Universidade Federal Fluminense

Instituto de Computação

Departamento de Ciência da Computação

Arthur de Oliveira Paiva

Programa Perfil: coletor de dados de pesquisadores para gerar relatórios e visualizações

Niterói

2022

Arthur de Oliveira Paiva

Programa Perfil: coletor de dados de pesquisadores para gerar relatórios e visualizações

Trabalho de Conclusão de Curso submetido ao Departamento de Ciência da Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Leonardo Gresta Paulino Murta

Co-orientadora: Profa. Vanessa Braganholo Murta

Niterói

2022

Arthur de Oliveira Paiva

Programa Perfil: coletor de dados de pesquisadores para gerar relatórios e visualizações

Trabalho de Conclusão de Curso submetido ao Departamento de Ciência da Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Aprovado em 13 de Julho de 2022:

Prof. Leonardo Gresta Paulino Murta
Orientador - UFF

Profa. Vanessa Braganholo Murta
Co-orientador - UFF

Prof. Alexandre Plastino de Carvalho
UFF

Prof. Célio Vinicius Neves de Albuquerque
UFF

Niterói
2022

Agradecimentos

Primeiro gostaria de agradecer aos meus pais, Maria José de Oliveira Paiva e Francisco Sales Martins Paiva que mesmo com as dificuldades da vida me criaram com muito amor, afeto e que acima de tudo me deram o luxo de ter todas as oportunidades necessárias para ter uma boa educação acadêmica da melhor qualidade possível.

A minha irmã, Robertha Josiara de Oliveira Paiva Giorgette por não poupar esforços ao sempre tentar me fazer feliz durante minha criação.

As minhas tias, Eulene Paiva e Glaice Paiva que junto aos meus pais até hoje ajudam na minha edução, conquistar meus objetivos e me mostram o caminho correto.

Ao meu primo, Victor Paiva por ser um exemplo na minha vida acadêmica e fazendo o papel de irmão mais velho desde que me entendo como pessoa.

Aos meus melhores amigos que me acompanham por quase toda minha vida, irmãos não consanguíneos, Cosme Melo, Marlon Luiz, Ronald Caetano, Vinicius Carvalho e Vitor Caetano, que sem eles minha vida não teria nem metade da graça.

Aos meus orientadores, Leonardo Murta e Vanessa Braganholo, por terem me acompanhando desde o inicio da minha iniciação científica. Sempre no meu ritmo e se esforçando o máximo possível para me guiar e cumprir com o calendário da UFF.

Aos professores integrantes da banca, Alexandre Plastino e Célio Albuquerque, por se comprometerem a julgar esse trabalho mesmo com um prazo curto.

Resumo

A Plataforma Lattes permite pesquisadores do Brasil publicarem seus currículos. Além de oferecer sua visualização em interface *web*, também é possível baixá-los em formato XML, o que pode ser usado para processamento por programas. Todavia neles não há todas as informações necessárias para certas análises, como o nível de Qualis. Por fim, o preenchimento manual do currículo pode fazer com que ele tenha dados errôneos. Na literatura já existem programas que extraem dados dos currículos Lattes, os analisam, encontram (e tentam corrigir ou minimizar) inconsistências e geram relatórios e visualizações. Entretanto, nenhum deles permite o usuário configurar seus próprios relatórios, mostra a mudança em grafos de colaboração de ano a ano e nem produz informações indispensáveis para os relatórios de gestão de um programa de pós-graduação, como o índice restrito total do programa. O programa Perfil 2.0, proposto neste trabalho, consegue gerar outras informações a partir das coletadas dos currículos Lattes e trata possíveis erros, inconsistências ou duplicações. A partir dessas informações, o programa Perfil 2.0 permite gerar diversos relatórios que podem auxiliar a progressão de um professor para o nível de titular, viabilizar análises quantitativas e responder a questionamentos usuais feitos aos programas de pós-graduação pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), além de possibilitar a visualização de informações dos pesquisadores por meio de *boxplots* e grafos de colaboração. Durante e após seu desenvolvimento, realizamos testes automatizados, testes manuais com massas de dados com 3 e 48 pesquisadores, testamos contra outros programas com funcionalidades parecidas e buscamos *feedback* de usuários para garantir que o programa funciona como desejado.

Palavras-chaves: Análise de Currículo Lattes. Progressão para Titular. Relatórios CAPES.

Abstract

The Lattes platform lets Brazillian researchers write their curriculum. Besides allowing visualization via its web interface, it also enables downloading them in the XML format, which can be used in software processing. However, they do not have all the necessary information, like Qualis level, to make certain analyses. Moreover, a curriculum written by the user could have data with errors. The literature already describes software that extracts data from the Lattes curriculum, analyzes them, find (and try to correct or minimize) inconsistencies, and generates reports and visualizations. Nevertheless, neither allow the user to configure their reports, show changes in collaboration graphs year by year, nor generate information for reports that are relevant to managing graduate programs, like the total restrict index. The Perfil 2.0 program can derive information from the data extracted from the Lattes curriculum and treat possible errors, inconsistencies, or duplications to generate several reports which can help request full professorship promotion, doing quantitative analyses, and generating reports to answer data requests made by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES). It also provides visualizations of information of researchers using boxplots and collaboration graphs. During and after its development, we did automated and manual tests using a corpus of 3 and 48 researchers. We also tested it against other software with similar features and asked for feedback from the users to ensure that it worked as expected.

Keywords: Lattes Curriculum Analysis. Full Professorship Promotion. CAPES Reports.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CAPES	*Coordenação de Aperfeiçoamento de Pessoal de Nível Superior*
CNPq	*Conselho Nacional de Desenvolvimento Científico e Tecnológico*
XML	*Extensible Markup Language*
HTML	*HyperText Markup Language*
DOI	*Digital Object Identifier*
ISSN	*International Standard Serial Number*
ISBN	*International Standard Book Number*
URL	*Uniform Resource Locator*

Sumário

1 Introdução

No Brasil, pesquisadores podem publicar seus currículos na Plataforma Lattes, que foi desenvolvida pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq). Desse modo, tem-se um padrão nacional para currículos de pesquisadores. Isso, por sua vez, permite que eles sejam analisados por diversas aplicações, pois além de sua interface *web*, essa plataforma possibilita a exportação dos currículos para o formato de arquivo XML, o que facilita programas os processarem e pode ajudar em diferentes situações, como, por exemplo concursos, fazer relatórios para a CAPES e relatórios para promoções a professor titular.

Entretanto, apenas com essa plataforma ainda não é possível fazer análises mais profundas. Por exemplo, é possível saber em qual periódico ou conferência um artigo foi publicado, mas não seu nível de Qualis. Até atributos mais diretos de um pesquisador, como, por exemplo, o número total de citações das suas publicações e o seu h-index, a plataforma não disponibiliza. Além disso, a plataforma aceita erros de preenchimento no currículo, o que torna ainda mais fácil de acontecer outro problema que é a duplicação de informações, fazendo com que se torne imprecisa a análise de um conjunto de currículos ao mesmo tempo, pois seja por erros de preenchimento ou pela falta de padronização dos mesmos, não é possível identificar diretamente produções conjuntas de um grupo de pesquisadores. Para completar, a plataforma não oferece meios de visualizar colaborações entre os pesquisadores dentro de um período de tempo, nem é permitido baixar os currículos de pesquisadores em lotes.

Já existem soluções para alguns desses problemas. Por exemplo, o programa ScriptLattes (MENA-CHALCO; JUNIOR, 2009) permite coletar os dados via páginas web, gerar relatórios de um grupo de pesquisadores em conjunto e permite gerar visualizações de colaboração. Segalin (2014) extrai, armazena e atualiza dados dos currículos Lattes de pesquisadores da Universidade Federal de Santa Catarina com uma certa frequência para realizar consultas definidas pela Pró-Reitoria de Pesquisa. Já Valverde et al. (2017) produzem relatórios para apoiar programas de pós-graduação com filtros temporais. Digiampietri et al. (2012) extraem informações de vários arquivos XML de currículos Lattes e montam um grande banco de dados para consulta. Rubim e Braganholo (2017) buscam encontrar inconsistências entre artigos de currículos Lattes de coautores, detectando duplicações e determinando onde e o quanto elas acontecem. Branco et al. (2018) coletam informações de diferentes plataformas e as comparam com as informações existentes nos currículos Lattes de pesquisadores. Mena-Chalco, Digiampietri e Cesar-Jr. (2012) montam redes de coautoria divididas em trienais. Por fim, Colonetti et al. (2016) montam *clusters* usando grafos de forma a agrupar pesquisadores com interesses semelhantes, usando para

isso as coautorias nos currículos Lattes.

Entretanto, nenhuma das soluções anteriores consegue gerar métricas indiretas como nível de Qualis quando o nome do local de publicação não seja similar, não coleta informações não disponíveis diretamente nos currículos Lattes, não trata o preenchimento incorreto (ou diferente) de nomes de periódicos, conferências e projetos, nem deixa o usuário configurar strings similares ou informar como quer que seus relatórios sejam gerados. No que diz respeito à visualizações de colaboração de pesquisadores, não é possível observar as diferenças de ano a ano e todo o período.

A abordagem proposta neste trabalho é voltada para as demandas do pesquisador brasileiro de instituições de ensino superior. Desse modo, além de coletarmos as informações dos currículos Lattes, também coletamos os dados necessários para gerar relatórios para a CAPES, para progressão para a carreira de professor titular, geramos perfis quantitativos de modo a comparar pesquisadores e tratamos possíveis erros, inconsistências ou duplicações encontradas nos currículos Lattes em todos os aspectos onde elas podem ocorrer dentre as informações coletadas. Também criamos a opção de gerar visualizações seja por *boxplot* ou grafos de colaboração que respeitam o horizonte de coleta do programa. Além disso, caso os relatórios pré-configurados no programa Perfil 2.0 não sejam suficientes, permitimos que o usuário configure outros relatórios a seu gosto.

O programa Perfil 2.0 foi implementado em Python 3.8 utilizando a biblioteca SQLAlchemy[1]. Ele é um projeto *open source*, disponível em <https://github.com/gems-uff/perfil>. Sendo composto de um conjunto de *scripts*, com diferentes funcionalidades, que podem ser executados de forma independente de acordo com a necessidade do usuário.

Antes do desenvolvimento do programa Perfil 2.0, havia dois programas, um desenvolvido usando a linguagem de programação Java e outro desenvolvido usando Python. Ambos coletavam as informações em currículos Lattes de pesquisadores, em formato XML, para gerar relatórios. O programa em Java gerava relatórios para a CAPES, enquanto o em Python gerava dados quantitativos dos pesquisadores, *boxplots* a partir desses dados e auxiliava o usuário a baixar os currículos Lattes. O programa Perfil 2.0, além de implementar todas as funcionalidades anteriores, gera relatórios para progressão para a carreira de professor titular, relatórios configurados pelo usuário e visualizações de grafos de colaboração entre os pesquisadores.

Para avaliar o programa Perfil 2.0, além de testes automatizados, geramos uma massa de dados pequena e outra contendo todos os pesquisadores do programa de Pós-Graduação em Computação (PGC) do Instituto de Computação da Universidade Federal Fluminense. Para a maioria dos *scripts* que compõem o programa, usamos essa massa de dados para procurar erros de execução e depois validamos os resultados com alguns

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

usuários. Algumas funcionalidades já estavam implementadas em dois programas diferentes, um escrito em Java, e outro escrito em Python. Então foi necessário testar nosso programa contra eles até obtermos resultados iguais ou melhores.

No final, concluímos que o programa atende ao que ele foi proposto, pois ele cumpre seu principal objetivo: integrar os programas antigos nos quais ele foi baseado, mas usando banco de dados. Além disso, implementa novas funcionalidades que permitem gerar outros tipos de relatórios e visualizações, reporta possíveis duplicações nos currículos Lattes e possibilita o usuário configurar diversos aspectos para melhor atender suas necessidades.

O restante desse trabalho está organizado da seguinte forma. O Capítulo 2 discute sobre a plataforma Lattes, a plataforma Google Scholar e os trabalhos relacionados. O Capítulo 3 fala sobre as classes do programa, seu banco de dados e como os dados e informações são coletados. Seguindo, o Capítulo 4 trata sobre os *scripts* disponíveis, suas funcionalidades e suas possíveis saídas e/ou visualizações. Já no Capítulo 5 informamos como testamos e validamos o programa Perfil 2.0. Finalmente, as conclusões e trabalhos futuros estão no Capítulo 6.

2 Análise de Currículos

Para coletar os dados sobre os pesquisadores com o objetivo de gerarmos relatórios e visualizações, usamos os seus currículos Lattes, que são disponibilizados na Plataforma Lattes. Entretanto, apenas as informações obtidas neles não suprem as necessidades do programa Perfil 2.0. Dessa forma, foi necessário consultar também a plataforma Google Scholar. A seguir, explanamos sobre essas plataformas e trabalhos relacionados que também buscam extrair dados da Plataforma Lattes para gerar relatórios e/ou visualizações.

2.1 Plataforma Lattes e currículo Lattes

A Plataforma Lattes[1] é um sistema criado pelo CNPq para integrar bases de dados de currículos, grupos de pesquisa e de instituições de ensino e seu nome foi dado em homenagem ao cientista Cesare Mansueto Giulio Lattes (AMORIN, 2003). Já o currículo Lattes é uma forma de padronizar os currículos de pesquisadores do Brasil por meio de formulários eletrônicos.

Os dados do currículos Lattes podem ser usados para diversas funções, como, por exemplo, avaliar candidatos a bolsas, selecionar membros de comitês, dar subsídios a pesquisas e avaliar programas de pós-graduação. Além do mais, os currículos Lattes podem ser acessados e editados via *web* ou baixados em um arquivo no formato XML.

Em sua estrutura, o currículo Lattes está organizado de forma hierárquica e em módulos, armazenando dados sobre o pesquisador, como nome, formação, área de atuação, seus projetos, suas produções bibliográficas (por exemplo, artigos em periódicos, conferências e livros), suas patentes, suas orientações, suas participações em bancas, entre outros. Essa estrutura hierárquica também se aplica ao arquivo XML gerado a partir do currículo, dadas as propriedades do currículo. A Figura 1 ilustra um exemplo de hierarquia das *tags* de um currículo Lattes convertido para XML.

2.2 Google Scholar

O Google Scholar[2] é uma ferramenta de busca voltada para textos acadêmicos, ou seja, artigos de periódicos, conferências, livros acadêmicos, teses, entre outros. Além de funcionar como uma ferramenta de busca, ele também permite visualizar quais obras citam outras obras, permite ver artigos relacionados a um dos resultados da busca, salvar obras como favoritas, acompanhar áreas de interesse e gerar citações.

[1] <https://lattes.cnpq.br/>
[2] <scholar.google.com>

Figura 1 – Exemplo de trecho de um currículo Lattes convertido para XML

Dentre suas outras funcionalidades, há a de classificar publicações usando fatores como fonte da publicação, autores, frequência e data das citações (ZIENTEK et al., 2018). Também possibilita autores criarem perfis com componentes sociais, dentre eles a funcionalidade de mostrar seus trabalhos, quem os cita, sua quantidade e seu *h-index* (HIRSCH, 2005).

2.3 Trabalhos relacionados

O programa Perfil 2.0 possui várias funcionalidades, então seus trabalhos relacionados compartilham pelo menos uma das suas funcionalidades ou várias delas. Por isso, decidimos agrupá-los pelo seu foco, dividindo-os em três categorias nas seções a seguir: *relatórios*, que discute trabalhos cujo o maior propósito é gerar relatórios a partir dos dados extraídos de currículos Lattes, *manipulação de dados*, que discute trabalhos cujo objetivo é extrair e processar diretamente os dados e *colaboração*, que discute trabalhos com o principal propósito de gerar visualizações representando a colaboração entre os pesquisadores.

2.3.1 Relatórios

O programa ScriptLattes (MENA-CHALCO; JUNIOR, 2009) cria relatórios a partir de um grupo de pesquisadores registrados na Plataforma Lattes. Primeiro ele baixa o currículo Lattes dos pesquisadores, depois extraí os dados necessários, elimina os redundantes para assim criar relatórios da produção e orientações, como também grafos de colaboração e mapear o local geográfico dos membros do grupo. Nota-se que seu foco está em grupos de pesquisadores e não pesquisadores individualmente, então a entrada consiste em um arquivo com o identificador do currículo Lattes dos pesquisadores, assim como o período a ser analisado. A partir dessa lista, os currículos Lattes são baixados em formato

HTML e deles são extraídas as informações, pois quando o ScriptLattes foi criado, usuários públicos não tinham acesso aos currículos em formato XML. Em seguida, produções duplicadas são eliminadas e, por fim, as saídas são geradas. O grafo de colaboração é gerado usando as produções científicas. Os pesquisadores são os nós e há arestas entre eles caso haja produção em comum. Quanto mais produções em comum, maior o peso da aresta. Já para montar o mapa, se faz uso dos endereços disponíveis nos currículos Lattes. Desse modo, o ScriptLattes mostra o mapa mundial com a localização de cada pesquisador e cada doutor formado pelo grupo. Ele também cria relatórios para produções e orientações em andamento e concluídas – elas são separadas por tipo e mostram informações quantitativas em ordem cronológica decrescente. Nesses relatórios, também há gráficos em barra para os valores da produção científica do grupo e hiperligações para ferramentas de busca de forma a encontrar citações ou trabalhos similares.

Segalin (2014) criou um sistema *web* que extrai dados dos currículos Lattes dos pesquisadores da Universidade Federal de Santa Catarina e popula um banco de dados com eles. Desse modo, o sistema permite que o usuário realize consultas pré-definidas pela Pró-Reitoria de Pesquisa da universidade e gere relatórios e gráficos a partir delas. Primeiro mapeou-se o arquivo XML do currículo Lattes para um modelo relacional, depois populou-se o banco de dados com os currículos Lattes dos pesquisadores da universidade de acordo com o mapeamento. Por fim, um web-service baixa diariamente os currículos dos pesquisadores e, uma vez por semana, o sistema operacional roda um *script* para atualizar o banco de dados do sistema.

O programa Ceifador (VALVERDE et al., 2017) extrai informações de currículos Lattes para gerar relatórios quantitativos e permitir consultas referentes a um programa de pós-graduação qualquer, contribuindo no controle de qualidade externo e interno do mesmo. Inicialmente, o programa coleta os dados dos currículos Lattes em formato XML populando um banco de dados. Depois, o usuário seleciona a consulta que deseja fazer e pode filtrar, modificar e/ou gerar relatórios com os dados retornados com o foco na avaliação quadrienal realizada pela CAPES.

2.3.2 Manipulação de dados

Digiampietri et al. (2012) criaram um banco de dados com mais de um milhão de currículos Lattes, com o objetivo de processar, organizar e analisar os seus dados. Para montar o banco de dados, primeiro se procurou e baixou 1.236.548 currículos de coautores, orientadores e orientandos em formato HTML (não fica clara a razão de usar HTML, visto que o processamento programático de XML é mais efetivo). Depois, para processá-los, foram retirados os caracteres especiais, espaços em branco excedentes e fins de linha. Em seguida, se usou expressões regulares para dividir o currículo em seções e cada seção teve seus dados extraídos por outras expressões regulares que as dividiam por categoria. Por fim,

esses dados foram inseridos no banco de dados, totalizando 88.402.026 registros. Usando esses registros, criaram-se redes sociais de pesquisadores a partir de coautorias (utilizando os nomes das publicações e os nomes dos autores), relações de orientador/orientado e áreas de interesse.

