Instituto Federal de Educação, Ciência e Tecnologia de Goiás
Departamento de Áreas Acadêmicas do Câmpus Jataí
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

SISTEMA AUTOMÁTICO DE CONSULTA AO BOLETIM DE SERVIÇOS DO IFG APOIADO EM PROCESSAMENTO DE LINGUAGEM NATURAL

–

Camilla Vanessa de Souza Bim

Jataí, julho de 2019.

TERMO DE AUTORIZAÇÃO PARA DISPONIBILIZAÇÃO NO REPOSITÓRIO DIGITAL DO IFG - ReDi IFG

Com base no disposto na Lei Federal nº 9.610/98, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia de Goiás, a disponibilizar gratuitamente o documento no Repositório Digital (ReDi IFG), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, em formato digital para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IFG.

Identificação da Produção Técnico-Científica

[ ]	Tese	[ ]	Artigo Científico
[ ]	Dissertação	[ ]	Capítulo de Livro
[ ]	Monografia – Especialização	[ ]	Livro
[x]	TCC - Graduação	[ ]	Trabalho Apresentado em Evento
[ ]	Produto Técnico e Educacional - Tipo: _____		

Nome Completo do Autor: Camilla Vanessa de Souza Bim
Matrícula: 20151020230192
Título do Trabalho: Sistema Automático de Consulta ao Boletim de Serviços do IFG Apoiado em Processamento de Linguagem Natural

Autorização - Marque uma das opções

1. (x) Autorizo disponibilizar meu trabalho no Repositório Digital do IFG (acesso aberto);
2. ( ) Autorizo disponibilizar meu trabalho no Repositório Digital do IFG somente após a data ___/___/_____ (Embargo);
3. ( ) Não autorizo disponibilizar meu trabalho no Repositório Digital do IFG (acesso restrito).

Ao indicar a opção **2 ou 3**, marque a justificativa:

( ) O documento está sujeito a registro de patente.
( ) O documento pode vir a ser publicado como livro, capítulo de livro ou artigo.
( ) Outra justificativa: _____

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O/A referido/a autor/a declara que:

i. o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;
ii. obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autor/a, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia de Goiás os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;
iii. cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia de Goiás.

Jataí/GO, 18/03/2024.

Local Data

gov.br Documento assinado digitalmente
CAMILLA VANESSA DE SOUZA BIM
Data: 18/03/2024 14:33:14-0300
Verifique em https://validar.iti.gov.br

_____
Assinatura do Autor e/ou Detentor dos Direitos Autorais

Instituto Federal de Educação, Ciência e Tecnologia de Goiás
Departamento de Áreas Acadêmicas do Câmpus Jataí
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

SISTEMA AUTOMÁTICO DE CONSULTA AO BOLETIM DE SERVIÇOS DO IFG APOIADO EM PROCESSAMENTO DE LINGUAGEM NATURAL

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Goiás - Câmpus Jataí como um dos pré-requisitos necessários para aprovação no Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas.

Camilla Vanessa de Souza Bim

Orientador: Dr. Gustavo de Assis Costa

Jataí, julho de 2019

Dados Internacionais de Catalogação na Publicação na (CIP)

Bim, Camilla Vanessa de Souza.

Sistema automático de consulta ao boletim de serviços do IFG apoiado em processamento de linguagem natural [manuscrito]/ Camilla Vanessa de Souza Bim. - Jataí: IFG, Coordenação de Tecnologia em Análise e Desenvolvimento de Sistemas, 2019.

49 f.; il.

Orientador: Prof. Dr. Gustavo de Assis Costa.

Bibliografias.

1. Processamento de Linguagem Natural. 2. Recuperação de informação. 3. Extração de informação textual. I. Costa, Gustavo de Assis. II. IFG, Câmpus Jataí. III. Título.

Ficha catalográfica elaborada pela Seção Téc.: Aquisição e Tratamento da Informação.

Bibliotecária – Rosy Cristina Oliveira Barbosa – CRB 1/2380 – Campus Jataí. Cód. F016/2024-1.

Camilla Vanessa de Souza Bim

Sistema automático de consulta ao boletim de serviços do IFG apoiado em processamento de linguagem natural

Trabalho de Conclusão de Curso apresentado à Coordenação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia de Goiás (IFG), Câmpus Jataí, como parte dos requisitos para a conclusão do Curso e obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Monografia defendida e aprovada, em 04 de julho de 2019, pela banca examinadora constituída pelos seguintes membros:

(assinado eletronicamente)
Prof. Dr. Gustavo de Assis Costa
Presidente da banca / Orientador
Instituto Federal de Educação, Ciência e Tecnologia de Goiás – Câmpus Jataí

(assinado eletronicamente)
Prof. Me. Leizer Fernandes Moraes
Membro
Instituto Federal de Educação, Ciência e Tecnologia de Goiás – Câmpus Jataí

(assinado eletronicamente)
Prof. Me. Roney Lopes Lima
Membro
Instituto Federal de Educação, Ciência e Tecnologia de Goiás – Câmpus Jataí

Jataí – GO
2019

Dedico este trabalho à memória do meu pai, Claudionor Bim.

AGRADECIMENTOS

Agradeço primeiramente a Deus. Sem ele, não estaria aqui hoje.

Agradeço ao meu pai, pois ele me ensinou tudo que eu precisava para seguir a vida com seriedade e mostrou que era possível ser alguém sem precisar prejudicar ninguém.

Agradeço a Amanda que me ajudou muito nesses últimos anos, sendo essencial para que eu chegasse a finalizar o curso.

Agradeço a todos os professores que se mostraram sempre prontos a ajudar, motivando os alunos, para que estes tivessem uma perspectiva diferente de vida, e também, ao Instituto Federal de Goiás, por ser uma Instituição de Ensino respeitada e de qualidade, proporcionando aos alunos ótimas condições de estudo.

Epígrafe

“Cada sonho que você deixa para trás, é um pedaço do futuro que deixa de existir”

(Steve Jobs)

RESUMO

A quantidade de textos gerada diariamente no âmbito de qualquer empresa ou instituição é enorme, e este cenário não seria diferente no Instituto Federal de Goiás. Há um número elevado de documentos que são gerados em virtude da dinâmica burocrática institucional, a qual depende da oficialização de seus atos por meio de resoluções, portarias, além de outros. Tais documentos são disponibilizados publicamente à sociedade, mas, alguns destes são de interesse específico da comunidade interna do IFG por se tratar de ações que envolvem diretamente o cotidiano institucional. Neste aspecto, alunos e servidores têm à sua disposição um serviço chamado Boletim de Serviços. Para ter acesso aos mesmos, é necessário acessar site específico. Neste site, sua organização é feita a partir do agrupamento por ano/mês em um único arquivo, fazendo com que os usuários tenham que verificar cada conteúdo que compõe o boletim de forma manual, um a um, a fim de encontrar a informação que precisa, demandando tempo e esforço, tornando-o um trabalho maçante e passível de insucesso. Para contornar tal problema, este trabalho propõe o desenvolvimento de uma solução de busca baseada no estado da arte em processos de tratamento automático de documentos textuais, empregando-se, para tanto, técnicas de Processamento de Linguagem Natural e Recuperação de Informação. Após realização do levantamento bibliográfico dos respectivos temas, foram elencadas algumas bibliotecas de programação que pudessem viabilizar a construção da solução e, a partir do uso experimental destas, pôde-se avaliar a eficácia das mesmas. No decorrer do trabalho serão descritas e discutidas tais ferramentas e como as mesmas contribuíram para o que se pretendia realizar. Como metodologia para se implementar uma solução final eficaz considerou-se a utilização da técnica de índice invertido e, alternativamente, o uso de uma ferramenta existente para armazenamento e busca em documentos, o Elasticsearch. Assim como esperado, o resultado foi bastante satisfatório e acreditamos que, após os últimos testes, a solução poderá estar disponível ao usuário final em um curto período de tempo.

Palavras chaves: Processamento de Linguagem Natural; Recuperação de Informação; Extração de Informação Textual

ABSTRACT

The amount of texts generated daily within any company or institution is enormous, and this scenario would not be different in the Federal Institute of Goiás. There are a large number of documents that are generated due to the bureaucratic institutional dynamics, which depend on their actions through resolutions, ordinances, and others. These documents are made publicly available to society, but some of these are of specific interest to the IFG's internal community because they are actions that directly involve institutional daily life. In this respect, students and staff have at their disposal a service called Service Bulletin. To have access to them, it is necessary to access a specific website. On this site, your organization is grouped by year/month into a single file, causing users to have to check each content that makes up the newsletter manually one by one to find the information they need, demanding time and effort, making it a dull and unsuccessful job. To overcome this problem, this work proposes the development of a search solution based on the state of the art in processes of automatic treatment of textual documents, using, for this purpose, techniques of Natural Language Processing and Information Retrieval. After the bibliographic survey of the respective themes, some programming libraries were listed that could make possible the construction of the solution and, from the experimental use of these, it was possible to evaluate their effectiveness. In the course of the work will be described and discussed such tools and how they contributed to what was intended. As a methodology to implement an effective final solution we considered the use of the inverted index technique and, alternatively, the use of existing document storage and search tool, Elasticsearch. As expected, the result was quite satisfactory and we believe that after the last tests, the solution may be available to the end user in a short period.

Key words: Natural Language Processing; Information Retrieval; Extraction of Textual Information

LISTA DE ILUSTRAÇÕES

Figuras

Quadros

Tabelas

LISTA DE ABREVIATURAS E SIGLAS

E.I.	Extração de Informação
I.E.	Índice Invertido
I.A.	Inteligência Artificial
I.N.	Inteligência Natural
NOSQL	Not Only SQL
PDF	Portable Document Format
PLN	Processamento de Linguagem Natural
RI	Recuperação de Informação

SUMÁRIO

CAPÍTULO I

1. INTRODUÇÃO

A pelo menos 5000 anos a humanidade realiza busca e recuperação de informações. Isso foi feito em tabuletas de argila, hieróglifos, livros, entre outros (YATES; NETO, 2013). Com o passar do tempo, a forma em como buscar e organizar informações mudou em vários aspectos, assim como, na maneira em que as informações estão dispostas.

Com o avanço da tecnologia e o considerável aumento de dados textuais, percebeu-se a necessidade de realizar a análise destes, porém, estes estão em grande parte dispostos de maneira textual e não estruturada. Para suprir essa necessidade e sobressair a estas dificuldades foi necessário aderir a técnicas de Inteligência Artificial. De acordo com Artero (2008, p. 18), "inteligência artificial (IA) é uma forma de simular a inteligência natural (IN) [...]". A IA vem trazendo resultados interessantes para a sociedade, como a criptografia, tradutores, como Google Tradutor, sistemas de análise de sentimento, entre outros exemplos.

Dentro da Inteligência Artificial, encontramos a subárea chamada de Processamento de Linguagem Natural (PLN), definida por Rosa (2011, p. 137) como "a habilidade de um computador processar a mesma linguagem que os humanos usam no dia a dia", e foi por meio de técnicas oriundas desta área adjunta a outras técnicas e ferramentas que esta solução se fundamentou para que fosse possível a sua execução.

Esta solução buscou resolver um problema que ocorre no Instituto Federal de Goiás (IFG). No IFG, os servidores precisam estar atentos a um conjunto de documentos denominados de Boletim de Serviços. Estes documentos são dispostos em site específico e, sempre que são publicados, exige do usuário a inspeção manual de cada arquivo em busca das informações necessárias, demandando tempo e podendo por vezes, não encontrar o que necessitam. No momento em que a solução foi implementada, somava-se cerca de 17 mil páginas[1] dispersas em mais de 200 documentos salvos no formato pdf (Portable Document Format). Notou-se assim, que a maneira em que são feitas as buscas, necessitava de melhorias, para que em um futuro não tão longe, sua escalabilidade não se torne um problema, mas sim, um simples detalhe.

[1] Informação obtida após análise realizada nos Boletins de Serviço contidos em site específico do IFG

Para que isso fosse possível, foram utilizadas bibliotecas que contemplam técnicas de PLN e Recuperação de Informação (RI). Estas por sua vez, possuem a finalidade de retornar resultados compatíveis de acordo com a entrada de dados. Por essa razão, os experimentos foram todos realizados a partir dos Boletins de Serviço.

Com isso, o objetivo deste trabalho, foi desenvolver uma ferramenta que fizesse a extração e indexação de termos específicos dos documentos que compõem os boletins de serviços do IFG, permitindo consultas em toda sua base textual possibilitando a busca por termos de interesse do usuário. Para atender o objetivo principal desta pesquisa, foram elencados objetivos específicos a fim de facilitar a pesquisa e a implementação da solução. Os objetivos visavam compreender as técnicas de PLN e de Índice Invertido, a realização de pesquisas em diferentes bibliotecas no intuito de encontrar alguma que possuísse suporte a técnica de Reconhecimento de Entidade Nomeada (NER) e que também, oferecesse suporte à implementação do Índice Invertido e estudar e avaliar técnicas de NER baseadas em corpus da língua portuguesa, mais especificamente a técnica NER. No decorrer do desenvolvimento do trabalho, foi percebido que a utilização da técnica NER não seria necessária, uma vez que a busca por termos se fazia de maneira aceitável, sem a utilização desta. O NER é responsável por extrair nomes de empresas, pessoas, lugares, a partir de um vocabulário dado (PROVOST; FAWCETT, 2016). Com isso, em se tratando de busca por informação, este método poderia ser utilizado para realizar buscas mais específicas, como por exemplo: Buscar todos os termos que foram nomeados/considerados como Lugares. Em consequência disso, este método foi descartado, pois o intuito era realizar busca por termos.

