UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
ESCOLA NACIONAL DE ADMINISTRAÇÃO PÚBLICA – ENAP
ESPECIALIZAÇÃO EM INFORMÁTICA: ÁREA DE CONCENTRAÇÃO: GESTÃO DE TECNOLOGIA DA INFORMAÇÃO

DANIEL DE SOUSA BRASILEIRO ARAUJO

IMPLEMENTAÇÃO DE SISTEMA DE RECUPERAÇÃO DE INFORMAÇÃO PARA O CONLEGIS

Brasília

2019

DANIEL DE SOUSA BRASILEIRO ARAUJO

IMPLEMENTAÇÃO DE SISTEMA DE RECUPERAÇÃO DE INFORMAÇÃO PARA O CONLEGIS

Monografia apresentada ao programa de Pós-Graduação em Informática - Área de Concentração: Gestão de Tecnologia da Informação, do Departamento de Ciência da Computação, da Universidade Federal de Minas Gerais, como requisito para a obtenção do certificado de Especialista em Gestão de Tecnologias da Informação.

Orientador: Rodrygo Luis Teodoro Santos

BRASÍLIA

2019

.

Araujo, Daniel de Sousa Brasileiro.

A663i Implementação de sistema de recuperação de informação para o Conlegis / Daniel de Sousa Brasileiro Araujo – Brasília, 2019.

xiii, 122 f.: il.

Monografia (especialização) – Universidade Federal de Minas Gerais. Departamento de Ciência da Computação.

Orientador: Rodrygo Luis Teodoro Santos

1. Computação – Monografias. 2. Recuperação da informação. 3. Conlegis. I. Orientador. II. Título.

UNIVERSIDADE FEDERAL DE MINAS GERAIS

INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
ESPECIALIZAÇÃO EM INFORMÁTICA: ÁREA DE CONCENTRAÇÃO GESTÃO EM TECNOLOGIA DA INFORMAÇÃO

Implementação de Sistema de Recuperação da Informação para o Conlegis

DANIEL DE SOUSA BRASILEIRO ARAÚJO

Monografia apresentada aos Senhores:

Prof. Rodrygo Luis Teodoro Santos
Orientador
DCC - ICEx – UFMG

Prof. José Nagib Cotrim Arabe
DCC - ICEx - UFMG

Prof. José Marcos Silva Nogueira
DCC - ICEx - UFMG

Belo Horizonte, 14 de março de 2019

DEDICATÓRIA

Dedico este trabalho a minha querida família.

À minha esposa Hanna que me ajudou a concluir o trabalho cuidando dos nossos pequenos enquanto o papai “fugia” para conseguir terminar essa missão. À minha mãe e minha sogra, Dida e Graci, que sempre estiveram presentes e nos ajudaram quando o papai e mamãe precisavam de uma folguinha para recuperar as energias. E aos meus filhos Brisa (4 anos) e Isaac (2 anos) os grandes bagunceiros culpados dessa logística enorme, pois sem eles não teríamos motivos nem inspiração pra tentar sermos melhores e buscar evoluir como profissionais e pessoas, já que fazemos tudo por eles mesmo...

Nós voamos mais alto quando somos unidos.

Amo todos vocês.

Daniel.

RESUMO

Esta monografia apresenta o processo de implementação de um Sistema de Recuperação da Informação - SRI para o Conlegis – aplicação governamental de armazenamento e busca de atos normativos relativos a área de gestão de pessoas da Administração Pública Federal - APF. Devido a necessidade de um mecanismo de busca mais simples e eficiente o Conlegis é um sistema subutilizado pelas unidades de gestão de pessoas. O objetivo do novo SRI Conlegis é a aplicação de técnicas e algoritmos modernos de recuperação de informação que poderão ser, posteriormente, integrados ao Conlegis. Isto permitirá uma melhor utilização do sistema atual e, consequentemente, um aperfeiçoamento dos processos referentes a consulta, disseminação e aplicação das normas de forma padronizada na APF. A partir do conjunto de documentos e da base de dados obtida da área negocial responsável pelo Conlegis, foi realizado o desenvolvimento de um novo mecanismo de busca sobre o motor da aplicação Elasticsearch combinando-o com uma implementação de uma interface web leve e minimalista de simples utilização para o usuário. O SRI Conlegis consolida estratégias de indexação e algoritmos de busca mais modernos e elaborados no intuito de retornar os atos normativos de maior relevância para os usuários conforme os termos de busca solicitados.

Palavras-chave: recuperação; informação; busca; conlegis; elasticsearch.

ABSTRACT

This monograph presents the implementation process of an Information Retrieval System (IRS) to Conlegis – a government application that stores and searches regulatory acts regarding to human resource management (HRM) at the Federal Public Administration (FPA). Because of the need for a more efficient and leaner search mechanism, Conlegis is underutilised by the HRM Departments. The objective of the new Conlegis IRS is the application of modern techniques and algorithms of information retrieval that may be integrated afterwards to the official Conlegis application. This will allow a better use of Conlegis consequently improving the processes of search, dissemination and application of regulatory acts in a standardized form across the FPA. From the corpus of documents and the database obtained along with the responsible area of Conlegis, a new search module was developed using the engine of the Elasticsearch platform combining it with a clean and light user interface. The Conlegis IRS aggregates modern indexing strategies and search algorithms designed to retrieve the most relevant regulatory acts to the user considering the requested search terms.

Keywords: information; retrieval; search; conlegis; elasticsearch.

LISTA DE FIGURAS

LISTA DE CÓDIGOS

SUMÁRIO

1 INTRODUÇÃO

Considerando a rápida evolução que todo o ecossistema de tecnologias relacionadas ao acesso, uso e distribuição da informação tiveram desde o surgimento do computador pessoal, e potencializada ainda mais com o surgimento da Internet, ficou aparente a necessidade de desenvolvimento de técnicas, algoritmos e aplicações mais eficientes para realizar o armazenamento, tratamento, análise e recuperação de informação.

A partir da popularização de grandes sites de busca e do crescente investimento de suas empresas no desenvolvimento de melhores formas de atender aos usuários, os famosos “motores de busca”, conceitualmente Sistemas de Recuperação de Informação, se aperfeiçoaram e passaram por um enorme avanço tecnológico na última década.

Este estudo se propõe em implementar um Sistema de Recuperação de Informação mais aderente às novas tendências e tecnologias para o Sistema Conlegis (1) que ainda possui um mecanismo baseado no modelo tradicional de consultas a bancos de dados.

1.1 Problema

Todas as unidades de recursos humanos dos órgãos e entidades da Administração Pública Federal devem estar sempre uníssonas e padronizadas no que se refere aos seus processos operacionais, atendimento aos servidores, esclarecimento de dúvidas e respostas a requerimentos formais. Portanto, todas estas ações devem ser feitas com completa aderência ao arcabouço legislativo vigente.

Para que isso ocorra, foi desenvolvido o Conlegis no intuito de servir de principal fonte de consulta às normas relativas aos recursos humanos e gestão de pessoas. Entretanto o Conlegis atualmente não é tão utilizado pelas unidades quanto deveria, pois, seu mecanismo de busca, além de nem sempre retornar os atos normativos necessários, é considerado muito complexo, de difícil utilização e de interface confusa.

A solução atual das áreas é a realização de buscas externas em outros sistemas de recuperação de informação como o Google (2), por exemplo, para então retornar a consulta um pouco mais estruturada para o Conlegis, provocando um enorme atraso nos processos operacionais, uma lacuna de completude e problemas de integridade de informação, além da subutilização de um sistema oficial de governo.

1.2 Objetivos

Neste contexto, este trabalho de monografia se propõe a atender ao objetivo geral e aos objetivos específicos a seguir.

1.2.1 Objetivo geral

O objetivo desta monografia será explorar o potencial das novas tecnologias e algoritmos de indexação, ranqueamento e categorização de documentos, assim como a aplicação de técnicas na análise, correção e entendimento das solicitações de pesquisa por meio da implantação de um novo Sistema de Recuperação de Informação - SRI para a ferramenta governamental de consulta aos atos normativos referentes à área de gestão de pessoas – Conlegis, denominado SRI Conlegis.

1.2.2 Objetivos específicos

Como objetivos específicos que serão atingidos durante a realização desta monografia, citam-se:

- Estudar, avaliar e definir os melhores critérios e parâmetros para a indexação de documentação legislativa referente aos recursos humanos;
- Avaliar quais os principais atributos que garantem qualidade e efetividade durante as consultas garantindo que o retorno dos documentos necessários possua a maior relevância possível para os usuários do SIPEC;
- Promover a melhoria dos processos de trabalho nos órgãos do SIPEC que necessitem de consulta de legislação pertinente;

- Propor uma metodologia para avaliação do sistema de busca e documentar a estratégia utilizada possibilitando uma forma de continuidade posterior no seu aperfeiçoamento.

1.3 Justificativa

O Conlegis é o repositório e sistema de consulta de todos os atos normativos, incluindo instruções normativas, portarias, portarias normativas e interministeriais, orientações normativas, ofícios circulares, notas técnicas e notas informativas relacionadas às atividades desempenhadas no âmbito do Sistema de Pessoal Civil da Administração Federal - SIPEC no governo federal.

O SIPEC, instituído pelo Decreto 67.326 de 1970 (3), é o sistema orgânico composto por todas as unidades organizacionais incubidas das atividades de administração de pessoal da Administração Pública direta e das Autarquias. Este sistema é coordenado pelo seu órgão central, a **Secretaria de Gestão e Desempenho de Pessoal - SGDP**

Dessa forma o sistema Conlegis é a fonte primária de consultas para os servidores, sejam dirigentes ou técnicos da área de gestão de pessoas do SIPEC. A proposta de aprimorar o sistema de recuperação de informação contida no Conlegis visa melhorar os processos de trabalho e a produtividade dos gestores e técnicos da área, bem como garantir transparência e acesso para os cidadãos.

Devido ao grande volume de atos regulatórios referentes à gestão de pessoas no âmbito do governo federal, desde 2012 a Orientação Normativa nº 7 determinou que o sistema CONLEGIS fosse fonte de consulta precípua aos órgãos integrantes do SIPEC quanto à orientação e ao esclarecimento de dúvidas concernentes à aplicação da legislação de recursos humanos, conforme seu artigo 16 apresentado abaixo:

> *"Art.16. Os órgãos integrantes do SIPEC deverão consultar o sistema de pesquisa CONLEGIS no endereço eletrônico https://CONLEGIS.planejamento.gov.br ou www.servidor.gov.br, link legislação, para conhecimento das manifestações exaradas pelo órgão Central."*
>
> (Orientação Normativa SEGEP/MP nº 7, de 12 de outubro de 2012)

Nesta linha, o fluxo de trabalho natural a ser realizado pelas unidades da SGP, no órgão central, bem como nas unidades organizacionais setoriais e seccionais do SIPEC, na Administração direta e nas Autarquias, durante a execução de suas atividades de orientação e esclarecimentos deve passar pela ferramenta Conlegis para agregação e consolidação dos atos normativos vigentes para determinado assunto.

Contudo, como o Conlegis é um sistema que existe desde 2007 e passou por poucas atualizações, sua funcionalidade de recuperação de informação encontra-se defasada e não atende mais as áreas negociais que preferem utilizar os sistemas de busca globais disponíveis na internet - ex. Google - como parte do processo para encontrar os números exatos dos atos normativos e outras informações mais precisas que ajudem em uma segunda busca posterior ao Conlegis.

Assim, ao passo que as técnicas e algoritmos de recuperação de informação foram sendo aperfeiçoados e os grandes portais de busca presentes no ciberespaço foram estendendo seu alcance de atuação, logo tornaram-se uma opção mais eficiente, simples e capaz de retornar informações mais relevantes que os sistemas governamentais dedicados a esta função.

A proposta deste trabalho, de implementar um novo sistema de recuperação de informação no Conlegis, atinge diretamente a SGDP/ME, na condição de órgão central do SIPEC, e seus órgãos integrantes, as unidades responsáveis pelas atividades referentes à gestão de pessoas. Ainda, a implantação de um SRI mais eficiente e utilizável pode proporcionar, também, uma melhor experiência de acesso às informações públicas para a sociedade como um todo.

1.4 Estrutura da monografia

A estrutura desta monografia divide-se em cinco tópicos.

O primeiro tópico consiste na introdução do conteúdo, na apresentação do problema a ser solucionado e na justificativa para o desenvolvimento deste trabalho.

O segundo tópico trata dos conceitos gerais estabelecidos desde a década de 50 até os dias atuais e apresenta uma revisão da literatura sobre a linha de pesquisa, a Recuperação de Informação.

No terceiro tópico é apresentada a metodologia utilizada para o desenvolvimento deste trabalho.

O quarto tópico apresenta a documentação detalhada de todo o processo de desenvolvimento do SRI Conlegis, a qual foi subdividida em três principais fases, a saber: o tratamento do corpus; o desenvolvimento dos algoritmos de indexação e de busca e a construção da interface para o usuário.

O quinto tópico apresenta as conclusões deste trabalho.

2 CONCEITOS GERAIS E REVISÃO DA LITERATURA

O termo “Recuperação de Informação” (do inglês, *Information Retrieval*), entendido da forma mais próxima da área de estudo atual, foi cunhado por Carl Mooers na década de 50. Os acadêmicos da época que debatiam sobre o assunto, embora vislumbrassem de forma ainda muito incipiente a imensidão de dados e informação hoje presentes em nosso mundo digital, possuíam um campo aberto para a concepção de novas idéias em um contexto que ainda tinha muito a ser explorado.

Dessa forma muitos dos princípios da recuperação de informação foram fundamentados entre as décadas de 50 e 60, nos primórdios da era da computação e andando lado a lado com o desenvolvimento também desta última. Entre estes princípios, a famosa Lei de Mooers (1960) (4), cujo criador lhe deu seu próprio nome, que diz:

> *“Um sistema de recuperação de informações terá a tendência de não ser usado se é mais irritante e problemático para um usuário obter a informação do que não obtê-la.”*

Apesar de ter como foco os aspectos negativos de um sistema de recuperação de informação, esta lei representa muito bem o conceito chave de todo o processo: quão eficiente é o processo de retornar ao usuário, que solicitou determinada pesquisa, informações que ele considera relevantes? Ou seja, quando tratamos de um sistema, processado tecnologicamente, o que está realmente em avaliação é a sua capacidade de entender, ou simular o entendimento, de determinada dúvida humana apresentando respostas que lhe sejam (a este humano - usuário) suficientes.

Aprofundando-se um pouco mais na relação estabelecida entre a solicitação de pesquisa (*query*), a entrada e o retorno ao usuário, objetos de informação, é possível apresentar de maneira simples a diferença entre o conceito de um sistema de recuperação de informação e um sistema comum de consulta a um banco de dados. Este último, apenas retorna dados específicos, aqueles que satisfizerem determinada consulta, de um conjunto maior de dados. Já um sistema de recuperação de informação não se preocupa em retornar registros de dados, mas

em proporcionar um conjunto de informações sobre determinado assunto que seja relevante ao usuário.

Em um sistema de recuperação de informação deve-se estabelecer o objeto mínimo que contém aquele pacote de informação retornado sobre determinada busca. Define-se este objeto como Documento. Segundo Le Coadic (2004) (5) documento é o termo genérico que designa os objetos portadores de informação. Assim, define-se como “documentos” todos os objetos retornados para o usuário após determinada busca por ele solicitada. E portanto, adequando este contexto à atualidade, as ferramentas tecnológicas existentes e as terminologias acadêmicas atuais, chegamos ao conceito moderno da recuperação de informação que utilizaremos neste documento, como descrito por Manning (2009) (6):

> *“Recuperação de informação (RI) é o processo de encontrar material (normalmente documentos) de natureza não estruturada (normalmente texto) que satisfaça uma necessidade de informação dentro de grandes coleções (normalmente armazenadas em computadores).”*

2.1 Modelos de recuperação de informação

Avançando algumas décadas no estudo e desenvolvimento de técnicas que implementam os sistemas de recuperação de informação, é interessante apontar o funcionamento de dois modelos bem diferenciados; os modelos booleanos e os modelos ranqueados para composição e tratamento de consultas.

Os modelos booleanos são tão antigos quanto os próprios fundamentos da programação e utilizam operadores lógicos e expressões regulares para a construção de consultas de forma a direcionar um retorno com informação precisa e proporcionar a transparência do processo como um todo. Em especial, os modelos booleanos estendidos chegam a proporcionar uma complexidade e refinamento de consultas altíssimo, sendo capazes de solicitar situações bem específicas como determinar a distância máxima entre duas ou mais palavras, exigir que elas estejam no mesmo parágrafo ou na mesma sentença.

Muitos profissionais, ao realizar pesquisas em conjuntos de documentos especializados em sua área de atuação, têm preferência por sistemas de recuperação com modelos booleanos uma vez que as buscas lhes trazem grande

controle e transparência sobre o que é, de fato, retornado. Ainda assim, é necessário que o usuário tenha conhecimento prévio, e por vezes avançado, da estruturação de expressões regulares bem como de termos específicos e seus radicais.

Já os modelos ranqueados, dentre eles os modelos linguísticos, probabilísticos e de espaço vetorial, que são mais modernos, permitem que uma busca seja feita de forma mais natural, com texto livre, sem a necessidade de conhecimento prévio em utilização de expressões regulares ou regras definidas para que a busca fique mais elaborada. Por outro lado, as técnicas que serão utilizadas, nestes modelos, são aplicadas automaticamente conforme foram pré-configuradas e o usuário não terá conhecimento dos tratamentos pelos quais seus termos de busca passaram. Ainda assim, mesmo que haja esta contrapartida entre transparência e facilidade de acesso, os modelos ranqueados, além de serem adequados a uma gama muito maior de perfis de usuários, acabam trazendo documentos igualmente ou, por vezes, mais relevantes que os booleanos, quando devidamente parametrizados.

2.2 Processo de recuperação de informação

A evolução constante das técnicas e algoritmos utilizados nos processos de recuperação de informação proporcionou, ao longo dos anos, um enorme arcabouço de métodos para:

- o tratamento e indexação do corpus (conjunto de documentos);
- a construção de formas diferentes para se realizar a pontuação dos documentos nos modelos ranqueados; e
- o entendimento dos termos de consulta.

A partir do momento que está definido o corpus que conterá os objetos de pesquisa de determinado sistema de recuperação de informação, diversas técnicas podem ser aplicadas ao processo de indexação que resulta em um índice que permitirá que os documentos sejam posteriormente recuperados. O processamento preliminar do corpus, conforme a necessidade da aplicação, pode envolver soluções

que realizam o tratamento dos termos contidos em cada documento, o agrupamento ou ajuste destes termos e sua contagem de forma diferenciada durante a geração do índice. Dentre as técnicas utilizadas podem-se citar: a demarcação e classificação de seções de fluxos textuais de entrada (tokenização, no inglês *tokenization*), a redução de radicais (*stemming* e *k-grams*), e a remoção de elementos de ligação (*stop words removal*), dentre outras.

O *ranking* de documentos resume-se na forma de pontuação a ser considerada para que, aos documentos de maior relevância, sejam atribuídos maior pontuação. Dessa forma, a partir da análise dos termos de busca solicitados, as pontuações são calculadas e os documentos retornados são ordenados para o usuário em forma decrescente de pontuação, ou seja, por relevância.

Quanto ao entendimento dos termos contidos na solicitação de pesquisa, diferentes necessidades de informação podem requerer diferentes técnicas de tratamento nas consultas, gerando *rankings* diferentes. Além disso, deve-se levar em consideração que os usuários normalmente não sabem descrever exatamente o que eles querem e esperam que o motor de busca seja capaz de complementar suas consultas podendo utilizar até variáveis que não se encontram no corpus para implementar técnicas mais eficientes como análise do usuário ou o contexto a que ele está inserido.

2.3 Recuperação de informação na atualidade

Atualmente os grupos reconhecidos como os maiores inovadores na área de recuperação de informação são as grandes empresas de tecnologia. Isto não podia ser diferente uma vez que os volumes de documentos que precisam ser analisados e retornados, de forma eficiente, na presente era digital é astronômico. Assim, empresas como Google, Amazon, Facebook participam ativamente como fontes dos novos artigos e publicações na área.

Nas últimas décadas, um dos algoritmos referência no campo de estudo foi o famoso PageRank (7) utilizado pela ferramenta de busca do Google desenvolvido pelos fundadores da Google, Larry Page e Sergey Brin em 1996. O PageRank aplicava um novo método para construção da pontuação dos documentos

baseada na quantidade de vezes que este documento era referenciado por outros documentos. O resultado da nova implementação foi uma combinação de técnicas de distribuição probabilística com uma nova variável que simulava a interação de usuários clicando nos links referenciados dentro de determinada página adicionando uma variável, o usuário, no processo. O PageRank foi um dos pilares para que a empresa se tornasse um dos sites mais utilizados de todos os tempos e, posteriormente, foi utilizado como base para o desenvolvimento de diversos outros algoritmos de ranqueamento.

2.4 Implementação de sistemas de recuperação de informação

A implementação de sistemas de recuperação hoje em dia já é muito facilitada por um conjunto de software, bibliotecas e APIs preparadas para implementar as estratégias e técnicas desenvolvidas no decorrer dos últimos anos. Esses pacotes vêm com os algoritmos prontos para implementação e ainda possibilitam o desenvolvimento de técnicas personalizadas por programação.

Desde o seu desenvolvimento, em 1999 por Doug Cutting, a ferramenta Lucene (8) tornou-se fundamental nos processos de ponta em recuperação de informação. Como uma ferramenta livre e de código aberto, o Lucene implementa um poderoso motor de busca, com capacidade de indexação e busca, implementando verificação ortográfica, técnicas de tokenização e outras funcionalidades avançadas, para qualquer fonte de informação que possa ser convertida em texto.

Utilizando o performático e escalável motor de busca do Lucene e adicionando uma gama maior de processos como a varredura em novos formatos, analisadores adicionais e bibliotecas de algoritmos pré-montados, uma miríade de novas ferramentas foi sendo desenvolvida e, atualmente são largamente utilizadas na implementação final de um sistema de recuperação de informação. Dentre estas ferramentas destacam-se o Apache Solr, Compass, CrateDB e o Elasticsearch cada um com sua modelagem de implementação e funcionalidades adicionais.

3 METODOLOGIA

Buscando atingir os objetivos propostos neste trabalho final, será adotada uma metodologia de execução das atividades separada em fases, conforme descritas a seguir:

I. **Acesso às fontes de informação - Corpus**: Nesta etapa a área responsável pela custódia do sistema, das bases de dados e da documentação utilizada pelo Conlegis deverá ser consultada, a permissão de uso dos dados deverá ser obtida e as fontes de informações brutas serão coletadas.

II. **Entrevistas com os principais utilizadores do Conlegis na Secretaria de Gestão e Desempenho de Pessoal - SGDP**: Os usuários atuais do sistema deverão ser consultados para que sejam apresentados o fluxo de trabalho, atual e o que deveria ser, bem como para levantar todos os problemas atuais com o SRI presente no Conlegis. Ainda deverá ser levantado com os usuários, quais os atributos das informações contidas nos documentos são mais importantes ou relevantes para o ranqueamento de documentos retornados de uma consulta.

III. **Tratamento das fontes de informações**: A partir da posse do corpus e de um conhecimento da expectativa e da necessidade das áreas negociais deve-se realizar o tratamento do conteúdo coletado e sua conversão para o padrão JSON que será utilizado pela ferramenta Elasticsearch.

IV. **Definição da estratégia e implementação do novo SRI**: A proposta, nesta fase do projeto, é que seja realizada algum tipo de metodologia ágil que combine o desenvolvimento, os testes, os ajustes e o processo de elaboração da documentação. Além disso, é importante, nesta etapa, que seja definido o modelo para avaliação da relevância nos documentos retornados;

V. **Documentação do processo de implantação do novo SRI**: Apesar de ser uma fase executada paralelamente aos processos de implantação do novo

SRI, é apresentada separadamente devido à importância de seu entregável, a documentação.

VI. **Entrega do código fonte e documentação para a SGDP:** Por fim, após entregues as obrigações acadêmicas deste projeto final - monografia - e finalizada a implantação e a elaboração da documentação do novo SRI para o Conlegis, o código desenvolvido e a documentação devem ser entregues à SGDP para implantação oficial, em produção, caso seja de interesse do governo.

4 DESENVOLVIMENTO DO PROJETO

O desenvolvimento do Sistema de Recuperação de Informação foi divido em três macroprocessos desde o início do tratamento do corpus (conjunto de documentos) recebido em arquivos .pdf, até a estruturação final da interface de busca para o usuário final.