Rubim e Braganholo (2017) buscaram achar inconsistências em currículos Lattes e determinar seus níveis usando heurísticas para encontrá-las e mapeá-las. Sua metodologia consistiu de, para cada artigo de um currículo, identificar seus coautores e, para cada coautor, procurar em seu currículo o artigo usando os seguintes atributos: título do artigo, volume, ano, página inicial, página final, DOI, ISSN (para artigos de periódicos), ISBN (para artigos de conferências), nome do local de publicação e coautores. Vale ressaltar que no caso de strings que não são identificadores, utilizou-se a maior subsequência comum (longest common subsequence – LCS) (BERGROTH; HAKONEN; RAITA, 2000). Contudo, primeiro tentou-se usar a string como ela foi extraída do currículo Lattes. Caso não fosse encontrada, era usado o algoritmo de similaridade entre strings baseado no LCS. Uma vez encontrado o artigo, comparava-se seus atributos para identificar inconsistências.

Branco et al. (2018) extraíram dados dos currículos Lattes para compará-los aos dados das plataformas Digital Bibliography & Library Project[3], Google Scholar e ResearchGate[4]. Seu foco foi usar os atributos de título da publicação, ano, autores, páginas e volumes para comparar artigos de conferência, artigos de periódicos, livros, capítulos de livros e outras publicações bibliográficas dos currículos Lattes. Para tal, primeiro se extraiu as informações dos currículos Lattes baixados em formato XML, inserindo-as em um banco de dados. Depois, conectou-se em um servidor *proxy* para acessar as plataformas, procurar e extrair delas os dados das publicações do pesquisador que teve seu currículo analisado, comparando-os com os obtidos do currículo Lattes dele usando maior subsequência comum. Por fim, persistiram todas as publicações que foram comparadas, mas identificadas como obras que não estão no currículo Lattes no banco de dados.

2.3.3 Colaboração

Mena-Chalco, Digiampietri e Cesar-Jr. (2012) fizeram redes de coautorias separadas em triênios utilizando currículos Lattes em extensão ao trabalho (DIGIAMPIETRI et al., 2012) visto na subseção anterior. Na rede de coautoria, cada pesquisador é um nó e cada aresta representa pelo menos uma publicação feita em coautoria. Para identificar as publicações em coautoria foram feitas comparações dois a dois por ano e tipo de produção utilizando casamento aproximado entre os títulos pela distância de Levenshtein (LEVENSHTEIN et al., 1966), mas antes separou-se as publicações em listas pela primeira letra do título, pois foi assumido que títulos com erros de digitação ou preenchimento só

[3] <www.dblp.org/>
[4] <www.researchgate.net/>

aconteceriam do meio para o final da string.

Colonetti et al. (2016) extraiu os dados de artigos em periódicos, artigos em conferências, área do conhecimento e especialidade de forma a criar *clusters* de pesquisadores a partir de coautorias em formato de grafo. O grafo possui como vértices os pesquisadores e seus nomes, as arestas são as coautorias entre eles e os pesos delas a quantidade de publicações entre os dois autores que elas ligam. Para fazer os grupos de pesquisadores foi usada uma metodologia própria – a aresta com o maior peso do grafo foi adicionada em um grupo, cada aresta dos vértices pertencentes à aresta anterior foi comparada a um valor mínimo. Se o peso dela fosse maior, adicionava-se ao mesmo grupo, então pegava-se de novo a aresta de maior peso do grafo que não está em nenhum grupo repetindo o processo até apenas restarem arestas menores ou iguais ao valor mínimo, que são colocadas em grupos individuais. Além da metodologia própria, também foram usados os métodos Cosine, Jaccard e Overlap para agrupamento.

2.4 Considerações finais

Apesar dos trabalhos anteriores coletarem os dados dos currículos Lattes, eles não geram nenhum outro dado derivado, como, por exemplo, o nível Qualis, não deixam o usuário criar um relatório customizado com as informações extraídas, seus grafos de colaboração não permitem visualizar as mudanças de ano por ano e não possibilitam a criação de *boxplots* para análises coletivas. No capítulo seguinte explicamos como o programa Perfil 2.0 é modelado e como ele coleta os dados para atender essas necessidades.

3 Modelagem e Coleta de Dados

Para analisar os currículos Lattes que o usuário deseja, optamos por armazená-los em uma pasta chamada */lattes* dentro do diretório do programa. Para guardar essas informações de forma propícia para o processamento, escolhemos fazer uso de orientação a objetos atrelada a um banco de dados relacional por meio de um mapeamento objeto relacional. Entretanto, somente coletar as informações dos currículos Lattes não atende completamente o escopo do programa, sendo necessário obter outras informações que derivam ou não delas. Nas seções a seguir, discutimos a modelagem de classes, o banco de dados relacional, como os dados são coletados e as razões que impulsionaram essas tomadas de decisões. Vale destacar que os currículos Lattes selecionados para terem seus dados extraídos vêm de uma lista de pesquisadores explicada na Seção 4.1.

3.1 Modelagem

Para o projeto, decidimos usar um mapeamento objeto relacional. Um dos motivos para essa escolha foi facilitar a conversão entre objetos em memória e tuplas no banco de dados relacional, sem ter que escrever código SQL, e, consequentemente, economizando esforço e tempo, e minimizando erros.

Optamos por desenvolver o programa na linguagem de programação Python 3.8, por ser uma linguagem popular e de sintaxe clara, facilitando manutenção futura. Para fazer o mapeamento objeto relacional, escolhemos a biblioteca SQLAlchemy. Essa biblioteca permite o mapeamento de diversas formas, em função da necessidade do usuário. Explicamos a seguir cada classe, associação e atributo, conforme o modelo exibido na Figura 2.

3.1.1 Researcher

Researcher é a classe que quando instanciada representa um pesquisador, ou seja, as pessoas que estão tendo os dados dos seus currículos Lattes coletados e analisados. Todas as outras classes estão relacionadas direta ou indiretamente a ela. Pode-se dizer que ela é a classe principal do modelo. Seus atributos são descritos a seguir.

name. O atributo *name* se refere ao nome do pesquisador. O nome usado é aquele que está declarado no currículo Lattes.

last_lattes_update. O atributo *last_lattes_update* indica quando foi a última atualização do currículo Lattes do pesquisador. Vale lembrar que não se trata da última atualização real, mas sim da última atualização do currículo que está salvo na pasta */lattes* do programa.

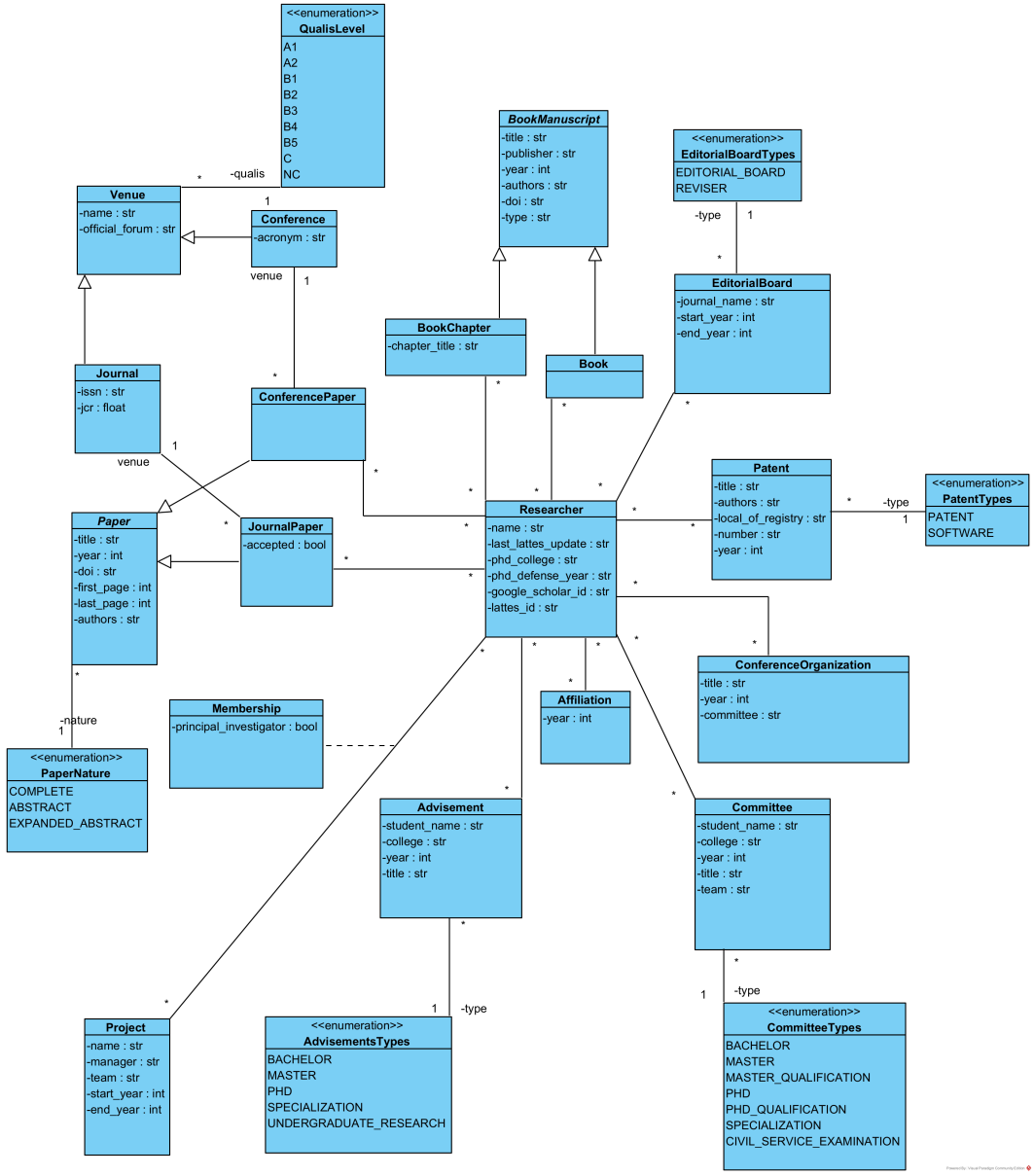


Figura 2 – Diagrama de classes

phd_college. O atributo *phd_college* informa em qual instituição de ensino superior o pesquisador obteve seu título de doutor.

phd_college_year. O atributo *phd_college_year* diz o ano em que o pesquisador obteve seu título de doutor, ou seja, quando ele concluiu seu doutorado.

google_scholar_id. O atributo *google_scholar_id* é o identificador do pesquisador na plataforma *https://scholar.google.com.br*.

lattes_id. O atributo *lattes_id* é o identificador do currículo Lattes do pesquisador.

3.1.2 Affiliation

Affiliation é a classe que relaciona *Researcher* aos anos nos quais ele estava credenciado ao Programa de Pós Graduação que está sendo analisado. Seu atributo é descrito a seguir.

year. O atributo *year* se refere ao ano que um dado pesquisador estava credenciado ao Programa de Pós Graduação.

3.1.3 Venue

Venue é a superclasse de *Journal* e *Conference*. Ela representa o meio em que uma publicação foi publicada. Seus atributos são descritos a seguir.

name. O atributo *name* é o nome do meio de publicação que o objeto *Venue* representa, como está no currículo Lattes analisado.

official_forum. Dado que um mesmo meio de publicação pode ter seu nome escrito de diferentes formas em diversos currículos Lattes, o atributo *oficial_forum* é o nome usado, no objeto *Venue*, para referenciar a todos eles de forma igual. Ele é obtido através de passos envolvendo maior subsequência comum, sinônimos e similares que são explicados na Seção 3.3 deste capítulo.

qualis. O atributo *qualis* é o nível do Qualis do meio de publicação que o objeto *Venue* representa. Ele é um *Enum* que pode assumir os valores A1, A2, B1, B2, B3, B4, B5, C e NC. Ele é obtido através de uma consulta a um arquivo na pasta */resources* do programa que é explicado na Seção 3.3 deste capítulo.

3.1.4 Journal

Journal é uma classe que herda de *Venue* e representa periódicos. Além dos atributos herdados de *Venue*, essa classe possui os atributos descritos a seguir.

issn. O atributo *issn* é o Número Internacional Normalizado para Publicações Seriadas, ou seja, o identificador do periódico.

jcr. O atributo *jcr* se refere ao fator de impacto do periódico segundo o *Journal Citation Reports*. Ele é obtido através de uma consulta a um arquivo na pasta */resources* do programa que é explicado na Seção 3.3 deste capítulo.

3.1.5 Conference

Conference é uma classe que herda de *Venue* e representa congressos. Além dos atributos herdados de *Venue*, essa classe possui o seguinte atributo.

acronym. Em alguns currículos Lattes, os nomes de algumas conferências vem acompanhado de seus acrônimos. O atributo *acronym* é justamente esse acrônimo separado do nome da conferência.

3.1.6 Paper

Paper é a superclasse de *JournalPaper* e *ConferencePaper*. Ela representa um artigo científico. Seus atributos são descritos a seguir.

title. O atributo *title* é o título do artigo.

year. O atributo *year* é o ano em que o artigo foi aceito ou publicado.

doi. O atributo *doi* é o *Digital Object Identifier* da publicação, ou seja, o identificador digital do artigo.

first_page. O atributo *first_page* indica o número da primeira página do artigo no meio de publicação em que ele foi publicado. Por exemplo, caso ele tenha sido publicado em uma revista, indica em qual página daquela revista o artigo começa.

last_page. O atributo *last_page* indica o número da última página do artigo no meio de publicação em que ele foi publicado. Por exemplo, caso ele tenha sido publicado em uma revista, indica em qual página daquela revista o artigo termina.

authors. O atributo *authors* lista todos os nomes dos autores do artigo, separados pelo caractere ";".

nature. O atributo *nature* é a natureza do artigo. Ele é um *Enum* que pode assumir os valores artigo *COMPLETE*, *ABSTRACT* ou *EXPANDED_ABSTRACT*.

3.1.7 JournalPaper

JournalPaper é a classe que herda de *Paper* e representa artigos aceitos ou publicados em periódicos, ou seja, artigos publicados em *Venues* que são *Journal*. Além dos atributos herdados de *Paper*, essa classe possui o seguinte atributo.

accepted. O atributo *accepted* informa se o artigo está em estado de aceito ou se já foi publicado. Então assume os valores de True, se aceito, e False, se já foi publicado.

3.1.8 ConferencePaper

ConferencePaper é a classe que herda de *Paper* e representa artigos publicados em conferências, ou seja, artigos publicados em *Venues* que são *Conference*. Além dos atributos herdados de *Paper*, essa classe possui o seguinte atributo.

3.1.9 BookManuscript

BookManuscript é a superclasse de *Book* e *BookChapter*. Ela representa um livro. Ela ser tratada como uma classe abstrata por seus filhos facilita consultas que precisam retornar tanto *Book* e *BookChapter* ou apenas um deles. Seus atributos são descritos a seguir.

title. O atributo *title* é o título do livro.

publisher. O atributo *publisher* é o nome da instituição que publicou o livro.

year. O atributo *year* é o ano em que o livro foi publicado.

authors. O atributo *authors* lista todos os nomes dos autores do livro, separados pelo caractere ";".

doi. O atributo *doi* é o *Digital Object Identifier* do livro.

3.1.10 Book

A classe *Book* herda de *BookManuscript*, assim tratando *BookManuscript* como se fosse uma classe abstrata. Ela representa um livro que foi publicado e possui os mesmos atributos de *BookManuscript*.

3.1.11 BookChapter

A classe *BookChapter* herda de *BookManuscript*, assim tratando *BookManuscript* como se fosse uma classe abstrata. Ela representa um capítulo de um livro que foi publicado e além dos atributos herdados de *Book* ela possui o seguinte atributo.

chapter_title. O atributo *chapter_title* é o título do capítulo de livro.

3.1.12 Project

A classe *Project* representa projetos nos quais os pesquisadores cujos currículos Lattes que estão sendo analisados participaram. Seus atributos são descritos a seguir.

name. O atributo *name* é o nome do projeto.

manager. O atributo *manager* é o nome do coordenador do projeto.

team. O atributo *team* lista todos os nomes dos integrantes do projeto, separados pelo caractere ";".

start_year. O atributo *start_year* é o ano em que o projeto foi iniciado.

end_year. O atributo *end_year* é ano em que o projeto terminou.

3.1.13 Membership

Membership é a classe que relaciona um *Researcher* com um *Project*, ou seja, ela informa se um dado *Researcher* participou de um dado *Project*. Seu atributo é descrito a seguir.

principal_investigator. O atributo *principal_investigator* indica se o *Researcher* da relação era ou não o coordenador do *Project* da relação. Tendo valor True, caso ele seja o coordenador, e False, caso não seja.

3.1.14 Patent

A classe *Patent* representa patentes dos pesquisadores cujos currículos Lattes estão sendo analisados. Seus atributos são descritos a seguir.

title. O atributo *title* é o título da patente.

authors. O atributo *authors* lista os nomes dos autores da patente, separados pelo caractere ";".

local_of_registry. O atributo *local_of_registry* indica o escritório em que a patente foi registrada.

number. O atributo *number* é o código que identifica a patente dentre as outras existentes.

year. O atributo *year* é ano em que a patente foi registrada.

type. O atributo *type* indica o tipo de patente que o objeto *Patent* representa. Ele é um *Enum* que pode assumir os valores *PATENT* e *SOFTWARE*, sendo *SOFTWARE* para patentes de *software* e *PATENT* para os demais tipos de patente.

3.1.15 EditorialBoard

A classe *EditorialBoard* representa o trabalho que um dado pesquisador fez atuando como membro de um corpo editorial ou avaliador *ad-hoc* de algum periódico. Seus atributos são descritos a seguir.

journal_name. O atributo *journal_name* indica o nome do periódico onde o pesquisador atuou.

start_year. O atributo *start_year* é o ano em que o trabalho começou.

end_year. O atributo *end_year* é o ano em que o trabalho terminou.

type. O atributo *type* indica o tipo de trabalho em periódico que o objeto *EditorialBoard* representa. Ele é um *Enum* que pode assumir os valores *EDITORIAL_BOARD* e *REVISER*. Sendo *EDITORIAL_BOARD* para corpo editorial e *REVISER* para avaliador *ad-hoc*.

3.1.16 ConferenceOrganization

A classe *ConferenceOrganization* representa o trabalho que um dado *Researcher* fez ao ajudar na organização de algum evento ou conferência. Seus atributos são descritos a seguir.

title. O atributo *title* é o nome da conferência.

year. O atributo *year* é o ano que o trabalho foi realizado.

committee. O atributo *committee* lista o nome dos participantes que ajudaram a realizar o trabalho, separados pelo caractere ";".

3.1.17 Advisement

A classe *Advisement* representa as orientações concluídas de um dado *Researcher*. Seus atributos são descritos a seguir.

student_name. O atributo *student_name* é o nome do estudante que foi orientado.

college. O atributo *college* é o nome da instituição em que ocorreu a orientação.

year. O atributo *year* é o ano que a orientação foi finalizada.

title. O atributo *title* é o título do documento que foi produzido pela orientação.

type. O atributo *type* é o tipo da orientação que o objeto *Advisement* representa. Ele é um *Enum* que pode assumir os valores *BACHELOR*, *MASTER*, *PHD*, *SPECIALIZATION* ou *UNDERGRADUATE_RESEARCH*.