O processo de desenvolvimento desta solução tecnológica foi conduzido de acordo com os conceitos de uma pesquisa tecnológica, que “visa a materialização de um produto, protótipo, processo ou um estudo de viabilização desses" (VALERIANO, 1998, [n.p.]). Com isso, realizou-se um levantamento literário acerca de PLN, RI e bibliotecas como Spacy, PyPDF2, tanto como um estudo mais profundo em relação à linguagem de programação Python.

A solução resultante deste trabalho, teve como intuito principal, auxiliar os servidores do Instituto Federal de Goiás neste aspecto. Ou seja, fornecer aos servidores uma ferramenta que os auxilie na busca por termos com a finalidade de trazer resultados rápidos e precisos.

1.1 METODOLOGIA E FERRAMENTAS UTILIZADAS

Nos tópicos seguintes, serão demonstrados como foram realizados os processos de desenvolvimento da pesquisa, ou seja, de que maneira foi gerenciado a implementação das atividades da solução e quais ferramentas auxiliaram no desenvolvimento.

1.1.1 Levantamento Bibliográfico

O levantamento bibliográfico foi realizado a partir da leitura de periódicos com assuntos similares ao do tema abordado neste trabalho, como PLN e RI na intenção de obter o máximo de informações possíveis sobre a área trabalhada e também, verificar de que maneira os pesquisadores estão utilizando as técnicas de PLN e RI. A linguagem escolhida para a execução deste trabalho foi a linguagem de programação Python, por conta disso, foi realizado um estudo aprofundado sobre a linguagem, visto a necessidade utilizar recursos importantes disponíveis na mesma, como por exemplo, a estrutura de dados dict ou, mais comumente chamada de dicionário. Alguns autores se fizeram necessários nesta etapa do trabalho, sendo, suas obras de grande importância para a implementação da solução. O quadro 1 cita os principais autores e obras que foram fonte principal neste trabalho.

Quadro 1 - Principais Referências Bibliográficas

Conceito	Autor(es)	Fonte(s)
Processamento de Linguagem Natural	GRUS, J.	Data Science do Zero: Primeiras Regras com o Python
Recuperação de Informação	BAEZA-YATES, R. RIBEIRO-NETO	Recuperação de Informação: Conceitos e Tecnologia das Máquinas de Busca
Linguagem de Programação Python	BARRY, P.	Use a Cabeça! Python

Fonte: (Elaborado pela autora)

1.1.2 Spyder

Para o desenvolver desta solução tecnológica, utilizei de um Ambiente Integral de Desenvolvimento (IDE) o qual facilitou os testes e auxiliou para uma implementação mais objetiva. Tendo em vista que esta solução também está inserida no campo de Ciência dos Dados, foi pertinente a busca por uma IDE que trouxesse um amparo maior para soluções voltadas para esta área. Antes dela, foram realizados testes em diversas IDE's como PyCharm, Jupyter-Notebook, até chegar a IDE Spyder.

O Spyder é um poderoso ambiente científico escrito em Python, para Python, e projetado por e para cientistas, engenheiros e analistas de dados. Ele oferece uma combinação única de recursos avançados de edição, análise, depuração e criação de perfil de uma ferramenta de desenvolvimento abrangente com exploração de dados, execução interativa, inspeção detalhada e recursos de visualização de um pacote científico (SPYDER, 2019). Ao utilizar esta IDE, foi possível obter algumas ferramentas importantes que permitiram uma análise melhor do código, como por exemplo: bibliotecas e variáveis não utilizadas, maior interação com cada variável alocada na memória, entre outras facilidades que esta IDE oferece. A escolha também se fez conveniente pelo fato de a IDE ter sido projetado principalmente para cientistas de dados, área na qual, este trabalho está inserido.

Para que fosse possível trabalhar com a IDE Spyder utilizei de Anaconda, que facilitou o trabalho, visto que esta plataforma é voltada para a utilização das linguagens de programação Python e R. Com ela, é possível utilizar diversas IDE's nas quais são inseridas automaticamente no momento em que Anaconda é instalada, como Jupyter Notebook, Orange3, Spyder, entre outros.

> Com mais de 11 milhões de usuários em todo o planeta, Anaconda permite aos cientistas de dados o desenvolvimento, o teste e o treinamento em uma única máquina, auxiliando-os com download de pacotes, o gerenciamento de bibliotecas, e também, a análise e a visualização de dados (ANACONDA, 2019).

1.1.3 Taiga

Para auxiliar no gerenciamento do projeto foi utilizado a ferramenta Taiga. Ela, possui em sua essência conceitos da metodologia ágil Scrum, Kanban, entre outras funcionalidades. Neste processo de gerenciamento, foram utilizadas as duas funções de Taiga citadas anteriormente. De acordo com Rubin (2017), o Scrum é formado por um pequeno conjunto de valores centrais, princípios e práticas. A partir da metodologia ágil Scrum, foi possível criar sprints, onde estas, possuíam data limite para entrega de atividades, auxiliando na entrega, de acordo com o cronograma estipulado no projeto. Com a execução de todas as atividades, logo, de todas as sprints criadas, o backlog, ou seja, o produto final seria finalizado. O Kanban, também auxiliou na gestão do projeto, ele, possibilitou uma visão objetiva de como estava o andamento do projeto a partir de uma tabela que demonstrava as novas atividades, em andamento, prontas e finalizadas.

Este trabalho está dividido em seis capítulos, descritos resumidamente a seguir:

Capítulo I – Introdução: Apresenta a justificativa do trabalho, assim como os objetivos, a metodologia utilizada na qual abrange as ferramentas utilizadas para a implementação, modelagem e programação e a organização dos capítulos deste trabalho.

Capítulo II – Fundamentação teórica: Aborda o referencial teórico com os principais conceitos de Processamento de Linguagem Natural, suas técnicas e bibliotecas utilizadas.

Capítulo III – Tecnologia e Ferramentas Utilizadas: Apresenta as tecnologias e ferramentas utilizadas trazendo uma abordagem teórica juntamente com relatos de como a tecnologia foi utilizada na solução.

Capítulo IV – Implementação: Apresenta a implementação do sistema, desta forma é efetuada uma descrição sintética da aplicação desenvolvida.

Capítulo V – Testes, avaliações e limitações: Relata os resultados, bem como suas limitações, obtidos por meio de testes realizados na solução.

Capítulo VI – Conclusões e trabalhos futuros: Apresenta as conclusões e sugestões de trabalhos futuros.

CAPÍTULO II

2. FUNDAMENTAÇÃO TEÓRICA

2.1 Introdução

Neste capítulo será apresentada toda a fundamentação teórica que envolve o processo investigativo em questão. Temas como Inteligência Artificial, Processamento de Linguagem Natural, Recuperação de Informação, linguagem de programação Python, entre outros, serão apresentados e detalhados, com a finalidade de demonstrar a sua utilidade neste projeto.

2.2 Inteligência Artificial

A Inteligência Artificial foi a base para que esta solução tecnológica pudesse acontecer. Com uma vasta área, hoje, a I.A. é utilizada por milhares de pessoas, uma vez que, tecnologias advindas de seus conceitos, são empregadas diariamente por milhares de pessoas. O fato pode ser percebido com base em uma pesquisa realizada em 2018 pelo IBGE[2] que revelava que mais 138 milhões de brasileiros pessoas possuíam smartphone, sendo estes, dotados de tecnologias que possuem por vezes em sua essência a Inteligência Artificial, ou seja, mais da metade da população, tem acesso direto a tecnologia, seja na hora de digitar mensagens, tirar fotos, reconhecimento facial, entre outros.

Todas essas formas de utilizar a I.A., demonstram uma surpreendente habilidade de fazer as coisas acontecerem de maneira mais inteligente, apesar de toda a tecnologia envolvida o usuário a utiliza de maneira transparente. Para uma maior compreensão sobre a I.A., a Smart Read diz que ela corresponde a um ‘comportamento inteligente’ ou à ‘capacidade de raciocínio’ dos artefatos, ou seja, é algo que pode ser definido como a ‘inteligência’ que qualquer aparelho e/ou máquina criada pelo Homem revela ter (2017).

Tecnicamente, a I.A. engloba muitas teorias, métodos, tecnologias e áreas como Machine Learning, Deep Learning, PLN, robótica, entre outras (S.A.S., 2018). Ela funciona ao combinar grandes quantidades de dados com processamento rápido e interativo e algoritmos inteligentes, permitindo ao software aprender automaticamente com padrões ou informações nos dados (S.A.S., 2018).

---

[2] Informação obtida através do link <https://www.tudocelular.com/mercado/noticias/n120572/internet-pesquisa-brasileiros-online.html> Acesso em: 07 de jul. 2019.

Essas são algumas das inúmeras definições encontradas atualmente, sobre o que é a Inteligência Artificial (IA). Embora esteja presente no dia a dia de milhares de pessoas, sua definição, muitas vezes ainda gera controvérsias.

Para o desenvolvimento da solução tecnológica descrita neste trabalho, utilizou-se de PLN, sendo esta, considerada como uma subárea de I.A., a qual foi aplicada nesta solução tecnológica, algumas de suas técnicas onde serão tratadas com mais detalhes nos próximos tópicos.

2.3 Processamento de Linguagem Natural

O Processamento de Linguagem Natural (PLN) foi a base para a realização deste trabalho, uma vez que, foi a partir de seus métodos e técnicas que possibilitaram a busca em termos. Porém, é interessante compreender no que consiste a linguagem natural, para posteriormente, dar sequência em PLN. A linguagem natural de qualquer nação pode ser representada de várias formas, como falada, escrita, ou mesmo, por sinais. Neste contexto, a forma escrita é a que se dará mais atenção. Em relação aos aspectos de uma linguagem natural, é possível perceber vários pontos, conforme demonstra o quadro 2 a seguir:

Quadro 2 - Níveis de Entendimento da Linguagem Natural

Fonético	Relacionamento das palavras com os sons da língua do ponto de vista da articulação e da recepção auditiva
Morfológico	Construção das palavras a partir unidades de significado primitivas e de como classificá-las em categorias morfológicas
Sintático	Relacionamento das palavras entre si, cada uma com o seu papel estrutural nas frases, a construção de frases
Semântico	Relacionamento das palavras com seus significados e de como eles são compostos para que a frase tornem-se compreensíveis
Pragmático	Uso de frases e sentenças em diferentes contextos, afetando o significado

Fonte: (MARTINS et al., 2010)

Martins et al. (2010) consegue explanar claramente quando faz uma comparação sucinta, porém, compreensível entre máquinas e linguagem natural:

> Normalmente, computadores estão aptos a compreender instruções escritas em linguagens de programação como Python, C, Java, mas requerem mais aprofundamento quando se trata da linguagem humana,

> pois, no primeiro caso, as regras são fixas, com estruturas lógicas bem definidas, permitindo ao computador como proceder a cada comando, ao contrário da linguagem humana, onde uma simples frase pode conter ambiguidades e interpretações que dependem do contexto, culturas, entre outros conceitos abstratos (MARTINS et al., 2010).

Apesar da complexidade, houve a necessidade em compreender a linguagem natural e aplicar técnicas que permitissem a resolução de vários problemas ou tarefas ligadas a ela. Com isso, os pontos citados no quadro 2, pertinentes a linguagem natural, foram adaptados para que estes fossem utilizados na área de PLN, conforme demonstra quadro 3.

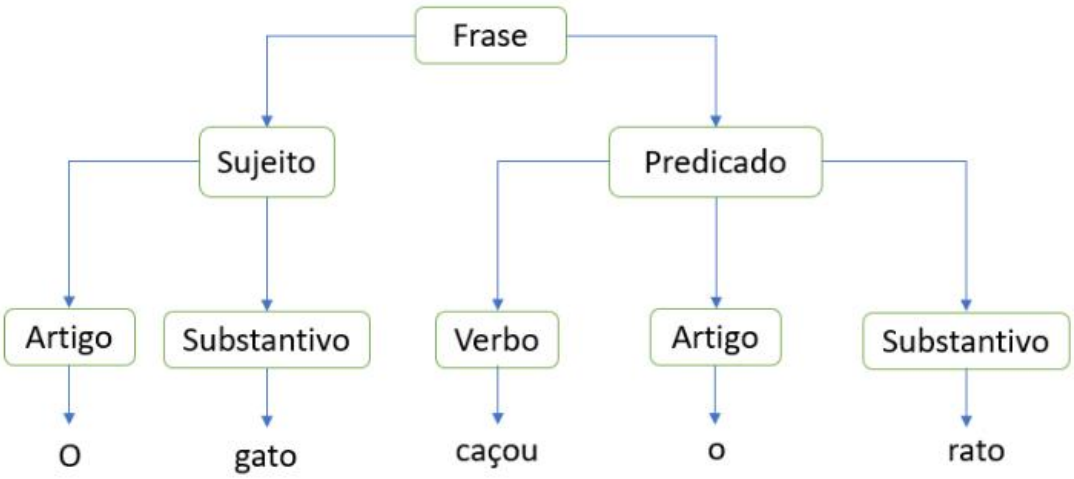
Quadro 3 - Etapas do Processamento de Linguagem Natural

Fonologia	Esta etapa só entra se o computador precisar ouvir a pessoa, identifica as palavras a partir do som
Análise morfológica	Identifica e analisa a estrutura das palavras. É impraticável ter um dicionário com todas as palavras, portanto, são implementadas regras para palavras regulares e adiciona-se algumas palavras irregulares dentro de domínios
Análise léxica	Determina o significado das palavras individuais
Análise sintática	Reconhece e analisa frases e as regras da gramática. Uma árvore sintática é usada para fazer o reconhecimento de cima para baixo (top-down)

Fonte: (STROSKI, 2018)

A figura 1 mostra como funciona a Análise Sintática utilizando o Top Down.

Figura 1 - Exemplo do Método Top-Down



Fonte: (STROSKI, 2018)

Neste caso, outras etapas são analisadas também, conforme demonstra quadro 4.