- A primeira fase, e mais complexa de todo o desenvolvimento, foi o processo de conversão do corpus de seu formato original para um formato estruturado que pudesse ser, posteriormente, indexado;
- A segunda fase consistiu da estratégia utilizada para a indexação dos documentos por meio da ferramenta elasticsearch e da elaboração de algoritmos de relevância de documentos durante as ações de busca;
- A terceira fase foi a implementação de uma interface gráfica de fácil utilização que garantisse uma boa experiência ao usuário e que promovesse a ponte entre o poderoso motor de busca do Elasticsearch e uma forma atrativa de visualização das respostas da consulta.

4.1 Tratamento do Corpus

Uma vez que o Elasticsearch utiliza o padrão de dados abertos JSON e os arquivos recebidos são um conjunto de arquivos no formato PDF, foi necessário que um processo de conversão inteligente fosse construído para que os documentos fossem transformados para texto puro e da forma mais estruturada possível.

Este processo foi composto de duas subfases, a saber, primeiramente foi realizado o processo de conversão de arquivos PDF para arquivos TXT. A segunda fase foi composta do desenvolvimento de um algoritmo capaz de converter o conjunto de texto para um formato minimamente estruturado que garantisse alguns atributos importantes para o processo de indexação dos documentos.

4.1.1 Conversão de PDF para TXT

O conjunto de documentos - corpus - obtido junto a área negocial do Conlegis consistiu de 13.417 arquivos PDF. Os documentos em PDF podem

apresentar metadados de atributos diversos conforme o seu processo original de criação. Apesar disso, os documentos obtidos do Conlegis não tinham praticamente nenhum metadado relevante para auxiliar um SRI. Além disso, alguns arquivos foram gerados nativamente em PDF enquanto outros foram apenas digitalizados. Dentre os digitalizados, existem aqueles que passaram por processos de detecção de caracteres, OCR (*Optical Character Recognition* - "Reconhecimento Ótico de Caracteres), apresentando uma camada textual sobre a camada de imagem, enquanto outros foram apenas digitalizados sem OCR possuindo a camada de imagem sem nenhuma camada textual.

Para a conversão da documentação do formato PDF para um TXT puro foi utilizado um software livre desenvolvido para versões de Linux chamado pdf2txt. Essa aplicação não realiza procedimentos de OCR nos arquivos PDF, apenas extrai as camadas textuais do arquivo PDF que já existem previamente no texto.

Foi optado por não realizar nenhum tipo de processamento de OCR adicional no conjunto de documentos pois aqueles que não apresentavam nenhuma camada textual eram poucos e a maioria consistia de digitalização de documentos antigos e de baixa qualidade, o que reduziria sobremaneira a precisão do OCR e, consequentemente, dos textos extraídos do PDF.

Como toda a implementação do projeto foi realizada em ambiente de sistema operacional Windows, foi utilizada a aplicação Cygwin (Anexo I, Item 2) que funciona como uma espécie de container de terminal Linux (bash) dentro do sistema operacional da Microsoft. Dentro do Cygwin foi instalado o pdftotext (Anexo I, Item 3) e um pequeno script shell foi programado para executar o processamento massivo de extração da camada textual para um arquivo TXT de todos os 13.417 arquivos PDF. Segue abaixo o script shell utilizado para o processo de conversão de TXT para PDF dentro de um terminal Cygwin.

```
#!/bin/bash
FILES=fullcorpus/*.pdf
fout="fullcorpustxt/"
for f in $FILES
do
 in="${f%.pdf}.txt"
 out="${in#fullcorpus/}"
```

```
 echo "Processing $fout$out..."
 pdftotext -enc UTF-8 "$f" "$fout$out"
done
```

Código 1. [Shell Script] Processamento em lote dos arquivos PDF para TXT

Para fins de armazenamento da documentação foram criadas três pastas, duas das quais podem ser observadas no script acima, com o seguinte propósito:

- Pasta "*fullcorpus*": Armazena toda a documentação original, em PDF, obtida junto a área negocial do Conlegis. Contém 13.417 arquivos, em extensão .pdf com um total de aproximadamente 4.5 GB;
- Pasta "*fullcorpustxt*": Armazena os arquivos de texto puro extraídos dos PDFs originais e que foram salvos com o mesmo nome do original PDF, porém com extensão ".txt". Contém 13.417 arquivos TXT e um total de aproximadamente 119 MB;
- Pasta "fullcorpusjson": Armazena os arquivos de texto puro estruturado no padrão JSON, será utilizado durante a próxima etapa do processo de conversão. Em sua última versão, contabilizaram, aproximadamente 124 MB.

A aplicação "*pdftotext*" foi chamada de forma simplificada e o único atributo configurado foi o tipo de codificação dos caracteres em UTF-8 para se evitar problemas de compatibilidade nos passos posteriores.

A saída do comando gera um arquivo com o mesmo nome do arquivo .pdf original, alterando apenas sua extensão para ".txt". A linha `for f in $FILES` serve para rodar o processo em lote, de forma iterativa, até processar todos os 13.417 arquivos.

4.1.2 Conversão de TXT para JSON

A fase seguinte da implementação foi o processamento dos arquivos de texto puro que foram gerados na etapa anterior, a aplicação de uma estratégia de estruturação do conteúdo e a geração dos arquivos finais em formato estruturado JSON (*Javascript Object Notation*) (9), um padrão aberto e simples para troca rápida

de dados entre sistemas e que será posteriormente utilizado para a indexação da documentação no Elasticsearch, proporcionando o desenvolvimento do SRI.

Em uma primeira fase de implementação, possuindo apenas o conjunto de documentos originais em formato PDF, foi necessária a estruturação de um conversor mais completo e com maior grau de “inteligência” para que os dados brutos extraídos da camada textual fossem estruturados da melhor forma possível para proporcionar uma boa resposta às buscas que seriam solicitadas.

Entretanto, após o desenvolvimento do conversor, foi também obtida da área negocial do Conlegis, uma extração do banco completa que estrutura o sistema web atual. Este banco, além das diversas tabelas de apoio ao sistema, possuía tabelas com dados estruturados de praticamente todos os normativos presentes no corpus anterior.

Este segundo acesso ao conteúdo já estruturado do banco permitiu o desenvolvimento de um algoritmo combinador do processo anterior com a extração estruturada do banco proporcionando um conjunto de arquivos JSON com uma qualidade de dados muito superior à obtida anteriormente. Dessa forma, os dois processos de desenvolvimento serão apresentados principalmente por serem, em parte, complementares.

4.1.2.1 Desenvolvimento inicial do conversor TXT para JSON

O desenvolvimento deste conversor foi desenvolvido na linguagem de programação Python (Anexo I, Item 4) e compôs grande parte do desenvolvimento inicial do SRI, uma vez que o algoritmo construído precisou de um nível complexo de tratamento necessário para a estruturação dos dados que seriam futuramente utilizados pelo Elasticsearch. Durante o desenvolvimento do projeto diversas melhorias foram sendo implementadas no conversor conforme o conjunto de documentos gerado por ele foi sendo indexado e testado no Elasticsearch.

Este algoritmo de conversão será apresentado de forma resumida, focando apenas em determinadas etapas do processo que foram mais essenciais para o processo. O conteúdo completo do conversor poderá ser visitado nos anexos desta documentação.

Durante o processo de conversão foi necessário que arquivos fossem lidos e gerados com alteração de extensões e acesso a diferentes pastas do ambiente de desenvolvimento. O código abaixo apresenta duas funções para tratamento dos nomes dos arquivos para renomeação da extensão, bem como das pastas onde eles se encontram. A função "`strip_left`" ajusta o prefixos dos arquivos alterando locais de armazenamento e acesso e a função "`strip_right`" ajusta as extensões dos arquivos no processo de geração dos novos formatos de saída.

```
#Função para ajustar as pastas de entrada e saída.
def strip_left(text, prefix):
       if not text.startswith(prefix):
       return text
       # else
       return text[len(prefix):]

#Função para alterar a extensão mantendo o nome do arquivo.
def strip_right(text, suffix):
 if not text.endswith(suffix):
  return text
 # else
 return text[:len(text)-len(suffix)]
```

Código 2. [Python] Funções para tratamento de nome de arquivo.

A função a seguir foi utilizada para encontrar uma palavra específica dentro do conjunto de dados carregado do arquivo texto sendo utilizada no processo de detecção do tipo de normativo (decreto, lei, portaria, etc…). A lógica utilizada foi primeiramente a execução da varredura do nome do arquivo para detecção do tipo de arquivo dentre uma lista (*array*) de tipos de normativos pré-estabelecidos. Caso não seja encontrado, no nome do arquivo (*filename*), nenhum dos tipos de normativos pré-estabelecidos a função faz a varredura de todo o conteúdo textual do arquivo, linha por linha, e o documento é categorizado assim que surgir a primeira linha com algum dos tipos de normativos estabelecidos na lista.

Esta função também foi utilizada no processo de detecção do título do documento, uma vez que ele define o tipo de normativo, a função detecta a linha que contém este tipo para que esta linha seja utilizada como título do documento.

```
#Função para encontrar uma palavra específica em uma string
def findWholeWord(w):
```

```
        return re.compile(r'\b({0})\b'.format(w), flags=re.IGNORECASE).search
```

Código 3. [Python] Função para detecção de determinada string.

A função *txttojson* é a responsável pela estruturação dos metadados do JSON, acrescentando ao documento os atributos que serão utilizados de forma estruturada pelo Elasticsearch, assim como no processo de visualização dos dados pelo usuário. Trechos desta função serão destacados para um melhor entendimento do processo de conversão.

Abaixo, é apresentada a lista pré-estabelecida utilizada no algoritmo que contém os tipos de normativos que serão utilizados no procedimento de detecção. Estas categorias de normativos foram retiradas observando os tipos disponíveis na aplicação atual do Conlegis. Não foram utilizados todos os tipos de normativos que são apresentados na aplicação atual do sistema sendo apenas construída uma lista reduzida contendo todos os tipos de normativos mais relevantes. Para os documentos que não se enquadram em nenhum outro tipo de normativo foi criada a categoria "*outros*".

```
def txttojson (filename):

(...)

  #DETECÇÃO DO TIPO DE NORMATIVO E DA LINHA DE TÍTULO
  normativos = ["decreto-lei", "decreto legislativo", "decreto", "ofício circular",
"ofício-circular", "ofício-circular conjunto", "ofício", "despacho", "emenda",
"acórdão tcu", "acórdão", "instrução normativa", "lei complementar", "lei
delegada", "lei", "memorando-circular", "memorando", "medida provisória", "nota
técnica", "nt n°", "comunica geral", "comunica", "portaria", "súmula", "resolução",
"orientação consultiva", "orientação normativa", "parecer", "ni n°", "nota
informativa"]

(...)
```

Código 4. [Python] Lista de tipos para procedimento de detecção automática.

O trecho abaixo apresenta a estruturação do arquivo JSON que será gravado, ao final do processo de conversão. A junção das strings "*header*" e "*jsoncontent*" compõe o conteúdo completo do arquivo, que ao final do processo, receberá o mesmo nome do arquivo original com a alteração apenas da sua extensão de ".txt" para ".json".

```
def txttojson (filename):

  (...)

  header = '{"index":{"_index":"legis","_type":"normativos"}}\n'

  jsoncontent = header + '{"filename": "' + namepdf + '", "fileurl": "' +
fileurl + '", "tipo_normativo": "' + tipo_normativo + '", "tipos_ref": ' +
tipos_ref_str + ', "titulo": "' + title + '", "preview": "' + preview +
'", "dados": "' + data + '"}\n'

  (...)

  fileout.write(jsoncontent)
```

Código 5. [Python] Montagem do conteúdo do arquivo JSON.

A estrutura acima apresenta a coleção final de atributos construídos no processo de conversão. A necessidade de alguns dos atributos acima foi surgindo no decorrer do desenvolvimento do projeto conforme novas necessidades foram surgindo ou observou-se que algumas etapas da implementação do SRI poderiam ser facilitadas por meio da extração direta do dado do atributo ao invés de implementar a funcionalidade na ferramenta de apresentação ou utilizar funções do Elasticsearch. O atributo "*fileurl*", por exemplo, poderia ser implementado de diversas formas mas retirá-lo pronto do documento foi a solução mais simples e rápida de desenvolvimento. Em uma aplicação final para utilização do Conlegis, este processo pode ser revisto e aprimorado.

Os atributos, os dados apresentados de forma estruturada, adicionados ao JSON são os seguintes:

- ***index***: Sempre atribuído para o índice "legis". Este atributo mostra em qual índice este documento será trabalhado, é necessária sua repetição em todos os documentos pois este atributo é um pré-requisito para o processamento em lotes de arquivo no Elasticsearch.
- ***_type***: parâmetro reservado do Elasticsearch, pode ser utilizada para segmentação da base de documentos em diversas sub-bases de diferentes categorias.
- ***filename***: Armazena o nome do arquivo original, na versão PDF

- ***fileurl***: Apresenta a URL completa do documento publicada no sistema conlegis. Este atributo é gerado a partir do filename e do prefixo de domínio do Conlegis. A URL é convertida para codificação de caracteres nos códigos do padrão Windows-1252, sendo que este é o utilizado pelo sistema atual do Conlegis.

- ***tipo_normativo***: Registra o tipo de normativo que foi categorizado o documento após o processo de detecção explicado anteriormente. Pode ser dos tipos: `"decreto-lei", "decreto legislativo", "decreto", "ofício-circular", "ofício-circular conjunto", "ofício", "despacho", "emenda", "acórdão tcu", "acórdão", "instrução normativa", "lei complementar", "lei delegada", "lei", "memorando-circular", "memorando", "medida provisória", "nota técnica", "comunica geral", "comunica", "portaria", "súmula", "resolução", "orientação consultiva", "orientação normativa", "parecer" e "nota informativa".`

- ***tipos_ref***: Apresenta uma lista (array) de todos os tipos de normativos que foram referenciados no documento.

- ***titulo***: Apresenta a linha do arquivo completa que possui o tipo de normativo detectado como categoria daquele documento.

- ***preview***: Atributo que consiste nos primeiros 300 caracteres do conteúdo do atributo “dados”. É utilizado para fins de apresentação gráfica, sinopse, do documento.

- ***dados***: Contém todo o conteúdo da camada textual extraída do PDF com alguns processos de transformação embutidos como a remoção de caracteres diferentes dos alfanuméricos. Foi adicionado no intuito de facilitar a indexação dos *tokens* dos documentos, bem como melhorar as estratégias de elaboração dos algoritmos de busca que serão apresentados mais à frente.

Alguns tipos de normativos, menos comuns, não foram tratados pelo algoritmo. Estes foram categorizados todos como “*outros*” ou são erroneamente

classificados. Observa-se que a estratégia de categorização utilizada tem uma boa eficácia mas apresenta falhas, eventualmente, no caso de alguns normativos que apresentam nome de arquivo fora do padrão e caso sua camada textual apresente referências a algum outro tipo de normativo antes do título oficial do documento. Ainda assim, apesar da falha de referência do título correto e do tipo de normativo, o processo de busca será construído para ler também todos os *tokens* gerados pelo conteúdo do arquivo no campo “dados”. Dessa forma, um documento erroneamente classificado não deixa de ser apresentado na busca, mas poderá ter sua pontuação de relevância reduzida.

Ainda, os documentos que não apresentavam camadas textuais geradas originalmente por OCR, tiveram seu título e dados constituídos apenas pelo nome do arquivo (*filename*) já que, neste caso, esta foi a única fonte de texto disponível para o documento.

O trecho abaixo do arquivo *txt2json7.py* apresenta as bibliotecas python importadas para facilitar o processo de conversão.

```
import logging
import os, os.path
import sys
import re
import urllib.parse
from multiprocessing.dummy import Pool as ThreadPool
```

Código 6. [Python] Bibliotecas utilizadas no conversor TXT para JSON.

A biblioteca `logging` adiciona funções básicas para estruturação de log de operações realizadas. As bibliotecas `os`, `os.path` e `sys` são utilizadas para internalização das variáveis passadas via parâmetros de linha de comando assim como a leitura dos arquivos dentro da pasta. A biblioteca `re` adiciona funções de tratamento de strings. A biblioteca `urllib.parse` adiciona funções nos tratamentos dos dados que serão convertidos para link de internet. A biblioteca `Pool` é utilizada no processamento em paralelo e permite a indexação em paralelo de diversos arquivos, simultaneamente.

4.1.2.2 Desenvolvimento do segundo conversor combinador

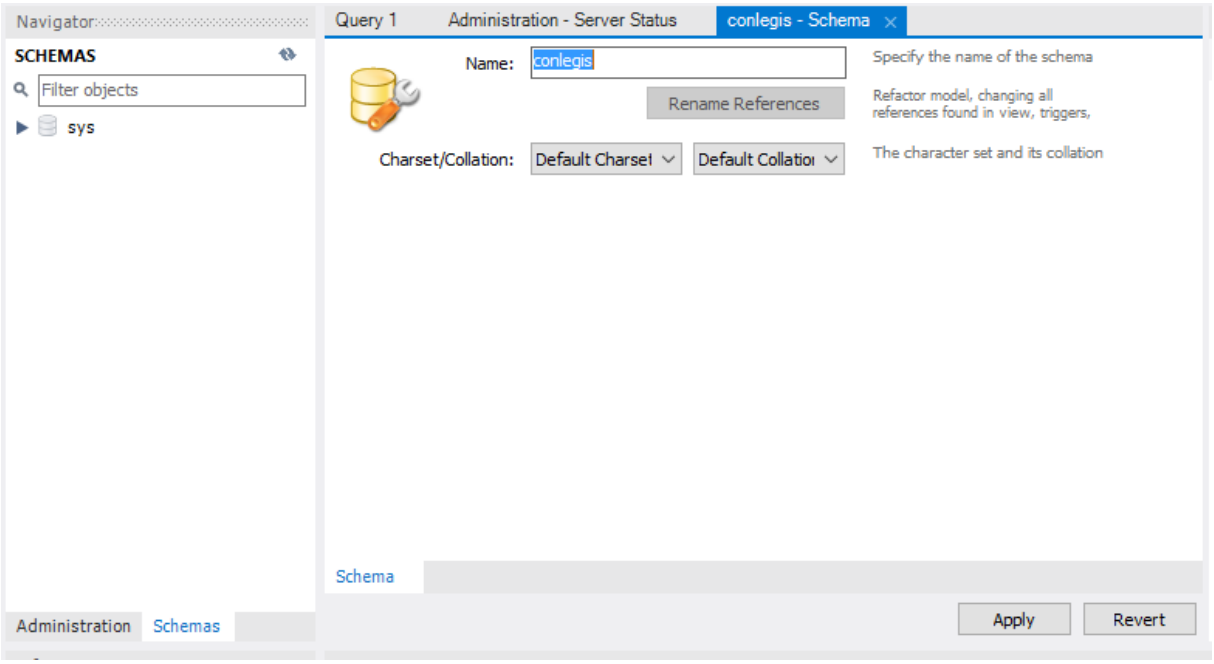
Com a obtenção da base de dados utilizada para o sistema oficial do Conlegis, foi possível obter muita informação, já estruturada, sobre o corpus de atos normativos disponibilizado. O arquivo disponibilizado contém um *dump* (extração de dados) completo da base utilizada em um arquivo de formato ".sql". O cabeçalho do arquivo - apresentado no código 7 abaixo - mostra que a base de dados ali presente foi extraída de um *schema* implementado sobre a plataforma de banco de dados MySQL.

```
-- MySQL dump 10.13  Distrib 5.7.17, for Linux (x86_64)
--
-- Host: 10.209.40.100    Database: dbp_54828_conlegis
-- ------------------------------------------------------
-- Server version 5.7.13
```

Código 7. [SQL] Cabeçalho do arquivo de extração da base do Conlegis.

Os dados foram carregados utilizando a ferramenta MySQL Workbench (Anexo I, Item 11) em conjunto com o MySQL Server (Anexo I, Item 10). Foi necessária a criação preliminar de um *schema* novo de base de dados, que no caso, foi denominado "*conlegis*".

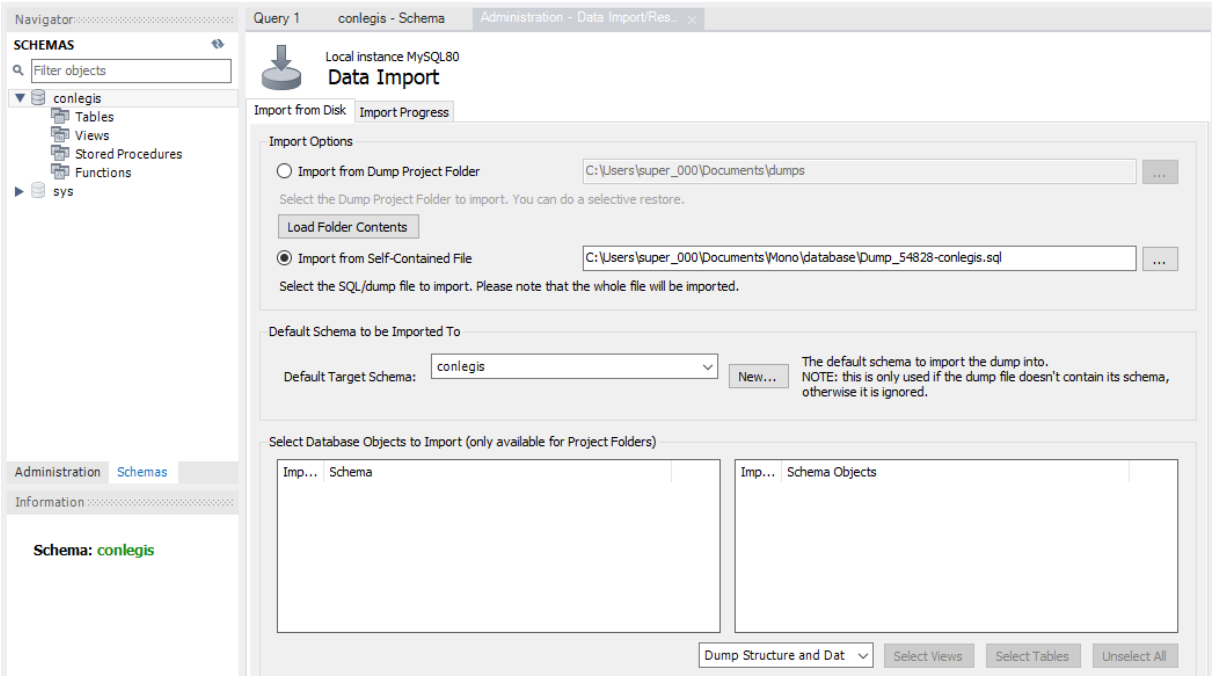
Figura 1. Criação de schema no MySQL Workbench



.

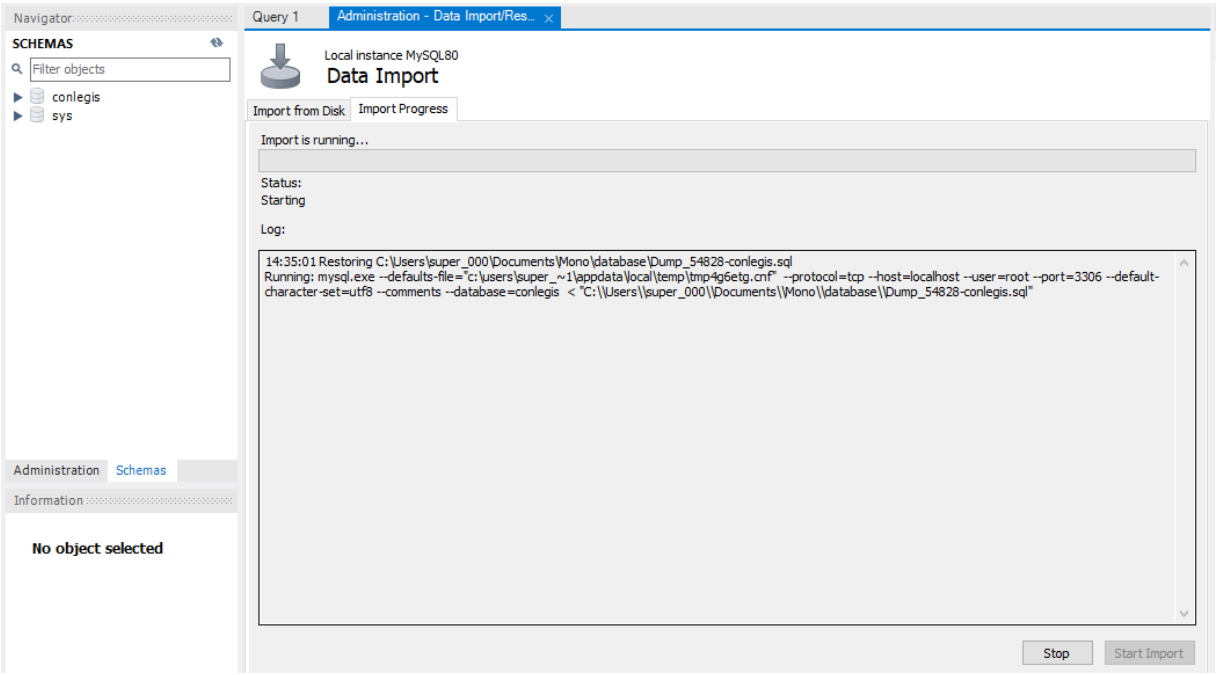
O MySQL Workbench - versão 8.0.14 - facilita o processo de importação de bases, adicionando uma interface gráfica de fácil utilização como intermediário da aplicação mysql que realiza a importação de um banco de dados completo em formato “.sql” para o novo *schema conlegis*. Este processo está disponível na aba “*Server*”, item “*Data Import*”.