3.1.18 Committee

A classe *Committee* representa as participações em bancas de um dado *Researcher*. Seus atributos são descritos a seguir.

student_name. O atributo *student_name* é o nome do estudante que foi avaliado pela banca.

college. O atributo *college* é o nome da instituição em que ocorreu a banca.

year. O atributo *year* é o ano em que ocorreu a banca.

title. O atributo *title* é o título do documento que foi avaliado pela banca. No caso de banca de concurso público, esse atributo assume como valor o nome da vaga do concurso.

team. O atributo *team* lista os participantes da banca, separados pelo caractere ";".

type. O atributo *type* é o tipo da banca que o objeto *Committee* representa. Ele é um *Enum* que pode assumir os valores *BACHELOR*, *MASTER*, *MASTER_QUALIFICATION*, *PHD*, *PHD_QUALIFICATION*, *SPECIALIZATION* ou *CIVIL_SERVICE_EXAMINATION*.

3.2 Banco de dados

Ao coletar as informações de um currículo Lattes, observamos a necessidade de armazená-las para poder analisá-las e gerar relatórios e visualizações em seguida. Dentre as possíveis formas de armazenamento, decidimos usar um banco de dados que segue o modelo relacional, pois esse modelo consegue representar dados obtidos dos currículos Lattes como atributos de tabelas e, por sua vez, essas tabelas e seus relacionamentos simbolizam como esses dados interagem entre si e com os pesquisadores, de forma a suprir as necessidades do projeto. Um exemplo é um dado pesquisador que possui artigos em periódicos, e esses artigos, por sua vez, possuem dados como *doi*, *year* e *venue*. Esse último atributo representa a relação do artigo com o local onde ele foi publicado, ou seja, uma entrada na tabela *journal*, que representa periódicos e revistas.

A Figura 3 representa o esquema do banco de dados usado no projeto. *book*, *paper* e *venue* são tabelas pai, pois seus filhos possuem comportamento e/ou atributos iguais. A tabela *researcher* representa os pesquisadores e a *affiliation* os anos que eles foram credenciados no programa de pós-graduação. As tabelas *journal_paper* e *conference_paper* simbolizam as publicações em periódicos e conferencias, e as tabelas *journal* e *conference* representam onde foram feitas essas publicações, em periódicos/revistas ou conferências, respectivamente. *published_book* e *published_book_chapter* representam, respectivamente, livros e capítulos de livros publicados. Já a tabela *project* diz respeito aos projetos e *patent* as patentes. E as tabelas *researcher_journal_paper*, *researcher_conference_paper*, *researcher_published_book*, *researcher_published_book_chapter*, *researcher_project* e *researcher_patent* indicam como as tabelas citadas anteriormente se relacionam com *researcher*. Por fim, *researcher_advisement*, *researcher_committee*, *researcher_conference_management* e *researcher_editorial_board* são para orientações, bancas, organização de conferências e atuação em corpos editoriais de periódicos e revistas, respectivamente.

Como visto anteriormente, a biblioteca que usamos para o mapeamento objeto relacional foi a SQLAlchemy e o SQLite foi usado como sistema de gerenciamento de banco de dados, optando por sua implementação *in-memory*. Além de ser mais rápido do que ler e escrever em disco, essa implementação atende às nossas necessidades, visto que o programa não foi pensado com o intuito de ser rodado múltiplas vezes seguidas neste momento. Outras vantagens de usar um banco de dados *in-memory* é não ter a necessidade de se preocupar com atualizar, modificar e/ou apagar seus dados, dado que há a possibilidade do usuário usar diferentes currículos Lattes com diferentes informações entre as execuções do programa.

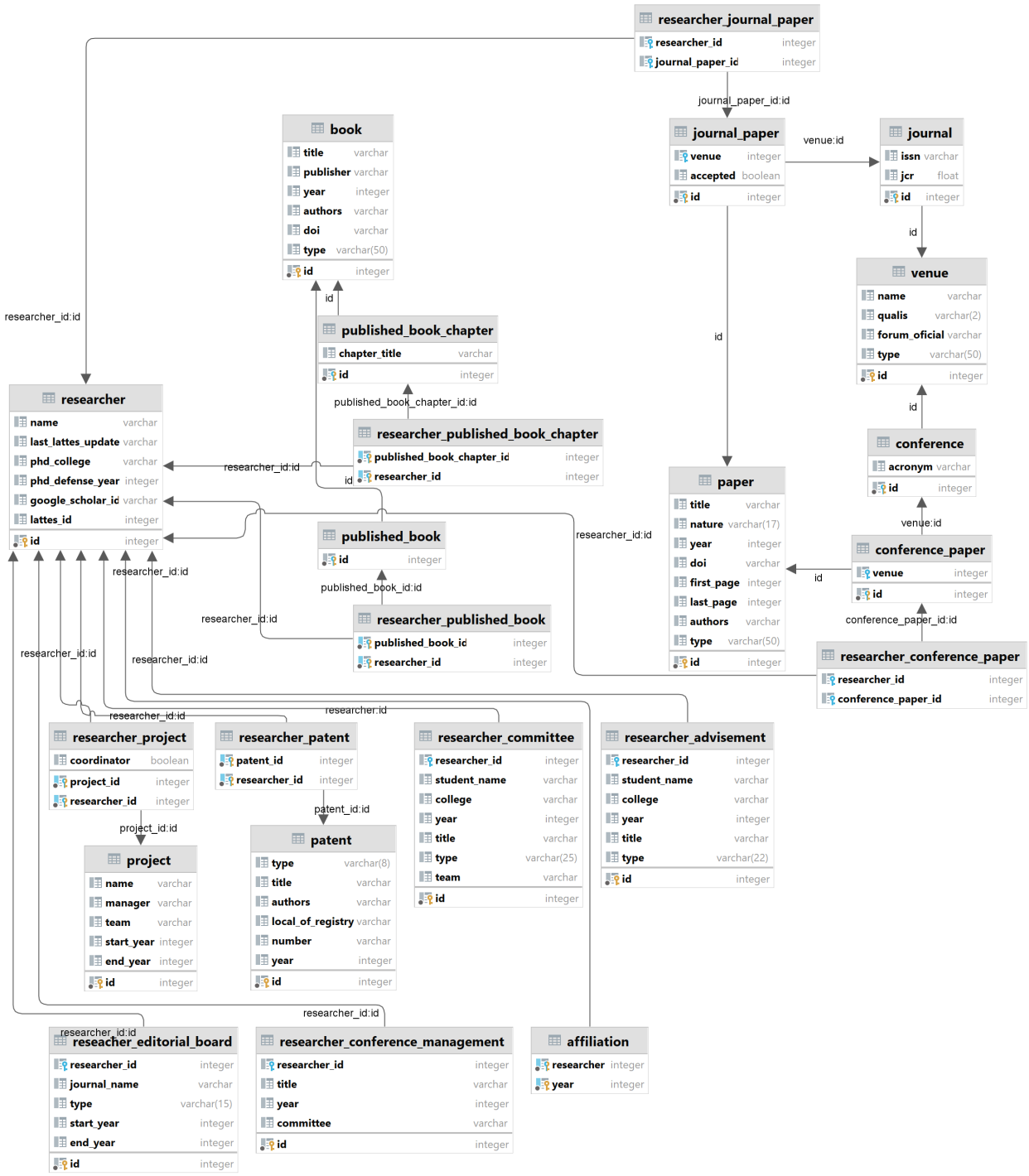


Figura 3 – Esquema do banco de dados

3.2.1 Tratamento de duplicações

Idealmente, não deveria haver entradas com dados duplicados em cada tabela do banco de dados, pois além de ocupar mais espaço de memória desnecessariamente, pode gerar erros nos relatórios e/ou visualizações. Por exemplo, um pesquisador tem seu currículo Lattes analisado e depois um dos coautores de um de seus artigos também tem seu

currículo Lattes analisado, então gostaríamos de armazenar o mesmo artigo somente uma vez no banco de dados com o menor número de dados errados possível. Entretanto, dada as características do currículo Lattes, tentar garantir essa unicidade de dados pode levar a outros erros. Ao analisar o perfil de pesquisadores, quantificando suas informações, um erro de preenchimento em alguma parte do currículo Lattes de um pesquisador analisado primeiro pode causar problemas para os seguintes ao unificar esses dados em apenas uma entrada na tabela.

Dado esse dilema, decidimos por implementar soluções configuráveis, que colocam na mão do usuário a capacidade de decidir ou não, quando realmente ocorre uma duplicação, se ele deseja garantir a unicidade e em quais tabelas. Como visto nas subseções a seguir, dividimos o problema da unicidade em duas partes: a primeira consiste do reporte de duplicação em um currículo Lattes de um pesquisador específico e a outra consiste na unificação das entradas das tabelas desejadas, ou seja, não cadastrar novamente dados consideradas iguais.

3.2.1.1 Duplicação no Lattes

O preenchimento do currículo Lattes é feito manualmente pelo pesquisador, o que pode levar a erros, dentre eles, o de duplicação de dados. Para minimizar tais acontecimentos, uma das soluções implementadas foi a de reportar possíveis dados repetidos no arquivo de log como avisos, o que é feito durante a execução do *script* que extraí os dados dos currículos Lattes e popula o banco de dados, que será discutido no Capítulo 4.

Entretanto, as possíveis duplicatas ainda são registradas no banco de dados do programa, porque as saídas geradas seriam contestáveis e erradas caso originem de falsos negativos. Então, optamos por indicar os casos encontrados em um log. Um dos possíveis usos para esses avisos no arquivo de log é de informar os donos dos respectivos currículos Lattes.

Vale ressaltar que não são todas as tabelas que necessitam da verificação de duplicação, mas apenas aquelas que fazem o relacionamento de um pesquisador com suas produções ou trabalhos, e que podem ter coautores. A seguir, apresentamos as tabelas nas quais esse critério é aplicável, junto com quais atributos são usados para checar uma possível duplicação. Ou seja, sempre que há mais de um dado com os mesmos valores para esses atributos, indicamos o ocorrido no log para que o usuário possa verificar se é uma situação correta ou um erro de preenchimento do Lattes.

- ***researcher_published_book***: *researcher_id* e *published_book.doi*. Caso *published_book.doi* não esteja disponível, o substituímos por *published_book.year* e *published_book.title*.

- ***researcher_published_book_chapter***: *researcher_id* e *published_book_chapter.doi*. Caso *published_book_chapter.doi* não esteja disponível, o substituímos por *published_book_chapter.year* e *published_book_chapter.chapter_title*.
- ***researcher_editorial_board***: *researcher_id*, *journal_name*, *type* e *start_year*. O atributo *type* indica se o pesquisador fez parte do corpo editorial ou se foi revisor, como visto na Seção 3.1.15 deste capítulo.
- ***researcher_conference_management***: *researcher_id*, *title* e *year*.
- ***researcher_patent***: *researcher_id* e *patent.number*.
- ***researcher_journal_paper***: *researcher_id* e *journal_paper.doi*. Caso *journal_paper.doi* não esteja disponível, o substituímos por *journal_paper.title* e *journal_paper.nature*. O atributo *journal_paper.nature* indica se o artigo é resumo, resumo expandido ou completo, como visto na Seção 3.1.6 deste capítulo.
- ***researcher_conference_paper***: *researcher_id* e *conference_paper.doi*. Caso *conference_paper.doi* não esteja disponível, o substituímos por *conference_paper.title* e *conference_paper.nature*. O atributo *conference_paper.nature* indica se o artigo é resumo, resumo expandido ou completo, como visto na Seção 3.1.6 deste capítulo.
- ***researcher_project***: *researcher_id* e *project.name*.
- **researcher_advisement**: *researcher_id*, *student_name*, *type* e *year*. O atributo *type* indica se a orientação foi de graduação, mestrado, doutorado, especialização ou iniciação científica, como na Seção 3.1.17 deste capítulo.
- ***researcher_committee***: *researcher_id*, *student_name*, *type* e *year*. O atributo *type* indica se a banca foi de graduação, mestrado, doutorado, qualificação de mestrado, qualificação de doutorado, especialização ou concurso público, como visto na Seção 3.1.18 deste capítulo.

3.2.1.2 Unificação

Algumas informações acabam se repetindo ao analisar currículos Lattes de múltiplos pesquisadores que colaboram. Elas tendem a pertencer a tabelas que muita das vezes têm mais de um autor, como artigos, por exemplo. Então, pode ser interessante para o usuário ter apenas uma entrada dentro do banco de dados representando todas as ocorrências espalhadas pelos currículos Lattes que estão sendo analisados. Por exemplo, quando se deseja gerar um relatório para a CAPES referente à produção do programa de pós-graduação, não faria sentido que um mesmo artigo aparecesse múltiplas vezes, só porque é resultado da colaboração de diversos pesquisadores do programa. Nessa circunstância, apesar do artigo estar declarado em diversos Lattes que estão sendo analisados, o desejável

seria que uma única ocorrência fosse inserida no banco de dados, e essa ocorrência fosse vinculada aos vários pesquisadores que colaboraram na publicação do artigo.

Por outro lado, esse processo de unificação é propenso a erros, já que os dados não são necessariamente idênticos. Ao unificar, os dados preenchidos por diferentes pesquisadores sofrerão pequenos ajustes que podem ser indesejados em determinadas situações. Por exemplo, suponha que dois candidatos de um concurso para professor foram coautores em um artigo, e um dos candidatos registrou o artigo com erros no seu Lattes. Ao fazer a unificação, a versão com erro pode contribuir com dados para a versão unificada, que fica atrelada a ambos os pesquisadores. Como consequência, o pesquisador que lançou corretamente o artigo pode ser prejudicado no concurso. Numa situação como essa, o ideal seria não unificar os dados, pois não há interesse em fazer análises conjuntas, como são feitas para contabilizar a produção de um programa de pós-graduação. Nessa análise, o risco de algum erro não justifica os potenciais benefícios da unificação.

Por padrão, o programa Perfil 2.0 replica no banco todas as informações de todos os currículos Lattes, garantindo assim que a análise reflita 100% os dados registrados no Lattes. Contudo, o usuário pode indicar que deseja garantir a unicidade de algumas tabelas. Para fazê-lo, basta acessar o arquivo *config.py* e mudar os valores das variáveis que começam com *unify* para *True*. A seguir, indicamos para quais tabelas há essa possibilidade e quais atributos são analisados para identificar que duas entradas são equivalentes. Vale ressaltar que toda vez que uma unificação ocorre, a mesma é reportada no arquivo de log.

- ***published_book***: *doi*. Caso *doi* não esteja disponível, o substituímos por *title* e *year*.
- ***published_book_chapter***: *doi*. Caso *doi* não esteja disponível, o substituímos por *chapter_title* e *year*.
- ***patent***: *number*.
- ***journal_paper***: *doi*. Caso *doi* não esteja disponível, o substituímos por *title* e *nature*. O atributo *nature* indica se o artigo é resumo, resumo expandido ou completo, como visto na Seção 3.1.6 deste capítulo.
- ***conference_paper***: *doi*. Caso *doi* não esteja disponível, o substituímos por *title* e *nature*. O atributo *nature* indica se o artigo é resumo, resumo expandido ou completo, como visto na Seção 3.1.6 deste capítulo.
- ***project***: *name*.

Dessas tabelas, *project* é um caso especial, pois o atributo *project.name* pode não ser exatamente aquele que está escrito no Lattes de um dado pesquisador. Isso ocorre

pois alguns pesquisadores colocam informações adicionais no nome do projeto, como, por exemplo, o valor do projeto ou o edital em que o projeto foi aprovado. Por conta disso, usamos o retorno de um procedimento envolvendo dicionários de sinônimos, similaridades e maior subsequência comum, que é explicado na Seção 3.3.

Vale notar que algumas das tabelas que fazemos detecção de duplicação (ver Seção 3.2.1.1) não são passíveis de unificação. Isso ocorre pois algumas das tabelas anteriores são as mais utilizadas em relatórios, ou também fazem parte de publicações. Todavia, se o recurso de unificação for útil, essa estratégia pode ser aplicada as tabelas *researcher_editorial_board*, *researcher_conference_management*, *researcher_committee* e *researcher_advisement*.

3.3 Coleta de dados

Para popular o banco de dados e inicializar os objetos que compõem o programa, é necessário a coleta de vários dados. Em sua maioria, esses dados vêm diretamente dos currículos Lattes dos pesquisadores que estão sendo analisados. Entretanto, em certo caso, que é explicado no Capítulo 4, também é necessário utilizar a plataforma Google Scholar para captura de dados. Ademais, alguns dados não são obtidos diretamente dos currículos Lattes, mas sim a partir da análise de seus dados e consultas a arquivos que ficam na pasta */resources* do programa ou estruturas de dados criadas em tempo de execução do programa. Nas seções a seguir buscamos explicar cada um desses tipos de coleta de dados.

3.3.1 Currículo Lattes

Como dito anteriormente, a parte massiva dos dados é obtida diretamente dos currículos Lattes. Para isso, primeiro o programa recupera os identificadores dos currículos Lattes a serem analisados no arquivo de pesquisadores. Depois, para cada identificador, procura o currículo Lattes correspondente na pasta */lattes* do programa, dado que o nome do arquivo com o currículo Lattes é o identificador concatenado com .zip. Caso o programa encontre o arquivo que contém o currículo Lattes, ele abre o arquivo .zip e depois o arquivo .xml que é de fato o currículo Lattes de um pesquisador.

Para processar os currículos Lattes, que são arquivos .xml, decidimos usar a ferramenta lxml. Essa ferramenta apresenta uma interface de programação de aplicação parecida com a ElementTree, nativa do Python. Ou seja, ela transforma os currículos Lattes em árvores e localiza os elementos necessários usando XML Path Language (CONSORTIUM, 2017), assim extraindo os dados necessários de cada currículo Lattes.

3.3.2 Google Scholar

Para um dos relatórios de saída do programa, que é visto mais detalhadamente no Capítulo 4, é necessário coletar alguns dados da plataforma Google Scholar de cada pesquisador. Como visto na Seção 3.1.1 deste capítulo, o identificador do Google Scholar de um dado *Researcher* é obtido ao ler o arquivo de pesquisadores que o usuário preencheu. Então, quando é preciso capturar os dados do Google Scholar, se recupera esse dado do objeto *Researcher* e se consulta o URL daquele pesquisador. Por fim, usando a biblioteca Beautiful Soup[1], o programa lê aquela página HTML e depois extrai os dados necessários da página.

3.3.3 Pasta /resources

Anteriormente, foi visto que alguns atributos dos objetos *Researcher*, *lattes_id* e *google_scholar_id* são obtidos a partir do arquivo de pesquisadores que é preenchido pelo usuário. Contudo, além deles, há outros dados que são colhidos a partir de consultas a arquivos que ficam na pasta */resources* do programa. Eles são *Affiliations.year*, *Venue.qualis*, *Journal.jcr*, *Venue.official_forum* e *Project.name* caso seu *unify* tenha valor True.

Arquivos de credenciados. Para saber os anos que pesquisadores (*Researchers*) estavam credenciados no programa de pós-graduação, o programa acessa a pasta */resources/affiliations*. Dentro dela, há arquivos de texto que são nomeados com o ano que eles correspondem e o conteúdo de cada um deles são os nomes dos pesquisadores credenciados naquele ano. Vale ressaltar que esses nomes são separados por uma quebra de linha. O programa varre cada arquivo identificando os pesquisadores (*Researchers*) que estavam credenciados no respectivo ano e, caso ache algum, cadastra aquela relação, ou seja, um objeto *Affiliation*, no banco de dados.