Quadro 4 - Etapas do Processamento de Linguagem Natural – Top-Down

Análise semântica	Verifica o significado das frases mapeando a estrutura sintática, precisa ter informações sobre o mundo real para eliminar ambiguidade e sentenças sem lógica, como “sorvete quente”
Análise do discurso	Observa como as sentenças se relacionam em um texto
Análise pragmática	Verifica se a interpretação semântica está correta e determina significados não claros.

Fonte: (STROSKI, 2018)

A partir da base exposta de PLN é possível então, dar sequência e compreender melhor como surgiu a área, que hoje é essencial para a resolução de problemas que lidam com a linguagem natural. Foi a partir de um artigo produzido por Alan Turing em 1950, no qual o mesmo tratava de Inteligência Artificial, que o termo Processamento de Linguagem Natural (PLN) começou a ser utilizado. A partir daí outros nomes foram aparecendo e, trazendo consigo, novidades como por exemplo, os tradutores automáticos.

Algumas literaturas consideram PLN como uma subárea da Inteligência Artificial, já outras, tratam-na como uma área multidisciplinar, a qual agrupa competências variadas como a Linguística, a Ciência da Computação, a Lógica, a Matemática, entre outras (CENTERATTO, 2015). Assim como o próprio nome sugere, ela realiza o processamento de dados inerentes à linguagem natural, seja ela qual for, inglês, português, francês, entre outras, de acordo com a linguagem de programação e biblioteca específica.

Atualmente, PLN é vista naturalmente em nosso cotidiano, como por exemplo, o reconhecimento de fala, a tradução automática de textos, a análise de sentimentos, a geração de textos automaticamente, entre outros. Para Centeratto (2015), umas das principais metas de PLN é a construção de modelos computacionais que possibilitem a comunicação entre o homem e a máquina, via linguagem natural. Grus (2016, p. 239), diz que o PLN “refere-se a técnicas computacionais envolvendo a linguagem”, enquanto Rosa (2011) indica que “todas as definições incorporam a noção de armazenamento em computador e manipulação de dados linguísticos” (p. 137). A partir disso, é possível perceber que, PLN tem como objetivo analisar, manipular e tratar dados, visando facilitar o dia a dia de usuários que se deparam diariamente,

seja de maneira transparente ou não, ao acesso e ao manuseio de informações que lidam com a linguagem natural.

A área de PLN é vasta e pode ser utilizada para diversos fins, conforme mencionado anteriormente. Para isso, dependendo de cada situação, há técnicas mais e menos indicadas. Neste caso, a intenção foi realizar a indexação, ou seja, a inclusão de grandes quantidades de texto em índices, nos quais, possibilitassem uma busca em termos de maneira rápida. Com isso, foi realizado alguns pré-processamentos que abstraem a estrutura da língua, reduzindo o vocabulário, deixando apenas o que é informação relevante (RODRIGUES, 2017). Para isso, algumas tarefas foram realizadas como demonstra o quadro 5 a seguir.

Quadro 5 – Tarefas Para o Pré-processamento de dados

TAREFA	CARACTERÍSTICA
Normalização	Abrange tokenização, transformação de letras maiúsculas para minúsculas,
Remoção de Stop Words	Método aplicado para remover palavras muito frequentes, sendo consideradas irrelevantes em uma busca, como por exemplo: "a", "de", "que", entre outras.

Fonte: Adaptado de RODRIGUES (2017)

Outras vertentes advindas de PLN foram utilizadas neste trabalho, como a RI, na qual será explanada no próximo tópico.

2.4 Recuperação de Informação

Após a invenção da web, criada por Tim Berners-Lee a busca por informação tornou-se mais frequente, não ficando limitada apenas a bibliotecas, devido a criação de bilhões de documentos elaborados por usuários no qual compõem hoje o maior repositório humano de conhecimento da história (BAEZA-YATES; RIBEIRO-NETO, 2013).

De acordo com Amaral (2016) mais de 1,7 bilhões de pessoas estão usando redes sociais atualmente, 100 horas de vídeos são carregados no Youtube por minuto, sendo 2,3 trilhões de GB de dados criados por dia, enfim, com isso, é possível perceber o quão abrangente é o repositório de documentos criados diariamente e como a RI pode auxiliar na extração de dados e transformá-los em informações para os usuários.

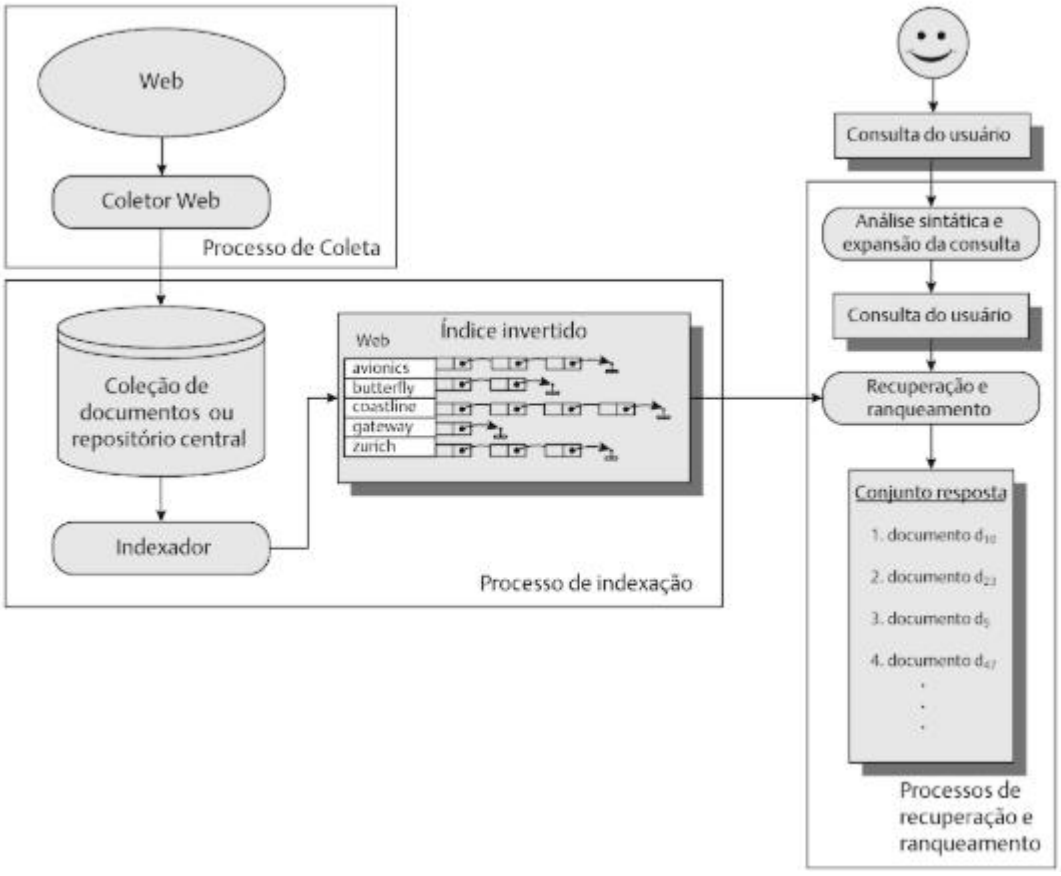
Para Baeza-Yates e Ribeiro-Neto (2013), RI é uma “área que cresceu muito além dos seus objetivos iniciais de indexação de textos e de buscas por documentos úteis em uma coleção”. Diante disso, é interessante trazer uma definição mais ampla sobre RI:

> RI é uma área abrangente da Ciência da Computação, na qual trata da representação, armazenamento, organização e acesso a itens de informação, como documentos, páginas Web, registros estruturados e semiestruturados, objetivando fornecer aos usuários facilidade de acesso às informações de seu interesse (BAEZA-YATES; RIBEIRO-NETO, 2013).

Ao se tratar da recuperação de informações, é interessante destacar que a relevância dos resultados lida com um julgamento pessoal e pode mudar com o tempo, à medida que informações novas tornam-se disponíveis (BAEZA-YATES; RIBEIRO-NETO, 2013).

A partir da figura 2, é possível perceber como funciona uma arquitetura de alto nível de um sistema de RI.

Figura 2 - Arquitetura de alto nível do software de um sistema de RI. A coleta é um módulo adicional necessário para os sistemas de RI para a *Web,* como as máquinas de busca.



Fonte (YATES; NETO, 2013)

É possível verificar na figura 2, que em um primeiro momento, o processo de coleta acontece. Esta coleta de dados ou mesmo, extração de dados, é feita a partir de métodos e bibliotecas específicas que serão demonstradas nos capítulos posteriores. O processo seguinte,

mostra que é necessário realizar a indexação e enviá-la para um Índice Invertido. É interessante ressaltar que neste trabalho, foram realizadas duas soluções em que ambas utilizam o Índice Invertido. Uma, para que a técnica de Índice Invertido fosse de fato implementado juntamente com um método de busca, e outra, voltada para o usuário final, onde, há um banco de dados voltado para documentos, porém, que utiliza automaticamente o Índice Invertido em sua essência. No processo seguinte, o usuário realiza uma busca e é retornado resultados de acordo com a busca do usuário. Em ambas as soluções haverá esta arquitetura conforme descrito, porém, no caso da solução realizada com o Elasticsearch, é possível ver mais claramente o processo de ranqueamento acontecer. No Capítulo V todo este conjunto de atividades poderão ser percebidos mais facilmente.

2.4.1 Índice Invertido

A utilização da estrutura de dados Índice Invertido no desenvolvimento de um projeto no qual envolve busca por termos é essencial para que estas buscas sejam realizadas de maneira eficiente. Há também outra maneira de realizar busca, chamada busca sequencial. Esta por sua vez tem a missão de retornar as informações desejadas pelo usuário, não se preocupando com a eficiência do algoritmo, conforme Baeza-Yates e Ribeiro-Neto (2013) descrevem-na quando dizem que ela é a forma mais básica de busca e não exige qualquer estrutura construída ou mantida sobre o texto.

Quando lidamos com a questão eficiência, alguns pontos são levados em consideração. O quadro 6 torna claro tais pontos.

Quadro 6 - Forma que é medido a eficiência em um sistema de busca

Tempo de Indexação	Tempo necessário para criar o índice. Em geral, o tempo de indexação é linear em função do tamanho do texto.
Espaço de Indexação	Espaço utilizado durante a geração do índice.
Armazenamento do Índice	Espaço necessário para armazenar o índice, uma vez que tenha sido gerado.
Latência da Consulta	Intervalo de tempo entre a chegada da consulta no sistema de R.I. e a geração da resposta.
Taxa de Transferência da Consulta	Número médio de consultas processadas por seguro.

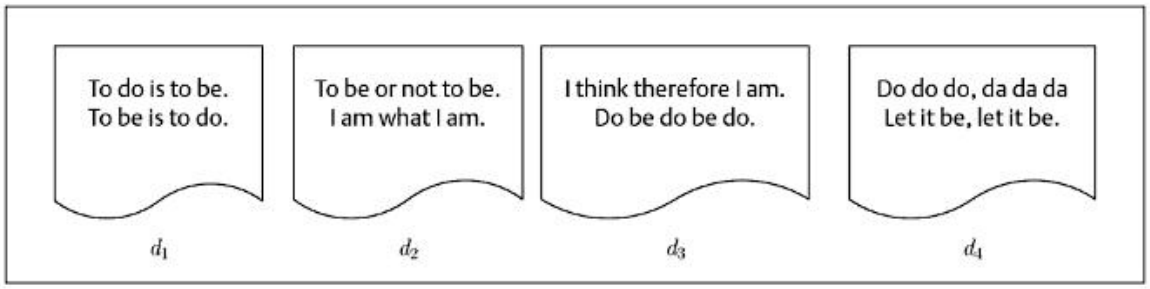
Fonte: (BAEZA-YATES; RIBEIRO-NETO, 2013, p. 340)

De acordo com Ramakrishnan e Gehrke (2011, p. 775) “Um Índice Invertido é uma estrutura de dados que permite a rápida recuperação de todos os documentos que contém um termo da consulta”. Em outras palavras, “um índice invertido (ou arquivo invertido) é um mecanismo orientado a palavras para a indexação de uma coleção de texto a fim de acelerar a tarefa de busca” (BAEZA-YATES; RIBEIRO-NETO, 2013, p. 342).

O processo de indexação descrito, armazena os tokens e a posição exata em que aparecem no texto. Estes elementos são inseridos em um dict ou dicionário (estrutura de dados do tipo chave/valor da linguagem de programação Python) permitindo assim, que a busca seja realizada dentro do índice, trazendo como resultado, o texto no qual encontra-se o termo.

Para haver uma maior compreensão acerca do Índice Invertido a figura 3 mostra uma pequena coleção composta por quatro documentos denominados como d1, d2, d3 e d4. Neste exemplo, todos os documentos serão submetidos à técnica de Índice Invertido.

Figura 3 - Representação de Uma Coleção de Documentos



Fonte: (BAEZA-YATES; RIBEIRO-NETO, 2013, p. 35)

Um Índice Invertido é composto por dois elementos: O vocabulário e as ocorrências, sendo o primeiro, o conjunto de todas as palavras diferentes no texto e o segundo, a quantidade de vezes que tal palavra aparece em cada documento (BAEZA-YATES; RIBEIRO-NETO, 2013).

A princípio é feita a técnica de tokenização, ou seja, para cada vocabulário (ou token) é feita uma análise de quantas vezes e em qual documento determinado vocabulário foi encontrado. É possível perceber na figura 4 como funciona a indexação dos documentos da figura 3 em um Índice Invertido. Nela, consta quantos documentos o vocabulário 'to' aparece e demonstra detalhadamente quantas ocorrências aparece em cada documento. Por fim, é possível verificar a lista invertida, a quantidade e a referência do documento em que o vocabulário é encontrado.