Figura 2. Data import no MySQL Workbench



Fonte: Elaborado pelo autor

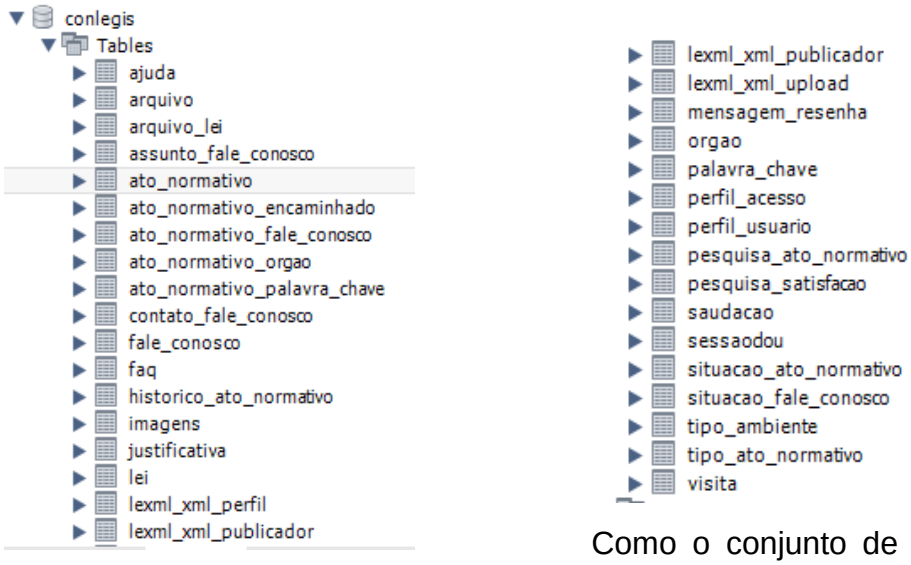
Figura 3. Data import – Log – no MySQL Workbench



Fonte: Elaborado pelo autor.

Deste conjunto de dados, foram utilizadas informações derivadas de quatro tabelas: *ato_normativo*; *arquivo*; *situacao_ato_normativo* e *tipo_ato_normativo*. As figuras abaixo mostram a estrutura da base de dados obtida e as tabelas que foram utilizadas como insumos para a execução do algoritmo conversor combinador.

Figura 4. Tabelas obtidas do sistema Conlegis.



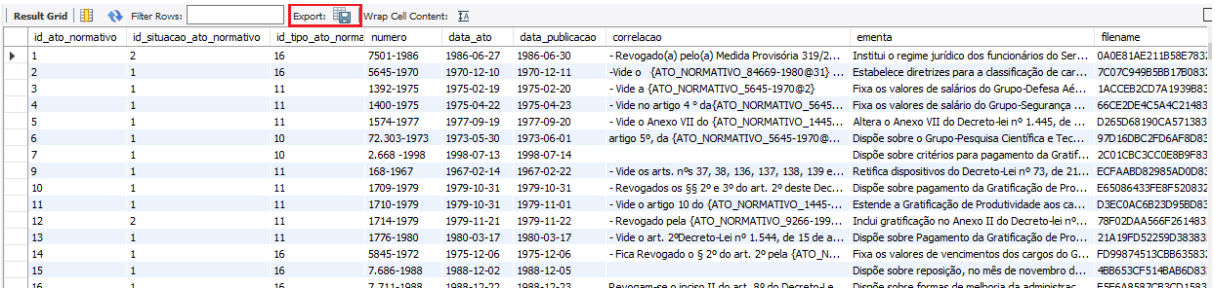
Como o conjunto de atos normativos

não é muito grande para estruturas de bancos de dados, foi possível estruturar uma consulta, apresentada abaixo, que trouxesse todos os atos normativos registrados na tabela "*atos_normativos*" combinada com a tabela "*arquivo*", esta última contendo o nome do arquivo .pdf no sistema, que servirá futuramente para construir o link de internet (URL) oficial utilizado pelo Conlegis. Ainda, o MySQL Workbench permite que toda a tabela resultante da consulta realizada seja exportada em diversos formatos, que no caso, foi utilizado como formato JSON. Foram extraídas também, em formato JSON, as tabelas referência para os códigos de situação dos atos normativos (*situacao_ato_normativo*) e para os códigos de tipos de atos normativos (*tipo_ato_normativo*).

```
select A.id_ato_normativo, A.id_situacao_ato_normativo,
A.id_tipo_ato_normativo, A.numero, A.data_ato, A.data_publicacao,
A.correlacao, A.ementa, B.nome as filename
from ato_normativo A left join arquivo B
on A.id_ato_normativo = B.id_ato_normativo
```

Código 8. [SQL] Query que combina as tabelas ato_normativo e arquivo.

Figura 5. Função de exportação do resultado de busca para JSON no MySQL Workbench.



id_ato_normativo	id_situacao_ato_normativo	id_tipo_ato_norma	numero	data_ato	data_publicacao	correlacao	ementa	filename
1	2	16	7501-1986	1986-06-27	1986-06-30	- Revogado(a) pelo(a) Medida Provisória 319/2…	Institui o regime jurídico dos funcionários do Ser…	0A0E81AE211B58E783…
2	1	16	5645-1970	1970-12-10	1970-12-11	-Vide o {ATO_NORMATIVO_84669-1980@31}…	Estabelece diretrizes para a classificação de car…	7C07C949B5BB17B083…
3	1	11	1392-1975	1975-02-19	1975-02-20	- Vide a {ATO_NORMATIVO_5645-1970@2}	Fixa os valores de salários do Grupo-Defesa Aé…	1ACCEB2CD7A1939B83…
4	1	11	1400-1975	1975-04-22	1975-04-23	- Vide no artigo 4 º da{ATO_NORMATIVO_5645…	Fixa os valores de salário do Grupo-Segurança …	66CE2DE4C5A4C21483…
5	1	11	1574-1977	1977-09-19	1977-09-20	- Vide o Anexo VII do {ATO_NORMATIVO_1445…	Altera o Anexo VII do Decreto-lei nº 1.445, de …	D265D68190CA571383…
6	1	10	72.303-1973	1973-05-30	1973-06-01	artigo 5º, da {ATO_NORMATIVO_5645-1970@…	Dispõe sobre o Grupo-Pesquisa Científica e Tec…	97D16DBC2FD6AF8D83…
7	1	10	2.668 -1998	1998-07-13	1998-07-14		Dispõe sobre critérios para pagamento da Gratif…	2C01CBC3CC0E8B9F83…
9	1	11	168-1967	1967-02-14	1967-02-22	- Vide os arts. nºs 37, 38, 136, 137, 138, 139 e…	Retifica dispositivos do Decreto-Lei nº 73, de 21…	ECFAABD82985AD0D83…
10	1	11	1709-1979	1979-10-31	1979-10-31	- Revogados os §§ 2º e 3º do art. 2º deste Dec…	Dispõe sobre pagamento da Gratificação de Pro…	E65086433FE8F520832…
11	1	11	1710-1979	1979-10-31	1979-11-01	- Vide o artigo 10 do {ATO_NORMATIVO_1445-…	Estende a Gratificação de Produtividade aos ca…	D3EC0AC6B23D95BD83…
12	2	11	1714-1979	1979-11-21	1979-11-22	- Revogado pela {ATO_NORMATIVO_9266-199…	Inclui gratificação no Anexo II do Decreto-lei nº…	78F02DAA566F261483…
13	1	11	1776-1980	1980-03-17	1980-03-17	- Vide o art. 2ºDecreto-Lei nº 1.544, de 15 de a…	Dispõe sobre Pagamento da Gratificação de Pro…	21A19FD52259D38383…
14	1	16	5845-1972	1975-12-06	1975-12-06	- Fica Revogado o § 2º do art. 2º pela {ATO_N…	Fixa os valores de vencimentos dos cargos do G…	FD99874513CBB63583…
15	1	16	7.686-1988	1988-12-02	1988-12-05		Dispõe sobre reposição, no mês de novembro d…	4BB653CF514BAB6D83…
16	1	16	7.711-1988	1988-12-22	1988-12-23	Revogam-se o inciso II do art. 88 do Decreto-Le…	Dispõe sobre formas de melhoria da administraç…	E8E6A8587CB3CD1583…

Fonte: Elaborado pelo autor

A partir dos dados extraídos dos PDFs e das informações estruturadas do banco de dados foi desenvolvido o algoritmo "combinador" uma vez que este combina algumas das funções e trechos de código do algoritmo de conversão anterior e também por utilizar as fontes JSON extraídas do banco MySQL e combiná-las com os arquivos TXT resultantes da conversão original dos arquivos PDF para gerar os arquivos finais JSON que serão processados e indexados no Elasticsearch.

Assim como foi apresentado o algoritmo conversor da primeira implementação no item anterior, as partes principais do algoritmo combinador serão apresentadas e o código completo estará disponível para consulta no Apêndice A desta documentação.

O código do combinador, assim como o do conversor, foi desenvolvido na linguagem Python. As bibliotecas que foram importadas foram as seguintes:

```
import logging
import sys
import re
import os, os.path
from pathlib import Path
import urllib.parse
import json
```

Código 9. [Python] Bibliotecas utilizadas no conversor combinador.

As bibliotecas `logging`, `sys`, `re`, `os` e a `urllib.parse` foram apresentadas e explicadas no tópico que trata do desenvolvimento do conversor. Além dessas foram adicionadas apenas a biblioteca `Path`, utilizada para checar se o arquivo TXT correspondente a referência extraída do banco existe e a biblioteca `json` que adiciona funções para processamento de estruturas em formato JSON. No combinador a biblioteca `json` foi utilizada devido a entrada de arquivos ser composta pelos arquivos JSON extraídos da base de dados do conlegis além dos arquivos TXT utilizados no conversor.

A lógica do combinador foi composta da seguinte maneira:

I. Carrega os arquivos que foram extraídos da base de dados: *tiposnormativos.json*, *situacaonormativos.json* e *database-full.json* onde:

   a. *tiposnormativos.json* é a versão JSON da tabela de referência que associa os códigos dos tipos de atos normativos com seu nome textual;

   b. *situacaonormativos.json* é a versão JSON da tabela de referência que associa os códigos das situações dos atos normativos com seu nome textual;

c. *database-full.json* é a versão JSON da tabela com os dados de todos os atos normativos extraída da base MySQL combinada com a tabela que armazena o nome do arquivo PDF para download de cada um dos normativos.

II. Processa cada item JSON do arquivo *database-full.json*, detecta o nome do arquivo PDF a que ele se refere e busca se existe o arquivo TXT equivalente;

III. Se possuir o nome do arquivo no item e for detectado o TXT equivalente ele extrai todas as informações do JSON e monta os campos `"filename"`, `"fileurl"`, `"tipo_normativo"`, `"sit_normativo"`, `"titulo"`, `"data_publicacao"`, `"ano"`, `"ementa"` e `"correlacao"`;

IV. Os campos `"tipo_normativo"`, `"sit_normativo"` são adicionados pelo cruzamento dos códigos (`id_situação_ato_normativo` e `id_tipo_normativo`) com as tabelas de referência equivalentes e são apresentados em formato textual.

V. Os campos título e data_publicação são construídos pela análise e processamento dos campos `"numero"` e `"data_publicacao"`, este último convertido para formato textual, da base de dados.

VI. O conteúdo completo do arquivo TXT equivalente é extraído e adicionado ao campo `"dados"`.

VII. O cabeçalho para processamento em lote do Elasticsearch é adicionado e o conteúdo de todos os campos são registrados no *<nomedoarquivo>.json*.

VIII. O processo se repete em todos os itens do arquivo *database-full.json* inicial.

Devido ao tamanho da função “*combiner”*, que implementa o processo acima, esta não será apresentada no corpo do trabalho, mas o código completo do algoritmo, incluindo a função “*combiner”,* está disponível no Apêndice A, Item 3, para

consulta com os comentários associados referenciando os passos apresentados acima.

Um ponto interessante de se observar é que por mais que seja comum um texto de referência como título do documento sempre que se busca um conjunto de informações em qualquer sistema de recuperação de informação tradicional, este campo simplesmente não existe na aplicação atual do Conlegis. Por isso o título teve que ser composto através de informação derivada de outros campos, no caso os que trazem informação sobre o título do ato, o número do ato e a data do ato.

Essas informações foram utilizadas para gerar o título como o exemplo abaixo:

Dadas as seguintes informações: `tipo_normativo = "Lei", numero = "8.112", data_ato = 1990-12-11;`

O algoritmo combina os campos e formata o seguinte título:

“Lei n° 8.112, de 11 de dezembro de 1990”

Este processo no algoritmo é apresentado em detalhes no trecho código abaixo:

```
def combiner (jsondump):

(...)

      mes_ext = {1: 'Janeiro', 2 : 'Fevereiro', 3: 'Março', 4: 'Abril', 5: 'Maio', 6: 'Junho', 7: 'Julho', 8: 'Agosto', 9: 'Setembro', 10: 'Outubro', 11: 'Novembro', 12: 'Dezembro', 13: '[não informado]'}

      if doc["data_ato"] == None:
            anoa = "[não informado]"
            mesa = "13"
            diaa = "[não informado]"
      else:
            anoa, mesa, diaa = doc["data_ato"].split("-")

(...)

      title = tipo_normativo + " n° " + numeroato + ", de " + diaa + " de " + mes_ext[int(mesa)] + " de " + anoa
```

Código 10. [Python] Script para formatação do título do ato normativo.

Por fim, os novos arquivos gerados JSON serão compostos pelos itens abaixo:

```
def combiner (jsondump):

(...)

      header = '{"index":{"_index":"legis2","_type":"atosnormativos"}}\n'

(...)

      jsoncontent = header + '{"filename": "' + filename + '", "fileurl":
"' + fileurl + '", "tipo_normativo": "' + tipo_normativo + '",
"sit_normativo": "' + sit_normativo + '", "titulo": "' + title + '",
"data_publicacao": "' + data_publicacao + '", "ano": "' + anop + '",
"ementa": "' + ementa + '", "correlacao": "' + correlacao + '", "dados":
"' + data + '"}\n'
```

Código 11. [Python] Construção do conteúdo JSON no combinador.

Nesta nova versão utilizando o algoritmo combinador os atributos, os dados apresentados de forma estruturada, adicionados ao JSON são os seguintes:

- ***index***: Sempre atribuído para o índice `"legis2"`. Este atributo mostra em qual índice este documento será trabalhado, é necessária sua repetição em todos os documentos pois este atributo é um pré requisito para o processamento em lotes de arquivo no Elasticsearch.
- ***_type***: parâmetro reservado do Elasticsearch, pode ser utilizada para segmentação da base de documentos em diversas sub-bases de diferentes categorias.
- ***filename***: Armazena o nome do arquivo original, na versão PDF.
- ***fileurl***: Apresenta a URL completa do documento publicada no sistema conlegis. Este atributo é gerado a partir do filename e do prefixo de domínio do Conlegis. A URL é convertida para codificação de caracteres nos códigos do padrão Windows-1252, sendo que este é o utilizado pelo sistema atual do Conlegis.
- ***tipo_normativo***: Registra o tipo de normativo extraído do banco de dados oficial do sistema Conlegis. Pode ser dos tipos: `"Ata"`, `"Ato`

Declaratório Executivo", "Ato Regimental", "Boletim Eletrônico-Contato", "Boletim Informativo STF", "Circular", "Comunica Geral", "Conjunto de Pareceres Conclusivos", "Decisão TCU", "Decreto", "Decreto-Lei", "Decreto Legislativo", "Despacho", "Emenda Constitucional", "Instrução Normativa", "Lei", "Lei Complementar", "Medida Provisória", "Memorando", "Mesa Nacional de Negociação Permanente", "Norma Interna", "Norma Operacional", "Nota Técnica", "Ofício", "Ofício-Circular", "Ofício-Circular Conjunto", "Orientação Consultiva", "Orientação Geral", "Orientação Normativa", "Parecer", "Parecer AGU", "Parecer CONJUR", "Portaria", "Portaria Conjunta", "Portaria Interministerial", "Portaria Normativa", "Projeto de Lei", "Resolução", "Resposta Fax", "Resposta Ofício", "Súmula Administrativa", "Súmula da Advocacia-Geral da União", "Nota Informativa", "Norma de Execução", "Lei Delegada", "Deliberação", "Constituição Federal E Emendas Constitucionais", "Constituição Federal", "Atos da AGU", "Ação Direta de Inconstitucionalidade", "Nota CONJUR", "Nota AGU", "Comunica SIAPE", "Manual de Redação da Presidência da República", "Exposição de Motivos", "Exposição de Motivos Interministerial", "Resolução Normativa", "Comunica", "Edital", "Nota Técnica_", "Recurso Extraordinário", "Acórdão TCU_", "Acórdão TCU", "Acórdão", "Anteprojeto de Lei", "Ato", "Súmula Vinculante", "Nota Técnica Conjunta", "emprego público", "Nota", "Formulação", "Súmula", "Diversos", "Lei 8.112 - ANOTADA", "Nota Informativa Conjunta", "Portaria AGU", "TESTE", "Súmula do TCU", "Nota Técnica Consolidada", "PARECER/PGFN", "PERFIL PROFISSIOGRÁFICO PREVIDENCIÁRIO - PPP", "SÚMULA AGU", "Decisão STF/Reclamação", "Manual de Perícia Oficial", "Agenda de Decisão SEGRT", "Comissão de

Ética", "Memorando Circular", "Portaria Gsiste", "OFÍCIO/PGFN", "Formulação/DASP", "PARECER/DECOR/CGU/AGU.", "Termo Aditivo", "Manual de Processo Administrativo Disciplinar", "Instrução Normativa Conjunta", "Mensagem" e "Ato Declaratório".

- ***sit_normativo***: Apresenta a situação a qual o normativo em questão se encontra, extraída do banco de dados oficial do Conlegis. Pode ser dos tipos: "Em Vigor", Revogado (a)", "Reeditado (a)", "Retificado (a)", "Revogado e Reeditado", "Republicado (a)", "Alterado (a)", "Rejeitado (a)", "Convertida em Lei", "Revogado (a) Parcialmente", "Prejudicado (a)", "Reeditada e Revogada", "Sem Eficácia", "Insubsistente", "Suspensa a Eficácia", "Revigorado(a)", "Insubsistente Parcialmente", "Revisto e Atualizado", "Aditado (a)", "Tornado (a) sem Efeito", "Anulada", "Revogado(a) Tacitamente" e "Vigência Encerrada".

- ***titulo***: Apresenta o título do documento composto pelo tipo de documento, o seu número e a sua data.

- ***data_publicacao***: Apresenta a data na qual o ato normativo foi publicado no diário oficial.

- ***ano***: Apresenta o ano de origem do normativo.

- ***ementa***: Atributo extraído da base de dados oficial do Conlegis que contempla o resumo sobre o assunto do documento.

- ***correlacao***: Apresenta a correlação do normativo com outros normativos.

- ***dados***: Contém todo o conteúdo da camada textual extraída do PDF com alguns processos de transformação embutidos como a remoção de caracteres diferentes dos alfanuméricos. Foi adicionado no intuito de facilitar a indexação dos *tokens* dos documentos, bem como melhorar as estratégias de elaboração dos algoritmos de busca.

4.2 Construção do Index no Elasticsearch

Depois de realizado o processo de conversão do corpus para o formato JSON foi criado um índice simples no Elasticsearch para que fosse carregado o corpus já no formato estruturado e compatível com a aplicação da Elastic.

O Elasticsearch (Anexo I, Item 5) é um poderoso motor de busca e análise distribuído que utiliza a arquitetura REST (*Representational State Transfer*, do inglês, Transferência de Estado Representacional) que foi introduzida e teve seus princípios definidos por Roy Thomas (2000) (10) visando facilitar a troca de dados pelo protocolo HTTP. Hoje o REST é amplamente utilizado como web service (serviço disponibilizado via http/web) proporcionando interoperabilidade entre sistemas de forma prática e eficiente. Assim, o Elasticsearch armazena os dados dos documentos e pode ser configurado de inúmeras formas proporcionando diversas funcionalidades e inovações nos processos de recuperação da informação contida nos documentos.

Durante o processo de construção de índices para o desenvolvimento do SRI Conlegis, foi utilizada a estratégia de estruturação de índices iniciais e simples para servir de ambientes de teste para detecção de problemas, validação de correções e mapeamento das necessidades de melhoria dos algoritmos que realizaram a conversão dos arquivos TXT para JSON. Dessa forma o índice foi sendo aperfeiçoado aos poucos e sendo complementado de funcionalidades necessárias para realizar o tratamento do corpus específico do Conlegis.

Sobre cada uma das implementações de índices foi, então, carregado o conjunto de atos normativos que compõem o corpus do projeto. Para fins de registro de documentação e para facilitar o entendimento caso a implementação seja futuramente replicada, a evolução do índice será apresentada toda de uma vez, nos três próximos itens, e o processo de carregamento do lote de documentos (atos normativos) será apresentada no quarto.

4.2.1 Configuração utilizada no índice simples

A configuração abaixo apresenta o primeiro índice gerado no Elasticsearch, de forma bem simples, para testar o conjunto de documentos e

observar os primeiros resultados da implementação de uma busca utilizando a ferramenta Elasticsearch. A implementação do índice abaixo, bem como dos subsequentes nesta documentação, foram realizadas por meio da ferramenta disponibilizada pela elastic chamada Kibana. O Kibana (Anexo I, Item 6) é uma ferramenta de apoio que permite utilização de um ambiente simples de desenvolvimento assim como várias ferramentas próprias para análise, descoberta e monitoramento de um cluster Elasticsearch.

```
PUT legis
{
  "settings": {
      "analysis": {
        "analyzer": {
             "default": {
               "type": "standard",
               "max_token_length": 15,
               "stopwords": "_brazilian_"
             },
             "default_search": {
               "type": "standard",
               "stopwords": "_brazilian_"
             }
        }
      },
      "similarity": {
        "default": {
         "type": "BM25"
        }
      }
  }
}
```

Código 12. [JSON] Criando índice simplificado no Kibana/Elasticsearch.

Pela configuração acima, foi utilizado o analisador do tipo “padrão” (`"type": "standard"`) do Elasticsearch. Este analisador traz uma pré configuração de separação de *tokens* baseado em regras gramaticais básicas e comuns a várias línguas e também converte automaticamente para minúsculas todas as palavras. No caso utilizado, também foi adicionado o tratamento de palavras irrelevantes (`"stopwords": "_brazilian_"`) da língua portuguesa do Brasil, assim como um tamanho máximo de *token* de 15 caracteres.

Para todas as estratégias de indexação foi utilizado o algoritmo de similiaridade BM25 (mais conhecido como Okapi BM25 pois este foi o primeiro sistema no qual foi implementado) baseado no modelo probabilístico de recuperação desenvolvido entre 1970 e 1980 por Robertson e Jones (11). Este parâmetro foi escolhido pelo algoritmo BM25 ser considerado uma das funções estado-da-arte de recuperação da informação baseada em TF-IDF (*Term Frequency – Inversed Document Frequency*) e ter sido comprovado como altamente eficaz para buscas textuais.

Os primeiros testes apresentaram muitos problemas derivados da precisão do conversor TXT para JSON, passando este por várias evoluções, principalmente na categorização de documentos e estruturação dos atributos necessários. A detecção de título, em particular, ainda apresenta falhas eventuais mas um aprofundamento posterior do algoritmo pode permitir uma maior precisão deste atributo.

Outro ponto importante que precisou ser solucionado no projeto foi o tratamento de termos de valores numéricos (Ex: lei 8.112). Uma vez que o analyzer do tipo “standard” não realiza nenhum tipo de tratamento em cima do ponto e apenas carrega o número como um todo ele gera, no exemplo acima, apenas os *tokens*: (lei, 8.112). Neste caso, se o usuário do sistema inserir a busca: (lei 8112 - sem o ponto) o Elasticsearch retorna um conjunto documental diferente do outro pois acaba calculando a pontuação (*_score*) dos documentos de forma diferente. Dessa forma, o Elasticsearch pontua a relevância dos documentos conforme o *token* solicitado na busca (ex: *token* 1 - “8.112” ou *token* 2 - “8112”).