Arquivos de Qualis. Dado que o valor de Qualis de uma conferência ou de um periódico pode mudar, o currículo Lattes não informa esse valor. Entretanto, para gerar algumas saídas do programa, é imprescindível armazená-lo. Visto essa demanda, disponibilizamos arquivos dentro da pasta */resources/qualis* que contêm o valor de Qualis de periódicos e conferências, sendo eles qualis-journals.xlsx e qualis-conferences.xlsx, respectivamente. Então, para recuperar esses valores de Qualis, ao identificar o *official_forum* de um *Venue*, se consulta dicionários que foram construídos usando os arquivos citados anteriormente. Esses dicionários permitem obter o valor de Qualis referente à *Venue* que será armazenada no banco de dados, seja ela um *Journal* ou uma *Conference*.

Arquivos de *Journal Citation Reports*. Assim como o valor de Qualis, o fator de impacto declarado no *Journal Citation Reports* também é disponibilizado em um arquivo dentro da pasta */resources*. Esse arquivo, nomeado jcr.xlsx, contém um nome de periódico,

[1] <www.crummy.com/software/BeautifulSoup/bs4/doc/>

seu ISSN e o fator de impacto segundo o *Journal Citation Reports* em cada linha. Apesar da possibilidade de usar os nomes dos periódicos, decidimos usar o ISSN como chave para construir um dicionário onde o JCR é o valor indexado pela chave. Essa decisão foi tomada pois os nomes de periódicos vindo dos currículos Lattes podem não ser exatamente os mesmos do arquivo. Desse modo, ao coletar um ISSN originado de um currículo Lattes, se consulta o dicionário, obtendo-se o valor do JCR que será cadastrado no banco de dados junto àquele *Journal*.

3.3.4 Tratamento de Strings Diferentes

Como visto nas Seções 3.1 e 3.2 deste capítulo, os nomes de periódicos, conferências e projetos nos currículos Lattes de diferentes pesquisadores nem sempre são iguais. Isso torna mais difícil obter uma única string para usar para os mesmos casos, ou seja, o atributo *official_forum* para objetos *Venue* e o atributo *name* para *Project*, caso suas configurações *unify* tenham o valor True.

Visando solucionar esse problema, desenvolvemos alguns passos para tentar garantir que pelo menos uma parte dos casos equivalentes seja capturada. A solução envolve primeiro checar se é uma correspondência direta (a sequência de caracteres ser a mesma). Caso contrário, se consulta um dicionário de sinônimos, ou seja, strings diferentes, mas que apontam para o mesmo nome (e.g. caso a conferência de um artigo em um currículo Lattes esteja nomeada como "Brazilian Symposium on Bioinformatics (BSB2011)"ou "Brazilian Symposium on Bioinformatics 2012", ela será considerada como "Brazilian Symposium on Bioinformatics (BSB)"). Caso essa consulta também venha a falhar, tenta-se consultar um dicionário de similaridades que é construído à medida em o programa é executado. No último caso, se ainda assim falhar, é feita uma tentativa de encontrar similares usando o algoritmo de detecção da maior subsequência comum. O Algoritmo 1 ilustra um pseudocódigo de como esses passos se comportam.

No Algoritmo 1, *nome* é a string extraída do currículo Lattes, *nomeOficial* é um recurso somente para demonstrar a correspondência direta, *sinônimos* é o dicionário de sinônimos, *similares* o dicionários de similaridades e *detectarSimilar* é a função que calcula a maior subsequência comum entre *nome* e todas as strings de *estruturaDeSimilares*, que é um dicionário para *Venues* e uma lista de strings para *Project*.

3.3.4.1 Sinônimos

Na pasta */resources/synonyms* se encontram três arquivos que devem ser preenchidos pelo usuário. Eles são journals_synonyms.xlsx, conferences_synonyms.xlsx, projects_synonyms.xlsx e correspondem respectivamente aos sinônimos para os nomes de periódicos, conferências e projetos. A primeira coluna desses arquivos corresponde aos nomes que serão usados no programa. No caso de objetos *Venue*, eles são o atributo

Algorithm 1 Algoritmo de casamento de strings.
Entrada: variável *nome*.
Saída: string usada para representar todas as suas equivalentes

```
se nome = nomeOficial então                          ▷ correspondência direta
    return nome
fim se
se nome em sinônimos então          ▷ correspondência no dicionário de sinônimos
    return sinônimos[nome]
fim se
se nome em similares então           ▷ correspondência no dicionário de similares
    return similares[nome]
fim se
similar = detectaSimilar(nome, estruturaDeSimilares, similaridadeMinima, similares)
▷ maior subsequência comum
se similar ≠ null então
    return similar
fim se
```

official_forum. Já em objetos *Project*, é o nome usado para unificar o atributo *name* caso o *unify* tenha valor True. As demais colunas são as strings que devem ser consideradas como sinônimos da primeira. Por exemplo, os textos escritos nas colunas B5, C5, D5 e E5 são todos sinônimos para o *official_forum* listado na coluna A5.

Quando o programa vai criar o dicionário de sinônimos, ele consulta cada um desses arquivos, inicializando um dicionário para cada um. Para cada linha, ele cria uma chave para cada valor que não está na primeira coluna e indexa o valor que está na primeira coluna por essas chaves. Por exemplo, tendo como chaves as strings em B5, C5, D5 e E5 todas elas têm o texto em A5 como valor.

3.3.4.2 Similares

Assim como para sinônimos, há um dicionário de similaridades para periódicos, conferências e projetos. Contudo, eles são inicializados vazios e são preenchidos conforme o programa lê os currículos Lattes dos pesquisadores. Esse processo é feito dentro de uma função que checa qual string é a mais similar ao ser comparada com outras, e é melhor explicado a seguir.

Como visto anteriormente, caso não se consiga nenhuma correspondência em nenhum dicionário, o programa tenta encontrar qual string, entre várias, é a mais similar ao nome capturado no currículo Lattes. No caso de periódicos e conferências se utiliza seus dicionários de Qualis e para projetos, os projetos já cadastrados no banco de dados para comparação. Para cada par de string, se faz uma conta envolvendo a maior subsequência comum (BERGROTH; HAKONEN; RAITA, 2000) entre os dois e checa-se se o resultado é o maior encontrado até então, e se é maior que a similaridade mínima estipulada pelo

Algorithm 2 Algoritmo de detecção de similaridade usando a maior subsequência comum.
Entrada: *nome*, *dicionarioDeSimilares*, *similaridadeMinima*, *estruturaDeSimilares*.
Saída: string mais similar dentre as contidas em *estruturaDeSimilares*, ou *null*

```
maiorSimilaridade ← 0
stringSimilar ← null
for all texto em estruturaDeSimilares faça
    similaridade ← calculaSimilaridade(nome, texto)
    se (similaridade > similaridadeMinima) E (similaridade > maiorSimilaridade) então
        maiorSimilaridade ← similaridade
        stringSimilar ← texto
    fim se
fim para
se stringSimilar ≠ null então
    adicionaAoDicionario(nome, stringSimilar, dicionarioDeSimilares)
fim se
return stringSimilar
```

usuário, que será vista em mais detalhes no Capítulo 4. Por fim, se adiciona no dicionário de similaridade o nome capturado no currículo Lattes como chave e a string com maior similaridade como valor, usando-a também como retorno da função. O pseudocódigo pode ser visto no Algoritmo 2.

No Algoritmo 2, *calculaSimilaridade* é a função que, dado duas string, calcula a similaridade entre elas usando a maior subsequência comum e retornando um valor entre 0 e 1, que representa o grau de similaridade considerando o percentual de caracteres em comum dentre todos os caracteres. Além disso, *nome* é o nome do periódico, conferência ou projeto que foi coletado em um currículo Lattes, *similaridadeMinima* é um valor entre 0 e 1 estabelecido pelo usuário, que será explicado no Capítulo 4. Finalmente, *adicionaAoDicionario* é a função que adiciona uma nova entrada no dicionário *dicionarioDeSimilaridades*, que é o dicionário passado para função, seja ele de periódicos, conferências ou projetos.

3.4 Considerações finais

Neste capítulo, vimos como o programa Perfil 2.0 foi modelado e como ele faz para coletar os dados tanto do currículo Lattes quanto do Google Scholar. No próximo capítulo, abordamos como ele faz para gerar suas saídas, ou seja, os relatórios e visualizações. Além disso, aproveitamos para falar também dos *scripts* para utilização do programa.

4 Relatórios e Visualizações

O principal objetivo do programa Perfil 2.0 é gerar relatórios e visualizações. Por esse motivo, na pasta raiz, o usuário tem acesso a diversos *scripts* escritos em Python que ele pode executar em um terminal de comando de um sistema operacional. Esses *scripts* sempre iniciam fazendo a coleta de dados, descrita no Capítulo 3. Em seguida, fazem o processamento, que varia de *script* para *script*. No final da execução, a partir das informações obtidas dos currículos Lattes, o usuário obtém arquivos .xslx ou representações visuais dos dados.

Os *scripts* que geram visualizações são *visualize.py*, *visualize.1.5.py*, responsáveis por produzirem *boxplots*, e *generate_collaboration_graphs.py*, que exibe grafos de colaboração. Já os arquivos .xlsx gerados pelo programa são relatórios com diferentes propósitos e informações. Por exemplo, o *script write_profile.py* escreve informações quantitativas sobre idade acadêmica, projetos, orientações, bancas e publicações de pesquisadores. Esse relatório pode ser usado para analisar os perfis de candidatos de concurso. Além do *script write_profile.py*, também implementamos os *scripts generate_datacapes.py*, *generate_researcher_progression_report.py* e *generate_configured_reports.py*, sendo esse último um caso especial, pois ele permite que o usuário configure as informações que deseja como saída. A Tabela 1 ilustra os *scripts* do programa Perfil 2.0 com um resumo de suas respectivas funções.

Ademais, existem *scripts* auxiliares com distintas funções: *download.py* auxilia o usuário a baixar os currículos Lattes dos pesquisadores, *database_test.py* serve para testar se o código de coleta de informações dos currículos Lattes está sendo executado corretamente, *populate_database.py* popula o banco de dados in-memory e *config.py* contém os parâmetros modificáveis para a execução do programa. Sobre esse último, os parâmetros gerais serão explicados na seção a seguir, mas o *script* também contém parâmetros específicos de determinados *scripts* que serão descritos nas seções de seus respectivos *scripts*.

Um guia de utilização dos scripts do programa Perfil 2.0 pode ser encontrado no seguinte endereço: <https://github.com/gems-uff/perfil#readme>.

4.1 config.py

O *script config.py* contém as variáveis de configuração do programa Perfil 2.0. Nele, o usuário pode modificar o comportamento do programa. As variáveis podem ser gerais, ou seja, afetam o programa como um todo, ou podem ser específicas de outros

Tabela 1 – *Scripts* do programa Perfil 2.0.

Nome do *script*	**Função**
config.py	Variáveis de configuração do programa.
download.py	Auxilia a baixar os currículos Lattes.
database_test.py	Roda os testes automatizados.
populate_database.py	Popula o banco de dados.
write_profile.py	Preenche o arquivo de pesquisadores com informações quantitativas.
generate_datacapes.py	Gera relatórios do Programa de Pós-Graduação.
generate_researcher_progression_report.py	Gera relatórios para auxiliar na progressão para o cargo de Professor Titular.
generate_configured_reports.py	Gera relatórios configurados pelo usuário.
visualize.py	Gera *boxplots* a partir dos dados quantitativos de um arquivo de pesquisadores preenchidos.
visualize-jcr.1.5.py	Gera *boxplot* de publicações com JCR maior que 1,5 a partir dos dados quantitativos de um arquivo de pesquisadores preenchidos.
generate_collaboration_graphs.py	Gera grafos de colaboração.

scripts. Aqui explicamos apenas as variáveis gerais, pois as específicas são explicadas em seus respectivos *scripts*. Vale ressaltar que as variáveis que indicam caminhos de pastas ou arquivos já vêm preenchidas com uma convenção própria, mas o usuário é livre para modifica-las de acordo com suas necessidades. A Tabela 2 mostra as variáveis gerais do *script config.py*. A primeira coluna contém nome da variável e a segunda coluna o resumo de sua função. Já as Tabelas 3, 4, 5, 6, 7 e 8 mostram as variáveis que atuam nos *scripts generate_datacapes.py* e *generate_configured_reports.py*, *generate_datacapes*, *generate_researcher_progression_report.py*, *generate_configured_reports.py*, *visualize.py* e *visualize-jcr1.5.py*, e *generate_collaboration_graphs.py*, respectivamente. Suas colunas contém o mesmo tipo de informação que as colunas da Tabela 2.

Tabela 2 – Variáveis gerais do *config.py*.

Variável	**Função**
resources_path	Diretório da pasta de recursos do programa Perfil 2.0 (*/resources*).
output_path	Diretório da pasta dos relatórios gerados pelo programa Perfil 2.0.
researchers_file	Caminho do arquivo de pesquisadores.
start_year	Ano inicial do horizonte de coleta. (Inclusive).
end_year	Ano final do horizonte de coleta. (Inclusive).
unify_conference_paper	Ativa ou desativa o unificador de *ConferencePaper*.

unify_journal_paper	Ativa ou desativa o unificador de *JournalPaper*.
unify_project	Ativa ou desativa o unificador de *Project*.
unify_book	Ativa ou desativa o unificador de *Book*.
unify_chapter	Ativa ou desativa o unificador de *BookChapter*.
unify_patent	Ativa ou desativa o unificador de *Patent*.
conferences_minimum_similarity	Valor (entre 0 e 1) mínimo para *Conference.name* serem considerados similares.
journals_minimum_similarity	Valor (entre 0 e 1) mínimo para *Journal.name* serem considerados similares.
conferences_papers_title _minimum_similarity	Valor (entre 0 e 1) mínimo para *ConferencePaper.title* serem considerados similares.
journals_papers_title _minimum_similarity	Valor (entre 0 e 1) mínimo para *JournalPaper.title* serem considerados similares.
project_name_minimum _similarity	Valor (entre 0 e 1) mínimo para *Project.name* serem considerados similares.
QualisLevel	Classe de enumeração com os níveis Qualis.
df_jcr	Caminho para o arquivo que contém os JCR.
jcr	Dicionário com o JCR.
df_qualis_conferences	Caminho para o arquivo que contém o Qualis de conferências.
conferences_qualis	Dicionário com o Qualis de conferência.
df_qualis_journals	Caminho para o arquivo que contém o Qualis de periódicos.
journals_qualis	Dicionário com o Qualis de periódicos.
conferences_synonyms	Dicionário com os sinônimos de conferências.
journals_synonyms	Dicionário com os sinônimos de periódicos.
projects_synonyms	Dicionário com os sinônimos de projetos.
lattes_dir	Diretório que contém os currículos Lattes.
affiliations_dir	Diretório com os arquivos dos pesquisadores credenciados.
similarity_dir	Diretório dos arquivos de similaridade gerados pelo programa.

Tabela 3 – Variáveis dos *scripts generate_datacapes.py* e *generate_configured_reports.py* em *config.py*.

Variável	**Função**
qualis_journal_points	Dicionário com os valores dos níveis de Qualis de periódicos.

qualis_conference_points	Dicionário com os valores dos níveis de Qualis de conferências.

Tabela 4 – Variáveis do *script generate_datacapes.py* em *config.py*.

Variável	**Função**
datacapes_minimum_similarity_titles	Valor (entre 0 e 1) mínimo para títulos de artigos serem considerados os mesmos na hora fazer a produção do Programa de Pós Graduação.
generate_datacapes_output_dir	Diretório dos arquivos gerados pelo *script generate_datacapes.py*

Tabela 5 – Variáveis do *script generate_researcher_progression_report.py* em *config.py*.

Variável	**Função**
generate_researcher_progression_report_output_dir	Diretório dos arquivos gerados pelo *script generate_researcher_progression_report.py*

Tabela 6 – Variáveis do *script generate_configured_reports.py* em *config.py*.

Variável	**Função**
reports_as_new_worksheets	Ativa ou desativa os relatórios do usuário serem abas em um único arquivo.
new_worksheet_if_conflict	Ativa ou desativa a criação de abas nos relatórios do usuário caso haja conflito.
configured_reports	Dicionário com os relatórios do usuário.
generate_reports_output_dir	Diretório dos arquivos gerados pelo *script generate_configured_reports.py*

Tabela 7 – Variáveis dos *scripts visualize.py* e *visualize-jcr1.5.py* em *config.py*.

Variável	**Função**
subject	Pesquisador a ser comparado nas *boxplots*.

print_subject	Liga ou desliga a comparação com um pesquisador.
build_dir	Diretório de saída das *boxplots* geradas pelo programa.

Tabela 8 – Variáveis do *script* *generate_collaboration_graphs.py* em *config.py*.

Variável	**Função**
collaboration_graphs_alpha	Valor (entre 0 e 1) da força inicial dos grafos de colaboração.
collaboration_graphs_alpha_decay	Taxa (entre 0 e 1) em que a força inicial dos grafos de colaboração chegam a 0.

A seguir, detalhamos o propósito de cada variável geral do *script* *config.py*.

resources_path. Essa variável indica o caminho da pasta *resources*, ou seja, onde os arquivos explicados na Seção 3.3.3 do Capítulo 3 se encontram (arquivos de credenciados, arquivos de Qualis, arquivo de JCR e arquivos de sinônimos).

output_path. Essa variável indica o caminho da pasta de saída, ou seja, onde o programa armazena os arquivos de gerados, sejam eles os dicionários de sinônimos, explicados no Capítulo 3, ou os relatórios gerados por outros *scripts*. Vale ressaltar que apesar desse caminho ser usado para construir o caminho de saída de outros *scripts*, o usuário é livre para modificá-lo.

researchers_file. Essa variável indica o caminho para o arquivo que contém os pesquisadores que o usuário deseja que tenham suas informações coletadas pelo programa, ou seja, o arquivo de pesquisadores. Um maior detalhamento desse arquivo é visto na Seção 4.1.1 deste capítulo.

lattes_dir. Essa variável indica o caminho onde os currículos Lattes dos pesquisadores se encontram.

start_year e end_year. Apesar do programa coletar todos os dados dos currículos Lattes, os relatórios consideram apenas os dados dentro dos anos estabelecidos como horizonte de coleta. As variáveis *start_year* e *end_year* indicam o ano inicial e final, respectivamente. Destacamos que ambos são inclusos no horizonte de análise, ou seja, o intervalo de análise é dado por [*start_year*, *end_year*].

Variáveis de unificação. As variáveis *unify_conference_paper*, *unify_journal_paper*, *unify_project*, *unify_book*, *unify_chapter* e *unify_patent* são respectivamente as variáveis que configuram se é desejado a unificação das classes *ConferencePaper*, *JournalPaper*, *Project*, *Book*, *BookChapter* e *Patent*, respectivamente. Caso seu valor seja True, a funcionali-

dade estará ligada. Caso contrário, desligada. Um maior detalhamento desta funcionalidade é apresentado na Seção 3.2.1 do Capítulo 3.

journals_minimum_similarity, conferences_minimum_similarity e project_name_minimum_similarity. Essas variáveis são usadas para decidir se o programa considera um nome de periódico, conferência ou projeto, respectivamente, similar ou não no Algoritmo 2, explicado na Seção 3.3.4 do Capítulo 3. Ou seja, indicam o limiar de similaridade que deve ser usado nas comparações. As variáveis devem ter valor entre 0 e 1.

journals_papers_title_minimum_similarity e conferences_papers_title_minimum_similarity. Essas variáveis são usadas para indicar qual o limiar mínimo de similaridade que o programa deve usar para decidir se títulos de artigo de periódico ou conferência, respectivamente, são similares ou não. São usadas somente se as variáveis *unify_journal_paper* e/ou *unify_conference_paper* estiverem com o valor True, e devem ter o valor entre 0 e 1.