Figura 4 - Índice Invertido Básico e Matriz de Termos por Documentos

Vocabulário	n_i	d_1	d_2	d_3	d_4	Ocorrências representadas como listas invertidas
to	2	4	2	-	-	[1,4],[2,2]
do	3	2	-	3	3	[1,2],[3,3],[4,3]
is	1	2	-	-	-	[1,2]
be	4	2	2	2	2	[1,2],[2,2],[3,2],[4,2]
or	1	-	1	-	-	[2,1]
not	1	-	1	-	-	[2,1]
I	2	-	2	2	-	[2,2],[3,2]
am	2	-	2	1	-	[2,2],[3,1]
what	1	-	1	-	-	[2,1]
think	1	-	-	1	-	[3,1]
therefore	1	-	-	1	-	[3,1]
da	1	-	-	-	3	[4,3]
let	1	-	-	-	2	[4,2]
it	1	-	-	-	2	[4,2]

Fonte: (BAEZA-YATES; RIBEIRO-NETO, 2013, p. 343)

A partir da utilização do Índice Invertido, também é possível aplicar o chamado Índice Invertido Completo. Com ele, a lista invertida também é contemplada com o local onde o

vocabulário se encontra no documento. Desta forma, é possível realizar busca em termos e com a possibilidade de retornar trechos do texto, também chamados de snippet, a partir do vocabulário encontrado.

Com a aplicação do Índice Invertido Completo, a lista invertida ficaria da seguinte forma:

[1,4,[1,4,6,9]],[2,2,[1,5]]
[1,2,[2,10]],[3,3,[6,8,10]],[4,3,[1,2,3]]
[1,2,[3,8]]
[1,2,[5,7]],[2,2,[2,6]],[3,2,[7,9]],[4,2,[9,12]]
[2,1,[3]]
[2,1,[4]]
[2,2,[7,10]],[3,2,[1,4]]
[2,2,[8,11]],[3,1,[5]]
[2,1,[9]]
[3,1,[2]]
[3,1,[3]]
[4,3,[4,5,6]]
[4,2,[7,10]]
[4,2,[8,11]]

Fonte: (BAEZA-YATES; RIBEIRO-NETO, 2013)

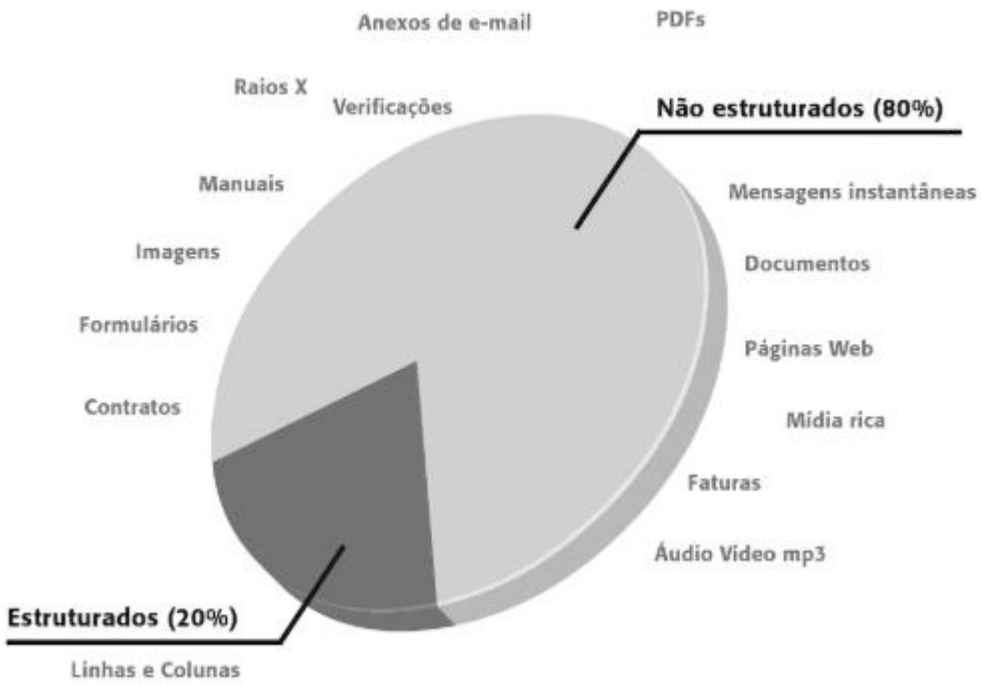
2.5 Dados

Os dados foram a matéria-prima desta solução tecnológica, visto que, os Boletins de Serviço do Instituto Federal de Goiás são formados por dados, sendo classificados como não-estruturados. Grus afirma que “vivemos em um mundo que está soterrado por dados” (2016, p.1) e quando ele diz isso, está se referindo a todos os dados que estão sendo criados a cada instante, por meio redes sociais, jogos online, entre outros. Estes dados estão dispostos de duas formas: Estruturados e não estruturados, como segue definição.

> Dados estruturados são organizados em linhas e colunas em um formato definido de forma rígida, fazendo com que a recuperação e o processamento sejam feitos de maneira mais eficiente. Já os dados não estruturados não são organizados em linhas e colunas, tornando a consulta e recuperação difícil de ser realizada. (SOMASUNDARAM; SHRIVASTAVA; EDUCATION SERVICES, 2011).

A figura 5 ilustra de maneira quantitativa o percentual de dados estruturados e não estruturados dispostos na rede.

Figura 5 - Tipo de Dados



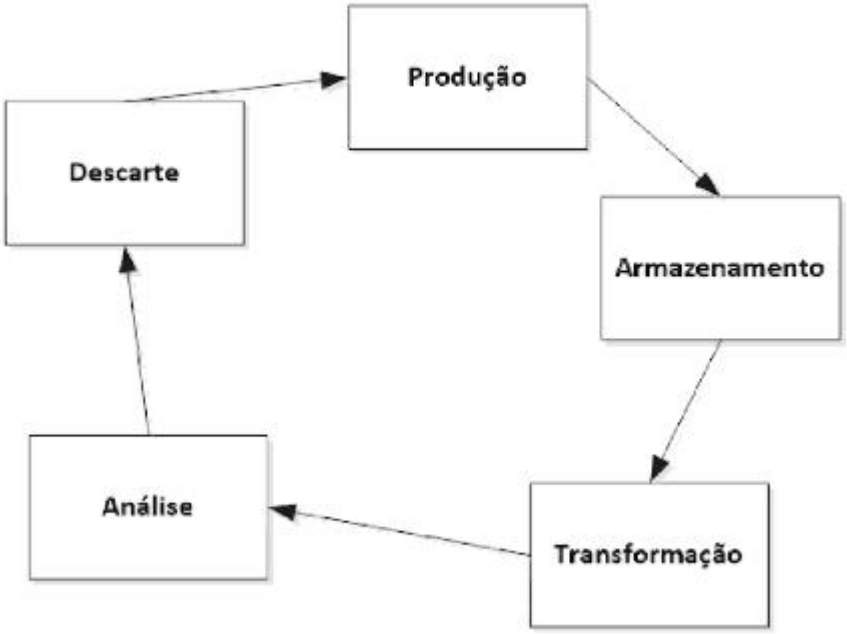
Fonte: (SOMASUNDARAM; SHRIVASTAVA; EDUCATION SERVICES, 2011)

É possível perceber que o aumento de dados não estruturados é tendencioso, sendo, não apenas interessante, mas necessário, fazer uso de técnicas específicas para analisá-los de maneira correta.

2.6 Ciência dos Dados

A Ciência dos Dados é uma área que existe há 30 anos na qual abrange a computação, estatística e o setor de negócios (WEBER; MELO, 2019). O campo visa estudar, analisar e transformar dados. Para Amaral (2016, p.4) esta ciência "trata de estudar o dado em todo o seu ciclo de vida, da produção ao descarte". Além disso, é um campo que está em evidência, como observa Grus (2016). Sendo assim, é possível compreender que esta ciência tem como matéria prima dados em geral e o cientista de dados tem como responsabilidade transformar tais dados em algo útil para os usuários a partir da utilização de técnicas específicas. Amaral (2016) traz em seu material o ciclo de vida de um dado. Nele, podemos verificar como funciona o ciclo e em seguida, será possível compreender como ele foi útil na elaboração desta solução tecnológica.

Figura 6 - Ciclo de Vida de um Dado



Fonte: (AMARAL, 2016, p.6)

A figura 6 aponta 5 passos onde este ciclo é representado. O primeiro passo, onde o dado é produzido. A produção deste dado pode ser feita de diversas formas e situações. Pierson (2019, p.7) relata que “eles vêm de cada computador, dispositivo móvel, câmera e sensor imaginável - e agora até de relógios e tecnologias em roupas”. Após a produção do dado, o mesmo é armazenado. Com isso, é possível realizar a transformação do mesmo. No caso desta solução, os dados oriundos do boletim de serviço, foram passados por um processo de tratamento, onde, utilizou-se da biblioteca Spacy para tratar os boletins extraindo informações de cada token para dar sequência no processo. Além disso, também foi utilizado do método STOP WORD, onde é responsável por filtrar as palavras, removendo palavras consideradas irrelevantes (exemplo: é, ser, este, esta, ...) para estes propósitos de busca.

Em seguida, na etapa de Análise é feita a extração de informação e conhecimento dos dados, conforme define Amaral (2016). Por fim, poderá ocorrer o momento de descarte, quando não é mais necessário a utilização do dado.

CAPÍTULO III

3. TECNOLOGIA E FERRAMENTAS UTILIZADAS

3.1 Linguagem de Programação Python

Python é uma linguagem de programação criada em 1990 por Guido Van Rossum. De acordo com Pereira (2018), seu principal objetivo era o de ser uma linguagem mais acessível ao meio acadêmico e alternativa a linguagens como Matlab e IDL. Python é uma linguagem de altíssimo nível orientada a objeto, de tipagem dinâmica e forte, interpretada e interativa. Ela é poderosa, fácil de aprender, rápida, com código aberto e empresas como Google, Youtube e Microsoft as utilizam (PYTHON, 2019).

A escolha por sua utilização baseia-se no fato da mesma possuir bibliotecas que foram pertinentes a elaboração desta solução, bibliotecas estas, que serão expostas posteriormente neste capítulo. A versão da linguagem de programação Python utilizada para a realização desta solução foi a 3.x, por ser a mais atual e também pelo fato dos desenvolvedores a considerarem a mais indicada para projetos por continuar em desenvolvimento ativo.

3.2 Biblioteca PyPDF2

A biblioteca PyPDF2 faz parte do arcabouço de bibliotecas pertencentes a linguagem de programação Python. Com ela, é possível extrair informação do documento, dividir documentos página por página, mesclar documentos página por página, mesclar várias páginas em uma única página, entre outras funcionalidades (PYPI, 2019). Neste trabalho, esta biblioteca foi muito importante para que pudéssemos realizar a etapa de extração de informação dos documentos PDF chamados Boletins de Serviço, como o número da página e também, os dados inseridos em cada boletim.

3.3 Biblioteca Spacy

Spacy é uma biblioteca livre e de código aberto para o processamento avançado de linguagem natural (NLP) em Python e foi publicado pela empresa Explosion. Ela permite o processamento de texto de maneira inteligente e também, a utilização de métodos nos quais possuem em sua essência técnicas de PLN sendo considerada a mais rápida do mundo e excelente em tarefas de extração de informação em grande escala (SPACY, 2019).

A aplicação desta biblioteca foi de extrema importância, uma vez que foi possível a extração de recursos linguísticos, permitindo através disso, a aplicação da técnica de tokenização e também, a utilização dos métodos start e stop words. A técnica de tokenização realiza a separação de palavras ou sentenças em unidades, facilitando o manuseio nos dados. Já com o método start, foi possível localizar determinados tokens a partir de seu índice dentro do documento. O método stop words, ignora palavras que são consideradas irrelevantes em um contexto como este, voltado para a busca de termos. Com isso, a biblioteca Spacy, trouxe bastante flexibilidade, auxiliando no desenvolvimento.

3.4 Biblioteca os

A biblioteca os foi utilizada para que fosse possível o código relacionar-se com o sistema operacional. De acordo com Barry (2017, p.9) essa biblioteca "fornece um modo independente da plataforma para interagir com o sistema operacional subjacente". Com ela, foi possível ter acesso a diretórios específicos onde os boletins de serviço se encontravam.

Desta forma, a biblioteca os, juntamente com a PyPDF2, fizeram parte da primeira etapa deste trabalho, realizando a extração de dados dos boletins de serviço.

3.5 Flask

O Flask é um framework, também encontrado em algumas literaturas como um microframework. O fato do Flask ser um microframework ajuda, por ser leve. Para Flask (2019), "O 'micro' em microframework significa que o Flask visa manter o núcleo simples, mas extensível". Ele é baseado em Werkzeug e Jinja2. Nele, há uma gama de bibliotecas disponíveis, facilitando assim, a utilização para quem deseja programar web utilizando a linguagem de programação Python. Para Fragoso (2018), Flask tem a flexibilidade da linguagem de programação Python e dispõe de um modelo simples para o desenvolvimento web. Já Barry diz que "Flask não é tão completo quanto alguns de seus concorrentes, como Django, o pai de todas as estruturas web do Python, mas é pequeno, leve e fácil de usar" (2018, p. 203).

Com ele, foi possível trabalhar com o desenvolvimento Web, uma vez que ele possui em sua essência, rotas, que auxiliam o direcionamento de páginas. Para este trabalho, Flask foi extremamente válido, uma vez que conseguiu através de suas ferramentas, ser útil ao que era necessário para este projeto, por isso, a escolha. Além disso, foi possível trabalhar ao lado do servidor e realizar a persistência dos boletins de maneira rápida com Elasticsearch.

