. 153–164.

Ranasinghe, WM Tharanga D e Chung Jun Min (2018). "Institutional repository based open access scholarly publishing system: A conceptual model". Em: *Library Philosophy and Practice*, pp. 1–10. URL: <https://core.ac.uk/download/pdf/188141043.pdf>.

Rosielli e Silva e Marcelo Ferreira e Milton Cinelli e Monique Vandresen, Christofer Ramos e (2017). "USABILIDADE DA PLATAFORMA LATTES APRESENTA NÍVEIS INADEQUADOS DE DE DESEMPENHO E SATISFAÇÃO DO USUÁRIO". Em: *Ergodesign HCI* 5.Especial, pp. 153–164. ISSN: 2317-8876. DOI: <10.22570/ergodesignhci.v5iEspecial.337>. URL: <https://periodicos.puc-rio.br/index.php/revistaergodesign-hci/article/view/337>.

Salgueiro, Mariana DA et al. (2022). "Searching for Researchers: an Ontology-based NoSQL Database System Approach and Practical Implementation". Em: *Journal of Information and Data Management* 13.5. URL: <https://journals-sol.sbc.org.br/index.php/jidm/article/view/2601>.

Seland, Darryl (2018). "Garbage in garbage out". Em: *Quality* 57.5, pp. 6–6.

StackOverflow (2024). *2024 Developer Survey*. URL: <https://survey.stackoverflow.co/2024/technology#2-web-frameworks-and-technologies>.

Technical University of Moldova (2024). *UTM's Institutional Repository Integrated into BASE: A Significant Achievement for International Visibility of Academic Research*. URL: <https://utm.md/en/blog/2024/06/12/utms-institutional-repository-integrated-into-base-a-significant-achievement-for-international-visibility-of-academic-research/>.

Venters, Colin C et al. (2018). "Software sustainability: Research and practice from a software architecture viewpoint". Em: *Journal of Systems and Software* 138, pp. 174–188. DOI: <10.1016/j.jss.2017.12.026>. URL: <https://www.sciencedirect.com/science/article/pii/S0164121217303072>.

Whitley, Kirsten N. (1997). "Visual programming languages and the empirical evidence for and against". Em: *Journal of Visual Languages & Computing* 8.1, pp. 109–142.

Winickoff, David et al. (2021). "Collaborative platforms for emerging technology". Em: *OECD Science, Technology and Industry Policy Papers*.

Wiseso, Linggis Galih, Mahmud Imrona e Andry Alamsyah (2020). "Performance Analysis of Neo4j, MongoDB, and PostgreSQL on 2019 National Election Big Data Management Database". Em: *2020 6th International Conference on Science in Information Technology (ICSITech)*, pp. 91–96. DOI: <10.1109/ICSITech49800.2020.9392041>. URL: <https://ieeexplore.ieee.org/abstract/document/9392041>.

Zarkesh, Maryam e Allison Marcela Beas (2004). "UCLA Community College Review: Performance Indicators and Performance-Based Funding in Communitq Colleges". Em: *Community College Review* 31.4, pp. 62–76. DOI: <10.1177/009155210403100404>. eprint: <https://doi.org/10.1177/009155210403100404>. URL: <https://doi.org/10.1177/009155210403100404>.

Zhang, Cheng e David Budgen (2011). "What do we know about the effectiveness of software design patterns?" Em: *IEEE Transactions on Software Engineering* 38.5, pp. 1213–1231.