QualisLevel. É uma classe de enumeração que contém todos os níveis Qualis possíveis que podem ser atribuídos para um *Venue*. Cada um dos seus atributos deve ser um nível Qualis. O valor de cada atributo deve ser seu respectivo nível Qualis como string.

Caminhos dos arquivos de resources e dicionários. As variáveis *df_jcr*, *df_qualis_journals* e *df_qualis_conferences* indicam, nos parâmetros passados, o diretório dos arquivos com os valores de *Journal Citation Reports*, Qualis de periódicos e Qualis de conferências, respectivamente. Já as variáveis *jcr*, *journals_qualis* e *conferences_qualis* são, na mesma ordem, os dicionários dos mesmos. As variáveis *conferences_synonyms*, *journals_synonyms* e *projects_synonyms* são os dicionários de sinônimos para conferências, periódicos e projetos, respectivamente. Ademais, *affiliations_dir* é o caminho para o diretório dos arquivos de credenciados. O uso dessas variáveis pode ser visto melhor na Seção 3.3.3 do Capítulo 3.

similarity_dir é a variável que indica onde os dicionários de similaridade serão gerados após o programa terminar de popular o banco de dados. Um maior detalhamento está presente na Seção 3.3.4 do Capítulo 3.

qualis_journal_points e qualis_conference_points. Essas variáveis são dicionários que possuem o *Enum* de nível de Qualis como chaves e números de ponto flutuante como valores. Elas servem para recuperar o valor que a CAPES dá para publicações em cada um dos níveis do Qualis. Um maior detalhamento pode ser visto na Seção 3.1.3 do Capítulo 3.

4.1.1 Arquivo de pesquisadores

O arquivo de pesquisadores é um arquivo Excel (.xlsx) que deve ser construído ou editado pelo usuário. Ele deve conter o cabeçalho: Nome, ID Lattes e ID Scholar em sua primeira linha. Abaixo do cabeçalho, o usuário deve escrever o nome de cada pesquisador na primeira coluna, o ID do currículo Lattes daquele pesquisador na segunda coluna e,

por fim, o ID da plataforma Google Scholar na terceira coluna, sendo um pesquisador por linha. Um exemplo de como o preenchimento deve ser feito pode ser visto na Figura 5.

4.2 download.py

O *script download.py* foi reaproveitado da antiga versão do programa Perfil 2.0. Ele ajuda o usuário a baixar os currículos Lattes dos pesquisadores listados no arquivo de pesquisadores (Seção 4.1.1). Vale notar que a interação do usuário acontece somente usando o terminal de comando do sistema operacional e os diretórios do computador. Ao executar o *script*, o programa abre a página web, no navegador padrão do usuário, para baixar o currículo Lattes do primeiro pesquisado. Então, o usuário deve baixá-lo e salvá-lo na pasta apontada pela variável *lattes_dir*. Em seguida, o programa irá abrir a página para baixar o currículo Lattes do segundo pesquisador, o usuário deve fazer o mesmo. Esse processo ocorrerá sucessivamente para todos os pesquisadores do arquivo de pesquisadores.

4.3 populate_database.py

O *script populate_database.py* coleta e popula o banco de dados in-memory com as informações dos pesquisadores que estão no arquivo de pesquisadores. Ele também gera os arquivos de similares para conferências, periódicos e projetos que o usuário pode usar para editar os arquivos de sinônimos. Esses arquivos são descritos na Seção 3.3.4 do Capítulo 3. Quando o usuário decide que a identificação de um evento similar foi feita de forma correta, ele pode passar a string detectada como similar para o arquivo de sinônimos, o que evitará que o cálculo de similaridade para uma string igual, encontrada em outro currículo, tenha que ser efetuado.

Os arquivos de similares são planilhas contendo na sua primeira coluna as strings usadas para todos os casos similares e as demais colunas naquela linha são os casos considerados similares. Tomando a Figura 4 como exemplo, International Workshop on Theory and Practice of Provenance (TaPP), USENIX Workshop on the Theory and Practice of Provenance (TaPP) e International Workshop on Theory and Practice of Provenance (TAPP) foram considerados nomes de conferência similares. Para todos eles, a string International Workshop on Theory and Practice of Provenance (TaPP) (especificada na primeira coluna do arquivo) foi usada pelo programa ao invés de usar a string que foi encontrada no currículo do pesquisador.

4.4 write_profile.py

O *script write_profile.py* é responsável por quantificar diversas informações dos pesquisadores listados no arquivo Excel e escrevê-las no mesmo arquivo. Ele não gera nenhum arquivo novo, apenas altera o arquivo de pesquisadores existente, acrescentando novas colunas. Por isso, é imprescindível que o arquivo de pesquisadores não esteja aberto durante a execução do *script*. Além dos dados coletados dos currículos Lattes, esse *script* também coleta dados da plataforma Google Scholar de cada pesquisador, o que faz com que uma conexão com a internet seja necessária ao executá-lo.

Como se pode ver na Figura 6, com exceção das informações escritas pelo usuário, vistas na Seção 4.1 deste capítulo, todas as métricas são quantitativas. Vale notar que essa figura não contempla todas as métricas do arquivo, ao todo elas são: "Nome", "ID Lattes", "ID Scholar", "Ano do Doutorado", "Idade Acadêmica", "Participações em Projetos", "Projetos Coordenados", "Projetos", "Orientações de Mestrado", "Orientações de Doutorado", "Orientações", "Bancas de Mestrado", "Bancas de Doutorado", "Bancas", "Publicações em Congressos", "Publicações em Periódicos", "Publicações", "Publicações JCR", "Publicações JCR > 1,5", "Aceitações JCR > 1,5", "Artigos JCR > 1,5", "Citações", "H-Index(total)" e "H-Index (anual)". O valor de cada métrica é correspondente ao período do horizonte de coleta, mas para cada uma delas, há uma com "(total)" concatenado ao seu nome, e outra com "(anual)" também concatenado ao nome, isso é, com exceção de "H-Index (total)" e "H-Index (anual)". As métricas que possuem o texto "(total)" em seu cabeçalho dizem respeito a todo o currículo Lattes do seu pesquisador e não somente aos anos dentro do horizonte de coleta. Já métricas com o texto "(anual)" em seu cabeçalho, são as respectivas métricas anteriores divididas pelo valor em Idade Acadêmica. Vale notar que "Citações" e "H-Index" são os dados extraídos do Google Scholar. A seguir explicamos cada métrica.

Nome. O nome do pesquisador.

ID Lattes. O identificador do currículo Lattes do pesquisador.

ID Scholar. O identificador do pesquisador na plataforma Google Scholar.

Ano do Doutorado. O ano em que o pesquisador concluiu seu doutorado.

Idade Acadêmica. O ano em que o *script write_profile.py* foi executado subtraído de Ano do Doutorado.

International Workshop on Science Gateways (IWSG)	International Workshop on Scientific Workflows (SWF)
International Conference on Quality of Information and Communications Technology (QUATIC)	International Conference on the Quality of Information and Communications Technology (QUATIC)
IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)	International Conference on Software Analysis, Evolution and Reengineering (SANER)
International Conference on the Practice and Theory of Automated Timetabling (PATAT)	3 rd International Conference on the Practice and Theory of Automated Timetabling

Figura 4 – Exemplo de células de um arquivo de conferência similares

	A	B	C
1	**Nome**	**ID Lattes**	**ID Scholar**
2	Célio Vinicius Neves de Albuquerque	4641684220602580	cB78oMQAAAAJ
3	Leonardo Gresta Paulino Murta	1565296529736448	VEbJeB8AAAAJ
4	Vanessa Braganholo Murta	0060120644445370	QT9-AZ4AAAAJ

Figura 5 – Exemplo de como o arquivo de pesquisadores deve ser preenchido pelo usuário

Participações em Projetos. O número de participações em projetos do pesquisador.

Projetos Coordenados. O número de projetos que o pesquisador coordenou.

Projetos. A soma de "Participações em Projetos" com "Projetos Coordenados".

Orientações de Mestrado. O número de orientações de mestrado concluídas feitas pelo pesquisador.

Orientações de Doutorado. O número de orientações de doutorado concluídas feitas pelo pesquisador.

Orientações. A soma de "Orientações de Mestrado" com "Orientações de Doutorado".

Bancas de Mestrado. O número de bancas de mestrado que o pesquisador participou.

Bancas de Doutorado. O número de bancas de doutorado que o pesquisador participou.

Bancas. A soma de "Bancas de Mestrado" com "Bancas de Doutorado".

Publicações em Congressos. O número de artigos completos do pesquisador publicados em conferência.

Publicações em Periódicos. O número de artigos completos do pesquisador publicados em periódicos.

Publicações. A soma de "Publicações em Congressos" com "Publicações em Periódicos".

Publicações JCR. O número de artigos completos do pesquisador publicados em periódicos com JCR maior que 0.

Publicações JCR > 1,5. O número de artigos completos do pesquisador publicados em periódicos com JCR maior que 1,5.

Aceitações JCR > 1,5. O número de artigos aceitos do pesquisador em periódicos com JCR maior que 1,5.

Artigos JCR > 1,5. A soma de "Publicações JCR > 1,5" com "Aceitações JCR > 1,5".

Citações. O número de citações do pesquisador de acordo com a plataforma Google Scholar.

H-Index (total). O *H-index* do pesquisador de acordo com a plataforma Google Scholar

H-Index (anual). O "H-Index (total)" dividido por Idade Acadêmica.

Nome	ID Lattes	ID Scholar	Ano do Doutorado	Idade Acadêmica	Participações em Projetos (total)	Projetos Coordenados (total)	Projetos (total)	Orientações de Mestrado (total)
Célio Vinicius Neves de Albuquerque	4641684220602580	cB78oMQAAAAJ	2000	22	9	12	21	31
Leonardo Gresta Paulino Murta	1565296529736448	VEbJeB8AAAAJ	2006	16	16	10	26	24
Vanessa Braganholo Murta	0060120644445370	QT9-AZ4AAAAJ	2004	18	11	11	22	26

Figura 6 – Exemplo das informações escritas na saída do *script* write_profile.py

4.5 generate_datacapes.py

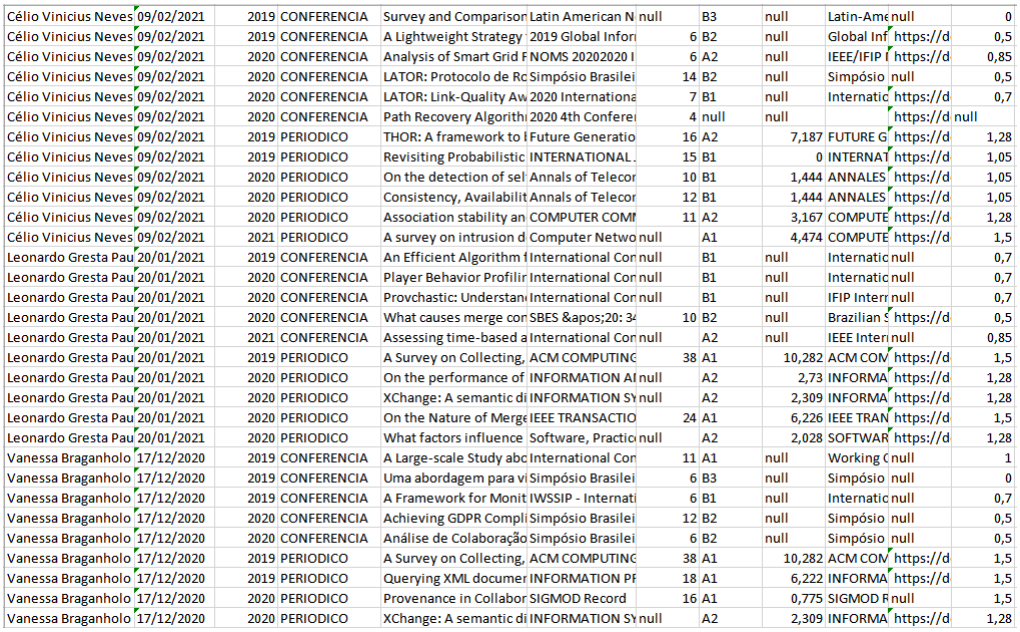
Com uma certa frequência, a CAPES pede para os programas de pós-graduação gerarem relatórios sobre sua produção apenas considerando os pesquisadores que estavam credenciados dentro de um período. O *script generate_datacapes.py* gera cinco desses relatórios que podem ser encontrados na pasta */output/datacapes* do programa. Esse diretório pode ser alterado modificando a variável *generate_datacapes_output_dir* do *script config.py*.

Esses relatórios podem ser divididos em dois grupos: relatórios que consideram a produção de cada pesquisador individualmente e relatórios que consideram a produção do programa como um todo. Então, após o *script* produzir os relatórios que levam em conta a produção de cada pesquisador, ele precisa remover os artigos duplicados (os que foram escritos em coautoria por pesquisadores credenciados no programa). Para isso, um dos valores que ele checa é a similaridade entre os títulos dos artigos. Logo, no *script config.py* há a variável *datacapes_minimum_similarity_titles* com o valor mínimo de comparação para dizer se títulos similares devem ser considerados o mesmo artigo ou não. Esse valor deve ser especificado entre 0 e 1.

4.5.1 Relatórios que Consideram Pesquisadores Individualmente

Dos relatórios gerados, há dois que consideram os pesquisadores individualmente. Seus arquivos são nomeados *producao_docentes.xlsx* e *sumario_docentes.xlsx*. O primeiro apresenta as informações qualitativas contendo as produções válidas para a CAPES de cada pesquisador naquele período, já o segundo apresenta as informações quantitativas referentes às mesmas produções. A seguir, detalhamos cada um desses relatórios.

producao_docentes.xlsx. Usando a Figura 7 como exemplo de um arquivo *producao_docentes.xlsx* gerado pelo *script*, a primeira coluna contém os nomes dos pesquisadores, a segunda coluna contém a data da última atualização do currículo Lattes daquele pesquisador, a terceira coluna contém o ano de publicação do artigo referente àquela linha do arquivo, a quarta coluna informa se aquele artigo foi publicado em um periódico ou conferência, a quinta coluna contém o título do artigo, a sexta coluna é o nome do *venue* onde ele foi publicado (conforme consta no Lattes do pesquisador), a sétima coluna sua quantidade de páginas, a oitava coluna seu nível Qualis, a nona coluna o valor do seu *Journal Citation Reports*, a décima coluna o *official_forum* onde ele foi publicado (obtido



Célio Vinicius Neves	09/02/2021	2019	CONFERENCIA	Survey and Comparison	Latin American N	null	B3	null	Latin-Ame	null	0
Célio Vinicius Neves	09/02/2021	2019	CONFERENCIA	A Lightweight Strategy	2019 Global Infor	6	B2	null	Global Inf	https://d	0,5
Célio Vinicius Neves	09/02/2021	2020	CONFERENCIA	Analysis of Smart Grid F	NOMS 20202020 I	6	A2	null	IEEE/IFIP I	https://d	0,85
Célio Vinicius Neves	09/02/2021	2020	CONFERENCIA	LATOR: Protocolo de Rc	Simpósio Brasilei	14	B2	null	Simpósio	null	0,5
Célio Vinicius Neves	09/02/2021	2020	CONFERENCIA	LATOR: Link-Quality Aw	2020 Internationa	7	B1	null	Internatio	https://d	0,7
Célio Vinicius Neves	09/02/2021	2020	CONFERENCIA	Path Recovery Algorith	2020 4th Confere	4	null	null		https://d	null
Célio Vinicius Neves	09/02/2021	2019	PERIODICO	THOR: A framework to I	Future Generatio	16	A2	7,187	FUTURE G	https://d	1,28
Célio Vinicius Neves	09/02/2021	2019	PERIODICO	Revisiting Probabilistic	INTERNATIONAL	15	B1	0	INTERNAT	https://d	1,05
Célio Vinicius Neves	09/02/2021	2020	PERIODICO	On the detection of sel	Annals of Telecor	10	B1	1,444	ANNALES	https://d	1,05
Célio Vinicius Neves	09/02/2021	2020	PERIODICO	Consistency, Availabilit	Annals of Telecor	12	B1	1,444	ANNALES	https://d	1,05
Célio Vinicius Neves	09/02/2021	2020	PERIODICO	Association stability an	COMPUTER COMI	11	A2	3,167	COMPUTE	https://d	1,28
Célio Vinicius Neves	09/02/2021	2021	PERIODICO	A survey on intrusion d	Computer Netwo	null	A1	4,474	COMPUTE	https://d	1,5
Leonardo Gresta Pau	20/01/2021	2019	CONFERENCIA	An Efficient Algorithm f	International Cor	null	B1	null	Internatio	null	0,7
Leonardo Gresta Pau	20/01/2021	2020	CONFERENCIA	Player Behavior Profilir	International Cor	null	B1	null	Internatio	null	0,7
Leonardo Gresta Pau	20/01/2021	2020	CONFERENCIA	Provchastic: Understan	International Cor	null	B1	null	IFIP Interr	null	0,7
Leonardo Gresta Pau	20/01/2021	2020	CONFERENCIA	What causes merge con	SBES '20: 34	10	B2	null	Brazilian S	https://d	0,5
Leonardo Gresta Pau	20/01/2021	2021	CONFERENCIA	Assessing time-based a	International Cor	null	A2	null	IEEE Inter	null	0,85
Leonardo Gresta Pau	20/01/2021	2019	PERIODICO	A Survey on Collecting,	ACM COMPUTING	38	A1	10,282	ACM COM	https://d	1,5
Leonardo Gresta Pau	20/01/2021	2020	PERIODICO	On the performance of	INFORMATION AI	null	A2	2,73	INFORMA	https://d	1,28
Leonardo Gresta Pau	20/01/2021	2020	PERIODICO	XChange: A semantic di	INFORMATION SY	null	A2	2,309	INFORMA	https://d	1,28
Leonardo Gresta Pau	20/01/2021	2020	PERIODICO	On the Nature of Merge	IEEE TRANSACTIO	24	A1	6,226	IEEE TRAN	https://d	1,5
Leonardo Gresta Pau	20/01/2021	2020	PERIODICO	What factors influence	Software, Practic	null	A2	2,028	SOFTWAR	https://d	1,28
Vanessa Braganholo	17/12/2020	2019	CONFERENCIA	A Large-scale Study abc	International Cor	11	A1	null	Working C	null	1
Vanessa Braganholo	17/12/2020	2019	CONFERENCIA	Uma abordagem para vi	Simpósio Brasilei	6	B3	null	Simpósio	null	0
Vanessa Braganholo	17/12/2020	2019	CONFERENCIA	A Framework for Monit	IWSSIP - Internati	6	B1	null	Internatio	null	0,7
Vanessa Braganholo	17/12/2020	2020	CONFERENCIA	Achieving GDPR Compli	Simpósio Brasilei	12	B2	null	Simpósio	null	0,5
Vanessa Braganholo	17/12/2020	2020	CONFERENCIA	Análise de Colaboração	Simpósio Brasilei	6	B2	null	Simpósio	null	0,5
Vanessa Braganholo	17/12/2020	2019	PERIODICO	A Survey on Collecting,	ACM COMPUTING	38	A1	10,282	ACM COM	https://d	1,5
Vanessa Braganholo	17/12/2020	2019	PERIODICO	Querying XML documer	INFORMATION PF	18	A1	6,222	INFORMA	https://d	1,5
Vanessa Braganholo	17/12/2020	2020	PERIODICO	Provenance in Collabor	SIGMOD Record	16	A1	0,775	SIGMOD R	null	1,5
Vanessa Braganholo	17/12/2020	2020	PERIODICO	XChange: A semantic di	INFORMATION SY	null	A2	2,309	INFORMA	https://d	1,28

Figura 7 – Exemplo de um arquivo producao_docentes.xlsx

usando o dicionário de sinônimos ou cálculo de similaridade), a décima primeira coluna o endereço web para o artigo e a décima segunda coluna o valor dos pontos daquele nível Qualis. Vale ressaltar que os pontos de nível de Qualis são configuráveis no *script config.py*. Destacando que apesar da CAPES ter uma configuração de pontos, os programs de pós-gradução podem ter configurações diferentes, ou podem ocorrer mudanças na configuração da CAPES.

sumario_docentes.xlsx. A Figura 8 mostra um exemplo do que é escrito em um arquivo *sumario_docentes.xlsx* gerado pelo *script*. Todavia, ela não mostra todas as informações. Em sua completude, elas são: “Entidade”, “Periodicos JCR”, “Periodico A1”, “Periodico B2”, “Periodico B1”, “Periodico B2”, “Periodico B3”, “Periodico B4”, “Periodico B5”, “Periodico C”, “Periodico NC”, “Conferencia A1”, “Conferencia A2”, “Conferencia B1”, “Conferencia B2”, “Conferencia B3”, “Conferencia B4”, “Conferencia B5”, “Conferencia C”, “Conferencia NC”, “I-Restrito Periodico”, “I-Restrito Conferencia”, “I-Restrito Total”, “I-Geral Periodico”, “I-Geral Conferencia”, “I-Geral Total”, “Saturacao (trava)” e “Pontos Credenciamento”. “Entidade” se refere ao nome dos pesquisadores, “Periodicos JCR” é a quantidade de periódicos com o valor de *Journal Citation Reports* maior que 1,5 daquele pesquisador, os cabeçalhos com “PEeriodico” ou “Conferencia” seguido de um nível de Qualis são a quantidade de artigos com aquele nível publicados em periódicos ou conferências respectivamente, o texto “I-” indica índice, ou seja, índice restrito de periódicos, índice restrito de conferências, índice restrito total, índice geral de periódico, índice geral de conferência e índice geral total.