3.6 Elasticsearch

A criação do Elasticsearch foi realizada em 2010 por Shay Banon. A tecnologia se refere a um mecanismo de análise e pesquisa de texto completo de código aberto altamente escalável o qual permite o armazenamento, pesquisa e análise em grandes volumes de dados rapidamente (ELASTICSEARCH, 2019). Em outras palavras, o Elasticsearch tem como objetivo fornecer um método de se catalogar e efetuar buscas em grandes massas de informação por meio de interfaces REST, na qual recebe e fornece informações em formato JSON (LOURENÇO, 2016). O Elasticsearch foi construído em Java e com base no Apache Lucene, no qual o último consiste em um software de busca. Ele é considerado também como uma base de dados orientado a documentos, tornando o manuseio em dados complexos mais fácil. Outro ponto forte do Elasticsearch é devido ao mesmo ser Open Source, ou seja, ele pode ser distribuído gratuitamente por qualquer um e para qualquer finalidade. O Elasticsearch funciona de acordo com um conjunto de propriedades da arquitetura chamado RESTFUL, sendo este, baseado totalmente em chamadas HTTP, seguindo os princípios de REST, utilizando métodos GET, POST, PUT e DELETE, onde GET é utilizado para consultar, POST para inserir, PUT para atualizar e DELETE para deletar (LECHETA, 2015).

Outros conceitos considerados básicos, porém, que são fundamentais para uma boa compreensão da ferramenta, se dizem respeito ao Cluster, o Nó e o Índice. Um Cluster é uma coleção de um ou mais nós, chamados também de servidores, que juntos mantêm seus dados e fornecem recursos de indexação e pesquisa federados em todos os nós. Já um Nó, trata de um servidor único no qual faz parte do cluster, armazenando seus dados e participando dos recursos de indexação e pesquisas do cluster. O Índice é uma coleção de documentos que possuem características semelhantes (ELASTICSEARCH, 2019). Além de todos os detalhes supramencionados, possui a estrutura de Índice Invertido na ferramenta de busca, fazendo com os resultados sejam gerados rapidamente.

Trabalhando com esta ferramenta, juntamente com o microframework Flask e a linguagem de programação Python, foi possível então realizar a persistência, a indexação e as buscas em termos nos Boletins de Serviço do IFG.

3.7 Bootstrap

Visando agilizar o processo de estilização da página web criada, foi utilizado o framework Bootstrap. O Bootstrap é um kit de ferramentas de código aberto para desenvolvimento com HTML, CSS e JS (BOOTSTRAP, 2019).

3.8 Jinja2

Jinja2 é uma linguagem de templates, considerada rápida, moderna e amigável ao designer para Python, modelada a partir dos modelos do Django (JINJA2, 2019). Ela foi utilizada na página HTML denominada index.html para dar acesso aos *request* do Elasticsearch.

CAPÍTULO IV

4. IMPLEMENTAÇÃO

Neste capítulo serão demonstrados os passos que se fizeram necessários para que chegasse ao resultado final, realizando buscas através de um Índice Invertido.

O trabalho seguiu duas vertentes, ambas utilizaram a linguagem de programação Python, porém:

1. A primeira, utilizou-se da linguagem para a criação de um método representando um Índice Invertido e posteriormente, a implementação de um método de busca.
2. Enquanto na outra, a implementação se fez voltada para o desenvolvimento Web, utilizando do microframework Flask e também, de um servidor de buscas que além de outras carac funciona como banco de dados NOSQL (Not Only SQL, ou, traduzindo, Não Somente SQL) chamado Elasticsearch.

A princípio, algumas bibliotecas se fizeram necessárias para as duas soluções. Sendo assim, há a necessidade de realizar os seguintes passos tanto para implementar o Índice Invertido como também, utilizar o Elasticsearch. Os próximos tópicos irão demonstrar como implementar tanto a primeira versão, onde criamos o Índice Invertido de fato, quanto a segunda, utilizando o Elasticsearch, onde o Índice Invertido é embutido na própria ferramenta.

4.1 Download de Bibliotecas Específicas do Python

Para fazer uso de algumas bibliotecas do Python, é necessário primeiramente, fazer o download delas. Este trabalho foi desenvolvido no ambiente Linux utilizando o sistema operacional Ubuntu v. 18.04 devido a facilidade em encontrar bibliotecas compatíveis com o sistema, onde, ao mesmo tempo, eram essenciais para a implementação da solução.

Alguns downloads de bibliotecas foram feitos através do sistema de gerenciamento de pacotes chamado Pip e outros do Conda.

4.2 Método de Extração de Dados

Neste primeiro momento, a intenção foi extrair os dados de cada Boletim de Serviço. Estes dados são salvos em uma lista denominada documentosPDF, onde contém em cada índice, os dados de uma página do boletim, o número da página, o número do documento

(para que seja possível se localizar em cada documento salvo) e por fim, um contador, para que no final, possamos vincular o resultado ao link que dá acesso ao boletim na integra.

4.3 Método para a Criação do Índice Invertido

Com os boletins de serviço inseridos em uma lista denominada documentosPDF podemos iniciar o processo da criação do Índice Invertido, para que posteriormente, a busca de maneira rápida, possa ocorrer. A solução utilizou-se da programação orientada a objetos, por isso, a classe IndiceInvertido foi a responsável por conter o método criarIndiceInvertido(). A princípio, é inserido um laço de repetição a iteração ocorre por cada item contido na lista documentosPDF. É aproveitado este momento, para que seja feita a conversão de tipo dos dados dos boletins de serviço, de STR para spacy.tokens.doc.Doc, assim, é possível utilizar métodos da biblioteca Spacy, como o tokenização e também, o método start. Após esta conversão, outro laço de repetição é utilizado, para que desta vez, ele percorra cada boletim contido na lista boletinsNlp. Neste momento, inicia-se o processo de inserção de dados no Índice Invertido, sendo este, representado por um dicionário. A estrutura de dados dicionário é composta por pares denominados chave-valor. Neste contexto, as chaves foram utilizadas para armazenar os tokens, e os valores, para armazenar um id, o número da página e também, o número do documento. Antes das inserções, é feito também, uma análise em cada token, utilizando o método is_stop. Sendo assim, se o token verificado não fizer parte da lista de stop words, ele será inserido no dicionário. Em seguida, é feita a última verificação utilizando a estrutura condicional if. Caso o token já fizer parte de uma chave no dicionário, é inserido nesta chave apenas os valores Id, número de página, número do documento e o índice do token no documento. Caso contrário, é criada uma nova chave e os valores inerentes à chave são inseridos no valor. Com isso, o Índice Invertido está pronto para uso.

4.4 Método de Busca

A classe Busca possui o método buscar. Para realizar a chamada do método é necessário inserir três parâmetros: O dicionário no qual contém o Índice Invertido, a lista de documentos chamados de documentosPDF e também, a lista onde contém os caminhos dos boletins salvos no sistema operacional.

Criou-se uma variável chamada busca do tipo STR. Uma estrutura condicional foi utilizada para verificar se o vocabulário inserido na variável busca está inserido em alguma chave do dicionário.

Se sim, outra variável de nome resultado é criada para receber os resultados obtidos através do Índice Invertido. O valor armazenado na variável resultado é/são o(s) valor(es) pertinentes à chave do Índice Invertido. Para mostrar a quantidade de resultados encontrados, é feita outra verificação, onde, a partir do método len(), podemos contar quantos valores foram encontrados na chave do Índice Invertido. Após isso, é utilizado um laço de repetição para iterar sobre cada valor retornado. Uma nova variável é criada e recebe o documento do índice [0][0] transformando-a para o tipo spacy.tokens.doc.Doc. Outro laço de repetição é criado para que seja possível então, mostrar um snippet a partir do vocabulário buscado. Com isso, a informação do local da chave é utilizada. A informação do índice do arquivo e a da página também são utilizadas e demonstradas nos resultados da busca. O método stdout.write pertencente a biblioteca sys e foi utilizado para escrever pois ao contrário do print, ele não realiza a quebra de linha ao final da execução. Com isso, os resultados apareceram de forma mais elegante.

Caso o vocabulário buscado não esteja localizado em nenhuma chave, uma mensagem é retornada informando que não foi encontrado nenhuma chave contendo o vocabulário buscado.

4.5 Etapa 2: Criação da Interface Gráfica para o Usuário

Neste momento, será apresentado a elaboração da solução indicada para o usuário final, onde, foi elaborado uma Interface Gráfica utilizando-se das tecnologias Flask e Bootstrap, com o apoio do Elasticsearch na realização das buscas.

O tópico 4.2 demonstrou como realizar o método de extração de dados. Este método será reaproveitado, uma vez que precisaremos de todas aquelas informações que são extraídas deste método. Antes de seguir para o método que realiza a inserção de documentos/boletins no Elasticsearch, é necessário realizar a instalação do próprio Elasticsearch. O tópico 4.6 irá demonstrar como fazer isso.

4.6 Elasticsearch

Essa segunda etapa do projeto consistiu em determinar de maneira amigável uma forma para que os usuários finais pudessem também, trabalhar com a solução, de modo que a busca nos boletins de serviço se tornasse viável, uma vez que os boletins pudessem ser armazenados em algum banco de dados. Como os dados contidos nos boletins estão dispostos de maneira não estruturada, a tecnologia Elasticsearch se mostrou bastante útil e eficaz tanto na persistência dos dados como também nas buscas em si. A versão do Elasticsearch utilizada

neste trabalho foi a 6.6.0. Sendo assim, caso haja interesse em implementar esta solução, é interessante, que os testes sejam feitos com esta versão do Elasticsearch.

A instalação do Elasticsearch funciona de maneira tranquila. No site oficial, a empresa disponibiliza uma quantidade grande de documentação. Com isso, a maioria das dúvidas pertinentes à tecnologia foram sanadas a partir da utilização da documentação disponível no próprio site do Elasticsearch.

Como mencionado anteriormente, a solução se fez na IDE Spyder, sendo esta, instalada no Anaconda, que por sua vez, foi instalado no sistema operacional Ubuntu v. 18.04. Sendo assim, ao navegar pelo site do Elasticsearch, foi necessário ir até a aba Downloads e localizar o download referente a versão para Linux.

Após finalizado o download do arquivo é necessário inserir os seguintes comandos no terminal:

`tar xzvf elasticsearch-6.6.0.tar.gz` (extrai, manipula e descompacta o arquivo)

`./bin/elasticsearch` (executa o elasticsearch em primeiro plano)

Para saber se a instalação foi realizada com sucesso e também se o Elasticsearch está funcionando como esperado é necessário ir até um browser e digitar na URL: localhost:9200
Se o resultado for parecido com a figura 7, o seu servidor de buscas está pronto para uso.

Figura 7 - Demonstração do Servidor de Buscas Pronto para Uso


```
{
  "name" : "primeirono",
  "cluster_name" : "camillabim",
  "cluster_uuid" : "gI22B4uOQuuUwPA84GkXbg",
  "version" : {
    "number" : "6.6.0",
    "build_flavor" : "default",
    "build_type" : "tar",
    "build_hash" : "a9861f4",
    "build_date" : "2019-01-24T11:27:09.439740Z",
    "build_snapshot" : false,
    "lucene_version" : "7.6.0",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
}
```


Fonte: (Elaborado pela autora)

Para utilizar do Elasticsearch no código fonte é necessário importar a biblioteca.

```
from elasticsearch import Elasticsearch
```

4.7 Bootstrap

Para fazer uso do framework Bootstrap é necessário ir até o site do Bootstrap, clicar no botão *Download* e inserir os seguintes códigos na estrutura do código HTML no qual fornece a versão em cache do CSS e JS compilados do Bootstrap. Os links que estão disponíveis no site foram citados abaixo para facilitar a implementação.

```
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/
css/bootstrap.min.css" integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRx
T2MZw1T" crossorigin="anonymous">


<script src="https://code.jquery.com/jquery-
3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1
Pi6jizo" crossorigin="anonymous"></script>


<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.1
4.7/umd/popper.min.js" integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dI
HNDz0W1" crossorigin="anonymous"></script>


<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/j
s/bootstrap.min.js" integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM
+B07jRM" crossorigin="anonymous"></script>
```

Existem outras maneiras de fazer uso do Bootstrap, porém, esta mostrou-se bastante prática e eficiente.

4.8 Flask

Para implementar o microframework Flask é necessário que primeiramente a instalação seja feita:

```
pip install Flask
```

Após a instalação, o microframework estará pronto para uso, sendo necessário apenas importar o módulo na hora da codificação. Para isso, no código fonte a importação foi feita da seguinte maneira:

```
from flask import Flask, render_template, request
```

Assim, apenas os métodos que foram utilizados, foram importados.

4.9 Método para Inserção de Dados no Elasticsearch

O método inserindoEs localizado na classe Banco de Dados, foi o responsável por inserir os dados no Elasticsearch, para que a busca fosse possível posteriormente.
Com a biblioteca já importada, é necessário criar uma instância. Instância essa chamada de es. Após criada, um laço for irá percorrer por todos os documentos salvos na lista documentosPDF. Dentro do laço, crio uma variável do tipo dict, na qual servida para armazenar os dados e deixá-los de acordo com o que o Elasticsearch armazena, neste caso, são documentos no formato JSON. Neste dict, é armazenado dados de uma página do boletim, o número da página e também, o link para ter acesso ao boletim. A indexação do dicionário é feita e é armazenado no Elasticsearch. O dicionário por sua vez é limpo, deixando-o pronto para uma nova iteração e consequentemente um novo armazenamento. Com isso, os dados dos boletins são salvos no Elasticsearch, restando apenas, a implementação das rotas do Flask e também, da página index na qual os usuários terão acesso.