4.2.2 Índice com “*Character Filter*” (Filtro de Caracteres)

A solução para o problema da geração de *tokens* na análise dos números dos normativos foi a utilização dos componentes “`char_filter`” na construção dos índices do Elasticsearch. Um filtro de caracteres, no Elasticsearch, é um componente da análise que realiza um pré-processamento do fluxo de texto antes que ele seja passado ao tokenizador. Esse componente adiciona, remove ou

modifica caracteres conforme a funcionalidade estabelecida do `char_filter` ou conforme as regras estabelecidas para o componente de forma personalizada.

Os primeiros testes foram realizados sobre um componente `char_filter` do tipo “*Pattern Replace*” (12) que utiliza expressões regulares específicas para detectar e trocar caracteres. O trecho de código apresentado abaixo apresenta um filtro de caracteres do tipo patern_replace que detecta dois grupos numéricos separados por um ponto e os unifica retirando o ponto, ou seja, transforma “x.y”em “xy” onde “x” e “y”são valores numéricos.

```
"char_filter": {
   "my_char_filter": {
         "type": "pattern_replace",
         "pattern": "(\\d+)\\.(\\d+)",
         "replacement": "$1$2"
   }
}
```

Código 13. [JSON] Implementação de filtro de caracteres do tipo Pattern Replace.

Outro problema que precisou ser tratafoi foi o caso de digitação do termo de busca sem caracteres típicos no português brasileiro como [ç, á, ã, ó, ú, etc]. Ou seja, caso o usuário queira buscar o termo “*acórdão*”, por exemplo, mas digitar apenas o termo de busca “*acordao*”, sem os acentos, só seriam retornados documentos onde os *tokens* tivessem sido mapeados sem acento. Foi necessária a construção de um filtro de caracteres mais robusto e versátil para que houvesse essa adaptação à experiência do usuário em casos deste tipo.

Uma solução preliminar foi a utilização de um filtro do tipo “*Mapping*” (13). Este filtro de caracteres detecta termos analisados, seja na indexação ou na busca conforme a configuração dos componentes analisadores, e os mapeia para outro termo pré-definido.

A estrutura a seguir apresenta uma das implementações do filtro de caracteres para contornar prováveis problemas de digitação dos usuários. Além disso, a primeira linha da matriz de mapeamentos (". => ",) realiza a remoção dos pontos nos números dos normativos, assim como o filtro de caracteres anterior do tipo `pattern_replace`.

```
"char_filter" : {
   "ajustabusca" : {
      "type" : "mapping",
      "mappings" : [
         ". => ",
         "acordao => acórdão",
         "acordão => acórdão",
         "instrucao => instrução",
         "instruçao => instrução",
         "oficio => ofício",
         "orientacao => orientação",
         "orientaçao => orientação",
         "provisoria => provisória",
         "resolucao => resolução",
         "resoluçao => resolução",
         "tecnica => técnica",
         "sumula => súmula"
         ]
      }
   }
}
```

Código 14. [JSON] Implementação de filtro de caracteres do tipo Mapping.

4.2.3 Índice com Filtros Adicionais

Por fim, uma implementação mais elaborada utilizada para resolver, de outra maneira, os casos de erros de digitação é a utilização de filtros do tipo "*Stemmers*". Estes filtros são construídos especificamente para cada tipo de linguagem e buscam os radicais das palavras ao invés das palavras completas. Dessa forma, a busca se torna mais "inteligente" retornando palavras similares. Uma busca por "aposentados", por exemplo, pode retornar documentos que contenham termos de radicais similares como "aposentada, aposentar, aposentadoria, etc".

Em combinação com filtros de caracteres foi gerado um índice um pouco mais versátil e flexível para os usuários. As configurações abaixo apresentam a combinação de filtros de analisadores e de caracteres para gerar este índice desejado.

```
"analysis" : {
      "filter" : {
```

```
            "lowercase" : {
                  "type" : "lowercase"
            },
            "br_stemmer" : {
                  "name" : "brazilian",
                  "type" : "stemmer"
            },
            "br_stop" : {
                  "type" : "stop",
                  "stopwords" : "_brazilian_"
            }
      },
      "analyzer" : {
           "default_search" : {
                  "filter" : [
                        "br_stop",
                        "lowercase",
                        "br_stemmer"
                  ],
                  "char_filter" : [
                        "delponto"
                  ],
                  "type" : "custom",
                  "tokenizer" : "standard"
            },
           "default" : {
                  "filter" : [
                        "br_stop",
                        "lowercase",
                        "br_stemmer"
                  ],
                  "char_filter" : [
                        "delponto"
                  ],
                  "type" : "custom",
                  "tokenizer" : "standard"
           }
      },
      "char_filter" : {
            "delponto" : {
                  "type": "pattern_replace",
                  "pattern": "\\.",
                  "replacement": ""
            }
      }
}
```

Código 15. [JSON] Implementação de índice com filtros adicionais.

Na implementação final utilizada foram combinados os filtros"`lowercase, br_stemmer, br_stop`" (conforme a documentação de referência do Elasticsearch 6.7 – *Token Filters*) (14) que realizam um pré processamento do fluxo de texto antes da geração de *tokens*. Nesta implementação de índice o fluxo de texto passa pelas seguintes transformações antes da geração de *tokens*:

- todo o texto é transformado para letras minúsculas;
- os radicais das palavras para a língua portuguesa do Brasil são detectados e adicionados;
- os acentos, cedilhas e outros tipos de sinais gráficos são removidos;
- as palavras de parada (conjunções, preposições, artigos, etc) da língua portuguesa do Brasil são removidas;
- e, por fim, todos os pontos são removidos dos *tokens* gerados.

Observa-se que apesar de todo o tratamento realizado no fluxo de texto durante o processamento e geração do índice de *tokens* no Elasticsearch, o conteúdo dos documentos, em si, são armazenados da forma original que foram inseridos, permitindo que os usuários recebam o conteúdo original mas realizem suas buscas sobre o conteúdo tratado.

4.2.4 Processamento em lote dos arquivos JSON no Elasticsearch

O processamento dos arquivos foi realizado utilizando-se um algoritmo desenvolvido em Python para realização do carregamento em lote, mas isto só deve ser realizado após o índice ter sido estruturado e adicionado ao Elasticsearch por meio do Kibana. Os trechos mais relevantes do algoritmo são apresentados abaixo e o conteúdo completo pode ser consultado ao final nos anexos desta documentação.

```
(...)

# Define a function to bulk index the contents of a file.
def index(filename):
      f = open(filename, 'r', encoding='utf-8')
      logger.info('Bulk indexing file {0}...'.format(filename))
```

```
      response = requests.post('http://localhost:9200/legis/_bulk',
data=f.read().encode('utf-8'), headers={"Content-Type":
"application/json"}).json()
      f.close()
      flog = open('indexing.log', 'a')
      flog.write('{0}\n'.format(json.dumps(response)))
      flog.close()

(...)

# Bulk indexing.
logger.info("Starting bulk indexing...")
pool = ThreadPool(2)
pool.map(index, [os.path.join(sys.argv[1], filename) for filename in
os.listdir(sys.argv[1])])
logger.info("Finished")
```

Código 16. [Python] Processamento em lote dos arquivos JSON no Elasticsearch.

O método "`response = requests.post('...`" acima solicita ao Elasticsearch que o comando `_bulk` (que prepara o Elasticsearch para o recebimento de documentos em lote) seja ativado para o índice legis. Já a função `pool.map`, ao final inicial o processamento em paralelo de vários arquivos solicitando que estes sejam carregados no método acima.

O algoritmo acima é desenvolvido sobre os métodos request e response, devido ao Elasticsearch trabalhar sobre a plataforma REST e utiliza a biblioteca `json` para tratamento e envio dos dados no padrão correto para a aplicação.

4.3 Construção do algoritmo de busca (*Query*)

Assim como a parametrização do índice pode ser construída de muitas formas diferentes, o Elasticsearch também permite uma parametrização elaborada e complexa do algoritmo de busca adicionando condições booleanas, adicionando pesos nos atributos de busca e uma miríade de outras opções diferentes.

A ordenação padrão dos resultados no Elasticsearch é definida pelo parâmetro "*_score*" que define a pontuação de relevância do documento de acordo com o algoritmo de busca definido. Assim, a partir de determinados termos inseridos pelo usuário e dos parâmetros definidos no algoritmo de busca, o seu índice calcula

as pontuações equivalentes para os documentos contidos em sua biblioteca e ordena estes documentos em ordem decrescente pela pontuação (“*_score*”).

O desenvolvimento e a aplicabilidade das funções para definição de relevância são o ponto chave do desenvolvimento de motores de busca modernos, como os que podem ser implementados pela ferramenta Elasticsearch, e a sua principal evolução em comparação com as consultas simples sobre bancos de dados como aparenta ser a estrutura de busca atual do Conlegis.

No desenvolvimento do SRI Conlegis, o algoritmo de busca é passado ao índice do Elasticsearch por meio da interface da aplicação. A estruturação e os testes do algoritmo utilizado foram realizados diretamente em JSON por meio da ferramenta Kibana. Segue abaixo o código utilizado:

```
POST legis2/_search
{
"size" : 60,
"_source": ["filename","titulo", "tipo_normativo", "sit_normativo",
"fileurl", "ementa"],
  "query": {
    "multi_match" : {
      "query" : <termo_de_busca>,
      "type" : "most_fields",
         "fields" : [
           "ementa^4",
           "dados^3",
           "titulo^7",
           "tipo_normativo^2",
           "_all"
          ]
      }
  },
  "highlight" : {
    "fields" : {
      "dados" : {
        "number_of_fragments" : 3,
        "pre_tags": [ "<mark>" ],
        "post_tags": [ "</mark>" ]
      }
    }
  }
}
```

Código 17. [JSON] Implementaçao do algoritmo da query no Kibana/Elasticsearch.

A busca apresentada acima, de modelo "multi_match" (15) permite que os *tokens* gerados dos termos de busca sejam comparados com diversos campos dos documentos simultaneamente. O tipo do multi_match apresentado como "most_fields" cruza os termos com os campos solicitados e retorna uma combinação da pontuação obtida em cada um deles. Outros tipos que podem ser utilizados são "best_fields" que retorna a maior das pontuações obtida entre os campos e "cross_fields" que primeiramente compara os termos individualmente e, em seguida, compara os termos em todos os campos como se estes fossem um único campo unificado.

O algoritmo acima também apresenta pesos diferentes que devem ser considerados para a pontuação dos documentos. O trecho "ementa^4", "dados^3", "titulo^7", "tipo_normativo^2, informa ao índice que a "*ementa"* deverá ter sua pontuação elevada à quarta potência, o atributo "*dados"* à terceira, o atributo "*título*" à sétima e o "*tipo de normativo"* deverá ser elevado à segunda potência. Esta funcionalidade permite que as pontuações sejam manipuladas de acordo com o conhecimento do desenvolvedor sobre a importância dos campos. Neste exemplo, o campo mais importante a ser considerado será o *titulo* do arquivo, em seguida a *ementa* e assim por diante.

A possibilidade de montar diferentes estratégias conforme cada tipo de aplicação é essencial para a composição de um SRI eficiente para o usuário final. No caso da implementação acima foram consideradas as seguintes premissas para composição da estratégia acima:

- Caso o usuário saiba exatamente o tipo e número, ou pelo menos o número, do ato normativo que deseja buscar ele deverá inserir estes termos de busca e espera que a consulta retorne o seu ato normativo em uma das primeiras posições, por isso o título deve ter o maior peso dentre os campos buscados;
- Caso o usuário deseje buscar atos normativos por determinado assunto, deverá colocar os termos chaves do assunto que deseja buscar. Os normativos que apresentam estes termos, ou termos de radicais similares (ver Stemmer, tópico 4.2.3) na ementa deverão ser melhor pontuados que os

outros documentos que têm os mesmos termos dentro dos dados completos do ato normativo

- O tipo do normativo é adicionado com um peso potencializado caso o usuário queira reforçar que deseja informações de certo tipo específico de documento. Neste caso, ele deverá adicionar o termo na caixa de busca em conjunto com quaisquer outros termos que desejar e as pontuações daqueles tipos de documento deverão ser aumentadas.

O campo final que apresenta configurações de parâmetro de "*highlight*" é uma funcionalidade interessante do Elasticsearch que já traz marcações automáticas dos trechos de determinado campo, no caso do campo "dados", onde o termo de busca é encontrado. Esta funcionalidade será melhor entendida no tópico 4.4.2.1 seguinte que trata da implementação da interface do SRI Conlegis com *Highlights*.

4.3.1 Algoritmos de busca booleanos

O uso de *queries* booleanas (16), no Elasticsearch, consiste na combinação de múltiplas consultas diferentes, condicionadas conforme a necessidade da aplicação e vinculadas entre si. A utilização desses tipos de *queries*, implementadas por meio do termo, combina, une, remove ou intersecciona grupos de resultados diferentes de acordo com regras pré-estabelecidas para cada uma das *queries* que são encapsuladas com a aplicação dos termos "`must, must_not, should e filter`". Os resultados retornados pelas *queries* encapsuladas em "`must`" são de retorno obrigatório e tem sua pontuação aumentada, os resultados retornados em "`must_not`" devem ser removidos, aqueles retornados sob o termo "`should`" podem ou não ser retornados mas podem contribuir para a pontuação de relevância e os resultados retornados sob o termo "`filter`" são como cláusulas "`must`" mas não tem sua pontuação computada.

Em uma linguagem mais lógica é como se o calculo da pontuação dos documentos resultantes fosse calculado por meio de um fórmula onde as cláusulas "`must`" equivalessem ao operador AND, as "`should`" ao operador OR e as "`must_not`" ao operador NOR.

O código abaixo foi utilizado para a adição de filtros adicionais nos resultados de busca utilizando apenas cláusulas "`must`" encadeadas e funciona com a seguinte lógica:

- A primeira *query* (*must*) deve retornar obrigatoriamente os documentos que são relevantes para os ***<termos_de_busca>*** solicitados;
    - Dentre estes resultados devem ser retornados apenas aqueles que possuem o tipo de normativo exatemente igual ao ***<filtro_tipo_normativo>;***
    - Dentre estes resultados devem ser retornados apenas aqueles que possuem o tipo de normativo exatemente igual ao ***<filtro_situacao_normativo>.***

```
POST legis2/_search
{
"size" : 60,
"_source": ["filename","titulo", "tipo_normativo", "sit_normativo",
"fileurl", "ementa" ],
  "query": {
    "bool" : {
      "must" : [{
        "multi_match" : {
          "query" : "<termos_de_busca>" ,
          "type" : "most_fields",
          "fields" : [
            "ementa^4",
            "dados^3",
            "titulo^7",
            "tipo_normativo^2",
            "_all"
          ]
        }
      },{
      "bool": {
        "must" : [
          { "term" : { "tipo_normativo.keyword" :
"<filtro_tipo_normativo>" } },
          { "term" : { "sit_normativo.keyword" :
"<filtro_situacao_normativo>" } }
        ]
      }}
    ]
    }
  },
```

```
  "highlight" : {
    "fields" : {
      "dados" : {
        "number_of_fragments" : 3,
        "pre_tags": ["<mark>"],
        "post_tags": ["</mark>"]
      }
    }
  }
}
```

Código 18. [JSON] Implementação de Query Booleana com Filtros.

Esta estratégia de *query* deve ser implementada em conjunto com a interface HTML que deverá apresentar os filtros de “tipo de normativo” e de “situação de normativo” existente no conjunto de documentos. Além disso, deve ser estruturada uma forma dinâmica de se adicionar ou retirar o filtro conforme isto foi solicitado ou não pelo usuário.

A implementação HTML/Javascript dos filtros adicionais será apresentada no item 4.4.2.2 que explica a implementação da interface HTML com filtros de Tipo e de Situação de Normativo.

4.4 Interface HTML do SRI Conlegis

A implementação da interface gráfica de busca para o usuário foi realizada utilizando duas plataformas que rodam em cima da linguagem Javascript. A tecnologia base utilizada foi o Node.js (Anexo I, Item 7) que funciona como uma plataforma de aplicação que compila e interpreta rotinas desenvolvidas em Javascript. Devido a idade e versatilidade da linguagem Javascript, a quantidade de aplicações, bibliotecas e módulos pré construídos é muito grande e suas aplicações são fáceis de serem implementadas.

Além disso, o Node.js é extremamente leve e funciona muito bem para serviços que rodam sobre o protocolo http e que trafegam basicamente texto puro. Ainda, ele é multiplataforma o que facilita a sua integração posterior a qualquer tipo de sistema já estruturado.

Sobre o Node.js foram adicionados, ao projeto, pacotes já desenvolvidos, livres e disponíveis na comunidade por meio do npm (Anexo I, Item 7.1). O npm é um grande repositório e gerenciador de pacotes pré-moldados e construídos em Javascript. A partir da instalação do Node.js, os pacotes disponíveis no Npm podem ser facilmente baixados e instalados, por aplicação, via linha de comando.

Para o desenvolvimento do SRI Conlegis foi instalado o Node.js e, dentro da pasta de trabalho do projeto, foi criada uma pasta para a hospedagem da aplicação de serviço. Dentro desta pasta foram instalados os pacotes, disponíveis no npm: “express”, “body-parser” e “elasticsearch” (Anexo I, Item 7.2). Estes pacotes serão executados no lado do servidor e tem as seguintes funções:

- **express**: é uma biblioteca que monta um servidor web leve e simples de ser configurado.
- **body-parser**: biblioteca que trabalha em conjunto com a Express para tratamento do conteúdo dos dados enviados nas solicitações HTTP. Esta biblioteca será importante para facilitar a integração entre o Express e o Elasticsearch.
- **elasticsearch**: neste caso estamos falando da biblioteca Npm Javascript para conexão com o Elasticsearch. Esta biblioteca funciona como cliente que irá acessar e receber os dados da aplicação Elasticsearch que está rodando o índice do SRI Conlegis.

Além disso, a interface gráfica utilizará alguns scripts adicionais que serão executados em tempo de execução, no lado do cliente (Anexo I, Item 7.3). São eles:

- **axios**: traz funcionalidades adicionais para solicitações e respostas HTTP, principalmente no processo de transformação de dados para JSON.
- **vue**: pacote de ferramentas que tornam a página dinâmica e trabalham com os resultados de resposta do Elasticsearch de forma rápida, bonita e muito prática.

A implementação final da interface para o usuário foi construída em cima de dois componentes principais que, juntos, compõe a interface web de busca

utilizada para o SRI Conlegis. Um código Javascript que roda sobre a plataforma do Node.js e uma página HTML que serve de interface para o usuário.

4.4.1 Conexão da interface HTML ao Elasticsearch

O código Javascript - *index.js* - é responsável pela abertura e sustentação do túnel entre o Elasticsearch e a aplicação web assim como também recebe os inputs do usuário e os envia em formato de busca JSON estruturada para ser analisado e respondido pelo índice de documentos que foi estruturado. Este componente inicia o servidor *express* e carrega as bibliotecas *body-parser* e *elasticsearch* (conector para o servidor Elasticsearch disponibilizado via npm) para que o serviço possa interoperar com o índice de documentos que foi previamente construído.

O serviço iniciado pelo *index.js* interopera com o Elasticsearch por meio de *Cross-Origin Resource Sharing* (CORS) (17), do inglês, “Compartilhamento de Recursos de Origem Cruzada”. Como apresentado na Wikipedia:

> *“CORS é uma especificação de uma tecnologia de navegadores que define meios para um servidor permitir que seus recursos sejam acessados por uma página web de um domínio diferente. Esse tipo de acesso seria de outra forma negado pela “same origin policy”. CORS define um meio pelo qual um navegador e um servidor web podem interagir para determinar se devem ou não utilizar/permitir requisições cross-origem. É um acordo que permite grande flexibilidade, mas é mais seguro que permitir todos as requisições desse tipo.”*

Para fins acadêmicos de desenvolvimento desta aplicação, a política de CORS foi configurada dentro do arquivo *config.yml* do Elasticsearch permitindo que apenas meu servidor express seja capaz de acessar os dados do índice gerado, entretanto, no estabelecimento de conexão com o serviço por meio do módulo elasticsearch do node.js é enviada no cabeçalho da página uma conexão com permissão para qualquer origem. Além disso, os métodos permitidos podem, também, ser limitados apenas a requisições de consulta. Em uma implementação posterior deste sistema, deve ser avaliada a melhor forma de configuração das políticas de acesso CORS uma vez que a liberação total de acesso pode expor os

dados dos documentos e o cluster do Elasticsearch a outras ações além de consulta, como inserção de documentação ou alteração de parâmetros do índice.

```
http.cors.allow-origin: "http://localhost:3001"
http.cors.allow-headers : "X-Requested-With, X-Auth-Token, Content-Type,
Content-Length, Authorization"
http.cors.allow-credentials: true
http.cors.allow-methods: "OPTIONS, HEAD, GET, POST, PUT, DELETE"
```

Código 19. [YAML] Parâmetros do config.yml no Elasticsearch.

```
// use the bodyparser as a middleware
app.use(bodyParser.json())
// set port for the app to listen on
app.set( 'port', process.env.PORT || 3001 );
// set path to serve static files
app.use( express.static( path.join( __dirname, 'public' )));
// enable CORS
app.use(function(req, res, next) {
  res.header("Access-Control-Allow-Origin", "*");
  res.header('Access-Control-Allow-Methods', 'PUT, GET, POST, DELETE,
OPTIONS');
  res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With,
Content-Type, Accept");
  next();
});
```

Código 20. [Javascript] Solicitação de conexão CORS na aplicação.

4.4.2 Camada de apresentação HTML

Finalmente o processo de desenvolvimento chega ao componente principal para o usuário final, a Interface Gráfica.

No SRI Conlegis a interface foi construída em código HTML combinado com funcionalidades Javascript, em particular, utilizando os pacotes *vue* e *axios.* Como a implementação de marcações altera bastante o tamanho das caixas de retorno optou-se por desenvolver duas versões diferentes da interface, uma que retorna os resultados sem as marcações, e outra com as marcações.

O código desenvolvido para a aplicação é leve e composto apenas pela estrutura mínima necessária para renderização da camada de apresentação do usuário, de modo que ela poderá ser facilmente integrada em outros sistemas web já

desenvolvidos, alinhando-se com a proposta deste projeto de integrar o mecanismo de busca do SRI Conlegis ao sistema oficial disponibilizado. O código está disponível, nas duas versões no Apêndice A – Itens 13 e 14.

A interface responde as entradas a cada novo caracter adicionado na caixa de busca e os resultados são retornados instantaneamente, e não há a necessidade de um comando de envio (normalmente na tecla "*Enter*"). Ainda a interface aguarda qualquer mudança de estado dos filtros e reenvia a busca apresentando o novo conjunto retornado, que no caso dos filtros de Tipo e Situação, será um subconjunto dos atos anteriores.

Figura 6. Inserção de: "le"

Fonte: Elaborado pelo autor

Devido a limitações de hardware, o máximo de documentos retornados são 60 atos normativos. Este é um parâmetro facilmente configurável no código integração entre o Elasticsearch e o express (Apêndice A – Item 12) e em uma integração oficial pode ser redimensionado de forma adequada a performance do servidor dedicado para a disponibilização do serviço.

Na figura abaixo o usuário já digitou os caracteres “lei 8” e o SRI já está respondendo conforme a estratégia de consulta elaborada. Em primeiro lugar retornou um ato normativo que possuía o número “8” no seu título e o tipo “Lei” na ementa. Na segunda posição e nas subsequentes está trazendo um normativo que teve pontuação definida pelo tipo solicitado e pela comparação dos termos com o conteúdo presente nas ementas dos documentos.

Figura 7. Inserção de: “lei 8”

A figura abaixo mostra o retorno após a inserção completa dos termos desejados, no caso, “lei 8112”. O SRI retornou, em primeiro lugar, a própria lei desejada e os itens subsequentes foram retornados conforme as pontuações combinadas dos termos encontrados nos campos título, ementa, dados e tipo de normativo, como explicado no item 4.3 que trata da estratégia de consulta.

Figura 8. Inserção de: “lei 8112”.