ENTIDADE	PERIODICOS JCR	PERIODICO A1	PERIODICO A2	PERIODICO B1	PERIODICO B2	PERIODICO B3	PERIODICO B4	PERIODICO B5	PERIODICO C	PERIODICO NC	CONFERENCIA A1
Célio Vinicius Neves de Albuquerque	6	3	2	4	0	0	0	0	0	0	2
Leonardo Gresta Paulino Murta	7	4	3	1	0	0	0	0	0	0	1
Vanessa Braganholo Murta	3	3	1	0	0	0	0	0	0	0	1

Figura 8 – Exemplo das informações escritas no arquivo sumario_docentes.xlsx

2019	PERIODICO	Revisiting Probabilistic Sched	INTERNATIONAL JOURNAL (	15	B1	0	INTERNATIONAL JOURNAL OF WIRELESS INFORMATION NETWORKS
2019	CONFERENCIA	A Case Study of Association I	2019 IEEE Symposium on Co	6	A2	null	International Symposium on Computers and Communications (ISCC)
2019	CONFERENCIA	Towards a Blockchain-Based S	ICC 2019 2019 IEEE Internati	6	A1	null	IEEE International Conference on Communications (ICC)
2019	PERIODICO	A Survey on Collecting, Mana	ACM COMPUTING SURVEYS	38	A1	10	ACM COMPUTING SURVEYS
2019	PERIODICO	Querying XML documents usi	INFORMATION PROCESSING(	18	A1	6,2	INFORMATION PROCESSING & MANAGEMENT
2019	CONFERENCIA	A Cache Prefetch Policy base	Latin American Network Op	null	B3	null	Latin-American Network Operations and Management Symposium (LANOMS)
2020	PERIODICO	BiRD-A Novel Bi-Dimensional	IEEE Transactions on Cognit	13	A1	4,3	IEEE TRANSACTIONS ON COMMUNICATIONS (PRINT)
2020	CONFERENCIA	Provchastic: Understanding a	International Conference o	null	B1	null	IFIP International Conference on Entertainment Computing (ICEC)
2020	CONFERENCIA	LATOR: Protocolo de Roteam	Simpósio Brasileiro de Rede	14	B2	null	Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)
2020	PERIODICO	XChange: A semantic diff app	INFORMATION SYSTEMS	null	A2	2,3	INFORMATION SYSTEMS (OXFORD)
2020	CONFERENCIA	Player Behavior Profiling thro	International Conference o	null	B1	null	International Conference on Theory and Practice of Digital Libraries (TPDL)
2020	PERIODICO	Provenance in Collaborative S	SIGMOD Record	16	A1	0,8	SIGMOD RECORD
2020	PERIODICO	On the Nature of Merge Conf	IEEE TRANSACTIONS ON SO	24	A1	6,2	IEEE TRANSACTIONS ON SOFTWARE ENGINEERING
2020	CONFERENCIA	Analysis of Smart Grid Fault R	NOMS 20202020 IEEE/IFIP N	6	A2	null	IEEE/IFIP Network Operations and Management Symposium (NOMS)
2020	CONFERENCIA	What causes merge conflicts?	SBES '20: 34th Brazili	10	B2	null	Brazilian Symposium on Software Engineering (SBES)
2020	PERIODICO	On the detection of selfish m	Annals of Telecommunicati	10	B1	1,4	ANNALES DES TELECOMMUNICATIONS
2020	CONFERENCIA	Path Recovery Algorithm Usir	2020 4th Conference on Clo	4	null	null	
2020	PERIODICO	On the performance of hybri	INFORMATION AND SOFTW	null	A2	2,7	INFORMATION AND SOFTWARE TECHNOLOGY
2020	PERIODICO	Association stability and han	COMPUTER COMMUNICATI	11	A2	3,2	COMPUTER COMMUNICATIONS
2020	CONFERENCIA	Achieving GDPR Compliance	Simpósio Brasileiro de Banc	12	B2	null	Simpósio Brasileiro de Banco de Dados (SBBD)
2020	PERIODICO	What factors influence the lif	Software, Practice & Experi	null	A2	2	SOFTWARE, PRACTICE & EXPERIENCE (PRINT)
2020	PERIODICO	Dominoes: An Interactive Exp	IEEE TRANSACTIONS ON SO	1	A1	6,2	IEEE TRANSACTIONS ON SOFTWARE ENGINEERING
2020	CONFERENCIA	B-Move: A Transmission Sche	2020 IEEE 33rd International	6	B1	null	International Symposium on Computer-Based Medical Systems (CBMS)
2020	CONFERENCIA	LATOR: Link-Quality Aware ar	2020 International Confere	7	B1	null	International Conference on Systems, Signals and Image Processing (IWSSIP)
2020	CONFERENCIA	GRASP-based Feature Selecti	2020 4th Conference on Clo	8	null	null	
2020	CONFERENCIA	Análise de Colaboração em P	Simpósio Brasileiro de Banc	6	B2	null	Simpósio Brasileiro de Banco de Dados (SBBD)
2020	PERIODICO	An NDT Model for Block Desig	IEEE Wireless Communicati	5	B1	4,3	IEEE WIRELESS COMMUNICATIONS LETTERS
2020	CONFERENCIA	A Parallel Method for Anaton	2020 International Joint Cor	6	A1	null	IEEE International Joint Conference on Neural Networks (IJCNN)
2020	PERIODICO	Consistency, Availability and	Annals of Telecommunicati	12	B1	1,4	ANNALES DES TELECOMMUNICATIONS
2021	CONFERENCIA	Assessing time-based and rar	International Conference o	null	A2	null	IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)
2021	PERIODICO	A survey on intrusion detecti	Computer Networks (1999)	null	A1	4,5	COMPUTER NETWORKS (1999)

Figura 9 – Exemplo de um arquivo producao_anual.xlsx

4.5.2 Relatórios do Programa

Os outros relatórios gerados por esse *script* levam em conta o programa de pós-graduação como um todo, ou seja, não há artigos duplicados. Eles são *producao_anual.xlsx*, *producao_4n.xlsx* (esses dois são qualitativos, sendo o primeiro parecido com o *producao_docentes.xlsx*), e *sumario_anual.xlsx*, que é quantitativo e muito parecido com o *sumario_docentes.xlsx*.

producao_anual.xslx. Como pode ser observado na Figura 9, apesar de parecido com o arquivo *producao_docentes.xlsx* há diferenças entre eles. Dentre essas diferenças podemos citar: a primeira coluna indica o ano do artigo e ordena o arquivo, a segunda coluna diz se o artigo foi publicado em periódico ou conferência, a terceira coluna é o título do artigo, a quarta coluna onde ele foi publicado, a quinta coluna indica a sua quantidade de páginas, a sexta coluna o seu nível Qualis, a sétima coluna o valor do seu *Journal Citations Reports* e a oitava coluna o *official_forum* da onde ele foi publicado.

sumario_anual.xlsx. Se repararmos na Figura 10, que representa um exemplo do que é escrito em um arquivo *sumario_anual.xlsx*, não há grandes diferenças entre a formatação desse relatório e o *sumario_docentes.xslx*, apenas que "Entidade" não se refere aos pesquisadores, mas sim ao conjunto de pesquisadores credenciados ao programa de pós-graduação naquele ano. O restante das explicações continua o mesmo da Figura 8.

producao_4n.xlsx. Como se pode ver na Figura 11, esse relatório é qualitativo e sua primeira coluna diz se o artigo da linha foi publicado em conferência ou periódico. A segunda coluna contém o ISSN para periódicos ou acrônimo para conferências, a terceira coluna contém o título do artigo da linha, a quarta coluna contém os autores do artigo, com os nomes separados por um caractere ";", a quinta coluna contém o local onde o artigo foi publicado, a sexta coluna contém o *official_forum* desse local, a sétima coluna contém o ano de publicação do artigo, a oitava coluna contém seu número de páginas, a nona coluna contém o nível de Qualis e a décima coluna contém o valor do seu *Journal Citation Reports*. O objetivo desse arquivo é facilitar a seleção, pela coordenação do programa de pós-graduação, das 4N produções mais importantes do programa, onde N é o número de docentes credenciados no programa, conforme solicitado pela CAPES nas avaliações quadrienais.

4.6 generate_researcher_progression_report.py

Esse *script* gera relatórios com informações sobre os artigos publicados, orientações e participações em bancas dos pesquisadores do arquivo de pesquisadores e podem ser usados para auxiliar na elaboração do relatório necessário à progressão para o cargo de Professor Titular. Os artigos gerados pelo *script* se encontram na pasta */output/generate_researcher_progression_report*, mas esse diretório pode ser modificado na variável *generate_researcher_progression_report_output_dir* do *script config.py*. Ademais, os

ENTIDADE	PERIODICOS JCR	PERIODICO A1	PERIODICO A2	PERIODICO B1	PERIODICO B2	PERIODICO B3	PERIODICO B4	PERIODICO B5	PERIODICO C	PERIODICO NC	CONFERENCIA A1
2019	5	4	1	2	0	0	0	0	0	0	2
2020	8	4	4	3	0	0	0	0	0	0	1
2021	1	1	0	0	0	0	0	0	0	0	0

Figura 10 – Exemplo das informações escritas no arquivo sumario_anual.xlsx

PERIODICO	1068-9605	Revisiting	BALBI, HELGA D.;CARRANO,	INTERNAT	INTERNAT	2019	15	B1	0
CONFERENCIA	ISCC	A Case Stu	BALBI, HELGA;Passos, Diegc	2019 IEEE	Internatic	2019	6	A2	null
CONFERENCIA	ICC	Towards a	DE OLIVEIRA, MARCELA T.;N	ICC 2019 2	IEEE Inter	2019	6	A1	null
PERIODICO	0360-0300	A Survey (	João Felipe Nicolaci Pimen	ACM COM	ACM COM	2019	38	A1	10,282
CONFERENCIA	APPEEC	IEC 61850	MATTOS, DOUGLAS P. DE;M	2019 IEEE	PES AsiaPa	2019	6	null	null
CONFERENCIA	ICIN	Towards a	OLIVEIRA, MARCELA T.;CAR	2019 22nd	Conferen	2019	8	null	null
CONFERENCIA	ISCC	The Relati	NETO, PAULO CEZAR LACER	2019 IEEE	Internatic	2019	6	A2	null
CONFERENCIA		A Large-sc	João Felipe Nicolaci Pimen	Internatic	Working (	2019	11	A1	null
CONFERENCIA		A Counsel	Silvio Quincozes;Raul Cere	Latin Ame	Latin-Ame	2019	null	B3	null
PERIODICO	2332-7731	BiRD-A Ne	cledson souza;Passos, Dieg	IEEE Trans	IEEE TRAN	2020	13	A1	4,341
CONFERENCIA	ICEC	Provchast	Troy Costa Kohwalter;Leon	Internatic	IFIP Interr	2020	null	B1	null
CONFERENCIA		LATOR: Pr	Egberto Caballero;Vinícius (	Simpósio	Simpósio	2020	14	B2	null
PERIODICO	0306-4379	XChange:	OLIVEIRA, ALESSANDREIA;K	INFORMA	INFORMA	2020	null	A2	2,309
CONFERENCIA	FDG	Player Be	Sidney Araujo Melo;Troy Co	Internatic	Internatic	2020	null	B1	null
CONFERENCIA	SBBD	Achieving	Daniel Prett Campagna;Alti	Simpósio	Simpósio	2020	12	B2	null
PERIODICO	0038-0644	What fact	Daricélio Moreira Soares;M	Software,	SOFTWAR	2020	null	A2	2,028
PERIODICO	0098-5589	Dominoes	José Ricardo da Silva Junior	IEEE TRAN	IEEE TRAN	2020	1	A1	6,226
CONFERENCIA	CBMS	B-Move: A	FERREIRA, VINICIUS;MUCHA	2020 IEEE	Internatic	2020	6	B1	null
CONFERENCIA	IWSSIP	LATOR: Li	CABALLERO, EGBERTO;FERR	2020 Inter	Internatic	2020	7	B1	null
CONFERENCIA	CIoT	GRASP-ba	QUINCOZES, SILVIO E.;Passo	2020 4th	Conference	2020	8	null	null
CONFERENCIA	SBBD	Análise de	Maria Luiza Falci;Andrea Ma	Simpósio	Simpósio	2020	6	B2	null
PERIODICO	2162-2337	An NDT M	Diego Passos;DE SOUSA, CL	IEEE Wirel	IEEE WIRE	2020	5	B1	4,348
CONFERENCIA	IJCNN	A Parallel	LACERDA, PAULO;GONZALE	2020 Inter	IEEE Inter	2020	6	A1	null
PERIODICO	0003-4347	Consisten	Gabriel Carrara;Leonardo B	Annals of	ANNALES	2020	12	B1	1,444
CONFERENCIA	SANER	Assessing	Felipe Curty do Rego Pinto;	Internatic	IEEE Inter	2021	null	A2	null
PERIODICO	1389-1286	A survey (	QUINCOZES, SILVIO E.;Célic	Computer	COMPUTE	2021	null	A1	4,474

Figura 11 – Exemplo de um arquivo producao_4n.xlsx

```
Type the number of the researcher you wish to generate the .xlsx file for
1  -  Célio Vinicius Neves de Albuquerque
2  -  Leonardo Gresta Paulino Murta
3  -  Vanessa Braganholo Murta
4  - All of them
```

Figura 12 – Exemplo da interação com a linha de comando do *script* generate_researcher_progression_report.py

relatórios são nomeados com o nome do respectivo pesquisador.

Diferentemente de outros *scripts*, ele não necessariamente gera relatórios para todos os pesquisadores do arquivo de pesquisadores, pois utiliza a linha de comando para interagir com o usuário ou parâmetros para ter comportamentos diferentes. Caso o usuário execute o *script* sem nenhum parâmetro, o programa perguntará para qual pesquisador ele deve gerar o relatório, como pode ser visto na Figura 12.

Caso o usuário não deseje interagir com a linha de comando, o *script* aceita os seguintes parâmetros para mudar seu comportamento:

- ***–all*** ou ***-a***: gera relatórios para todos os pesquisadores do arquivo de pesquisadores. É o mesmo que selecionar a opção 4 da Figura 12.
- ***–researchers*** ou ***-r***: junto a esse parâmetro o usuário deve passar um ou mais identificadores separados por um espaço em branco. Esses identificadores devem ser o identificador do currículo Lattes de um pesquisador ou seu identificador no banco de dados. Para saber seu identificador no banco de dados basta usar a ordem do arquivo de pesquisadores (ou usar o comando a seguir). Então, o programa gera os relatórios apenas para esses pesquisadores.
- ***–ids*** ou ***-i***: mostra na linha de comando os identificadores que os pesquisadores no arquivo de pesquisadores possuem no banco de dados.

4.6.1 Relatórios

Como dito anteriormente, cada relatório terá o nome do pesquisador a que pertence e dentro de cada arquivo há quatro abas: “Publicações em periódico”, “Publicações em conferência”, “Orientações” e “Participações em banca”.

Como se pode ver na Figura 13, a aba “Publicações em periódico” possui os campos “Autores” com os nomes dos autores separados pelo caractere “;”, “Ano” para o ano do artigo, “Título” para o título dele, “Periódico” com o nome do periódico em que ele foi publicado, “ISSN”, o “Número de páginas”, o nível “Qualis” e “JCR” com o valor do seu *Journal Citations Reports*.

Na Figura 14 nota-se que a aba “Publicações em conferências” apresenta o mesmo comportamento de “Publicações em periódicos” para as colunas “Autores”, “Ano”, “Título”,

Autores	Ano	Título	Periódico	ISSN	Número de páginas	Qualis	JCR
João Felipe Nicolaci Pimentel;Juliana Freire;Leor	2019	A Survey on Collect	ACM COMPU	0360-0300	38	A1	10,282
Fábio Gomes dos Santos;Leonardo Machado;Rafa	2019	Querying XML docu	INFORMATIC	0306-4573	18	A1	6,222
Eduardo Jandre de Oliveira;Bruna Diirr;Vanessa E	2020	Provenance in Coll	SIGMOD Rec	0163-5808	16	A1	0,775
OLIVEIRA, ALESSANDREIA;KOHWALTER, TROY;KAI	2020	XChange: A semant	INFORMATIC	0306-4379	null	A2	2,309

Figura 13 – Exemplo da aba "Publicações em periódico" de um relatório gerado pelo *script* generate_researcher_progression_report.py

"Número de páginas" e "Qualis". A coluna "Conferência" contém o nome da conferência em que o artigo foi publicado e "Tipo" contém a natureza (*nature*) do artigo, que foi explicada na Seção 3.1.6 do Capítulo 3.

Pode-se observar pela Figura 15 que, na aba "Orientações", que são orientações concluídas, a coluna "Tipo" se refere ao tipo (*type*) da orientação, visto na Seção 3.1.17 do Capítulo 3, seguido do "Nome do aluno" e "Ano" com o ano que ela ocorreu.