O próximo tópico irá demonstrar como foi a implementação da página HTML index.

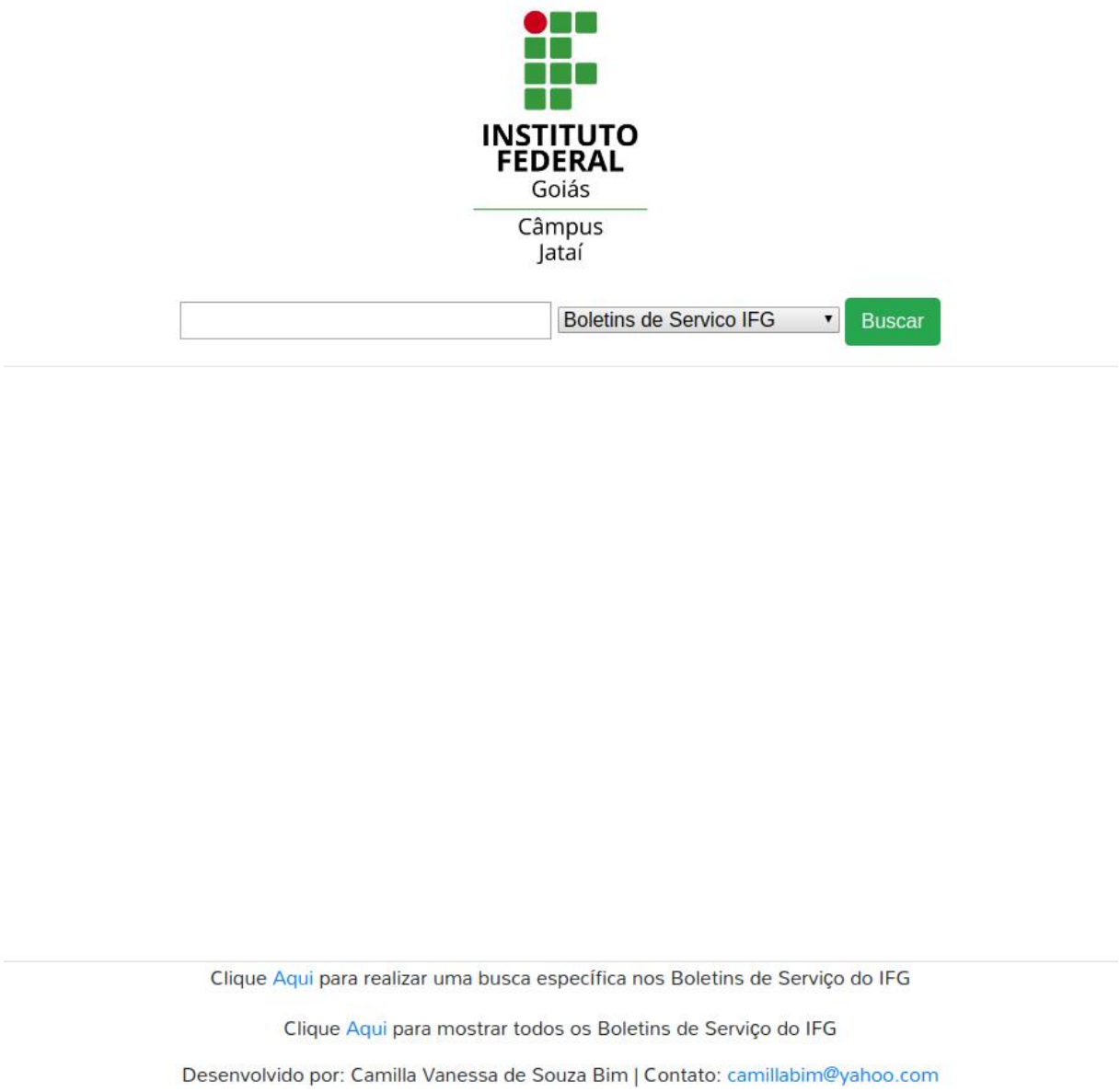
4.10 Implementação da Página index.html

Com as devidas marcações da linguagem HTML inseridas, o Jinja2 foi utilizado para . Com um laço de repetição, primeiramente foi verificado se o request da rota era nulo. Caso fosse, o retorno seria apenas a própria página index.html, caso contrário, algumas ações seriam realizadas. A primeira, seria retornar a quantidade de resultados encontrados a partir da utilização de uma estrutura condicional. Após isso, outro laço de repetição for foi feito, para que fosse possível então, iterar por cada documento retornado pelo Elasticsearch. Com isso, informações como o número da ocorrência encontrada, um link para ter acesso ao boletim na íntegra, o número da página encontrada, o termo, e também um snippet contendo a página do boletim são retornados para o usuário.

4.11 Implementação das rotas com Flask

Nesta etapa, primeiramente, importamos a biblioteca Flask. Em seguida, uma instância da classe Flask é criada, para isso, um argumento é inserido, neste caso, o nome do módulo ou do aplicativo. Após isso, o route() é utilizado para informar ao Flask qual URL deve chamar nossa função. A rota escolhida foi a / e a função recebeu o nome de index. Dentro da função, uma variável de nome busca foi criada para receber o valor advindo do formulário localizado na index.html. Com uma estrutura condicional é verificado se a variável busca é nula ou não. Caso seja, é retornado apenas o template index.html, caso contrário, uma busca é feita utilizando a linguagem própria do Elasticsearch. Ao executar o arquivo app.py o método run irá iniciar o servidor na porta 5000 e o aplicativo estará pronto para uso. A figura 8 demonstra como ficou a página inicial index.html.

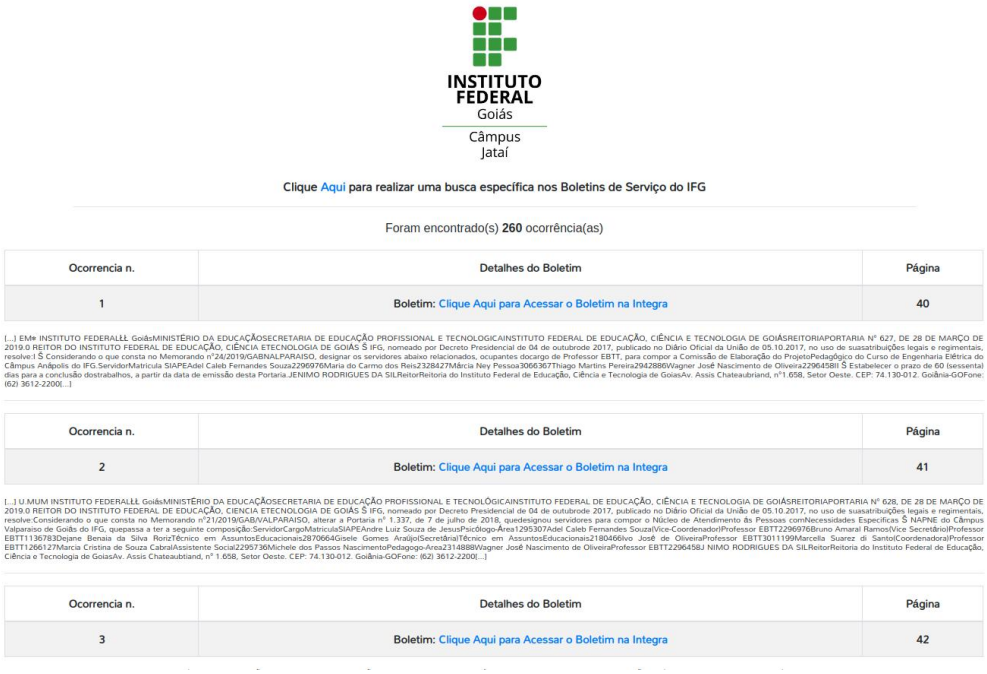
Figura 8 – Documento HTML: index.html



Fonte: (Elaborado pela autora)

A outra função, na qual utiliza a rota /buscartodos, também possui uma variável busca. Caso a variável busca seja nula, a função irá retornar todos os documentos contidos no Banco de Dados e irá redirecionar para a página buscarTodos.html, conforme demonstra figura 9.

Figura 9 – Documento HTML: buscarTodos.html



Fonte: (Elaborado pela autora)

É importante ressaltar, que, para que as buscas ocorram, é necessário iniciar também o Elasticsearch. Como mencionado anteriormente, este, utiliza a porta 9200.

CAPÍTULO V

5. TESTES E AVALIAÇÕES

Testes visando a eficácia desta solução foram realizados. Em ambas as situações, foram utilizados a quantidade de 5 Boletins de Serviço no formato PDF, totalizando o equivalente ao tamanho de 7,4 MB. O termo escolhido para a busca foi: julho

Com a primeira solução na qual se fez utilizando Flask e Elasticsearch, foram obtidos os seguintes resultados:

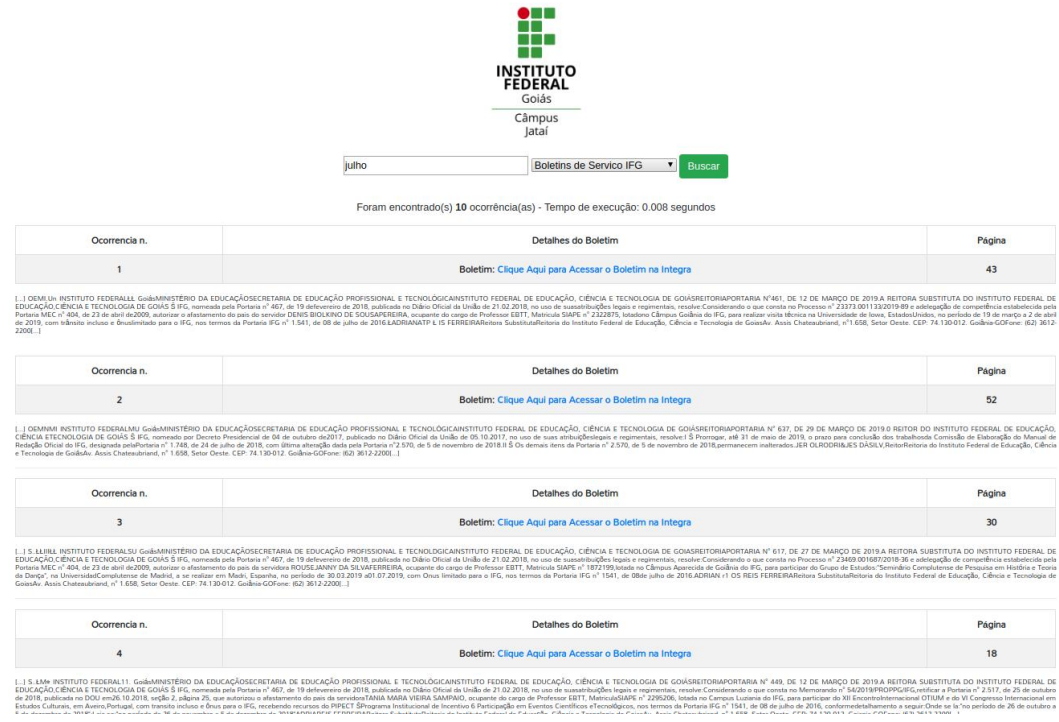
Tabela 1 - Medindo a Eficiência (1ª Solução)

Tempo de Indexação (segundos)	20.655
Latência da Consulta (segundos)	0,008
Taxa de Transferência da Consulta	125
Links Preciso	10
Precisão nos Retornos	10
Quantidade de Retornos	10

Fonte: (Elaborada pela Autora)

A figura 10 representa os resultados demonstrados na tabela 1.

Figura 10 – Print Screen da Tela do Sistema de Buscas



Fonte: (Elaborado pela autora)

Com este mesmo termo de busca, foram realizados testes na segunda solução onde foi implementado um Índice Invertido utilizando a linguagem de programação Python, conforme demonstra resultados na tabela 2.

Tabela 2 - Medindo Eficiência (2ª Solução)

Tempo de Indexação (segundos)	19.605
Latência da Consulta (segundos)	0,3
Taxa de Transferência da Consulta	3,333
Links Preciso	10
Precisão nos Retornos	10
Quantidade de Retornos	10

Fonte: (Elaborado pela autora)

A figura 11 demonstra como aconteceu a execução da segunda solução.