4.4.2.1 Implementação de *Highlights*

Conforme apresentado no tópico sobre o algoritmo de consulta item 4.3, foi solicitado ao índice do Elasticsearch que se fossem apresentados no resultado da consulta pequenos fragmentos de texto, “*highlights*”, onde fossem encontrados os

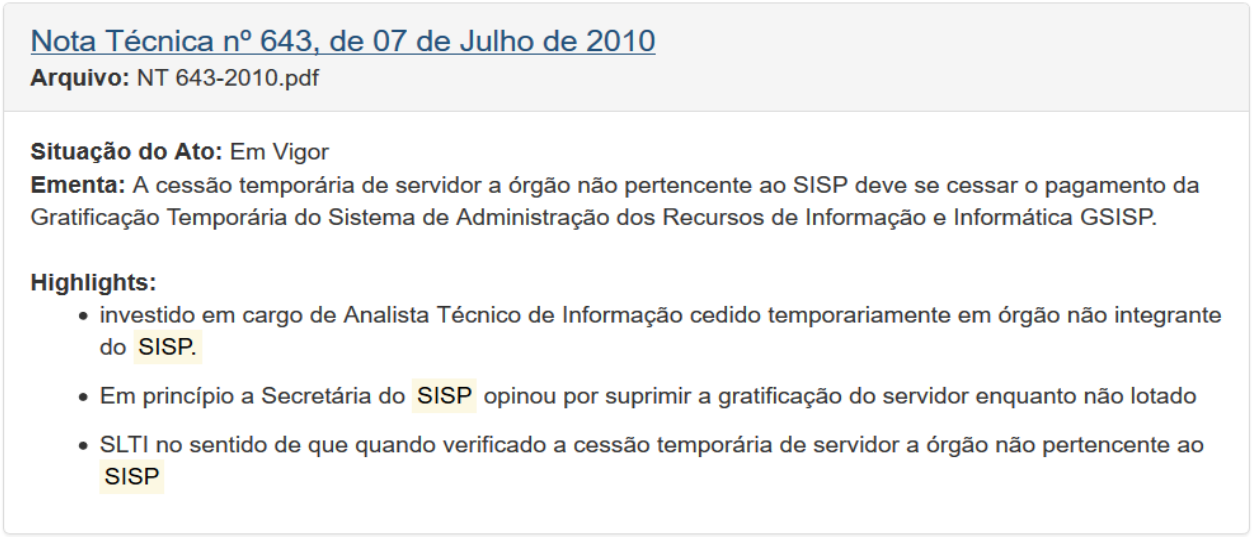
termos solicitados pelo usuário. Uma nova versão da implementação foi desenvolvida, e as figuras são apresentadas abaixo, com a utilização de *highlights*.

Como pode ser observado no código 18 foram solicitados um máximo de 3 fragmentos por texto, e as suas tags de marcação foram configuradas como *<mark>* e *</mark>* que renderizam na interface gráfica HTML o efeito de "caneta marca-texto". A figura 9 abaixo mostra a implementação das marcações (*highlights*) e também dos filtros de tipo e situação de normativos que serão abordados no item seguinte.

Figura 9. Inserção de: "lei 8112" na versão com highlights

A figura a seguir mostra mais um exemplo de como fica a estrutura de apresentação dos resultados de busca com *highlights*. Neste caso, apresenta-se o primeiro resultado retornado pelo termo de busca “*sisp*”.

Figura 10. Apresentação de um resultado de busca com highlights.



4.4.2.2 Implementação de filtros de Tipo e Situação do ato normativo

Os dados estruturados obtidos da base da dados do sistema original possuíam determinados campos bem estruturados e com uma qualidade de informação muito boa sendo possível a implementação de filtros adicionais que proporcionam uma melhor experiência para o usuário reduzindo a quantidade de normativos retornados para um subconjunto mais aderente à necessidade de pesquisa.

No SRI Conlegis, foram implementados dois filtros adicionais no momento da busca. O filtro de “*Tipo de Ato Normativo*” e o de “*Situação do Ato Normativo*”.

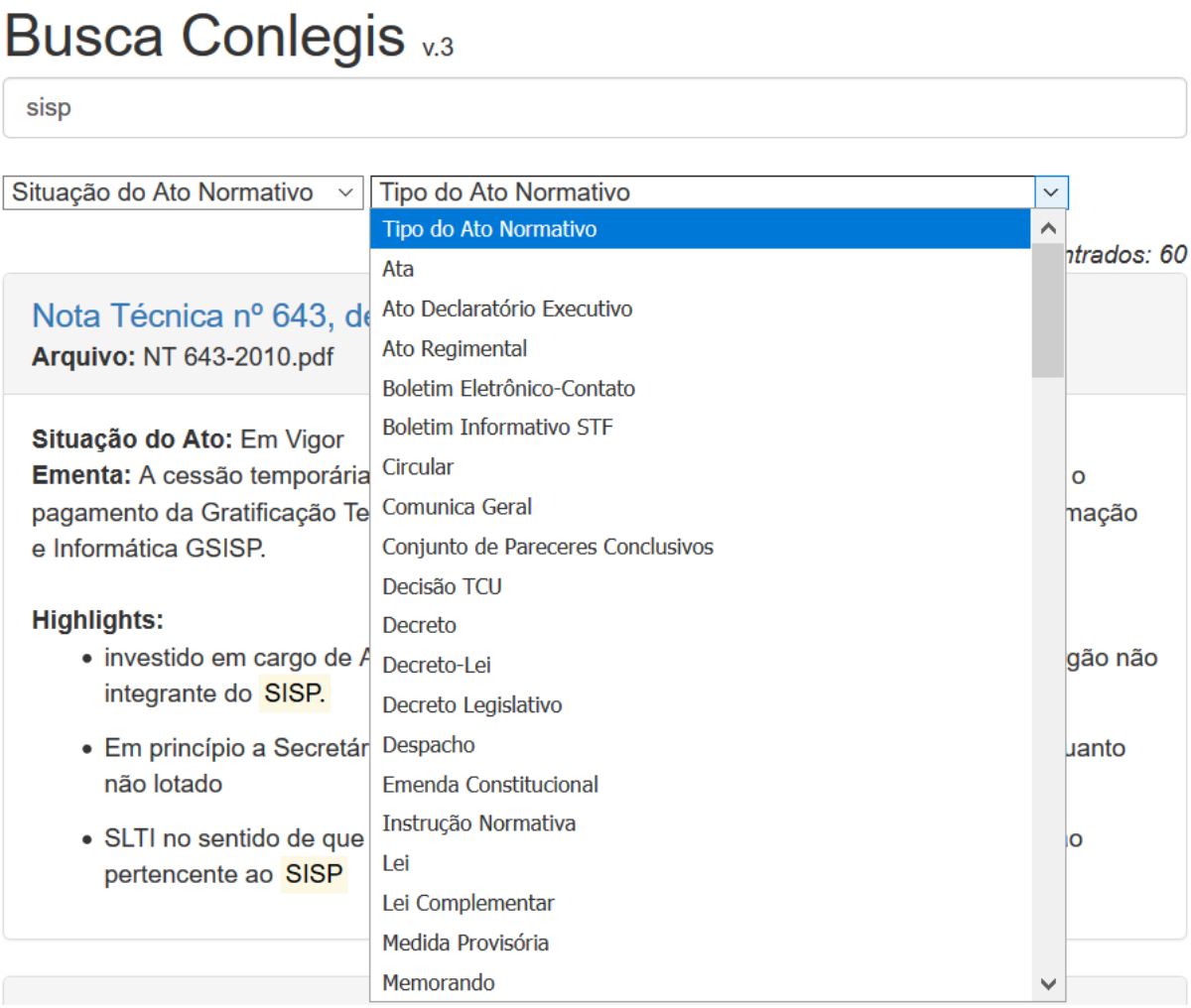
O filtro de Tipo categoriza os atos por seus tipos e, durante a utilização do sistema o usuário pode filtrar seus resultados para que sejam retornados apenas atos da categoria escolhida a qualquer momento. Da tabela de tipos da base de dados original do Conlegis, são disponibilizados os seguintes tipos:

- `Ata, Ato Declaratório Executivo, Ato Regimental, Boletim Eletrônico-Contato, Boletim Informativo STF,`

Circular, Comunica Geral, Conjunto de Pareceres Conclusivos, Decisão TCU, Decreto, Decreto-Lei, Decreto Legislativo, Despacho, Emenda Constitucional, Instrução Normativa, Lei, Lei Complementar, Medida Provisória, Memorando, Mesa Nacional de Negociação Permanente, Norma Interna, Norma Operacional, Nota Técnica, Ofício, Ofício-Circular, Ofício-Circular Conjunto, Orientação Consultiva, Orientação Geral, Orientação Normativa, Parecer, Parecer AGU, Parecer CONJUR, Portaria, Portaria Conjunta, Portaria Interministerial, Portaria Normativa, Projeto de Lei, Resolução, Resposta Fax, Resposta Ofício, Súmula Administrativa, Súmula da Advocacia-Geral da União, Nota Informativa, Norma de Execução, Lei Delegada, Deliberação, Constituição Federal E Emendas Constitucionais, Constituição Federal, Atos da AGU, Ação Direta de Inconstitucionalidade, Nota CONJUR, Nota AGU, Comunica SIAPE, Manual de Redação da Presidência da República, Exposição de Motivos, Exposição de Motivos Interministerial, Resolução Normativa, Comunica, Edital, Nota Técnica_, Recurso Extraordinário, Acórdão TCU_, Acórdão TCU, Acórdão, Anteprojeto de Lei, Ato, Súmula Vinculante, Nota Técnica Conjunta, emprego público, Nota, Formulação, Súmula, Diversos, Lei 8.112 - ANOTADA, Nota Informativa Conjunta, Portaria AGU, TESTE, Súmula do TCU, Nota Técnica Consolidada, PARECER/PGFN, PERFIL PROFISSIOGRÁFICO PREVIDENCIÁRIO - PPP, SÚMULA AGU, Decisão STF/Reclamação, Manual de Perícia Oficial, Agenda de Decisão SEGRT, Comissão de Ética, Memorando Circular, Portaria Gsiste, OFÍCIO/PGFN, Formulação/DASP, PARECER/DECOR/CGU/AGU., Termo Aditivo, Manual de Processo Administrativo

Disciplinar, Instrução Normativa Conjunta, Mensagem e Ato Declaratório.

Figura 11. Interface apresentando o filtro de Tipo expandido.



Uma evolução do filtro de tipos seria a disponibilização de apenas uma parte destas opções conforme algum critério pré-estabelecido, como uma pesquisa ou coleta de feedback implícito de quais tipos mais consultados, ou a aplicação de alguma regra, como a vigência ou atualidade do tipo de normativo.

A figura 9 mostra a interface do filtro e a figura 11, adiante, mostra a caixa seletora expandida.

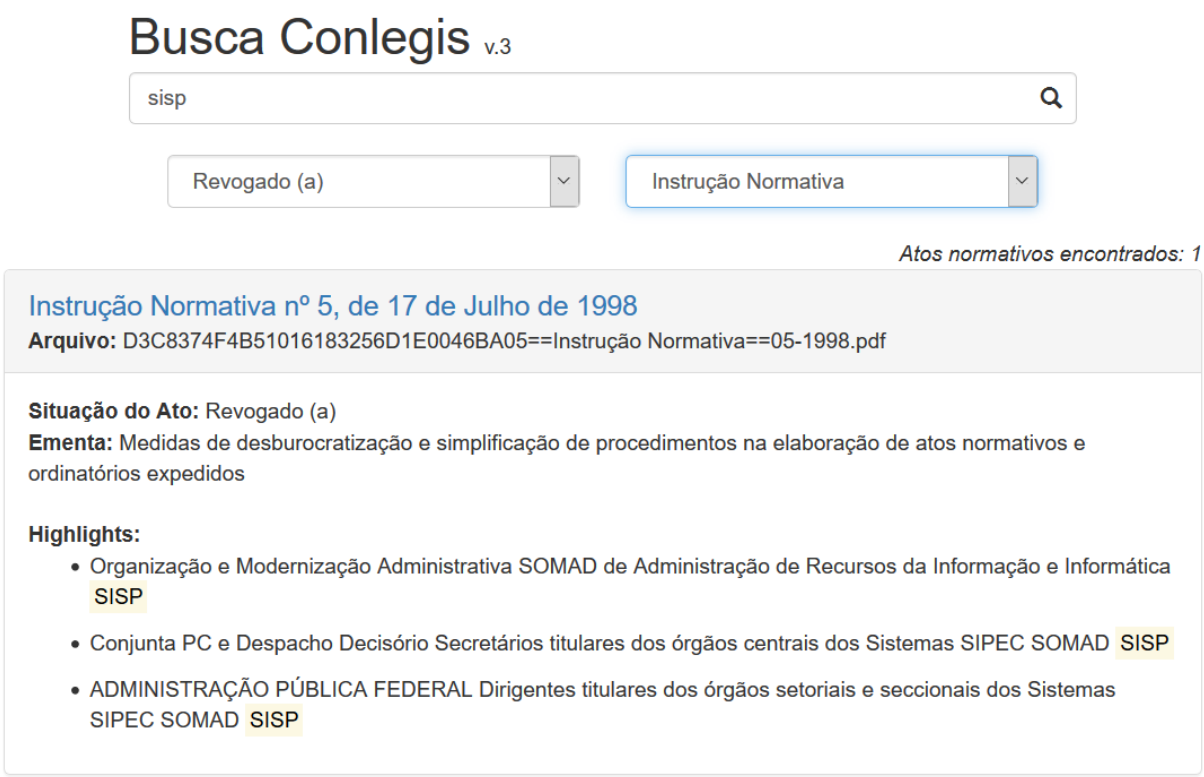
Já o filtro de Situação do ato normativo sinaliza a vigência da lei bem como os processos pelos quais determinado ato normativo já passou. Da tabela de

situação de ato normativo disponibilizada na base de dados original do Conlegis tem-se as seguintes situações:

- Em Vigor, Revogado (a), Reeditado (a), Retificado (a), Revogado e Reeditado, Republicado (a), Alterado (a), Rejeitado (a), Convertida em Lei, Revogado (a) Parcialmente, Prejudicado (a), Reeditada e Revogada, Sem Eficácia, Insubsistente, Suspensa a Eficácia, Revigorado(a), Insubsistente Parcialmente, Revisto e Atualizado, Aditado (a), Tornado (a) sem Efeito, Anulada, Revogado(a) Tacitamente e Vigência Encerrada.

A figura a seguir mostra o resultado da busca pelo termo "*sisp*" com os filtros aplicados de forma a mostrar apenas os atos normativos na situação "*Revogado (a)*" e do tipo "*Instrução Normativa*" resultando em apenas 1 ato que se enquadra neste cenário específico.

Figura 12. Busca por "sisp" com os dois filtros aplicados.



O desenvolvimento do código para implementação da interface apresentada foi uma derivação do código anterior, sem a utilização de filtros, porém com a adição de duas variáveis Javascript adicionais que são tratadas em tempo de execução e ajudam a formar a *query* de forma dinâmica. Esta *query* dinamicamente construída pela aplicação dos filtros, que no código são novas variáveis POST enviadas pelo formulário HTML, utiliza o algoritmo de consulta booleana apresentada no item 4.3.1.

O código abaixo apresenta a função que requisita a solicitação de busca junto ao Elasticsearch e como o corpo da consulta, que será enviado, é dinamicamente construído com os termos de busca e filtros. Observa-se que todo o algoritmo de *query*, desenvolvido e testado no Kibana, é encapsulado na variável chamada "*bodysearch3*" que é formada pela combinação dos resultados passados pela variável "*req.query[*]*".

```
app.get('/search3', function (req, res){

  body_tipo = '"match_all": {}'
  body_sit = '"match_all": {}'

  if (req.query['tipo_ato'] !== "Todos") {
    body_tipo =
    '"term" : { "tipo_normativo.keyword" : "' + req.query['tipo_ato'] +
'" }';
  }
  if (req.query['sit_ato'] !== "Todas") {
      body_sit =
      '"term" : { "sit_normativo.keyword" : "' + req.query['sit_ato'] +
'" }';
  }

  let bodysearch3 = '{' +
    '"size" : 60,' +
    '"_source": ["filename","titulo", "tipo_normativo", "sit_normativo",
"fileurl", "ementa" ],' +
    '"query": {' +
        '"bool" : {' +
            '"must" : [{' +
                '"multi_match" : {' +
                    '"query" : "' + req.query['q'] + '" ,' +
                    '"type" : "most_fields",' +
                    '"fields" : [' +
                        '"ementa^4",' +
                        '"dados^3",' +
```

```
                        '"titulo^7",' +
                        '"tipo_normativo^2",' +
                        '"_all"' +
                        ']' +
                    '}' +
                '},{' +
                '"bool": {' +
                    '"must" : [' +
                        '{'+ body_tipo + '},' +
                        '{'+ body_sit + '}' +
                    ']' +
                '}}' +
                ']' +
            '}' +
    '},' +
    '"highlight" : {' +
        '"fields" : {' +
            '"dados" : { ' +
                '"number_of_fragments" : 3, ' +
                '"pre_tags": ["<mark>"], ' +
                '"post_tags": ["</mark>"] ' +
            '}' +
        '}' +
    '}' +
'}'
  client.search({ index:'legis2', body:bodysearch3,
type:'atosnormativos' })
  .then(results => {
    res.send(results.hits.hits);
  })
  .catch(err=>{
    console.log(err)
    res.send([]);
  });
})
```

Código 21. [Javascript] Código da implementação query booleana na aplicação.

Observa-se que quando os valores dos filtros são deixados no seu item genérico, de valores “Todos” e “Todas” a cláusula booleana interna retorna todos os documentos, dentro daqueles compostos pela *query* `multi_match` por meio da *subquery* `"match_all": {}`. Contudo, uma vez que os filtros são modificados para qualquer outro dos valores pré-definidos eles irão compor a *query* utilizando `"term":{"tipo_normativo.keyword": req.query['tipo_ato']` ou `"term":{"sit_normativo.keyword": req.query['sit_ato']` dentro da

booleana interna. O trecho de código acima é implementado no *index.js* do lado interno do servidor.

Pelo lado externo da aplicação, a parte da interface web e os trechos de javascript, também foram adaptados para contemplar o novo conjunto de variáveis que é enviado. O trecho abaixo é implementado no código *template2.html* e no *template3.html* (com *highlights*) (disponíveis para consult no Apêndice A – Itens 13 e 14).

```
<script>
//template.html
// create a new Vue instance
var app = new Vue({
  el: '#app',
    // declare the data for the component (An array that houses the
results and a query that holds the current search string)
  data: {
        results: [],
        query: '',
      q_sit: 'Todas',
            q_tipo: 'Todos'
    },
    // declare methods in this Vue component. here only one method which
performs the search is defined
    methods: {
        // make an axios request to the server with the current search
query
        search: function() {
                  if (typeof sit_ato === 'undefined') {
                        situacao = "Todas";
                  } else {
                        situacao = sit_ato.value;
                  }
                  if (typeof tipo_ato === 'undefined') {
                        tipoato = "Todos";
                  } else {
                        tipoato = tipo_ato.value;
                  }
            axios.get("http://localhost:3001/search3?q=" + this.query +
"&sit_ato=" + sit_ato.value + "&tipo_ato=" + tipo_ato.value)
                .then(response => {
                    this.results = response.data;
                })
        }
    },
    // declare Vue watchers
    watch: {
```

```
        // watch for change in the query string and recall the search
method
        query: function() {
            this.search();
        },
            q_sit: function() {
                        this.search();
        },
            q_tipo: function() {
                        this.search();
        }
    }
})
```

Código 22. [Javascript] Algoritmo de construção do formulário dinâmico.

4.5 Comparativo das soluções

Este tópico se propõe a apresentar um breve comparativo entre as interfaces de busca, assim como dos resultados retornados, do sistema oficial do Conlegis e o novo SRI desenvolvido sobre a plataforma Elasticsearch para avaliar as diferenças entre as duas soluções. Com relação aos mecanismos de busca utilizados nos dois casos, a solução atual utilizada pelo Conlegis aparentemente deriva de uma consulta direta ao banco via *queries* SQL tradicionais enquanto o novo SRI opera sobre motor de busca do Elasticsearch.

4.5.1 Interface gráfica para o usuário

A solução atual do Conlegis solicita um conjunto de informações adicionais antes de consultar a documentação existente como o tipo de perfil do usuário – Unidade de Gestão de Pessoas, Servidor, Aposentado, Pensionista, Estudante ou Outros – e o preenchimento de um *captcha*. Além disso são disponibilizados três modelos de consulta e todos possuem interfaces complexas de utilização conforme apresentado nas figuras abaixo.

Figura 13. Pesquisa Textual do Conlegis.

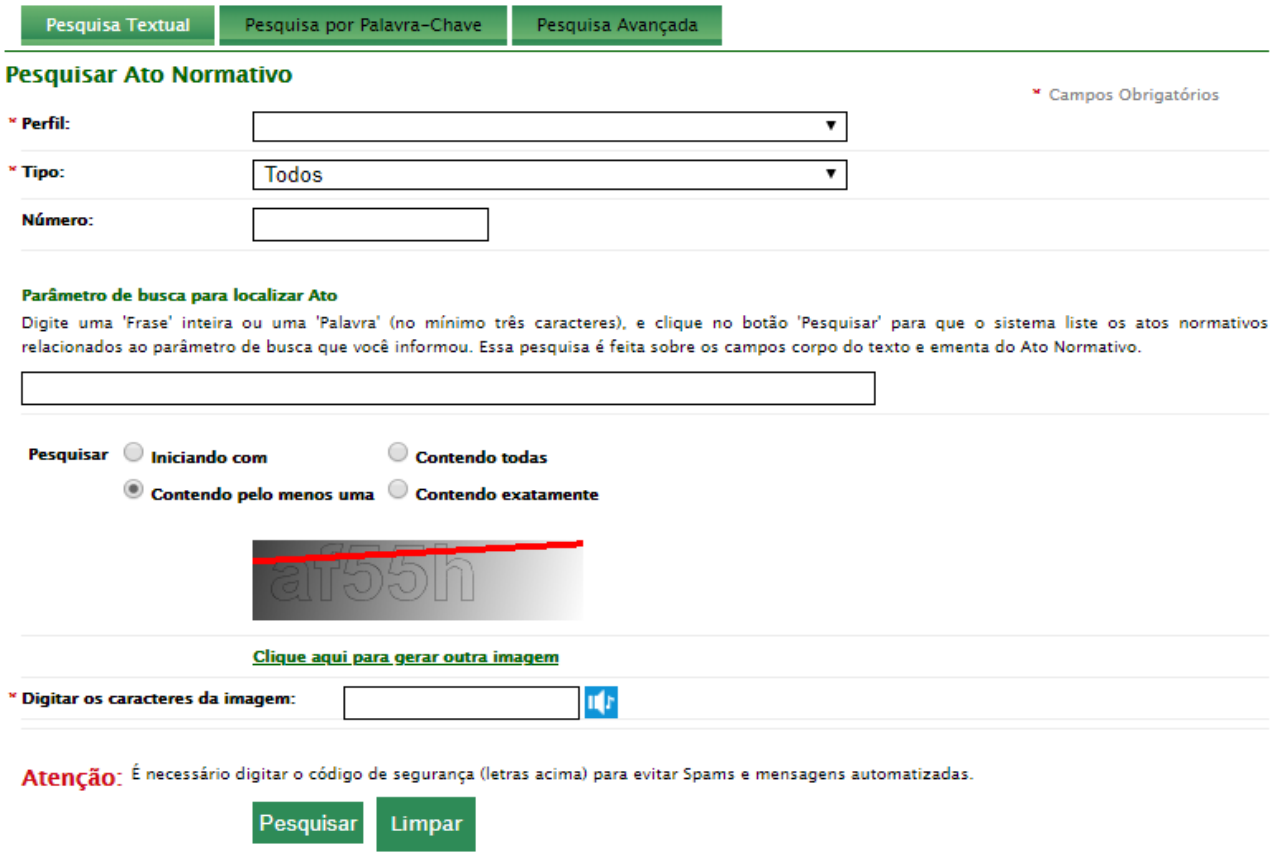


Figura 14. Pesquisa por Palavra-Chave do Conlegis.