Por fim, como exibido na Figura 16, a aba "Participações em banca" tem o "Tipo" (*type*) da banca, explicado na Seção 3.1.18 do Capítulo 3, "Ano" sendo o ano em que ela ocorreu, o "Nome da universidade" que é o local onde aconteceu e "Nome do aluno ou cargo" – sendo que o cargo é apenas no caso da banca ser de concurso público.

4.7 generate_configured_reports.py

Nem sempre os relatórios anteriores irão suprir as necessidades do usuário. Por isso, implementamos a possibilidade do usuário gerar relatórios configurados por ele mesmo. Nesses relatórios, o usuário pode decidir os nomes dos relatórios, se ele deseja reunir todos os relatórios em abas de um mesmo arquivo e quais informações cada relatório ou aba irá ter. Entretanto, diferente dos relatórios criados pelos outros *scripts*, o *script generate_configured_reports.py* só produz informações em grão fino, ou seja, ele apenas escreve diretamente atributos parecidos com os vistos na Seção 3.1 do Capítulo 3. Mas ainda há grande potencial nessa abordagem, pois usando ferramentas como Microsoft Excel, Google Spreadsheet, dentre outras, é possível extrair outras informações derivadas das originadas pelo *generate_configured_reports.py*.

Autores	Ano	Título	Conferência	Número de páginas	Qualis	Tipo
João Felipe Nicolaci Pimentel;Leona	2019	A Large-scale Study	International Confere	11	A1	artigo completo
Eduardo Jandre de Oliveira;Bruna Di	2019	Uma abordagem pa	Simpósio Brasileiro de	6	B3	artigo completo
Adriel dos Santos Araújo;Juan Lucas	2019	A Framework for M	IWSSIP - International	6	B1	artigo completo
Daniel Prett Campagna;Altigran Silv	2020	Achieving GDPR Co	Simpósio Brasileiro de	12	B2	artigo completo
Maria Luiza Falci;Andrea Magalhães;	2020	Análise de Colabor	Simpósio Brasileiro de	6	B2	artigo completo

Figura 14 – Exemplo da aba "Publicações em conferência" de um relatório gerado pelo *script* generate_researcher_progression_report.py

Tipo	Nome do aluno	Ano
mestrado	Victor Bezerra Alencar	2020
graduação	Marcelo Vieitas Da Fonseca Filho	2019
graduação	Thiago Augusto Sivirino	2019

Figura 15 – Exemplo da aba "Orientações" de um relatório gerado pelo *script* generate_researcher_progression_report.py

Para configurar relatórios o usuário deve ir no *script config.py* e modificar a variável *configured_reports*. Ela deve ser um dicionário com strings representando o nome do relatório/aba como chave e seu valor deve ser uma lista com as informações que o usuário deseja naquele relatório. Essas informações devem ser atributos das classes a seguir.

- ***Artigo***: *titulo_artigo, tipo_artigo, ano, doi, quantidade_paginas, autores, nome, qualis, qualis_pontos, forum_oficial*
- ***Periodico***: *titulo_artigo, tipo_artigo, ano, doi, artigo_aceito, quantidade_paginas, autores, nome, forum_oficial, qualis, qualis_pontos, issn, jcr*
- ***Conferencia***: *titulo_artigo, tipo_artigo, ano, doi, quantidade_paginas, autores, nome, qualis, qualis_pontos, forum_oficial, acronimo*
- ***Banca***: *tipo, ano, universidade, aluno_ou_cargo, titulo, membros*
- ***Livro***: *titulo, editora, ano, autores*
- ***Capitulo***: *titulo, titulo_livro, editora, ano, autores*
- ***Corpo_Editorial***: *nome, tipo, inicio, fim*
- ***Organizacao_Evento***: *nome, ano, membros*
- ***Orientacao***: *tipo, estudante, ano, universidade, titulo*
- ***Patente***: *nome, tipo, autores, local, numero, ano*
- ***Projeto***: *nome, responsavel, coordenador, equipe, inicio, fim*
- ***Pesquisador***: *nome, ultima_atualizacao_lattes, doutorado_universidade, doutorado_ano, id_google_scholar, id_lattes*

Tipo	Ano	Nome da universidade	Nome do aluno ou cargo
qualificação de doutorado	2020	Laboratório Nacional de Computação Científica	Maria Luiza Botelho Mondelli
mestrado	2019	Universidade Federal Fluminense	Daniel Pinheiro da Silva Júnior
doutorado	2019	Universidade Federal de Santa Catarina	Luiz Henrique Zambom Santana
doutorado	2019	Universidade Federal do Rio de Janeiro	Renan Francisco Santos Souza

Figura 16 – Exemplo da aba "Participações em banca" de um relatório gerado pelo *script* generate_researcher_progression_report.py

```
configured_reports = {
    "Relatorio1": [
        Pesquisador.nome,
        Periodico.titulo_artigo,
        Periodico.nome,
        Periodico.ano

    ],
    "Relatorio2": [
        Corpo_Editorial.tipo,
        Corpo_Editorial.nome
    ]
}
```

Figura 17 – Exemplo de variável configured_reports

Um exemplo pode ser visto na Figura 17. Na figura, há dois relatórios configurados (Relatório1 e Relatório2), cada um com um conjunto diferente de atributos. Cabe notar que na configuração dos relatórios, é necessário informar o nome da classe e o nome do atributo, separados por um ponto ".".

Chamamos a atenção para duas classes, *Artigo* e *Pesquisador*. Artigo se comporta como *Periodico* ou *Conferencia* possuindo menos atributos, mas sua vantagem está no fato dela agrupar informações comuns a publicações em periódicos e publicações em conferências. Já *Pesquisador*, diferentemente das outras classes, consegue coexistir em um relatório junto a outra classe. Ou seja, é possível requisitar informações de *Pesquisador* em um relatório que há qualquer outra classe (e.g. *Pesquisador* e *Patente*, ou *Pesquisador* e *Banca*, ou *Pesquisador* e *Capitulo*, ou *Pesquisador* e *Projeto*). Desse modo, será gerado um produto cartesiano de cada linha com as informações de *Pesquisador* pedidas pelo usuário, como pode ser visto na Figura 18. Nessa figura, é possível observar que Vanessa e Leonardo são coautores de uma publicação no ACM Computing Surveys, então essa publicação aparece duas vezes, com os dados de cada um dos pesquisadores que são coautores.

Os relatórios são gerados na pasta */output/configured_reports* do programa, mas se o usuário desejar, pode modificar o caminho de saída alterando a variável *generate_reports_output_dir* no *script config.py*. Além disso, caso ele deseje que os relatórios sejam escritos como abas em um mesmo arquivo, basta colocar o valor da variável *reports_as_new_worksheets*, também em *config.py*, como True.

Vale destacar que não permitimos mais de uma classe no mesmo relatório, a menos desse caso onde uma das classes é *Pesquisador*, pois isso geraria um produto cartesiano sem sentido. Por exemplo, suponha que um pesquisador participou das bancas B1, B2 e B3 e publicou os artigos A1 e A2. Isso faria surgir seis linhas no relatório, listando

Pesquisador.nome	Periodico.artigo	Pesquisador.doutorado_ano	Periodico.nome	Periodico.ano
Célio Vinicius Neves de Albuquerque	THOR: A framework to build an advanced metering infrastructure resilient to DAP failures in smart grids	2000	Future Generation Computer Systems	2019
Célio Vinicius Neves de Albuquerque	On the detection of selfish mining and stalker attacks in blockchain networks	2000	Annals of Telecommunications	2020
Leonardo Gresta Paulino Murta	A Survey on Collecting, Managing, and Analyzing Provenance from Scripts	2006	ACM COMPUTING SURVEYS	2019
Leonardo Gresta Paulino Murta	Dominoes: An Interactive Exploratory Data Analysis tool for Software Relationships	2006	IEEE TRANSACTIONS ON SOFTWARE ENGINEERING	2020
Vanessa Braganholo Murta	A Survey on Collecting, Managing, and Analyzing Provenance from Scripts	2004	ACM COMPUTING SURVEYS	2019
Vanessa Braganholo Murta	Querying XML documents using Prolog engines: When is this a good idea?	2004	INFORMATION PROCESSING & MANAGEMENT	2019

```
configured_reports = {
    "Relatorio1": [
        Pesquisador.nome,
        Periodico.titulo_artigo,
        Pesquisador.doutorado_ano,
        Periodico.nome,
        Periodico.ano

    ]
}
```

Figura 18 – Exemplo de um relatório gerado pelo *script* generate_configured_reports.py onde há *Pesquisador* e mais uma classe

os dados das possíveis combinações (B1, A1; B1, A2; B2, A1; B2, A2; B3, A1; B3, A2). Contudo, essas linhas gerariam somente redundância, sem nenhum ganho interpretativo. Desta forma, optamos por permitir que sejam definidos vários relatórios independentes e que, caso seja do desejo do usuário, esses relatórios sejam criados como abas de uma mesma planilha. Voltando ao exemplo, para um dado pesquisador, teríamos uma aba de bancas, com três linhas contendo os dados de B1, B2 e B3, e outra aba de artigos, com duas linhas contendo os dados de A1 e A2.

Como visto, apenas a classe *Pesquisador* pode existir em um mesmo relatório com outras classes. Caso haja quaisquer outras classes dentro de uma lista da variável *configured_reports* o programa irá exibir uma mensagem na linha de comando dizendo que houve conflito, e não irá produzir aquele relatório. Entretanto, caso o usuário vá no *script config.py* e coloque o valor da variável *new_worksheet_if_conflict* como True, quando houver conflito, em vez do programa exibir a mensagem e não gerar o relatório, ele irá gerar uma aba no relatório com as informações pedidas para cada classe que houve conflito. Por exemplo, caso uma lista seja *[Projeto.responsavel, Patente.nome, Patente.tipo, Livro.ano]*, o *script* irá produzir um relatório com três abas, uma para *Projeto.responsavel*, uma para *Patente.nome* e *Patente.tipo* e outra para *Livro.ano*.

4.8 visualize.py

Após executar o *script write_profile.py* pelo menos uma vez, é possível comparar um pesquisador, que não necessariamente está no arquivo de pesquisadores, com as informações obtidas. Para isso o programa usa *boxplots* e os separa pelas métricas anual, total e dentro

do horizonte de comparação que são explicadas na Seção 4.4 deste capítulo.

Para designar o pesquisador que terá seu currículo comparado ao resultado da saída do *script write_profile.py*, o usuário deve modificar a variável *subject* que pode ser encontrada no *script config.py*. Ela é um dicionário cujas chaves são Nome, ID Lattes e ID Scholar e os valores dessas chaves precisam ser respectivamente o nome do pesquisador desejado, o identificador do seu currículo Lattes e o seu identificador na plataforma Google Scholar. Vale ressaltar que tanto as chaves quanto os valores necessitam ser informados como strings. Um exemplo da comparação pode ser visto na Figura 19 onde o ponto vermelho representa o pesquisador em *subject*. O usuário pode ligar ou desligar essa funcionalidade ao atribuir à variável *print_subject* em *config.py* o valor True ou False, respectivamente.

A saída do *script visualize.py* é composta por imagens em formato .png que são geradas na pasta */output/build* do programa. Esse diretório de saída pode ser modificado alterando a variável *build_dir* no *script config.py*. Exemplos da saída podem ser observados nas Figuras 20 para anual, que são as suas respectivas informações divididas pelo valor de idade acadêmica do pesquisador, 21 para toda a carreira do pesquisador, 22 para o horizonte de coleta e 23 comparando a idade acadêmica e o H-Index.

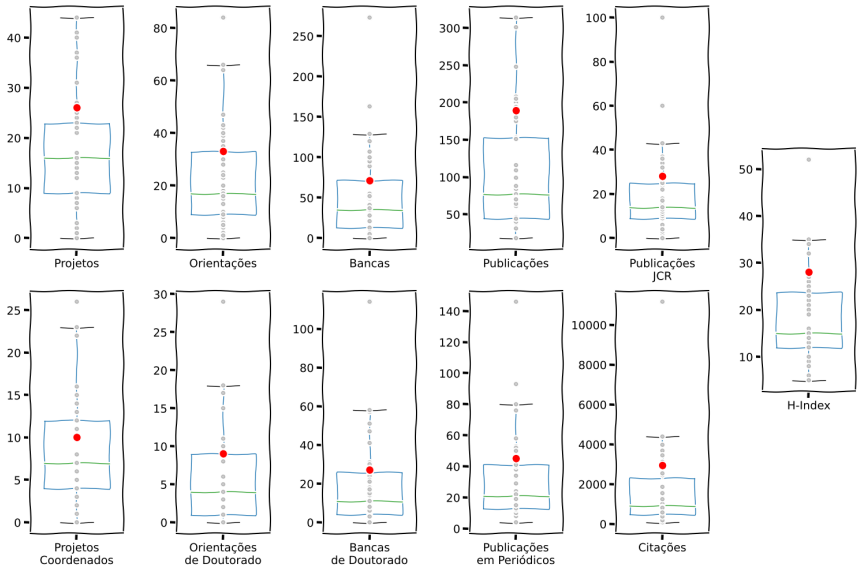


Figura 19 – Exemplo de *boxplot* para todo o currículo gerado pelo *script* visualize.py com comparação ao pesquisador em *subject*

4.8.1 visualize-jcr1.5.py

Também é possível gerar um *boxplot* no quesito de publicações com *Journal Citations Reports* maior que 1,5. O que o *script visualize.py* não faz. Destacando, ou não, o ponto de um determinado pesquisador. Para isso, basta que o usuário execute o *script visualize-jcr1.5.py* que irá produzir uma imagem .png como a Figura 24.

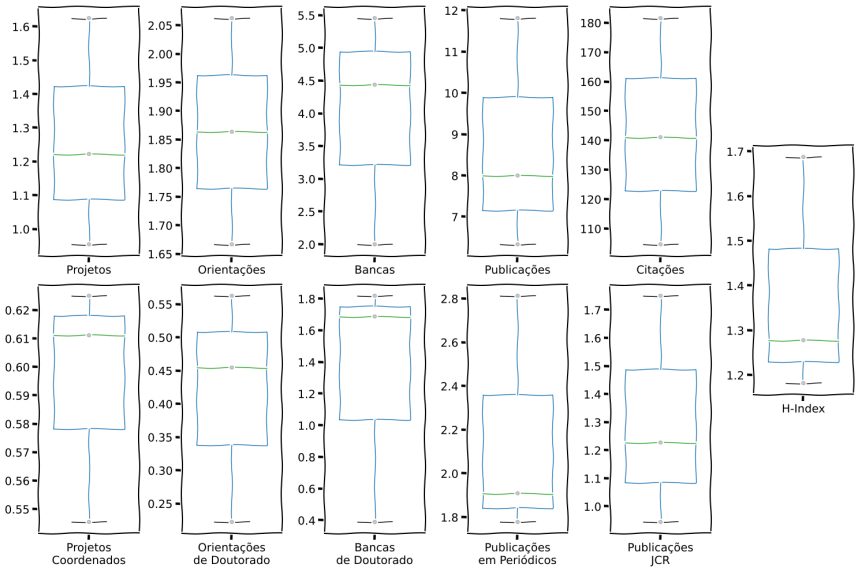


Figura 20 – Exemplo de *boxplot* anual gerado pelo *script* visualize.py

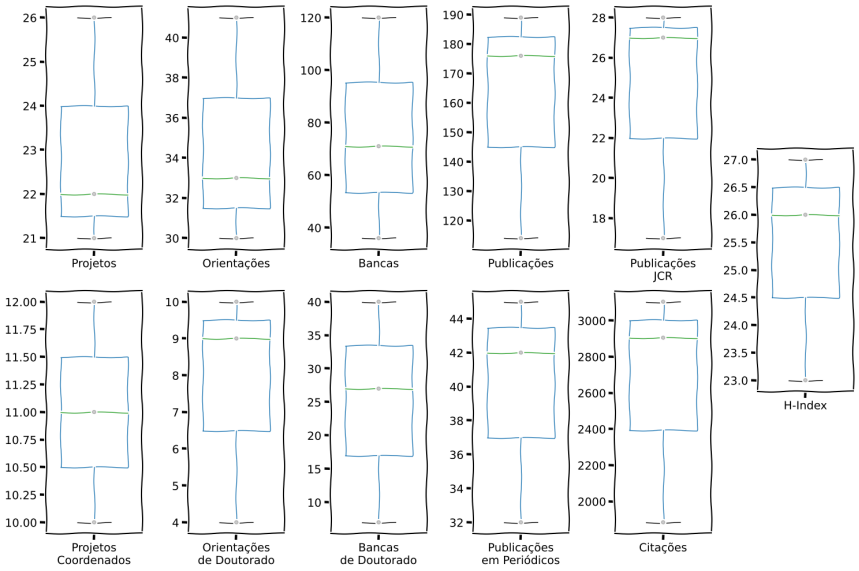


Figura 21 – Exemplo de *boxplot* para todo o currículo gerado pelo *script* visualize.py

4.9 generate_collaboration_graphs.py

O *script generate_collaboration_graphs.py* permite que o usuário visualize quais pesquisadores, do arquivo de pesquisadores, publicaram artigos juntos no decorrer dos anos. Além disso, ele monstra a intensidade dessa colaboração. Para tal, escolhemos uma representação usando grafos, tendo os pesquisadores como nós, as colaborações (coautoria) como arestas ligando esses nós e a intensidade da colaboração representada pela espessura das arestas. A espessura das arestas é acrescentada uma unidade para cada artigo de periódico ou conferência em que os pesquisadores escreveram juntos.

Desenvolvemos a visualização dos grafos usando a ferramenta Flask como back-end

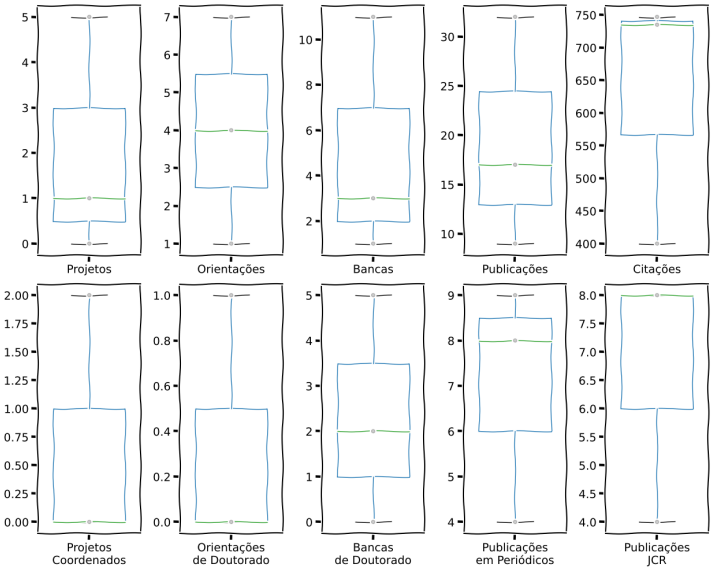


Figura 22 – Exemplo de *boxplot* para o horizonte de coleta gerado pelo *script* visualize.py

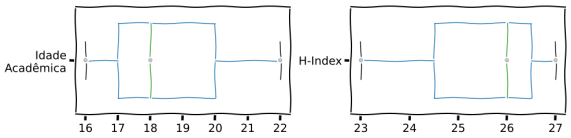


Figura 23 – Exemplo de *boxplot* para a idade acadêmiaca e H-index gerado pelo *script* visualize.py

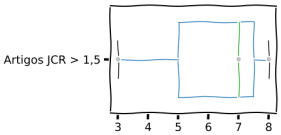


Figura 24 – Exemplo de *boxplot* para *Journal Citation Reports* maiores que 1,5 gerado pelo *script* visualize-jcr1.5.py

e o front-end utilizando D3.js. O código disponível em <https://bl.ocks.org/heybignick/3faf257bbbbc7743bb72310d03b86ee8> foi usado como base da implementação, assim retirando a responsabilidade de definir previamente o lugar de cada nó, pois, dado que é um grafo de força, todos os nós originam no centro e a força os impulsiona para suas posições. Para impedir que os nós se distanciem muito do centro do grafo, criamos as variáveis *collaboration_graphs_alpha* e *collaboration_graphs_alpha_decay* no *script* *config.py* para o usuário alterar conforme sua necessidade. A primeira é o valor da força inicial e a segunda a taxa em que a força inicial deve decair até chegar a zero, ou seja, os nós ficarem imóveis. Vale ressaltar que o valor das duas variáveis deve ser especificado entre 0 e 1.