Figura 11 – Print Screen dos resultados obtidos através da solução na qual implementa um Índice Invertido com Python

```
Qual entidade deseja buscar?: julho
Este elemento está contido..
Resultados:
foram encontradas 10 ocorrências
Ocorrência encontrada n.1:
 [ ... ] julho de 2016 , conformedetalhamento a seguir : Onde se [ ... ]
Encontrado no documento Link: /home/camilla/Documentos/ProjetoIndiceInvertidoPythonVersaoFinal/boletins/aaportariasmarçoterceirasemana2019.PDF na página: 18

Ocorrência encontrada n.2:
 [ ... ] julho de 2016.ŁADRIANATP Ł IS FERREIRAReitora SubstitutaReitoria do Instituto Federal [ ... ]
Encontrado no documento Link: /home/camilla/Documentos/ProjetoIndiceInvertidoPythonVersaoFinal/boletins/aaportariasmarçoterceirasemana2019.PDF na página: 43

Ocorrência encontrada n.3:
 [ ... ] julho de 2016.ADRIAN r1 OS REIS FERREIRAReitora SubstitutaReitoria do Instituto [ ... ]
Encontrado no documento Link: /home/camilla/Documentos/ProjetoIndiceInvertidoPythonVersaoFinal/boletins/aaportariasmarçoquintasemana2019.pdf na página: 30

Ocorrência encontrada n.4:
 [ ... ] julho de 2018 , quedesignou servidores para compor o Núcleo [ ... ]
Encontrado no documento Link: /home/camilla/Documentos/ProjetoIndiceInvertidoPythonVersaoFinal/boletins/aaportariasmarçoquintasemana2019.pdf na página: 41

Ocorrência encontrada n.5:
 [ ... ] julho de 2018 , com última alteração dada pela Portaria [ ... ]
Encontrado no documento Link: /home/camilla/Documentos/ProjetoIndiceInvertidoPythonVersaoFinal/boletins/aaportariasmarçoquintasemana2019.pdf na página: 52

Ocorrência encontrada n.6:
 [ ... ] julho de 2016.ONIMO RODRIGUES DA SILVReitorReitoria do Instituto Federal de [ ... ]
Encontrado no documento Link: /home/camilla/Documentos/ProjetoIndiceInvertidoPythonVersaoFinal/boletins/aaportariasmarçosegundasemana2019.PDF na página: 6

Ocorrência encontrada n.7:
 [ ... ] julho de 2015,atualizar a composição do Conselho Superior ( Consup [ ... ]
Encontrado no documento Link: /home/camilla/Documentos/ProjetoIndiceInvertidoPythonVersaoFinal/boletins/aaportariasmarçoquartasemana.pdf na página: 31

Ocorrência encontrada n.8:
 [ ... ] julho de 2019.ERiIMthODRÍÕTJES DA SILVReitorReitoria do Instituto Federal de Educação [ ... ]
Encontrado no documento Link: /home/camilla/Documentos/ProjetoIndiceInvertidoPythonVersaoFinal/boletins/aaportariasmarçoquartasemana.pdf na página: 33

Ocorrência encontrada n.9:
 [ ... ] julho de 2016.JERONIMO RODRIGUES DA SILVReitorReitoria do Instituto Federal de [ ... ]
Encontrado no documento Link: /home/camilla/Documentos/ProjetoIndiceInvertidoPythonVersaoFinal/boletins/aaportariasmarçoquartasemana.pdf na página: 35

Ocorrência encontrada n.10:
 [ ... ] julho de 2016.ÔNIMO RODRIGUES DA SILV.ReitorReitoria do Instituto Federal de [ ... ]
Encontrado no documento Link: /home/camilla/Documentos/ProjetoIndiceInvertidoPythonVersaoFinal/boletins/aaportariasmarçoquartasemana.pdf na página: 36

Tempo de execução: 0.3427566999998817 segundos
```

Fonte: (Elaborado pela autora)

É possível perceber que a primeira solução se sobressaiu na métrica Latência da Consulta e consequentemente na métrica Taxa de Transferência da Consulta. Devido a primeira solução ser mais indicada para o usuário final, esta conseguiu trazer um resultado satisfatório, onde indica que é possível trazer até 125 resultados por segundo.

A tabela 3 indica características pertinentes a cada solução e suas distinções.

Tabela 3 - Principais Diferenças Resultantes de Cada Solução

Tipo de Tecnologia	Amostra	Características	Tempo para Armazenamento/ Tempo de Indexação	Latência da Consulta	Retorno / Request	Armazenamento
Solução 1: Buscas utilizando Elasticsearch e Linguagem de Programação Python com Flask	5 Boletins de Serviço (no formato pdf, equivalente a 260 páginas) - tamanho: 7,4 MB	Utiliza em sua essência Índice Invertido; Sistema Web; Banco de Dados Orientado a Documentos	20,655 segundos	0,008 segundos	Retorna uma página que contém o vocabulário inserido no campo Busca	Elasticsearch
Solução 2: Buscas utilizando Índice Invertido com Linguagem de Programação Python	5 Boletins de Serviço (no formato pdf, equivalente a 260 páginas) - tamanho: 7,4 MB	Utiliza em sua essência Índice Invertido	19,605 segundos	0,3 segundos	Retorna um snippet contendo o vocabulário utilizado para busca	Memória RAM

Fonte: (Elaborado pela autora)

Posto isso, é interessante dizer que cada solução possui características que apontam para qual intuito cada uma é indicada. A primeira, sendo mais indicada ao usuário final, ou seja, aos servidores do Instituto Federal de Goiás, enquanto a segunda, é mais apropriada para interessados que desejam saber como é implementado um método de Índice Invertido e também, um método de Busca.

CAPÍTULO VI

6. CONCLUSÕES E TRABALHOS FUTUROS

6.1 Conclusão

A busca por melhorias, sejam elas no âmbito tecnológico ou não, fazem parte do cotidiano de empresas, instituições e pessoas. Os ambientes que souberem utilizar a tecnologia a seu favor, poderão se destacar dentre outros do mesmo segmento, fazendo com que processos que antes eram considerados maçantes, se tornem processos automatizados e eficientes. Com esta visão, o trabalho buscou solucionar o problema relacionado a busca de informações do Instituto Federal de Goiás.

Dados são gerados dia após dia e implementar algo visando a busca por informações de maneira rápida, pode influenciar positivamente o cotidiano de cada um dos stakeholders. Com isso, buscou-se apropriar-se de conhecimento acerca da área de PLN e também, R.I., afim de utilizar suas técnicas e métodos. Com isso, este trabalho tomou duas vertentes para pesquisa, uma direcionada ao entendimento de como funciona um processo de busca e implementando-o e outra, visando construir uma solução que pudesse então, se tornar útil aos servidores do Instituto Federal de Goiás. Em ambos as situações, a essência de PLN e RI estavam presentes, sendo estes, essenciais para que o trabalho fosse realizado.

Apesar das dificuldades que foram aparecendo, posso dizer que a grande totalidade da solução desenvolvida, trouxe uma vasta gama de novas informações, fazendo-me perceber o quão amplo é a área da Tecnologia da Informação. Espero que novos trabalhos, relacionados a este, sejam desenvolvidos e tragam cada vez mais novidades tecnológicas.

6.2 Trabalhos Futuros

A área de PLN e RI é muito ampla sendo possível seguir várias direções para execução de uma pesquisa. Para trabalhos futuros, o autor poderá solucionar o problema já relatado no projeto, onde percebeu-se que alguns boletins eram compostos de imagens, de modo que ficamos impossibilitados de extrair termos utilizando a biblioteca PyPDF2. Há algumas bibliotecas que fazem o trabalho de extração de termos em imagens, como a Tesseract, porém, devido ao grande tempo de processamento, acabou se tornando um trabalho inviável neste momento. Como a maioria dos boletins são formados por arquivos pdf, e os

mais atuais também, sendo estes, passíveis de realizar a extração de termos, foi resolvido que daríamos atenção a um projeto que desenvolvesse a busca em boletins de serviço no formato pdf. Posto isso, a indicação é que seja desenvolvido algo no sentido de realizar buscas em imagens que contenham textos e talvez, utilizar outro método de busca.

APÊNDICE A

Implementação da solução tecnológica utilizando Elasticsearch

Classe Boletim – Arquivo: boletim.py

```
import os
import PyPDF2

class Boletim():
    def __init__(self):
        self.caminho =
'/home/camilla/Documentos/ProjetoIndiceInvertidoPythonVersaoFinal/bo
letins/'
        self.diretorioBoletins = os.listdir(self.caminho)
        self.listaBoletins = []
        self.documentosPDF = []

    def criarListaBoletins(self):
        print("... Inserindo na listaBoletins os caminhos de cada
boletim contido no diretório ...")
        for i in self.diretorioBoletins:
            self.listaBoletins.append(self.caminho + i)

    def extracaoDados(self):
        contador = 0
        for numDoc, caminho in enumerate(self.listaBoletins):
            arquivo = open(caminho, 'rb')
            arquivoPdf = PyPDF2.PdfFileReader(arquivo)
            for numPagina in range(0, arquivoPdf.getNumPages()):
                print("NumPage: {} | numDoc: {} | Id:
{}".format(numPagina, numDoc, contador))

self.documentosPDF.append([arquivoPdf.getPage(numPagina).extractText
(), numPagina, numDoc, contador])
                contador +=1
```

Classe BancoDeDados – Arquivo: bdes.py

```
from elasticsearch import Elasticsearch
from boletim import Boletim
import time

class BancoDeDados():
    def __init__(self):
        self.es = Elasticsearch()
        self.b = Boletim()

    def inserindoEs(self):
        inicio = time.time()
        for num, boletim in enumerate(b.documentosPDF):
            dict = {}
            dict = {"Boletim": b.documentosPDF[num][0], "Pagina":
b.documentosPDF[num][1],
"Link":b.listaBoletins[b.documentosPDF[num][2]]}
            self.es.index(index='colecao', doc_type='boletim', id=num,
body=dict)
            print("Inserido doc: {} em Elasticsearch com
Sucesso".format(num))
            dict.clear()
        fim = time.time()
        print("Tempo para armazenamento: {}".format(fim-inicio))
```

Documento Flask (Rotas) – Arquivo: app.py

```
from flask import Flask, render_template, request
from elasticsearch import Elasticsearch
import timeit

es = Elasticsearch()
app = Flask('buscasNosBoletins')

@app.route('/', methods=["GET", "POST"])
def index():
    busca = request.args.get("busca")
    if busca is not None:
        inicio = timeit.default_timer()
        resp = es.search(index='colecao', doc_type='boletim', body={
  "size": 500,
    "query" : {
        "match_phrase" : {
            "Boletim" : busca
        }
    }
})
        fim = timeit.default_timer()
        tempoExecucao = (fim-inicio)
        t = round(tempoExecucao, 3)
```

```
        return render_template("index.html", busca=busca,
response=resp, tempo = t)
    return render_template("index.html")

@app.route('/buscartodos')
def buscarTodos():
    busca = request.args.get("busca")
    if busca == None:
        resp = es.search(index='colecao', doc_type='boletim',
body={"size": 10000,"query": {"match_all":{}}})
    else:
        resp = es.search(index='colecao', doc_type='boletim', body={
  "size": 500,
    "query" : {
        "match_phrase" : {
            "Boletim" : busca
        }
    }
})
    return render_template("buscarTodos.html", title='Buscas Boletim',
response=resp)

if __name__ == "__main__":
    app.run(debug=True, port=5000)
```

Documento HTML – Arquivo: buscarTodos.html

```
<!doctype html>
<html lang="pt">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1, shrink-to-fit=no">
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstr
ap.min.css"
    integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
    <link
href="https://fonts.googleapis.com/css?family=Mandali&display=swap"
rel="stylesheet">
    <link
href="https://fonts.googleapis.com/css?family=Mandali&display=swap"
rel="stylesheet">

    <style>
        footer{
          width: 100%;
          height: 28px;
          border-top: 1px solid #E0E0E0;
          bottom: 0px;
```

```
    left: 0px;
    text-align: center;
    font-family: 'Mandali', sans-serif;

}
p.response{
    max-width: 150ch;
    overflow: scroll;
    text-overflow: clip;
    white-space: normal;
    font-weight: bold;
    }
em{
    color: red;
}
h6{
    font-size: 8pt;
    text-align: justify;
    font-family: 'Mandali', sans-serif;
}
h5{
    font-size: 12pt;
    font-family: 'Mandali', sans-serif;
    text-align: center;

}
td{
    font-size: 11pt;
    font-family: 'Mandali', sans-serif;
    text-align: center;
              font-weight: bold;


}
button{
    font-size: 10pt;
    text-align: justify;
    font-family: 'Mandali', sans-serif;

}
em{
    color: red;
}
html {
    height: 90%;
    min-height: 90%;
}

body {
    display: flex;
    flex-direction: column;
    min-height: 100%;
}

footer {
    margin-top: auto;
```

```
       }

    </style>
    <title>Buscando Todos - Buscas em Boletins de Serviço utilizando
NLP</title>
  </head>
  <body>

     <div class="container">
             <div class="text-center">
                <img src="{{url_for('static',
filename='logo_ifg.png')}}" width="180px"/>
             </div>

<h5><b>Clique<a href="/"> Aqui<a/> para realizar uma busca
específica nos Boletins de Serviço do IFG</h5></b>

                <hr>
             </form>

       </div>
    {% if response == null %}
       {{response}}
    {% else %}
       <div class="container text-center">
          {% if response.hits.total > 0 %}
             <p>Foram encontrado(s) <b>{{response.hits.total}}</b>
ocorrência(as)</p>
          {% else %}
             <b><p>Não foram encontrado documentos, por favor,
refaça sua busca</p></b>
          {% endif %}
        </div>

          <div class="col-md-10 container-fluid center text-
justify">
             {% for i in response.hits.hits %}
                <table id="tabelaResultados" class="table table-
bordered table-striped">
                <thead>
                   <tr>
                      <td>Ocorrencia n.</td>
                      <td>Detalhes do Boletim</td>
                      <td>Página</td>
                   </tr>
                </thead>
                   <tbody>
                      <td>{{loop.index}}</td>
                      <td>Boletim: <a
href="file:\\\{{i._source.Link}}#page={{i._source.Pagina}}"
                      target="_blank">Clique Aqui para Acessar o
Boletim na Integra</a></td>
                      <td>{{i._source.Pagina}}</td>

                   </tbody>
```

```
                </table>
                <h6><td>[...] {{i._source.Boletim}}[...]</td></h6>
                <hr>
            {% endfor %}
          </div>
          {% endif %}

     <footer>
     <p>Clique<a href="/"> Aqui<a/> para realizar uma busca
específica nos Boletins de Serviço do IFG</p>
     <p>Clique <a href="/buscartodos"
value="AQUI"name="buscartodos">Aqui</a>
     para mostrar todos os Boletins de Serviço do IFG</p>
     <h5>Desenvolvido por: Camilla Vanessa de Souza Bim | Contato:
     <a href="mailto:camillabim@yahoo.com">
camillabim@yahoo.com</a></h5>
     </footer>

    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
    integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
    crossorigin="anonymous"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/pop
per.min.js"
    integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
    crossorigin="anonymous"></script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.
min.js"
    integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
    crossorigin="anonymous"></script>
  </body>
</html>
```