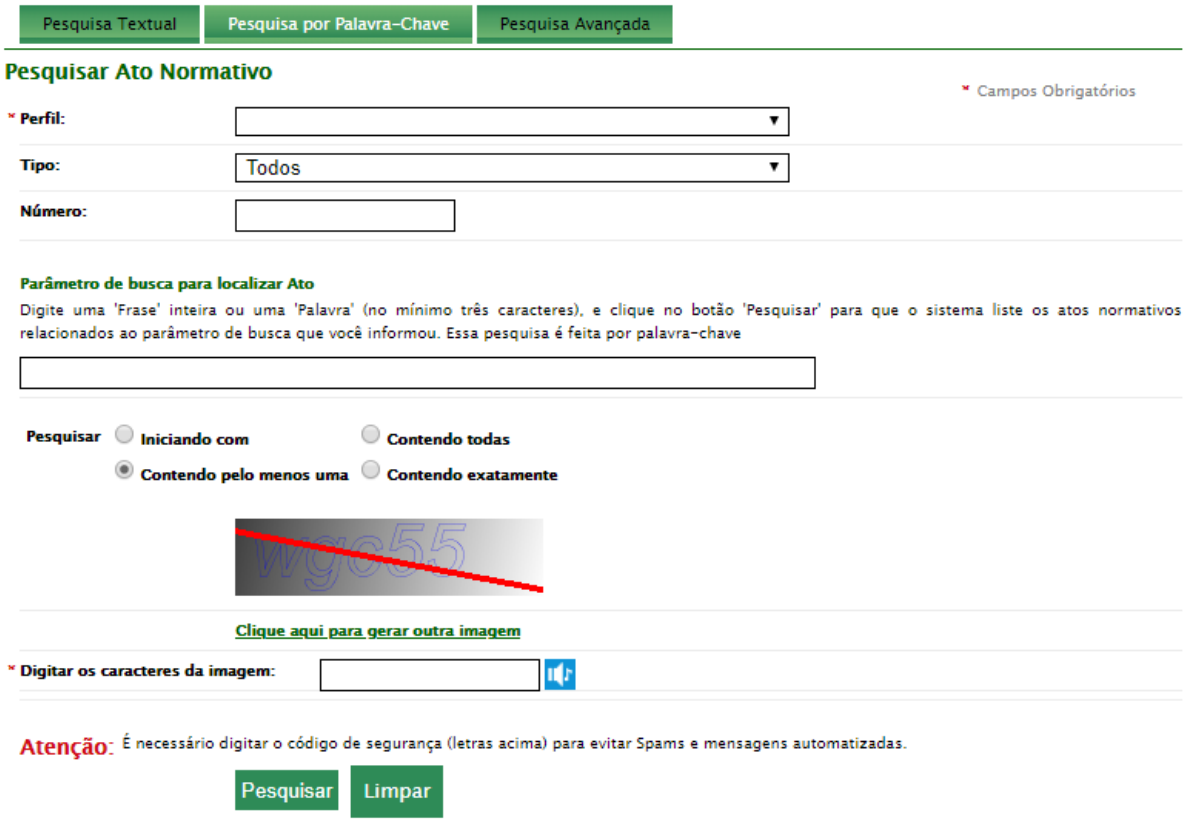
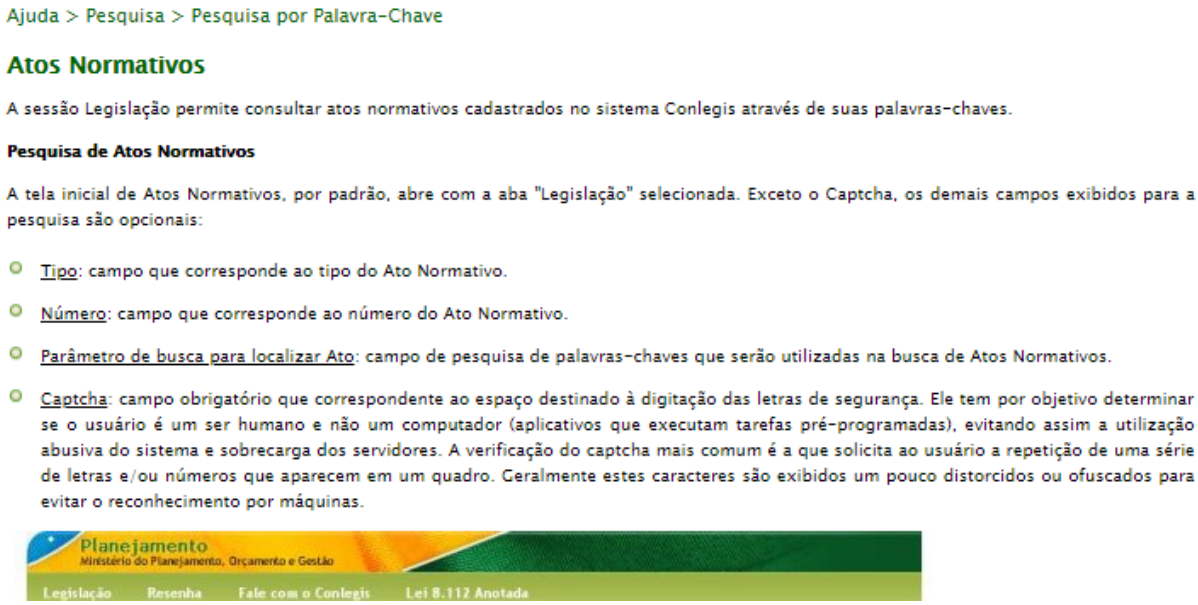
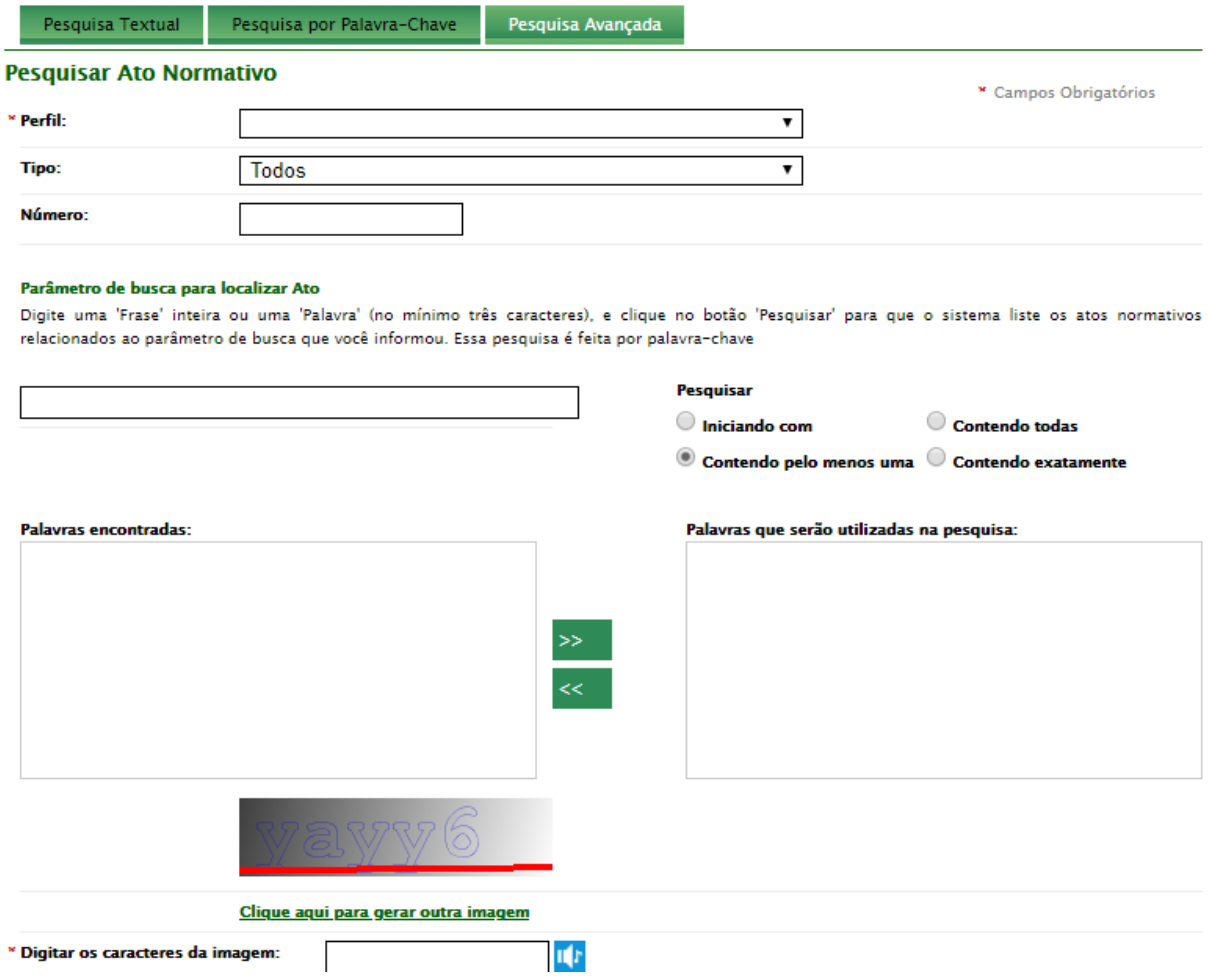


Figura 15. Manual de usuário para a Pesquisa por Palavra-Chave.



A utilização dos filtros não é explicada de forma intuitiva na página e é preciso a utilização do Manual do Usuário (figura 15) para se aprender a utilizar as diversas formas de pesquisa disponíveis no sistema.

Figura 16. Pesquisa Avançada do Conlegis.



A aplicação desenvolvida, o novo SRI Conlegis, se propõe a ser uma simples caixa de busca disponibilizada para todos os perfis de usuários. Além disso ela não possui nenhum outro componente adicional que não a própria busca e os filtros adicionais e uma área de resposta dos resultados diretamente abaixo para que possa ser integrada a qualquer outro sistema web de forma prática e simples.

Durante o processo de busca do SRI Conlegis todos os atributos disponíveis são verificados em conjunto e a estratégia de relevância é determinada pelos algoritmos e parametrizações desenvolvidos durante o desenvolvimento da solução. Não é necessário que o usuário especifique que tipo de termo ele está buscando (se é número, texto, palavra-chave, etc).

Figura 17. Caixa de pesquisa do SRI Conlegis.

4.5.2 Apresentação dos resultados da busca

A pesquisa textual padrão pelo termo "**8112**" no campo número do sistema atual retorna apenas um conjunto de 9 (nove) normativos. Os links dos normativos são adicionados a pequenos trechos cortados das Ementas e levam para outra página que apresenta um conjunto maior de informações sobre o normativo, incluindo o link para download do documento oficial. Os normativos são sempre ordenados por ordem decrescente de ano. O resultado é apresentado conforme as figuras a seguir.

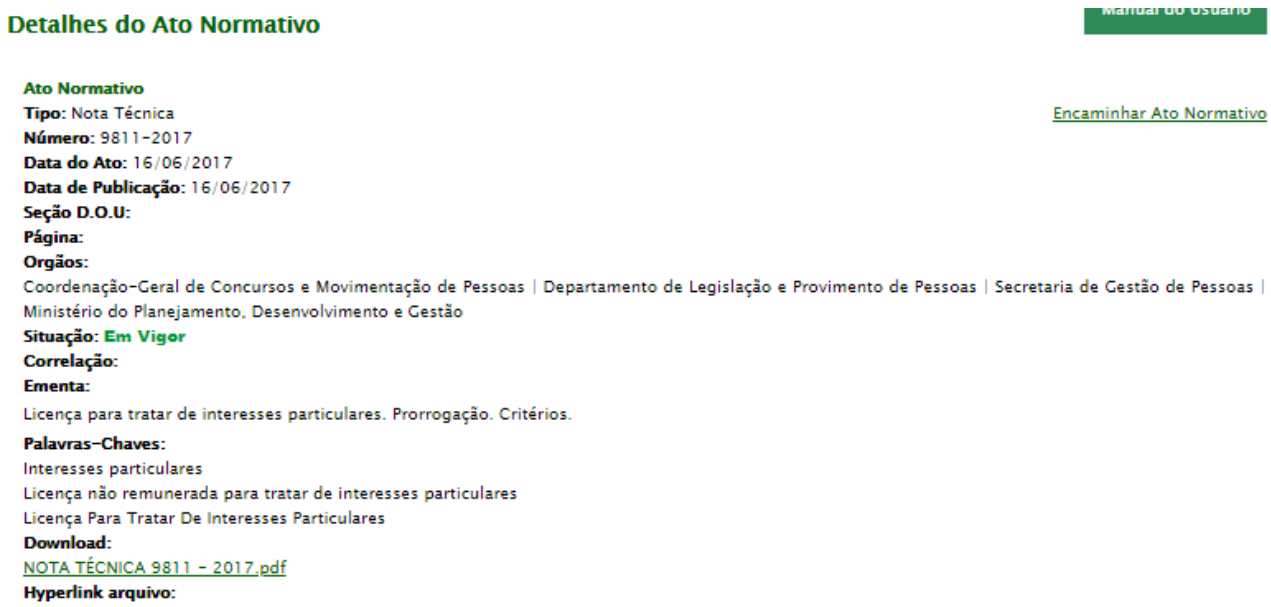
Figura 18. Pesquisa Textual do Conlegis - termo "8112".

Normativos

Clique na coluna desejada para ordenar o resultado da pesquisa.

Número	Tipo	Data Pub.	Ementa
9811-2017	Nota Técnica	16/06/2017	Licença para tratar de interesses particulares. Pr...
6811-2016	Nota Técnica	08/06/2016	Divulgação aos órgãos e entidades integrantes do S...
8.112-2013	Decreto	30/09/2013	Altera o Decreto no 6.558, de 8 de setembro de 200...
811-2012	Nota Informativa	09/10/2012	Informações sobre as condutas vedadas aos agentes ...
811-2011	Nota Informativa	19/12/2011	Revisão de enquadramento de empregado anistiado.
540811-2010	Comunica Geral	03/08/2010	ATUALIZACAO DE PIS/PASEP NO CADASTRO DO SERVIDOR
6.811-2009	Decreto	01/04/2009	Dispõe sobre o remanejamento de cargos em comissão...
4.811-2003	Decreto	20/08/2003	Aprova a Estrutura Regimental e o Quadro Demonstra...
8.112-1990	Lei	12/12/1990	Dispõe sobre o Regime Jurídico dos Servidores Púb...
	Total		**9/9**

Figura 19 Página do Ato Normativo 9811-2017.



Detalhes do Ato Normativo

Manual do Usuário

Ato Normativo
Tipo: Nota Técnica
Número: 9811-2017
Data do Ato: 16/06/2017
Data de Publicação: 16/06/2017
Seção D.O.U:
Página:
Orgãos:
Coordenação-Geral de Concursos e Movimentação de Pessoas | Departamento de Legislação e Provimento de Pessoas | Secretaria de Gestão de Pessoas | Ministério do Planejamento, Desenvolvimento e Gestão
Situação: Em Vigor
Correlação:
Ementa:
Licença para tratar de interesses particulares. Prorrogação. Critérios.
Palavras-Chaves:
Interesses particulares
Licença não remunerada para tratar de interesses particulares
Licença Para Tratar De Interesses Particulares
Download:
NOTA TÉCNICA 9811 - 2017.pdf
Hyperlink arquivo:

Encaminhar Ato Normativo

Ainda no sistema atual do Conlegis, buscando o termo "8112**" no "Parâmetro de busca para localizar Ato", e deixando o número em branco, retorna-se um conjunto de 66 normativos, também ordenados, entretanto a própria "*Lei 8.112*" não aparece entre eles. Ainda, os outros normativos retornados também não aparentam versar sobre nada a respeito da Lei 8.112 no campo *Ementa*.**

No novo SRI Conlegis a busca pelo termo "**8112**" retorna o máximo pré-estabelecido de 60 (sessenta) normativos. A Lei 8.112 aparece em primeiro lugar e os normativos restantes versam sobre esta lei em suas ementas. Todos os normativos são apresentados com o link direto para o download do documento oficial e a ementa é apresentada de forma completa. Existem as duas possibilidades de utilização da interface gráfica, com ou sem as marcações nos trechos dos normativos encontrados. Além disso os 60 normativos são apresentados em uma só página do navegador permitindo uma busca interna na documentação com a própria função de busca (*CTRL-F*) do navegador. Os resultados são apresentados conforme a figura a seguir.

Figura 20. Pesquisa no SRI Conlegis - termo “8112”.

4.5.3 Filtragem rápida de resultados

O sistema atual do Conlegis não permite nenhuma modificação dos resultados retornados com facilidade e interface de simples utilização. Por exemplo, se o usuário decidiu pesquisar determinada palavra chave apenas para os atos do tipo “Decreto” ele sairá da página de busca para uma página de apresentação dos resultados. Caso o usuário queira fazer a mesma consulta para atos do tipo “Instrução Normativa” ele precisará voltar para a página de busca e refazer todo o processo trocando a opção selecionada anteriormente para a desejada.

No novo SRI Conlegis é possível manipular de forma prática e simples os resultados aplicando os filtros de Tipo e de Situação de Normativos sobre o conjunto de atos retornados da busca. Uma vez solicitada uma busca sobre determinado(s) termo(s) basta aplicar os filtros de busca e os documentos são filtrados instantaneamente conforme a condição estabelecida.

4.5.4 Relevância no retorno dos atos normativos

O sistema atual do Conlegis não aparenta possuir critérios de pontuação/relevância sobre os atos normativos retornados. Toda a estratégia de busca deve ser realizada pelo usuário conforme a escolha dos diversos parâmetros ou filtros disponibilizados nas três formas diferentes de busca e deve-se entender bem o manual para montar as combinações necessárias. Ainda, ele não possui nenhuma forma de estabelecer prioridades de importância entre os atributos pesquisados.

O novo SRI Conlegis implementa automaticamente as estratégias de relevância e depende dos termos solicitados pelo usuário para construção da ordem de apresentação. A definição da ordem de importância de dada atributo já foi desenvolvida previamente (item 4.3). Por exemplo, se o usuário deseja obter normativos do ano de 2018 basta adicionar este termo na consulta em conjunto com os outros termos desejados. Se o usuário busca determinado assunto ou o próprio número da lei, todos estes termos podem ser solicitados em conjunto, combinados no mesmo campo de busca.

5 CONCLUSÕES

O desenvolvimento e implantação do SRI Conlegis foi um projeto complexo e trabalhoso, mas cujo produto final pode proporcionar uma grande melhoria no fluxo de trabalho realizado por dezenas de unidades prestadoras de serviço público nas áreas meio, de gestão de pessoas, na Administração Pública.

Além disso, um melhor mecanismo de busca para o Conlegis beneficia também qualquer outro usuário da sociedade, pesquisadores ou estudantes que precisam de informação especializada sobre os atos normativos expedidos relacionados à gestão de pessoas no Brasil.

A quantidade e qualidade dos dados disponíveis no Conlegis, que possui de atos de alta hierarquia, como leis e decretos, a atos corriqueiros e mais operacionais, como notas técnicas e pareceres, combinados a um mecanismo de busca eficaz, resulta em uma integralidade de informação excelente para apoio aos processos administrativos desempenhados no âmbito de governo que não é capaz de ser equiparado nem mesmo pelos motores de busca implementados atualmente nos sites mais comuns como Google, Bing ou DuckDuckGo.

Os objetivos do trabalho foram cumpridos e bem documentados no tópico quatro onde pode-se discorrer sobre os parâmetros importantes que contribuíram para um bom processo de indexação; observar quais os atributos mais importantes a serem considerados no algoritmo de busca de forma a trazer os atos normativos mais relevantes para o usuário e consolidar tudo em uma documentação bem completa e detalhada para apoiar o processo de implantação do SRI Conlegis ao sistema oficial, caso isto seja aceito e patrocinado pela área negocial.

Por fim, esta implementação será entregue às unidades negociais para avaliação, testes de uso e propostas de melhoria visando melhorar cada vez mais os processos realizados pelas áreas afins ao Conlegis no serviço público federal.

O ideal será que haja a continuidade deste trabalho pela unidade de Tecnologia da Informação mais próxima da área negocial responsável pelo Conlegis, provavelmente a Diretoria de Tecnologia da Informação na Secretaria de Gestão Corporativa do Ministério da Economia, de forma a promover cada vez mais o

Conlegis, e torná-lo um sistema forte e eficiente dentro da APF. Neste contexto, pontos que ainda podem ser aperfeiçoados no SRI Conlegis são:

- a adição de novos filtros dinâmicos nos resultados como ano, por exemplo;
- a implementação de documentos correlacionados aplicando referências cruzadas entre os atos normativos;
- o desenvolvimento de uma forma de feedback implícito conforme os cliques dos usuários nos normativos retornados adicionando um atributo extra sobre a quantidade de visualização que seja importante para o algoritmo de pontuação da relevância do documento;
- a avaliação da necessidade de portabilidade do processo de requisição de informação ao Elasticsearch, agora feito do lado do servidor, para o lado do cliente, reduzindo a carga de processamento da aplicação web;
- o *upgrade* do SRI Conlegis para um cluster distribuído mais robusto, escalável e de alta performance capaz de atender uma alta capacidade de requisições simultâneas e podendo ser reconfigurado para uma melhor performance e tempo de resposta.

Ademais, concluo com a certeza de que esta proposta tem o potencial para contribuir com a melhoria do serviço público, em especial nos processos de padronização de ações e na qualidade e excelência de atendimento aos servidores, pelas unidades que compõe o SIPEC.

Espera-se que este trabalho prossiga e que seja responsável por trazer bons frutos para a Administração Pública Federal.

REFERÊNCIAS

1. **SERPRO.** *Conlegis.* [Online] [Citado em: 4 de março de 2019.] https://conlegis.planejamento.gov.br/conlegis/legislacao/indexSaudacao.htm.

2. **Google LLC.** Google. *Google.* [Online] [Citado em: 4 de março de 2019.] https://www.google.com.br/.

3. *DECRETO No 67.326, DE 05 DE OUTUBRO DE 1970.* [Online] [Citado em: 04 de março de 2019.] http://www.planalto.gov.br/ccivil_03/decreto/1970-1979/D67326.htm.

4. **Mooers, Calvin N.** *Why Some Retrieval Systems Are Used and Others Are Not.* s.l. : American Documentation, 1960.

5. **LE COADIC, J. F.** A Ciência da Informação. Brasília : Briquet de Lemos, 2004.

6. **Manning, Christopher D., Raghavan, Prabhakar e Schütze, Hinrich.** An Introduction to Information Retrieval. [A. do livro] Christopher D. Manning. s.l. : Cambridge University Press, 2009.

7. **Page, Larry e Brin, Sergey.** PageRank. *Wikipedia.* [Online] [Citado em: 5 de março de 2019.] https://en.wikipedia.org/wiki/PageRank.

8. **The Apache Software Foundation.** Apache Lucene. [Online] [Citado em: 5 de março de 2019.] http://lucene.apache.org/.

9. **Crockford, Douglas.** JSON. *Wikipedia.* [Online] [Citado em: 5 de março de 2019.] https://en.wikipedia.org/wiki/JSON.

10. **Fielding, Roy Thomas.** *Architectural Styles and the Design of Network-based Software Architectures.* s.l. : University of California, 2000.

11. *Proceedings of the Seventh Text Retrieval Conference.* **Robertson, Stephen E. e Hancock-Beaulieu, Steve Walker & Micheline.** Gaithersburg, USA : Okapi at TREC-7, 1998.

12. Pattern Replace Char Filter. *Elasticsearch Reference 6.7.* [Online] [Citado em: 5 de março de 2019.] https://www.elastic.co/guide/en/elasticsearch/reference/6.7/analysis-pattern-replace-charfilter.html.

13. Mapping Char Filter. *Elasticsearch Reference 6.7.* [Online] [Citado em: 5 de março de 2019.] https://www.elastic.co/guide/en/elasticsearch/reference/6.7/analysis-mapping-charfilter.html.

14. Token Filters. *Elasticsearch Reference 6.7.* [Online] [Citado em: 5 de março de 2019.] https://www.elastic.co/guide/en/elasticsearch/reference/6.7/analysis-tokenfilters.html.

15. Multi Match Query. *Elasticsearch Reference 6.7.* [Online] [Citado em: 5 de março de 2019.] https://www.elastic.co/guide/en/elasticsearch/reference/6.7/query-dsl-multi-match-query.html.

16. Bool Query. *Elasticsearch Reference 6.7.* [Online] [Citado em: 5 de março de 2019.] https://www.elastic.co/guide/en/elasticsearch/reference/6.7/query-dsl-bool-query.html.

17. Cross-origin resource sharing. *Wikipedia.* [Online] [Citado em: 5 de março de 2019.] https://en.wikipedia.org/wiki/Cross-origin_resource_sharing.