Como se pode observar na Figura 25, os botões radio no topo são referentes aos anos do horizonte de coleta, vistos na Seção 4.1 deste capítulo, sendo o botão *Tudo* para

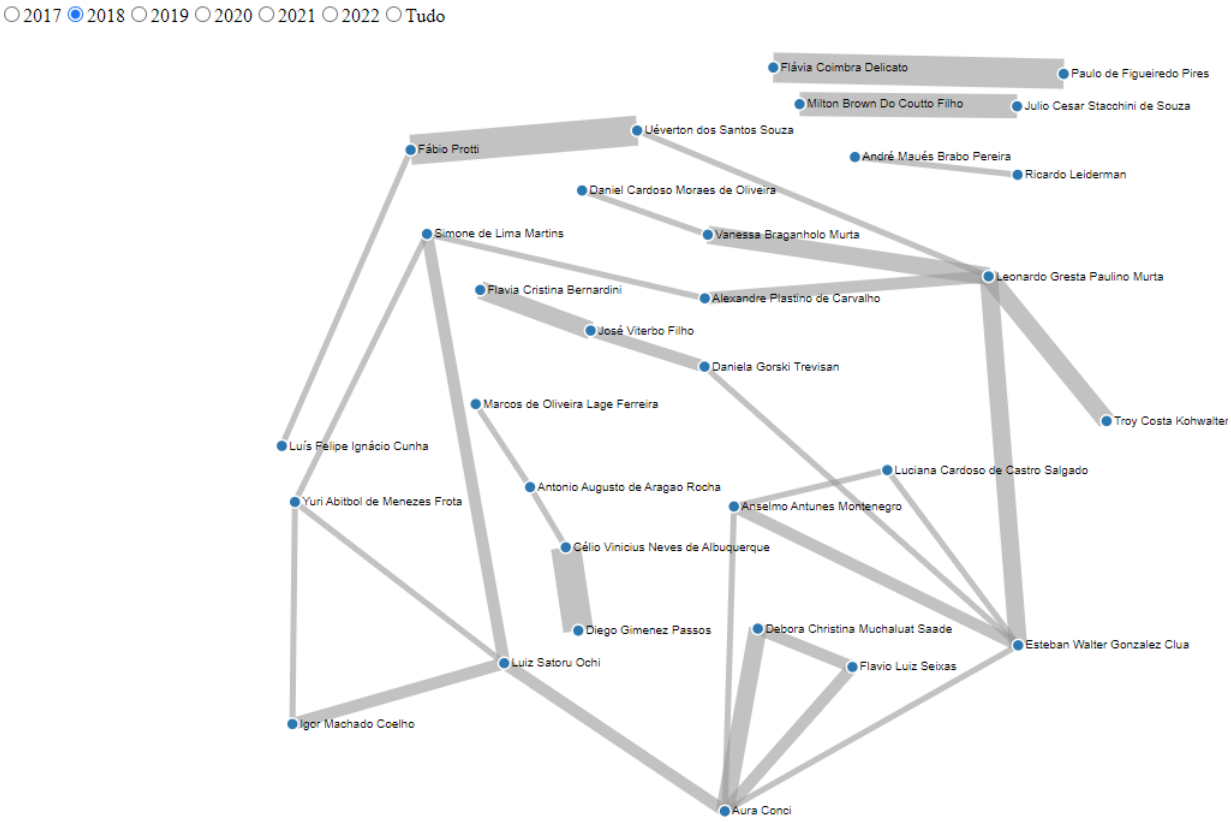


Figura 25 – Exemplo de uma visualização gerada pelo *script* generate_collaboration_graphs.py com o horizonde de coleta de 2017 até 2022

todo o horizonte. Ao clicar em um deles, o grafo de colaboração correspondente é exibido na tela no lugar do atual. Os nós estão em cor azul e o nome do pesquisador representado pelo nó é escrito ao seu lado. As arestas são cinzas e, como dito anteriormente, quanto mais grossa, maior a quantidade de artigos de coautoria entre os pesquisadores. Ademais, o usuário pode arrastar os nós para onde ele quiser dentro da visualização.

4.10 Considerações Finais

Neste capítulo foram apresentados os *scripts* do programa Perfil 2.0 disponíveis para o usuário. Dentre eles há *scripts* para alterar o comportamento do programa, *scripts* para gerar relatórios e *scripts* para gerar visualizações. Os scripts *config.py*, *populate_database.py*, *write_profile.py* e *generate_datacapes.py* integram os programas antigos nos quais o programa Perfil foi baseado. Já os scripts *generate_researcher_progression_report.py*, *generate_configured_reports.py* e *generate_collaboration_graphs.py* implementam novas funcionalidades. Os *scripts visualize.py* e *visualize-jcr1.5.py*, apenas foram alterados o suficiente para funcionarem com essa versão do programa. Por fim, o *script download.py* não foi modificado.

Desta forma, as contribuições deste trabalho são: (1) integração de dois programas independentes, escritos em linguagens de programação diferentes (Java e Python); (2) criação de uma representação orientada a objetos mapeada automaticamente para banco de dados relacional, que permite execução de consultas SQL sobre os dados dos currículos Lattes coletados; (3) criação de mecanismos para evitar duplicação, no banco de dados, de informações vindas de currículos Lattes; (4) ajustes nos *scripts* preexistentes para que fizessem uso da nova representação orientada a objetos; (5) criação de um mecanismo para gerar relatórios configurados de forma declarativa; (6) criação de um mecanismo para gerar relatórios que podem auxiliar na progressão de um pesquisador; e (7) criação de um mecanismo para visualizar grafos de coautorias de artigos entre pesquisadores.

5 Avaliação

O objetivo inicial do programa era unir sua antiga funcionalidade de escrever os perfis quantitativos dos pesquisadores, ou seja, a funcionalidade do *script write_profile.py*, que é visto na Seção 4.4 do Capítulo 4, junto à capacidade de armazenar e relacionar as informações dos currículos Lattes, vista no Capítulo 3, e a funcionalidade de gerar relatórios de produção de um programa de pós-graduação requisitados pela CAPES. Ao decorrer do desenvolvimento, outras funcionalidades foram adicionadas, como pode ser visto no Capítulo 4, mas ainda era necessário testar a corretude delas.

Para isso, primeiro foram criadas as massas de teste. Elas constituem de três arquivos de pesquisadores: (i) *teste.xlsx*, contendo 3 pesquisadores; (ii) *pgc.xlsx* que possui 48 pesquisadores, que são todos os pesquisadores cadastrados no programa de pós-graduação do Instituto de Computação da Universidade Federal Fluminense, isso é, no momento dos testes; e (iii) *collaboration-graphs.xlsx*, tendo 23 pesquisadores; e os lattes de todos esses pesquisadores. Essas massas de testes foram usadas da seguinte forma: primeiro se testava com *teste.xlsx*, que é a menor. Caso não houvesse erros, se testava com *pgc.xlsx*. Já a base *collaboration-graphs.xslx* foi utilizada para apresentações do *script generate_collaboration_graphs.py*, visto na Seção 4.9 do Capítulo 4, pois seus pesquisadores colaboraram entre si nos anos anteriores a esse trabalho.

Os *scripts generate_collaboration_graphs.py*, *generate_configured_reports.py* e *generate_reseacher_progression_report.py* foram apenas testados com as massas de dados em busca de erros de execução e depois apresentados aos clientes até serem aprovados. Evidenciando que os clientes eram professores envolvidos com a coordenação da pós-graduação do Instituto de Computação da Universidade Federal Fluminense ou estavam para aplicar para promoção de professor titular. Já as funcionalidades a seguir careceram um empenho maior para serem julgadas corretas.

5.1 Dados

Para o testar a parte do armazenamento de dados, foram utilizados 13 testes automatizados que verificam a leitura, extração e armazenamento das informações do currículo Lattes. Vale ressaltar que esses testes são executados como uma forma de integração contínua. No entanto, também testamos a duplicação de informações nos currículos Lattes, vista na Seção 3.2.1. Para tal, após verificarmos que não havia problemas ao usar o arquivo de pesquisadores *teste.xlsx*, populamos o banco de dados com o arquivo *pgc.xlsx* e enviamos os avisos que o programa escreveu no arquivo de log para os pesquisadores listados no arquivo, de forma que eles pudessem avaliar os resultados de duplicação detectados. Foram

encontrados 26 casos de duplicação, deles, 65,38% estavam corretos e 88,88% dos falsos positivos se tratavam de alguns casos de projetos finais, orientações e bancas, dos cursos de engenharia da Universidade Federal Fluminense, que, de acordo com o pesquisador que o currículo Lattes pertence, os alunos fazem dois projetos finais. Com o retorno dos pesquisadores, fizemos as modificações necessárias para que essa funcionalidade gerasse um menor número de falsos positivos. Entretanto, para a funcionalidade de unificação, só fizemos o teste com o arquivo *teste.xlsx*, dado o grande número de avisos no log gerados por causa da quantidade de pesquisadores do arquivo *pgc.xlsx*.

5.2 Perfil 1.0

Os testes feitos com o *script write_profile.py* se resumiram em executar esse *script* e o código antigo, com as mesmas configurações, verificando os resultados obtidos com as massas de testes disponíveis, pois sua saída é completamente quantitativa. Então, nós apenas consideramos correto quando ambos os resultados foram completamente iguais, desconsiderando erros de arredondamento em números fracionários. Vale notar que as variáveis de unificação, vistas em 3.2.1, foram colocadas como False, pois não existia funcionalidade parecida com elas no código antigo.

5.3 Datacapes

O *script generate_datacapes.py* é o responsável por gerar os relatórios de produção de um programa de pós-graduação. Antes deste trabalho, já existia um programa escrito na linguagem Java que possuía essa funcionalidade. Então, para garantir que o comportamento do *script* gerado por esse trabalho é correto, primeiro desligamos as variáveis de unificação, dado que o programa Java também não possuía essa funcionalidade. Em seguida, executamos os dois programas para os mesmos pesquisadores com as mesmas configurações.

Quando testamos usando a massa de dados do arquivo de pesquisadores *teste.xlsx*, verificamos todos os relatórios produzidos na saída, visto que o número de pesquisadores é pequeno. Só fomos para a próxima etapa quando todos os resultados foram iguais, desconsiderando erros de arredondamento em números fracionários.

A segunda etapa consistiu em executar ambos os programas usando os pesquisadores do arquivo *pgc.xlsx*. No entanto, devido ao grande número de pesquisadores desse arquivo, decidimos analisar primeiramente apenas o relatório *sumario_anual*, visto na Seção 4.5. Como houve discrepância entre os resultados, foi necessário investigarmos a fundo os relatórios *producao_anual* e *producao_docentes*. Desse modo, além de encontrarmos erros no nosso *script*, também descobrimos que a ordem de análise dos currículos Lattes dos

pesquisadores era diferente nos programas, e isso levava a resultados diferentes, pois o programa em Java podia escrever resultados errados de acordo com a ordenação do seus pesquisadores. Na verdade, na hora de descartar os artigos iguais, vindos de currículos Lattes diferentes, para pode gerar a produção anual do programa, apesar de considerar vários fatores, ele não considerava o valor do atributo JCR. Ou seja, caso um pesquisador tivesse preenchido um artigo com um ISSN que não consta no arquivo com os valores de JCR, visto na Seção 3.3.3 do Capítulo 3, por exemplo, no caso de periódicos com mais de um ISSN, o valor de JCR do artigo seria 0. E caso outro pesquisador possuísse o mesmo artigo no seu currículo, mas com o valor de ISSN que está no arquivo, o valor do JCR desse artigo estaria correto. Dependendo da sequência em que o programa em Java analisa essas informações, ele podia descartar o artigo com o valor de JCR correto e manter o com JCR errado. Então, além das correções para obtermos o mesmo resultado que o programa Java, corrigimos esse erro ao também considerar o valor de JCR para descartarmos um artigo quando o programa vai gerar a produção do programa de pós, fazendo com que a ordem de análise dos currículos dos pesquisadores do programa não afete sua exatidão. Dessa forma, diferente do programa desenvolvido em Java, o programa Perfil 2.0, ao final das comparações, caso dois artigos publicados em periódico sejam iguais em todos os outros quesitos, descarta aquele com menor JCR.

Após todos os testes, observamos que o programa Perfil 2.0 apresentou o comportamento esperado, atendendo ao que foi planejado na sua idealização. Portanto, julgamos que as funcionalidades descritas neste trabalho estão prontas para uso, mas que ainda há aspectos nos quais podemos melhorá-lo, como descrito no próximo capítulo.

6 Conclusão

Este trabalho apresentou o programa Perfil 2.0, que coleta dados de pesquisadores a partir do currículo Lattes, e, quando necessário, do Google Scholar. Com esses dados, são gerados relatórios e visualizações com o intuito de auxiliar tanto pesquisadores quanto programas de pós-graduação. Realizar essas tarefas manualmente pode ser exaustivo, pois as plataformas não possuem todos os dados necessários para análises mais profundas, possibilitam erros de preenchimento e não possuem uniformidade nos dados.

Nesta monografia apresentamos a modelagem do Programa Perfil 2.0, seu diagrama de classes e seu banco de dados, como os dados são obtidos e inseridos no banco de dados e os métodos para tentar garantir unicidade e a não duplicação dos dados. Também foram explicados os *scripts* disponíveis para o usuário, como utilizá-los e suas saídas. Por fim, também foram descritos os procedimentos que fizemos para garantir que o programa Perfil 2.0 funciona como pretendido.

Além do que foi visto na literatura, nosso programa é a capaz de obter outros dados a partir dos dados coletados do currículo Lattes. Por exemplo, ele consegue obter o nível Qualis de uma conferência dado o nome dela, e permite o usuário gerar relatórios configurados por ele mesmo. Entretanto, ele não possui interface gráfica, necessita de gerenciamento dos seus recursos, vistos na Seção 3.3.3, e poderia usar algoritmos mais elaborados para identificar coautorias. Ademais, os relatórios que não são os configurados pelo usuário precisam de alterações diretas no código caso haja alguma mudança em seus requisitos.

Para o futuro, gostaríamos de trabalhar em alguns pontos fortes do programa Perfil 2.0, como adicionar mais dados que estão nos currículos Lattes, mas que atualmente não são extraídos, de forma a aumentar as possibilidades do usuário ao fazer um relatório configurado. Ainda sobre os relatórios configurados, poderíamos separar artigos aceitos de publicados na hierarquia de herança do modelo de orientação a objeto, ampliando assim as opções de escolha. Também planejamos cobrir alguns pontos fracos, como fornecer uma interface gráfica, deixar o usuário escolher quais gráficos de *boxplot* gerar, fornecer uma possível persistência de dados e transformá-lo em uma aplicação *web*. Outro ponto a melhorar é a identificação de coautoria dos artigos, pois os nomes dos coautores podem estar diferentes de como está escrito em seus currículos Lattes. Então, poderíamos ter também um arquivo de sinônimos para remediar essa falta de padronização e também usar a unificação de artigos para os grafos de colaboração. Outra possibilidade é gerar automaticamente as variações mais comuns dos nomes dos usuários para povoar o arquivo de sinônimos. Ainda sobre os arquivos da pasta */resources*, assim como temos um arquivo

para os valores de JCR de periódicos, poderíamos ter um arquivo com o valor H5 das conferências. Por fim, uma nova versão do programa Perfil poderia implementar a opção de exportar os grafos de colaboração como imagem.

Referências

AMORIN, C. V. Organização do currículo: plataforma lattes. *Pesquisa Odontológica Brasileira*, v. 17, n. suppl. 1, p. 18–22, 2003. Citado na página 4.

BERGROTH, L.; HAKONEN, H.; RAITA, T. A survey of longest common subsequence algorithms. In: IEEE. *Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000.* [S.l.], 2000. p. 39–48. Citado 2 vezes nas páginas 7 e 24.

BRANCO, A. M. et al. Ferramenta para coleta e comparação de dados de publicações acadêmicas dos professores com o currículo lattes. Florianópolis, SC., 2018. Citado 2 vezes nas páginas 1 e 7.

COLONETTI, G. B. et al. Agrupando pesquisadores por coautoria de publicações segundo currículo lattes. Florianópolis, SC, 2016. Citado 2 vezes nas páginas 1 e 8.

CONSORTIUM, W. W. W. *XML Path Language (XPath) 3.1*. 2017. <https://www.w3.org/TR/2017/REC-xpath-31-20170321/>. Citado na página 21.

DIGIAMPIETRI, L. A. et al. Minerando e caracterizando dados de currículos lattes. In: SBC. *Anais do I Brazilian Workshop on Social Network Analysis and Mining*. [S.l.], 2012. Citado 3 vezes nas páginas 1, 6 e 7.

HIRSCH, J. E. An index to quantify an individual's scientific research output. *Proceedings of the National academy of Sciences*, National Acad Sciences, v. 102, n. 46, p. 16569–16572, 2005. Citado na página 5.

LEVENSHTEIN, V. I. et al. Binary codes capable of correcting deletions, insertions, and reversals. In: SOVIET UNION. *Soviet physics doklady*. [S.l.], 1966. v. 10, n. 8, p. 707–710. Citado na página 7.

MENA-CHALCO, J. P.; DIGIAMPIETRI, L. A.; CESAR-JR, R. M. Caracterizando as redes de coautoria de currículos lattes. In: SBC. *Anais do I Brazilian Workshop on Social Network Analysis and Mining*. [S.l.], 2012. Citado 2 vezes nas páginas 1 e 7.

MENA-CHALCO, J. P.; JUNIOR, R. M. C. Scriptlattes: an open-source knowledge extraction system from the lattes platform. *Journal of the Brazilian Computer Society*, Springer, v. 15, n. 4, p. 31–39, 2009. Citado 2 vezes nas páginas 1 e 5.

RUBIM, I. C.; BRAGANHOLO, V. Detecting referential inconsistencies in electronic cv datasets. *Journal of the Brazilian Computer Society*, Springer, v. 23, n. 1, p. 1–11, 2017. Citado 2 vezes nas páginas 1 e 7.

SEGALIN, V. d. S. Ipu: sistema web de mapeamento dos currículos de professores e pesquisadores da ufsc. 2014. Citado 2 vezes nas páginas 1 e 6.

VALVERDE, B. d. S. et al. Ceifador-uma abordagem para geração de relatórios de apoio à pós-graduação. Florianópolis, SC., 2017. Citado 2 vezes nas páginas 1 e 6.

ZIENTEK, L. R. et al. The use of google scholar for research and research dissemination. *New Horizons in Adult Education and Human Resource Development*, Wiley Online Library, v. 30, n. 1, p. 39–46, 2018. Citado na página 5.