Documento HTML – Arquivo: index.html

```
<!doctype html>
<html lang="pt">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1, shrink-to-fit=no">
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstr
ap.min.css"
          integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
    <link
href="https://fonts.googleapis.com/css?family=Mandali&display=swap"
```

```
rel="stylesheet">
    <style>
        footer{
           width: 100%;
           height: 28px;
           border-top: 1px solid #E0E0E0;
           bottom: 0px;
           left: 0px;
           text-align: center;
           font-family: 'Mandali', sans-serif;

       }
       p.response{
           max-width: 150ch;
           overflow: scroll;
           text-overflow: clip;
           white-space: normal;
           font-weight: bold;
           }
       em{
           color: red;
       }
       h6{
           font-size: 8pt;
           text-align: justify;
           font-family: 'Mandali', sans-serif;
       }
       h5{
           font-size: 12pt;
           font-family: 'Mandali', sans-serif;
           text-align: center;

       }
       td{
           font-size: 11pt;
           font-family: 'Mandali', sans-serif;
           text-align: center;
                     font-weight: bold;


       }
       button{
           font-size: 10pt;
           text-align: justify;
           font-family: 'Mandali', sans-serif;

       }
       em{
           color: red;
       }
       html {
           height: 90%;
           min-height: 90%;
       }

       body {
```

```
            display: flex;
            flex-direction: column;
            min-height: 100%;
        }

        footer {
            margin-top: auto;
        }

    </style>
    <title>Inicial - Buscas em Boletins de Serviço utilizando
NLP</title>
  </head>
  <body>

      <div class="container">
              <div class="text-center">
                  <img src="{{url_for('static',
filename='logo_ifg.png')}}" width="180px"/>
              </div>

              <form action"/" method="get" autocomplete="off">
                  <div class="col-md-12 text-center">
                      <input type="teste" name="busca" size=30
role="textbox" value="{{busca}}"/>
                      <select name="escopo">
                          <option value="0" disabled>Selecionar escopo
de busca</option>
                          <option value="1" selected>Boletins de
Servico IFG</option>
                      </select>
                      <input type="submit" value="Buscar" class="btn
btn-success">
                  </div>

                  <hr>
              </form>

        </div>
    {% if response == null %}
        {{response}}
    {% else %}
        <div class="container text-center">
            {% if response.hits.total > 0 %}
                <p>Foram encontrado(s) <b>{{response.hits.total}}</b>
ocorrência(as) -
                    Tempo de execução: {{tempo}} segundos </p>
            {% else %}
                <b><p>Não foram encontrado documentos, por favor,
refaça sua busca</p></b>
            {% endif %}
         </div>

            <div class="col-md-10 container-fluid center text-
justify">
                {% for i in response.hits.hits %}
```

```
                <table id="tabelaResultados" class="table table-
bordered table-striped">
                <thead>
                    <tr>
                        <td>Ocorrencia n.</td>
                        <td>Detalhes do Boletim</td>
                        <td>Página</td>
                    </tr>
                </thead>
                    <tbody>
                        <td>{{loop.index}}</td>
                        <td>Boletim: <a
href="file:\\\{{i._source.Link}}#page={{i._source.Pagina}}"
                                            target="_blank">Clique Aqui para
Acessar o Boletim na Integra</a></td>
                        <td>{{i._source.Pagina}}</td>

                    </tbody>

                </table>
                <h6><td>[...] {{i._source.Boletim}}[...]</td></h6>
                <hr>
            {% endfor %}
          </div>
          {% endif %}

     <footer>
     <p>Clique<a href="/"> Aqui<a/> para realizar uma busca
específica nos Boletins de Serviço do IFG</p>
     <p>Clique <a href="/buscartodos"
value="AQUI"name="buscartodos">Aqui</a> para mostrar todos os
Boletins de Serviço do IFG</p>
     <h5>Desenvolvido por: Camilla Vanessa de Souza Bim | Contato:
<a href="mailto:camillabim@yahoo.com"> camillabim@yahoo.com</a></h5>
     </footer>

   <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
          integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
          crossorigin="anonymous"></script>
   <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/pop
per.min.js"
          integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
          crossorigin="anonymous"></script>
   <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.
min.js"
          integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
          crossorigin="anonymous"></script>
  </body>
</html>
```

APÊNDICE B

Implementação do Índice Invertido Utilizando a Linguagem de Programação Python

Classe Boletim – Arquivo: boletins.py

```
import os
import PyPDF2

class Boletim():
    def __init__(self):
        self.caminho =
'/home/camilla/Documentos/ProjetoIndiceInvertidoPythonVersaoFinal/bo
letins/'
        self.diretorioBoletins = os.listdir(self.caminho)
        self.listaBoletins = []
        self.documentosPDF = []

    def criarListaBoletins(self):
        # INSIRO NA LISTABOLETINS TODOS OS BOLETINS COM O CAMINHO
CORRETO
        print("... Inserindo na listaBoletins os caminhos de cada
boletim contido no diretório ...")
        for i in self.diretorioBoletins:
            self.listaBoletins.append(self.caminho + i)

    def extracaoDados(self):
        contador = 0
        for numDoc, caminho in enumerate(self.listaBoletins):
            arquivo = open(caminho, 'rb')
            arquivoPdf = PyPDF2.PdfFileReader(arquivo)
            for numPagina in range(0, arquivoPdf.getNumPages()):
                print("NumPage: {} | numDoc: {} | Id:
{}".format(numPagina, numDoc, contador))

self.documentosPDF.append([arquivoPdf.getPage(numPagina).extractText
(), numPagina, numDoc, contador])
                contador +=1
```

Classe IndiceInvertido – Arquivo: indiceInvertido.py

```
from boletim import Boletim
import spacy
nlp = spacy.load("pt")

class IndiceInvertido():
    def __init__(self):
        self.b = Boletim()
        self.dicionario = {}
```

```
        self.boletinsNlp = []


    def criarIndiceInvertido(self, docPdf):
        for pagina in docPdf:
            self.boletinsNlp.append(nlp(pagina[0]))
            for boletim in self.boletinsNlp:
                for token in boletim:
                    if nlp.vocab[token.text].is_stop == False:
                        if token.text.lower() in self.dicionario:
                            print("Chave: {} | Id: {} | N. Pag: {} | N.
Doc: {} | Local: {}".format(
                                token.text.lower(), pagina[3],
pagina[1], pagina[2], token.i))

self.dicionario[token.text.lower()].append([pagina[3], pagina[1],
pagina[2], token.i])
                        else:
                            print("Nova Chave: {} | Id: {} | N. Pag: {} |
N. Doc: {} | Local: {}".format(
                                token.text.lower(), pagina[3],
pagina[1], pagina[2], token.i))
                            self.dicionario[token.text.lower()] =
([[pagina[3], pagina[1], pagina[2], token.i]])
                self.boletinsNlp.clear()
```

Classe Busca – Arquivo: busca.py

```
from indiceInvertido import IndiceInvertido
from boletim import Boletim
import sys
import spacy
nlp = spacy.load("pt")
import timeit

class Busca():
    def __init__(self):
        self.indiceInvertido = IndiceInvertido()
        self.b = Boletim()

    def buscar(self, indiceInvertido, documentosPDF, listaBoletins):
        busca = str(input("Qual termo deseja buscar?: "))
        if busca in indiceInvertido:
            inicio = timeit.default_timer()
            print("Este elemento está contido..")
            print("Resultados: ")
            resultado = indiceInvertido[busca]
            if len(resultado) == 1:
                print("Foi encontrada {}
ocorrencia.".format(len(resultado)))
            elif len(resultado) >= 1:
                print("foram encontradas {}
ocorrências".format(len(resultado)))
            for num, res in enumerate(resultado):
                documento = nlp(documentosPDF[resultado[num][0]][0])
```

```
                print("Ocorrência encontrada n.{}: ".format(num+1))
                sys.stdout.write(" [ ... ] ")
                for i in range(10):
                    sys.stdout.write(str(documento[resultado[num][3]+i]))
                    sys.stdout.write("  ")
                print(" [ ... ] ")
                print("Encontrado no documento Link: {} na página: {}
".format(
                        listaBoletins[resultado[num][2]],
resultado[num][1]+1))
                print(" ")
            fim = timeit.default_timer()
            print("Tempo de execução: {} segundos".format(fim-inicio))
        else:
            print("Não foi encontrado nenhuma chave com este termo")
```

main – Arquivo: main.py

```
import boletim
import indiceInvertido
import busca

def main():

    b = boletim.Boletim()
    b.criarListaBoletins()
    b.extracaoDados()

    i = indiceInvertido.IndiceInvertido()
    i.criarIndiceInvertido(b.documentosPDF)

    buscaIndice = busca.Busca()
    buscaIndice.buscar(i.dicionario, b.documentosPDF, b.listaBoletins)


if __name__ == '__main__':
    main()
```

REFERÊNCIA BIBLIOGRÁFICA

AMARAL, F. Introdução à Ciência de Dados: Mineração de Dados e Big Data. Rio de Janeiro: Editora Alta Books, 2016.

ANACONDA. Anaconda. Disponível em: <https://www.anaconda.com/> Acesso em 20 de abr. 2019.

BAEZA-YATES, R. RIBEIRO-NETO, B. Recuperação de Informação: Conceitos e Tecnologia das Máquinas de Busca. 2. ed. Porto Alegre: Editora Bookman, 2013.

BARRY, P. Use a Cabeça! Python. 2. ed. Rio de Janeiro: Editora Alta Books, 2018.

CENTERATTO, G. Fundamentos Linguísticos e Computação. Rio Grande do Sul: Editora Universitária da PUCRS, 2015.

ELASTICSEARCH. Documentação. Disponível em: <https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html> Acesso em 26 de mar. 2019.

FLASK. Prefácio. Disponível em: <http://flask.pocoo.org/docs/1.0/foreword/#what-does-micro-mean> Acesso em 02 de jun. 2019.

FRAGOSO, W. Guia Prático Python & Flask: Aprenda a Criar Aplicações Web usando o Mini Framework mais usado na atualidade. DevAcademy, 2018. Não paginada

GRUS, J. Data Science do Zero: Primeiras Regras com Python - Rio de Janeiro: Editora Alta Books, 2016.

JINJA2. Bem Vindo ao Jinja2. Disponível em: <http://jinja.pocoo.org/docs/2.10/> Acesso em 02 de jun. de 2019.

LECHETA, R. R. Web Services RESTful: Aprenda a criar web services RESTful em Java na nuvem do Google. São Paulo: Editora Novatec, 2015.

LOURENÇO, A. Elasticsearch: Consumindo dados real-time com ELK. [S.I.]. Editora Casa do Código, 2016.

MARTINS, D. KATAOKA, K. TRINDADE, L. Processamento de Linguagem Natural. Disponível em: <http://homes.dcc.ufba.br/~leotavo/index.html/artigo2.pdf> Acesso em: 13 de jun. 2019.

PEREIRA, E. Trilhas Python: Programação Multiparadigma e desenvolvimento Web com Flask. [S.I.]: Editora Casa do Código, 2018.

PROVOST, F. FAWCETT, T. Data Science para Negócios: O que Você Precisa Saber Sobre Mineração de Dados e Pensamento Analítico de Dados. Editora Alta Books, 2016.

PYPI. Encontre, instale e publique pacotes Python com o Índice de Pacotes Python. Disponível em: <https://pypi.org/> Acesso em 01 de jun. de 2019.

PYTHON. Python. Disponível em: <https://www.python.org/> Acesso em 30 de nov. 2018.

RAMAKRISHNAN, R., GEHRKE, J. Sistemas de Gerenciamento de Banco de Dados. 3. ed. EUA: McGrawHill Bookman, 2011.

ROSA, J. L. G. Fundamentos da Inteligência Artificial. Rio de Janeiro: Editora LTC, 2011.
RODRIGUES, J. O que é Processamento de Linguagem Natural?. Disponível em: <https://medium.com/botsbrasil/o-que-%C3%A9-o-processamento-de-linguagem-natural-49ece9371cff> Acesso em 15 de jun. 2019.

RUBIN, K.S. Scrum Essencial: Um Guia Prático para o Mais Popular Processo Ágil - Rio de Janeiro: Editora Alta Books, 2017.

S.A.S. Como a Inteligência Artificial Funciona?. Disponível em: <https://www.sas.com/pt_br/insights/analytics/inteligencia-artificial.html> Acesso em 14 de dez. 2018.

SMART READ. Inteligência Artificial: Compreender em que Consiste a I.A. e o que Implica a Aprendizagem de Máquinas. Babelcube Books, 2017.

SOMASUNDARAM, G., SHRIVASTAVA, A., EMC EDUCATION SERVICES. Armazenamento e Gerenciamento de Informações: Como armazenar, gerenciar e proteger informações digitais - Rio Grande do Sul: Bookman, 2011.

SPACY. Processamento de Linguagem Natural com Força Industrial. Disponível em: <https://spacy.io/> Acesso em 09 de nov. 2018.

SCHALCH, R. B. Wget: Fazendo Download de Arquivos com Python. Disponível em: <https://rschalch.github.io/wget-download-arquivos-com-python.html> Acesso em 26 de mar. 2019.

SPYDER. Spyder. Disponível em: <https://www.spyder-ide.org/> Acesso em 10 de dez. 2018.

STROSKI, P. N. O que é Processamento de Linguagem Natural? Disponível em: <http://www.electricalelibrary.com/2018/02/09/o-que-e-processamento-de-linguagem-natural/> Acesso em 16 de jun. 2019.

WEBER, A. MELO, N. Data Science em 12 Semanas: Mais Fácil do Que Você Imagina. [S.I.]. Editora DSB Books, 2019.