APÊNDICE A – CÓDIGOS

1. Processo de extração de Texto dos arquivos originais em PDF

```
#!/bin/bash
FILES=fullcorpus/*.pdf
fout="fullcorpustxt/"
for f in $FILES do
 in="${f%.pdf}.txt"
 out="${in#fullcorpus/}"
 echo "Processing $fout$out..."
 pdftotext -enc UTF-8 "$f"
"$fout$out" done
```

pdftextlegis.sh

2. Conversor de TXT para JSON

```
import logging
import os, os.path
import sys
import re
import urllib.parse
from multiprocessing.dummy import Pool as ThreadPool

logger = logging.getLogger(__name__)
logging.basicConfig(format='{levelname:.3s}: {message}', style='{',
level=logging.INFO)

if len(sys.argv) not in [2]:
  print('Usage: py {} <path> '.format(sys.argv[0]))
  sys.exit()

#Função para ajustar as pastas de entrada e saída.
def strip_left(text, prefix):
    if not text.startswith(prefix):
        return text
    # else
    return text[len(prefix):]

#Função para alterar a extensão mantendo o nome do arquivo.
def strip_right(text, suffix):
 if not text.endswith(suffix):
  return text
 # else
 return text[:len(text)-len(suffix)]

#Função para encontrar uma palavra específica em uma string
```

```
def findWholeWord(w):
    return re.compile(r'\b({0})\b'.format(w), flags=re.IGNORECASE).search

def txttojson (filename):
  nametxt = filename
  namepdf = strip_left(strip_right(filename, "txt") + 'pdf', sys.argv[1] +
"\\")
  namejson = "fullcorpusjson\\" + strip_left(strip_right(filename, "txt") +
'json', sys.argv[1] + "\\")

  filein = open(nametxt, 'r', encoding='utf8')
  fileout = open(namejson, 'w', encoding='utf8')

  header = '{"index":{"_index":"legis","_type":"normativos"}}\n'

  #DETECÇÃO DO TIPO DE NORMATIVO E DA LINHA DE TÍTULO
  normativos = [ "acórdão tcu", "acórdão", "decisão tcu", "decreto-lei",
"decreto legislativo", "decreto", "desp ", "despacho", "emenda", "exposição
de motivos", "formulação", "instrução normativa", "lei complementar", "lei
delegada", "lei", "manual", "memorando-circular", "memorando", "medida
provisória", "nota técnica", "nt n°", "ni n°", "nota informativa",
"comunica geral", "comunica", "portaria", "súmula", "resolução", "oficio
circular", "ofício circular", "ofício-circular", "ofício-circular
conjunto", "ofício", "of n", "orientação consultiva", "orientação
normativa", "parecer" ]

  tipo_normativo = "outros"
  pretitle = ""
  title = ""
  flag = ""

#Categorização por nome do arquivo
  for x in normativos:
    if findWholeWord(x)(nametxt):
      tipo_normativo = x
      flag = "(F)"
      break;

  if tipo_normativo == "desp ":
    tipo_normativo = "despacho"
  if tipo_normativo == "ni n°":
    tipo_normativo = "nota informativa"
  if tipo_normativo == "nt n°":
    tipo_normativo = "nota técnica"
  if tipo_normativo == "ofício circular":
     tipo_normativo = "ofício-circular"
  if tipo_normativo == "oficio circular":
     tipo_normativo = "ofício-circular"
  if tipo_normativo == "of n":
     tipo_normativo = "ofício"
```

```
#Categorização por linha de dados no documento
  if tipo_normativo == "outros":
    title = namepdf
    for line in filein:
      for x in normativos:
        if findWholeWord(x)(line):
          tipo_normativo = x
          flag = "(L)"
          title = re.sub(r'[^\w.]', ' ', line.rstrip('\n'))
          break
      if tipo_normativo == x:
        break
      pretitle += re.sub(r'[^\w.]', ' ', line.rstrip('\n')) + ' '
  else:
    for line in filein:
      if findWholeWord(tipo_normativo)(line):
        title = re.sub(r'[^\w.]', ' ', line.rstrip('\n'))
        break
      else:
        title = namepdf
      pretitle += re.sub(r'[^\w.]', ' ', line.rstrip('\n')) + ' '

  data = pretitle + title + ' ' + re.sub(r'[^\w.]', ' ', filein.read())

  tipos_ref = []
  for x in normativos:
    if findWholeWord(x)(data):
      if x == "ni n°" or x == "nt n°" or x == "desp " or x == "ofício
circular" :
        if x == "ni n°":
          tipos_ref.append("nota informativa")
        else:
          if x == "nt n°":
            tipos_ref.append("nota técnica")
          else:
            if x == "ofício circular" or "oficio circular":
              tipos_ref.append("ofício-circular")
            else:
              tipos_ref.append("despacho")
      else:
        tipos_ref.append(x)

  tipos_ref_str = '["' + '","'.join(tipos_ref) + '"]'

  fileurl = 'https://conlegis.planejamento.gov.br/conlegis/Downloads/file?'
+ urllib.parse.quote(namepdf, encoding='cp1252')

  preview = data[0:350] + '...'

  if tipo_normativo == "ofício":
    preview = '...' + data[150:500] + '...'
```

```
  if tipo_normativo ==  "ofício-circular":
    preview = '...' + data[150:500] + '...'
  if tipo_normativo ==  "despacho":
    preview = '...' + data[150:500] + '...'

  jsoncontent = header + '{"filename": "' + namepdf + '", "fileurl": "' +
fileurl + '", "tipo_normativo": "' + tipo_normativo + '", "tipos_ref": ' +
tipos_ref_str + ', "titulo": "' + title + '", "preview": "' + preview + '",
"dados": "' + data + '"}\n'

  print("Escrevendo arquivo tipo " + tipo_normativo + " " + flag + ": " +
namejson)
  fileout.write(jsoncontent)

  filein.close()
  fileout.close()

logger.info("Iniciando conversão em lote...")
pool = ThreadPool(2)
pool.map(txttojson, [os.path.join(sys.argv[1], filename) for filename in
os.listdir(sys.argv[1])])
logger.info("Terminado")
```

txt2json7.py

3. Conversor combinador JSON(MySQL)/TXT para JSON

```
import logging
import sys
import re
import os, os.path
from pathlib import Path
import urllib.parse
import json

logger = logging.getLogger(__name__)
logging.basicConfig(format='{levelname:.3s}: {message}', style='{',
level=logging.INFO)

#Função para alterar a extensão mantendo o nome do arquivo.
def strip_right(text, suffix):
      if not text.endswith(suffix):
        return text
      # else
      return text[:len(text)-len(suffix)]

#Carregamento dos arquivos iniciais .json
def combiner (jsondump):
```

```
      filetipos = open("database\\tiposnormativos.json", 'r',
encoding='utf8')
      filesituacoes = open("database\\situacaonormativos.json", 'r',
encoding='utf8')
      tip_norm_json = json.load(filetipos)
      sit_norm_json = json.load(filesituacoes)

      namedatabase = "database\\" + jsondump
      databasejson = open(namedatabase, 'r', encoding='utf8')

      docs = json.load(databasejson)
      print("Arquivo " + namedatabase + " carregado!")
      logger.info("Iniciando combiner...")

#Cabeçalho para processamento em lote no Elasticsearch
    header = '{"index":{"_index":"legis2","_type":"atosnormativos"}}\n'

#Processamento de item a item (atos normativos) do arquivo .json de entrada
      for doc in docs:
      data = ""

#Construção dos campos filename, fileurl e dados
      filename = doc["filename"]
      if filename != None:
            fileurl =
'https://conlegis.planejamento.gov.br/conlegis/Downloads/file?' +
urllib.parse.quote(filename, encoding='cp1252')
            nametxt = "fullcorpustxt\\" + strip_right(filename, "pdf") +
'txt'
            namejson = "fullcorpusjson\\" + strip_right(filename, "pdf") +
'json'

            checkpath = Path(nametxt)
            if checkpath.exists():
                  txtdatain = open(nametxt, 'r', encoding='utf8')
                  data = re.sub(r'[^\w.]', ' ', txtdatain.read())
                  flag = "data"
                  txtdatain.close()
            else:
                  flag = "nodata"
      else:
            fileurl = ""
            filename = ""

#Construção dos campos tipo_normativo e sit_normativo.
      for tipo in tip_norm_json:
            if tipo["id_tipo_ato_normativo"] ==
doc["id_tipo_ato_normativo"]:
                  tipo_normativo = tipo["descricao"]
                  break;
```

```
      for sit in sit_norm_json:
            if sit["id_situacao_ato_normativo"] ==
doc["id_situacao_ato_normativo"]:
                  sit_normativo = sit["descricao"]
                  break;

      jsondocout = open(namejson, 'w', encoding='utf8')

#Construção dos campos titulo, data_publicacao, ementa e correlacao
      mes_ext = {1: 'Janeiro', 2 : 'Fevereiro', 3: 'Março', 4: 'Abril', 5:
'Maio', 6: 'Junho', 7: 'Julho', 8: 'Agosto', 9: 'Setembro', 10: 'Outubro',
11: 'Novembro', 12: 'Dezembro', 13: '[não informado]'}

      if doc["data_ato"] == None:
            anoa = "[não informado]"
            mesa = "13"
            diaa = "[não informado]"
      else:
            anoa, mesa, diaa = doc["data_ato"].split("-")

      if doc["data_publicacao"] == None:
            data_publicacao = "00/00/0000"
      else:
            anop, mesp, diap = doc["data_publicacao"].split("-")

      numero = doc["numero"].split("-")
      numeroato = numero[0]

      if numeroato == "TCU":
            numeroato = numero[1]

      title = tipo_normativo + " n° " + numeroato + ", de " + diaa + " de "
+ mes_ext[int(mesa)] + " de " + anoa
      data_publicacao = diap + "/" + mesp + "/" + anop
      ementa = re.sub(r'[^\w.]', ' ', doc["ementa"])

      if doc["correlacao"] != None:
            correlacao = re.sub(r'[^\w.]', ' ', doc["correlacao"])
      else:
            correlacao = ""

#Gravacao do conteúdo do arquivo <filename>.json
      jsoncontent = header + '{"filename": "' + filename + '", "fileurl":
"' + fileurl + '", "tipo_normativo": "' + tipo_normativo + '",
"sit_normativo": "' + sit_normativo + '", "titulo": "' + title + '",
"data_publicacao": "' + data_publicacao + '", "ano": "' + anop + '",
"ementa": "' + ementa + '", "correlacao": "' + correlacao + '", "dados": "'
+ data + '"}\n'

      jsondocout.write(jsoncontent)
      print("Escrevendo arquivo " + "(" + flag + "):  " + title)
```

```
      jsondocout.close()

      filetipos.close()
      filesituacoes.close()

combiner(str(sys.argv[1]))
logger.info("Terminado")
```

combiner.py

4. Parâmetros de configuração do cluster Elasticsearch

```
bootstrap.memory_lock: false
cluster.name: elasticsearch
http.cors.enabled: true
http.port: 9200
http.cors.allow-origin: "http://localhost:3001"
http.cors.allow-headers : "X-Requested-With, X-Auth-Token, Content-Type,
Content-Length, Authorization"
http.cors.allow-credentials: true
http.cors.allow-methods: "OPTIONS, HEAD, GET, POST, PUT, DELETE"
node.data: true
node.ingest: true
node.master: true
node.max_local_storage_nodes: 1
node.name: <server name>
path.data: C:\ProgramData\Elastic\Elasticsearch\data
path.logs: C:\ProgramData\Elastic\Elasticsearch\logs
transport.tcp.port: 9300
xpack.license.self_generated.type: basic
xpack.security.enabled: false
```

elasticsearch.yml

5. Script JSON de construção do índice básico

```
PUT legis
{ "settings": {
      "analysis": {
        "analyzer": {
            "default": {
              "type": "standard",
              "max_token_length": 15,
              "stopwords": "_brazilian_"
            },
            "default_search": {
              "type": "standard",
```

```
            "stopwords": "_brazilian_"
          }
        }
      },
      "similarity": {
        "default": {
            "type": "BM25"
        }
      }
  }
}
```

Script Kibana/Elasticsearch

6. Script JSON de construção do índice com filtro de caracteres do tipo Mapping

```
PUT legis
{ "settings": {
      "analysis" : {
          "analyzer" : {
            "default_search" : {
              "char_filter" : [
                "ajustabusca"
              ],
              "stopwords" : "_brazilian_",
              "tokenizer" : "standard"
            },
            "default" : {
              "char_filter" : [
                "ajustabusca"
              ],
              "stopwords" : "_brazilian_",
              "tokenizer" : "standard"
            }
          },
          "char_filter" : {
            "ajustabusca" : {
              "type" : "mapping",
              "mappings" : [
                ". => ",
                "acordao => acórdão",
                "acordão => acórdão",
                "instrucao => instrução",
                "instruçao => instrução",
                "oficio => ofício",
                "orientacao => orientação",
                "orientaçao => orientação",
                "provisoria => provisória",
```

```
                "resolucao => resolução",
                "resoluçao => resolução",
                "tecnica => técnica",
                "sumula => súmula"
              ]
            }
          }
    },
      "similarity": {
        "default": {
              "type": "BM25"
        }
      }
  }
}
```

Script Kibana/Elasticsearch

7. Script JSON de construção do índice com filtro de caracteres do tipo Pattern Replace e filtros adicionais de Stemmer e Lowercase

```
PUT legis
{ "settings": {
    "analysis" : {
      "filter" : {
        "lowercase" : {
          "type" : "lowercase"
        },
        "br_stemmer" : {
          "name" : "brazilian",
          "type" : "stemmer"
        },
        "br_stop" : {
          "type" : "stop",
          "stopwords" : "_brazilian_"
        }
      },
      "analyzer" : {
        "default_search" : {
          "filter" : [
            "br_stop",
            "lowercase",
            "br_stemmer"
          ],
          "char_filter" : [ "delponto" ],
          "type" : "custom",
          "tokenizer" : "standard"
        },
        "default" : {
```

```
          "filter" : [
            "br_stop",
            "lowercase",
            "br_stemmer"
          ],
          "char_filter" : [ "delponto" ],
          "type" : "custom",
          "tokenizer" : "standard"
        }
      },
     "char_filter" : {
       "delponto" : {
         "type": "pattern_replace",
         "pattern": "\\.",
         "replacement": ""
       }
     }
   }
 }
}
```

Script Kibana/Elasticsearch

8. Processo de carga do corpus no Elasticsearch

```
# Version 1.0

import requests
import logging
import json
import pprint
import os, os.path
import sys
from multiprocessing.dummy import Pool as ThreadPool

logger = logging.getLogger(__name__)
logging.basicConfig(format='{levelname:.3s}: {message}', style='{',
level=logging.INFO)

if len(sys.argv) not in [2, 3, 4]:
      print('Usage: python {} <corpus-directory> [<index-
settings>]'.format(sys.argv[0]))
      sys.exit()

# Define a function to bulk index the contents of a file.
def index(filename):
      f = open(filename, 'r', encoding='utf-8')
      logger.info('Bulk indexing file {0}...'.format(filename))
      response = requests.post('http://localhost:9200/legis/_bulk',
```

```
data=f.read().encode('utf-8'), headers={"Content-Type":
"application/json"}).json()
      f.close()
      flog = open('indexing.log', 'a')
      flog.write('{0}\n'.format(json.dumps(response)))
      flog.close()

try:
      requests.get('http://localhost:9200')
except Exception as e:
      logger.error('Failed connecting to Elasticsearch on
http://localhost:9200.')
      sys.exit()

response = requests.get('http://localhost:9200/legis').json()
if 'legis' not in response:
      # Create a new index called 'legis'
      response = requests.put('http://localhost:9200/legis').json()
      if response['acknowledged'] == True:
      logger.info('Index `legis` created.')
      else:
      logger.error('Failed creating index `legis`.')
      sys.exit()

if len(sys.argv) == 3:
      # Before updating _settings, we should close the index.
      requests.post('http://localhost:9200/legis/_close').json()

      # Apply provided settings.
      f = open(sys.argv[2])
      my_settings = f.read()
      f.close()

      response = requests.put('http://localhost:9200/legis/_settings', data
= my_settings.encode('utf-8'), headers={"Content-Type":
"application/json"}).json()

      # Open index again.
      requests.post('http://localhost:9200/legis/_open').json()

      # Check if settings were updated.
      if 'acknowledged' not in response or response['acknowledged'] ==
False:
      logger.error('Failed updating index
settings:\n{}'.format(json.dumps(response, indent=2)))
      else:
      response =
requests.get('http://localhost:9200/legis/_settings').json()
      logger.info('Successfully updated index
settings:\n{}'.format(json.dumps(response, indent=2)))
```

```
# Bulk indexing.
logger.info("Starting bulk indexing...")
pool = ThreadPool(2)
pool.map(index, [os.path.join(sys.argv[1], filename) for filename in
os.listdir(sys.argv[1])])
logger.info("Finished")
```

index.py

9. Script JSON para a busca simples ao índice Legis - Antes do algoritmo combinador

```
GET legis/_search
{
  "size" : 60,
  "_source": ["filename","titulo", "tipo_normativo", "tipos_ref",
"fileurl", "preview"],
  "query": {
    "multi_match" : {
      "query" : "[termos de busca]",
      "type" : "most_fields",
      "fields": [
        "filename^2",
         "dados^4",
         "titulo^1.5",
         "tipo_normativo",
         "_all"
      ]
     }
   }
 }
```

Script Kibana/Elasticsearch

10. Script JSON de busca após algoritmo combinador - com *highlights*

```
GET legis2/_search
{
      "size" : 60,
      "_source": ["filename","titulo", "tipo_normativo", "sit_normativo",
"fileurl", "ementa" ],
      "query": {
      "multi_match" : {
            "query" : req.query['q'],
            "type" : "most_fields",
           "fields" : [
                 "ementa^4",
```

```
                    "dados^3",
                    "titulo^7",
                    "tipo_normativo^2",
                    "_all"
                    ]
                }
            },
            "highlight" : {
                "fields" : {
                    "dados" : {
                    "number_of_fragments" : 3,
                    "pre_tags": ["<mark>"],
                    "post_tags": ["</mark>"]
                    }
                }
        }
}
```

Script Kibana/Elasticsearch

11. Instalação das bibliotecas necessárias para implementação da interface web para o SRI Conlegis

```
#Após instalação do Node.js
#Dentro de pasta dedicada para a aplicação web
#
$ npm install express body-parser elasticsearch
```

Comandos shell

12. Javascript de integração entre o Elasticsearch e o express

```
//index.js
//require the Elasticsearch librray
const elasticsearch = require('elasticsearch');
// instantiate an elasticsearch client
const client = new elasticsearch.Client({
   hosts: [ 'http://localhost:9200']
});
//require Express
const express = require( 'express' );
// instanciate an instance of express and hold the value in a constant
called app
const app = express();
//require the body-parser library. will be used for parsing body requests
const bodyParser = require('body-parser')
//require the path library
```

```
const path = require( 'path' );

// ping the client to be sure Elasticsearch is up
client.ping({
     requestTimeout: 30000,
 }, function(error) {
 // at this point, eastic search is down, please check your Elasticsearch
service
     if (error) {
         console.error('elasticsearch cluster is down!');
     } else {
         console.log('Everything is ok');
     }
 });


// use the bodyparser as a middleware
app.use(bodyParser.json())
// set port for the app to listen on
app.set( 'port', process.env.PORT || 3001 );
// set path to serve static files
app.use( express.static( path.join( __dirname, 'public' )));
// enable CORS
app.use(function(req, res, next) {
  res.header("Access-Control-Allow-Origin", "*");
  res.header('Access-Control-Allow-Methods', 'PUT, GET, POST, DELETE,
OPTIONS');
  res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With,
Content-Type, Accept");
  next();
});

// defined the base route and return with an HTML file called tempate.html
app.get('/', function(req, res){
  res.sendFile('template.html', {
     root: path.join( __dirname, 'views' )
   });
})

app.get('/v2', function(req, res){
  res.sendFile('template2.html', {
     root: path.join( __dirname, 'views' )
   });
})

app.get('/v3', function(req, res){
  res.sendFile('template3.html', {
     root: path.join( __dirname, 'views' )
   });
})
// rota search para o template 1
```

```
app.get('/search', function (req, res){
  // declare the query object to search elastic search and return only 200
results from the first result found.
  // also match any data where the name is like the query string sent in
  let bodysearch = {
                "size" : 100,
                "_source": ["filename","titulo", "tipo_normativo",
"tipos_ref","preview" ],
                "query": {
                    "multi_match" : {
                        "query" : req.query['q'],
                        "type" : "most_fields",
                        "fields": [
                            "filename^4.5",
                            "dados^3",
                            "titulo^2",
                            "tipo_normativo^1.5",
                            "preview^3",
                            "_all"
                            ]
                        }
                    },
                    "highlight" : {
                        "fields" : {
                            "dados" : { "number_of_fragments" : 4,
"pre_tags": ["<mark>"], "post_tags": ["</mark>"] }
                        }
                    }
                }

  // perform the actual search passing in the index, the search query and
the type
  client.search({index:'legis',  body:bodysearch, type:'normativos'})
  .then(results => {
    res.send(results.hits.hits);
  })
  .catch(err=>{
    console.log(err)
    res.send([]);
  });
})

app.get('/search2', function (req, res){
  // declare the query object to search elastic search and return only 200
results from the first result found.
  // also match any data where the name is like the query string sent in
  console.log(req.query)

  body_tipo = '"match_all": {}'
  body_sit = '"match_all": {}'
```

```
  if (req.query['tipo_ato'] !== "Todos") {
    body_tipo = '"term" : { "tipo_normativo.keyword" : "' +
req.query['tipo_ato'] + '" }';
  }
  if (req.query['sit_ato'] !== "Todas") {
    body_sit = '"term" : { "sit_normativo.keyword" : "' +
req.query['sit_ato'] + '" }';
  }

  let bodysearch2 = '{' +
    '"size" : 60,' +
    '"_source": ["filename","titulo", "tipo_normativo", "sit_normativo",
"fileurl", "ementa" ],' +
        '"query": {' +
            '"bool" : {' +
                '"must" : [{' +
                    '"multi_match" : {' +
                        '"query" : "' + req.query['q'] + '" ,' +
                        '"type" : "most_fields",' +
                        '"fields" : [' +
                            '"ementa^4",' +
                            '"dados^3",' +
                            '"titulo^7",' +
                            '"tipo_normativo^2",' +
                            '"_all"' +
                            ']' +
                        '}' +
                '},{' +
                '"bool": {' +
                    '"must" : [' +
                        '{'+ body_tipo + '},' +
                        '{'+ body_sit + '}' +
                    ']' +
                '}}' +
                ']' +
            '}' +
        '}' +
    '}'

  // perform the actual search passing in the index, the search query and
the type
  client.search({index:'legis2',  body:bodysearch2, type:'atosnormativos'})
  .then(results => {
    res.send(results.hits.hits);
  })
  .catch(err=>{
    console.log(err)
    res.send([]);
  });

})
```

```
app.get('/search3', function (req, res){
  // declare the query object to search elastic search and return only 200
results from the first result found.
  // also match any data where the name is like the query string sent in
  console.log(req.query)

  body_tipo = '"match_all": {}'
  body_sit = '"match_all": {}'

  if (req.query['tipo_ato'] !== "Todos") {
    body_tipo = '"term" : { "tipo_normativo.keyword" : "' +
req.query['tipo_ato'] + '" }';
  }
  if (req.query['sit_ato'] !== "Todas") {
    body_sit = '"term" : { "sit_normativo.keyword" : "' +
req.query['sit_ato'] + '" }';
  }

  let bodysearch3 = '{' +
    '"size" : 60,' +
    '"_source": ["filename","titulo", "tipo_normativo", "sit_normativo",
"fileurl", "ementa" ],' +
        '"query": {' +
            '"bool" : {' +
                '"must" : [{' +
                    '"multi_match" : {' +
                        '"query" : "' + req.query['q'] + '" ,' +
                        '"type" : "most_fields",' +
                        '"fields" : [' +
                            '"ementa^4",' +
                            '"dados^3",' +
                            '"titulo^7",' +
                            '"tipo_normativo^2",' +
                            '"_all"' +
                            ']' +
                        '}' +
                '},{' +
                '"bool": {' +
                    '"must" : [' +
                        '{'+ body_tipo + '},' +
                        '{'+ body_sit + '}' +
                    ']' +
                '}}' +
                ']' +
            '}' +
        '},' +
        '"highlight" : {' +
            '"fields" : {' +
                '"dados" : { ' +
                    '"number_of_fragments" : 3, ' +
```

```
                    '"pre_tags": ["<mark>"], ' +
                    '"post_tags": ["</mark>"] ' +
                '}' +
            '}' +
        '}' +
  '}'
    console.log(bodysearch3)
  // perform the actual search passing in the index, the search query and
the type
  client.search({index:'legis2',  body:bodysearch3, type:'atosnormativos'})
  .then(results => {
    res.send(results.hits.hits);
  })
  .catch(err=>{
    console.log(err)
    res.send([]);
  });
})
// listen on the specified port
app .listen( app.get( 'port' ), function(){
  console.log( 'Express server listening on port ' + app.get( 'port' ));
} );
```

index.js

13. Aplicação (Interface Gráfica) do SRI Conlegis v.2

```
<!—template2.html -->
<link rel="stylesheet" type="text/css"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css
">
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/vue"></script>
<div class="container" id="app">
    <div class="row">
        <div class="col-md-6 col-md-offset-3"><br>
            <font size="6">Busca Conlegis </font>v.2
        </div>
    </div>
    <form action="" class="search-form">
    <div class="row">
        <div class="col-md-6 col-md-offset-3">
                <div class="has-feedback">
                    <label for="search" class="sr-only">Search</label>
                    <input type="text" class="form-control" name="search2"
id="search2" placeholder="busque aqui" v-model="query">
                    <span class="glyphicon glyphicon-search form-control-
feedback"></span><br>
                </div>
```

```
        </div>
    </div>
    <div class="form-row">
        <div class="col-md-8 col-md-offset-2">
            <div class="col-md-6">
                <select id="sit_ato" class="form-control" v-model="q_sit">
                    <option value="Todas">Situação do Ato
Normativo</option>
                    <option value="Em Vigor">Em Vigor</option>
                    <option value="Revogado (a)">Revogado (a)</option>
                    <option value="Reeditado (a)">Reeditado (a)</option>
                    <option value="Retificado (a)">Retificado (a)</option>
                    <option value="Revogado e Reeditado">Revogado e
Reeditado</option>
                    <option value="Republicado (a)">Republicado
(a)</option>
                    <option value="Alterado (a)">Alterado (a)</option>
                    <option value="Rejeitado (a)">Rejeitado (a)</option>
                    <option value="Convertida em Lei">Convertida em
Lei</option>
                    <option value="Revogado (a) Parcialmente">Revogado (a)
Parcialmente</option>
                    <option value="Prejudicado (a)">Prejudicado
(a)</option>
                    <option value="Reeditada e Revogada">Reeditada e
Revogada</option>
                    <option value="Sem Eficácia">Sem Eficácia</option>
                    <option value="Insubsistente">Insubsistente</option>
                    <option value="Suspensa a Eficácia">Suspensa a
Eficácia</option>
                    <option value="Revigorado(a)">Revigorado(a)</option>
                    <option value="Insubsistente
Parcialmente">Insubsistente Parcialmente</option>
                    <option value="Revisto e Atualizado">Revisto e
Atualizado</option>
                    <option value="Aditado (a)">Aditado (a)</option>
                    <option value="Tornado (a) sem Efeito">Tornado (a) sem
Efeito</option>
                    <option value="Anulada">Anulada</option>
                    <option value="Revogado(a) Tacitamente">Revogado(a)
Tacitamente</option>
                    <option value="Vigência Encerrada">Vigência
Encerrada</option>
                </select>
            </div>
            <div class="col-md-6">
                <select id="tipo_ato" class="form-control" v-
model="q_tipo">
                    <option value="Todos">Tipo do Ato Normativo</option>
                    <option value="Ata">Ata</option>
```

```
                    <option value="Ato Declaratório Executivo">Ato
Declaratório Executivo</option>
                    <option value="Ato Regimental">Ato Regimental</option>
                    <option value="Boletim Eletrônico-Contato">Boletim
Eletrônico-Contato</option>
                    <option value="Boletim Informativo STF">Boletim
Informativo STF</option>
                    <option value="Circular">Circular</option>
                    <option value="Comunica Geral">Comunica Geral</option>
                    <option value="Conjunto de Pareceres
Conclusivos">Conjunto de Pareceres Conclusivos</option>
                    <option value="Decisão TCU">Decisão TCU</option>
                    <option value="Decreto">Decreto</option>
                    <option value="Decreto-Lei">Decreto-Lei</option>
                    <option value="Decreto Legislativo">Decreto
Legislativo</option>
                    <option value="Despacho">Despacho</option>
                    <option value="Emenda Constitucional">Emenda
Constitucional</option>
                    <option value="Instrução Normativa">Instrução
Normativa</option>
                    <option value="Lei">Lei</option>
                    <option value="Lei Complementar">Lei
Complementar</option>
                    <option value="Medida Provisória">Medida
Provisória</option>
                    <option value="Memorando">Memorando</option>
                    <option value="Mesa Nacional de Negociação
Permanente">Mesa Nacional de Negociação Permanente</option>
                    <option value="Norma Interna">Norma Interna</option>
                    <option value="Norma Operacional">Norma
Operacional</option>
                    <option value="Nota Técnica">Nota Técnica</option>
                    <option value="Ofício">Ofício</option>
                    <option value="Ofício-Circular">Ofício-
Circular</option>
                    <option value="Ofício-Circular Conjunto">Ofício-
Circular Conjunto</option>
                    <option value="Orientação Consultiva">Orientação
Consultiva</option>
                    <option value="Orientação Geral">Orientação
Geral</option>
                    <option value="Orientação Normativa">Orientação
Normativa</option>
                    <option value="Parecer">Parecer</option>
                    <option value="Parecer AGU">Parecer AGU</option>
                    <option value="Parecer CONJUR">Parecer CONJUR</option>
                    <option value="Portaria">Portaria</option>
                    <option value="Portaria Conjunta">Portaria
Conjunta</option>
                    <option value="Portaria Interministerial">Portaria
```

```
Interministerial</option>
                    <option value="Portaria Normativa">Portaria
Normativa</option>
                    <option value="Projeto de Lei">Projeto de Lei</option>
                    <option value="Resolução">Resolução</option>
                    <option value="Resposta Fax">Resposta Fax</option>
                    <option value="Resposta Ofício">Resposta
Ofício</option>
                    <option value="Súmula Administrativa">Súmula
Administrativa</option>
                    <option value="Súmula da Advocacia-Geral da
União">Súmula da Advocacia-Geral da União</option>
                    <option value="Nota Informativa">Nota
Informativa</option>
                    <option value="Norma de Execução">Norma de
Execução</option>
                    <option value="Lei Delegada">Lei Delegada</option>
                    <option value="Deliberação">Deliberação</option>
                    <option value="Constituição Federal E Emendas
Constitucionais">Constituição Federal E Emendas Constitucionais</option>
                    <option value="Constituição Federal">Constituição
Federal</option>
                    <option value="Atos da AGU">Atos da AGU</option>
                    <option value="Ação Direta de
Inconstitucionalidade">Ação Direta de Inconstitucionalidade</option>
                    <option value="Nota CONJUR">Nota CONJUR</option>
                    <option value="Nota AGU">Nota AGU</option>
                    <option value="Comunica SIAPE">Comunica SIAPE</option>
                    <option value="Manual de Redação da Presidência da
República">Manual de Redação da Presidência da República</option>
                    <option value="Exposição de Motivos">Exposição de
Motivos</option>
                    <option value="Exposição de Motivos
Interministerial">Exposição de Motivos Interministerial</option>
                    <option value="Resolução Normativa">Resolução
Normativa</option>
                    <option value="Comunica">Comunica</option>
                    <option value="Edital">Edital</option>
                    <option value="Nota Técnica_">Nota Técnica_</option>
                    <option value="Recurso Extraordinário">Recurso
Extraordinário</option>
                    <option value="Acórdão TCU_">Acórdão TCU_</option>
                    <option value="Acórdão TCU">Acórdão TCU</option>
                    <option value="Acórdão">Acórdão</option>
                    <option value="Anteprojeto de Lei">Anteprojeto de
Lei</option>
                    <option value="Ato">Ato</option>
                    <option value="Súmula Vinculante">Súmula
Vinculante</option>
                    <option value="Nota Técnica Conjunta">Nota Técnica
Conjunta</option>
```

```
                    <option value="emprego público">emprego
público</option>
                    <option value="Nota">Nota</option>
                    <option value="Formulação">Formulação</option>
                    <option value="Súmula">Súmula</option>
                    <option value="Diversos">Diversos</option>
                    <option value="Lei 8.112 - ANOTADA">Lei 8.112 -
ANOTADA</option>
                    <option value="Nota Informativa Conjunta">Nota
Informativa Conjunta</option>
                    <option value="Portaria AGU">Portaria AGU</option>
                    <option value="TESTE">TESTE</option>
                    <option value="Súmula do TCU">Súmula do TCU</option>
                    <option value="Nota Técnica Consolidada">Nota Técnica
Consolidada</option>
                    <option value="PARECER/PGFN">PARECER/PGFN</option>
                    <option value="PERFIL PROFISSIOGRÁFICO PREVIDENCIÁRIO -
PPP">PERFIL PROFISSIOGRÁFICO PREVIDENCIÁRIO - PPP</option>
                    <option value="SÚMULA AGU">SÚMULA AGU</option>
                    <option value="Decisão STF/Reclamação">Decisão
STF/Reclamação</option>
                    <option value="Manual de Perícia Oficial">Manual de
Perícia Oficial</option>
                    <option value="Agenda de Decisão SEGRT">Agenda de
Decisão SEGRT</option>
                    <option value="Comissão de Ética">Comissão de
Ética</option>
                    <option value="Memorando Circular">Memorando
Circular</option>
                    <option value="Portaria Gsiste">Portaria
Gsiste</option>
                    <option value="OFÍCIO/PGFN">OFÍCIO/PGFN</option>
                    <option
value="Formulação/DASP">Formulação/DASP</option>
                    <option
value="PARECER/DECOR/CGU/AGU.">PARECER/DECOR/CGU/AGU.</option>
                    <option value="Termo Aditivo">Termo Aditivo</option>
                    <option value="Manual de Processo Administrativo
Disciplinar">Manual de Processo Administrativo Disciplinar</option>
                    <option value="Instrução Normativa Conjunta">Instrução
Normativa Conjunta</option>
                    <option value="Mensagem">Mensagem</option>
                    <option value="Ato Declaratório">Ato
Declaratório</option>
                </select><br>
            </div>
        </div>
    </div>
    </form>
    <div class="row">
        <div class="col-md-10 col-md-offset-1" align="right">
```

```
            <em>Atos normativos encontrados: {{ results.length }}</em>
        </div>
        <div class="col-md-10 col-md-offset-1" v-for="result in results">
            <div class="panel panel-default">
                <div class="panel-heading">
                <!-- Titulo e nome do arquivo  -->
                     <a :href="result._source.fileurl" target="_blank"
><font size="4">{{ result._source.titulo }}</font></a><br><b>Arquivo:</b>
{{ result._source.filename }}
                </div>
                <div class="panel-body">
                <!-- Situacao do Ato e Ementa -->
                    <b>Situação do Ato:</b> {{ result._source.sit_normativo
}}<br><b>Ementa:</b> {{ result._source.ementa }}
                </div>
            </div>
        </div>
    </div>
</div>
<script>
//template.html
// create a new Vue instance
var app = new Vue({
    el: '#app',
    // declare the data for the component (An array that houses the results
and a query that holds the current search string)
    data: {
        results: [],
        query: '',
        q_sit: 'Todas',
        q_tipo: 'Todos'
    },
    // declare methods in this Vue component. here only one method which
performs the search is defined
    methods: {
        // make an axios request to the server with the current search
query
        search: function() {
            if (typeof sit_ato === 'undefined') {
                situacao = "Todas";
            } else {
                situacao = sit_ato.value;
            }
            if (typeof tipo_ato === 'undefined') {
                tipoato = "Todos";
            } else {
                tipoato = tipo_ato.value;
            }
            axios.get("http://localhost:3001/search2?q=" + this.query +
"&sit_ato=" + sit_ato.value + "&tipo_ato=" + tipo_ato.value)
                .then(response => {
```

```
                    this.results = response.data;
                })
            }
        },
        // declare Vue watchers
        watch: {
            // watch for change in the query string and recall the search
method
            query: function() {
                this.search();
            },
            q_sit: function() {
                    this.search();
            },
            q_tipo: function() {
                    this.search();
            }
        }

})
</script>
<!--- some styling for the page -->
<style>
    .search-form .form-group {
        float: right !important;
        transition: all 0.35s, border-radius 0s;
        width: 32px;
        height: 32px;
        background-color: #fff;
        box-shadow: 0 1px 1px rgba(0, 0, 0, 0.075) inset;
        border-radius: 25px;
        border: 1px solid #ccc;
    }

    .search-form .form-group input.form-control {
        padding-right: 20px;
        border: 0 none;
        background: transparent;
        box-shadow: none;
        display: block;
    }

    .search-form .form-group input.form-control::-webkit-input-placeholder
{
        display: none;
    }

    .search-form .form-group input.form-control:-moz-placeholder {
        /* Firefox 18- */
        display: none;
    }
```

```
    .search-form .form-group input.form-control::-moz-placeholder {
        /* Firefox 19+ */
        display: none;
    }

    .search-form .form-group input.form-control:-ms-input-placeholder {
        display: none;
    }

    .search-form .form-group:hover,
    .search-form .form-group.hover {
        width: 100%;
        border-radius: 4px 25px 25px 4px;
    }

    .search-form .form-group span.form-control-feedback {
        position: absolute;
        top: -1px;
        right: -2px;
        z-index: 2;
        display: block;
        width: 34px;
        height: 34px;
        line-height: 34px;
        text-align: center;
        color: #3596e0;
        left: initial;
        font-size: 14px;
    }
</style>
```

template2.html

14. Aplicação (Interface Gráfica) do SRI Conlegis v.3 (com *Highlights)*

```
<!—template3.html -->
<link rel="stylesheet" type="text/css"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css
">
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/vue"></script>
<div class="container" id="app">
    <div class="row">
        <div class="col-md-8 col-md-offset-2"><br>
            <font size="6">Busca Conlegis </font>v.3
        </div>
    </div>
    <form action="" class="search-form">
```

```
    <div class="row">
        <div class="col-md-8 col-md-offset-2">
            <div class="has-feedback">
                <label for="search" class="sr-only">Search</label>
                <input type="text" class="form-control" name="search3"
id="search3" placeholder="busque aqui" v-model="query">
                <span class="glyphicon glyphicon-search form-control-
feedback"></span><br>

            </div>
        </div>
    </div>
    <div class="form-row">
        <div class="col-md-8 col-md-offset-2">
            <div class="col-md-6">
                <select id="sit_ato" class="form-control" v-model="q_sit">
                    <option value="Todas">Situação do Ato
Normativo</option>
                    <option value="Em Vigor">Em Vigor</option>
                    <option value="Revogado (a)">Revogado (a)</option>
                    <option value="Reeditado (a)">Reeditado (a)</option>
                    <option value="Retificado (a)">Retificado (a)</option>
                    <option value="Revogado e Reeditado">Revogado e
Reeditado</option>
                    <option value="Republicado (a)">Republicado
(a)</option>
                    <option value="Alterado (a)">Alterado (a)</option>
                    <option value="Rejeitado (a)">Rejeitado (a)</option>
                    <option value="Convertida em Lei">Convertida em
Lei</option>
                    <option value="Revogado (a) Parcialmente">Revogado (a)
Parcialmente</option>
                    <option value="Prejudicado (a)">Prejudicado
(a)</option>
                    <option value="Reeditada e Revogada">Reeditada e
Revogada</option>
                    <option value="Sem Eficácia">Sem Eficácia</option>
                    <option value="Insubsistente">Insubsistente</option>
                    <option value="Suspensa a Eficácia">Suspensa a
Eficácia</option>
                    <option value="Revigorado(a)">Revigorado(a)</option>
                    <option value="Insubsistente
Parcialmente">Insubsistente Parcialmente</option>
                    <option value="Revisto e Atualizado">Revisto e
Atualizado</option>
                    <option value="Aditado (a)">Aditado (a)</option>
                    <option value="Tornado (a) sem Efeito">Tornado (a) sem
Efeito</option>
                    <option value="Anulada">Anulada</option>
                    <option value="Revogado(a) Tacitamente">Revogado(a)
Tacitamente</option>
```

```
                    <option value="Vigência Encerrada">Vigência
Encerrada</option>
                </select>
            </div>
            <div class="col-md-6">
                <select id="tipo_ato" class="form-control" v-
model="q_tipo">
                    <option value="Todos">Tipo do Ato Normativo</option>
                    <option value="Ata">Ata</option>
                    <option value="Ato Declaratório Executivo">Ato
Declaratório Executivo</option>
                    <option value="Ato Regimental">Ato Regimental</option>
                    <option value="Boletim Eletrônico-Contato">Boletim
Eletrônico-Contato</option>
                    <option value="Boletim Informativo STF">Boletim
Informativo STF</option>
                    <option value="Circular">Circular</option>
                    <option value="Comunica Geral">Comunica Geral</option>
                    <option value="Conjunto de Pareceres
Conclusivos">Conjunto de Pareceres Conclusivos</option>
                    <option value="Decisão TCU">Decisão TCU</option>
                    <option value="Decreto">Decreto</option>
                    <option value="Decreto-Lei">Decreto-Lei</option>
                    <option value="Decreto Legislativo">Decreto
Legislativo</option>
                    <option value="Despacho">Despacho</option>
                    <option value="Emenda Constitucional">Emenda
Constitucional</option>
                    <option value="Instrução Normativa">Instrução
Normativa</option>
                    <option value="Lei">Lei</option>
                    <option value="Lei Complementar">Lei
Complementar</option>
                    <option value="Medida Provisória">Medida
Provisória</option>
                    <option value="Memorando">Memorando</option>
                    <option value="Mesa Nacional de Negociação
Permanente">Mesa Nacional de Negociação Permanente</option>
                    <option value="Norma Interna">Norma Interna</option>
                    <option value="Norma Operacional">Norma
Operacional</option>
                    <option value="Nota Técnica">Nota Técnica</option>
                    <option value="Ofício">Ofício</option>
                    <option value="Ofício-Circular">Ofício-
Circular</option>
                    <option value="Ofício-Circular Conjunto">Ofício-
Circular Conjunto</option>
                    <option value="Orientação Consultiva">Orientação
Consultiva</option>
                    <option value="Orientação Geral">Orientação
Geral</option>
```

```
                    <option value="Orientação Normativa">Orientação
Normativa</option>
                    <option value="Parecer">Parecer</option>
                    <option value="Parecer AGU">Parecer AGU</option>
                    <option value="Parecer CONJUR">Parecer CONJUR</option>
                    <option value="Portaria">Portaria</option>
                    <option value="Portaria Conjunta">Portaria
Conjunta</option>
                    <option value="Portaria Interministerial">Portaria
Interministerial</option>
                    <option value="Portaria Normativa">Portaria
Normativa</option>
                    <option value="Projeto de Lei">Projeto de Lei</option>
                    <option value="Resolução">Resolução</option>
                    <option value="Resposta Fax">Resposta Fax</option>
                    <option value="Resposta Ofício">Resposta
Ofício</option>
                    <option value="Súmula Administrativa">Súmula
Administrativa</option>
                    <option value="Súmula da Advocacia-Geral da
União">Súmula da Advocacia-Geral da União</option>
                    <option value="Nota Informativa">Nota
Informativa</option>
                    <option value="Norma de Execução">Norma de
Execução</option>
                    <option value="Lei Delegada">Lei Delegada</option>
                    <option value="Deliberação">Deliberação</option>
                    <option value="Constituição Federal E Emendas
Constitucionais">Constituição Federal E Emendas Constitucionais</option>
                    <option value="Constituição Federal">Constituição
Federal</option>
                    <option value="Atos da AGU">Atos da AGU</option>
                    <option value="Ação Direta de
Inconstitucionalidade">Ação Direta de Inconstitucionalidade</option>
                    <option value="Nota CONJUR">Nota CONJUR</option>
                    <option value="Nota AGU">Nota AGU</option>
                    <option value="Comunica SIAPE">Comunica SIAPE</option>
                    <option value="Manual de Redação da Presidência da
República">Manual de Redação da Presidência da República</option>
                    <option value="Exposição de Motivos">Exposição de
Motivos</option>
                    <option value="Exposição de Motivos
Interministerial">Exposição de Motivos Interministerial</option>
                    <option value="Resolução Normativa">Resolução
Normativa</option>
                    <option value="Comunica">Comunica</option>
                    <option value="Edital">Edital</option>
                    <option value="Nota Técnica_">Nota Técnica_</option>
                    <option value="Recurso Extraordinário">Recurso
Extraordinário</option>
                    <option value="Acórdão TCU_">Acórdão TCU_</option>
```

```
                    <option value="Acórdão TCU">Acórdão TCU</option>
                    <option value="Acórdão">Acórdão</option>
                    <option value="Anteprojeto de Lei">Anteprojeto de
Lei</option>
                    <option value="Ato">Ato</option>
                    <option value="Súmula Vinculante">Súmula
Vinculante</option>
                    <option value="Nota Técnica Conjunta">Nota Técnica
Conjunta</option>
                    <option value="emprego público">emprego
público</option>
                    <option value="Nota">Nota</option>
                    <option value="Formulação">Formulação</option>
                    <option value="Súmula">Súmula</option>
                    <option value="Diversos">Diversos</option>
                    <option value="Lei 8.112 - ANOTADA">Lei 8.112 -
ANOTADA</option>
                    <option value="Nota Informativa Conjunta">Nota
Informativa Conjunta</option>
                    <option value="Portaria AGU">Portaria AGU</option>
                    <option value="TESTE">TESTE</option>
                    <option value="Súmula do TCU">Súmula do TCU</option>
                    <option value="Nota Técnica Consolidada">Nota Técnica
Consolidada</option>
                    <option value="PARECER/PGFN">PARECER/PGFN</option>
                    <option value="PERFIL PROFISSIOGRÁFICO PREVIDENCIÁRIO -
PPP">PERFIL PROFISSIOGRÁFICO PREVIDENCIÁRIO - PPP</option>
                    <option value="SÚMULA AGU">SÚMULA AGU</option>
                    <option value="Decisão STF/Reclamação">Decisão
STF/Reclamação</option>
                    <option value="Manual de Perícia Oficial">Manual de
Perícia Oficial</option>
                    <option value="Agenda de Decisão SEGRT">Agenda de
Decisão SEGRT</option>
                    <option value="Comissão de Ética">Comissão de
Ética</option>
                    <option value="Memorando Circular">Memorando
Circular</option>
                    <option value="Portaria Gsiste">Portaria
Gsiste</option>
                    <option value="OFÍCIO/PGFN">OFÍCIO/PGFN</option>
                    <option
value="Formulação/DASP">Formulação/DASP</option>
                    <option
value="PARECER/DECOR/CGU/AGU.">PARECER/DECOR/CGU/AGU.</option>
                    <option value="Termo Aditivo">Termo Aditivo</option>
                    <option value="Manual de Processo Administrativo
Disciplinar">Manual de Processo Administrativo Disciplinar</option>
                    <option value="Instrução Normativa Conjunta">Instrução
Normativa Conjunta</option>
                    <option value="Mensagem">Mensagem</option>
```

```
                    <option value="Ato Declaratório">Ato
Declaratório</option>
                </select><br>
            </div>
        </div>
    </div>
    </form>
    <div class="row">
        <div class="col-md-10 col-md-offset-1" align="right">
            <em>Atos normativos encontrados: {{ results.length }}</em>
        </div>
        <div class="col-md-10 col-md-offset-1" v-for="result in results">
            <div class="panel panel-default">
                <div class="panel-heading">
                <!-- Titulo e nome do arquivo  -->
                     <a :href="result._source.fileurl" target="_blank"
><font size="4">{{ result._source.titulo }}</font></a><br><b>Arquivo:</b>
{{ result._source.filename }}
                </div>
                <div class="panel-body">
                <!-- Situacao do Ato, Ementa e Highlights -->
                    <b>Situação do Ato:</b> {{ result._source.sit_normativo
}}<br><b>Ementa:</b> {{ result._source.ementa }}
                <br><br><b>Highlights:</b><br>
                    <div v-if="(typeof result.highlight !== 'undefined')">
                        <ul v-for="dado in result.highlight.dados">
                            <li v-html="dado"></li>
                        </ul>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
<script>
//template.html
// create a new Vue instance
var app = new Vue({
    el: '#app',
    // declare the data for the component (An array that houses the results
and a query that holds the current search string)
    data: {
        results: [],
        query: '',
        q_sit: 'Todas',
        q_tipo: 'Todos'
    },
    // declare methods in this Vue component. here only one method which
performs the search is defined
    methods: {
        // make an axios request to the server with the current search
```

```
query
        search: function() {
            if (typeof sit_ato === 'undefined') {
                situacao = "Todas";
            } else {
                situacao = sit_ato.value;
            }
            if (typeof tipo_ato === 'undefined') {
                tipoato = "Todos";
            } else {
                tipoato = tipo_ato.value;
            }
            axios.get("http://localhost:3001/search3?q=" + this.query +
"&sit_ato=" + sit_ato.value + "&tipo_ato=" + tipo_ato.value)
                .then(response => {
                    this.results = response.data;
                })
        }
    },
    // declare Vue watchers
    watch: {
        // watch for change in the query string and recall the search
method
        query: function() {
            this.search();
        },
        q_sit: function() {
                this.search();
        },
        q_tipo: function() {
                this.search();
        }
    }

})
</script>
<!--- some styling for the page -->
<style>
    .search-form .form-group {
        float: right !important;
        transition: all 0.35s, border-radius 0s;
        width: 32px;
        height: 32px;
        background-color: #fff;
        box-shadow: 0 1px 1px rgba(0, 0, 0, 0.075) inset;
        border-radius: 25px;
        border: 1px solid #ccc;
    }

    .search-form .form-group input.form-control {
        padding-right: 20px;
```

```
        border: 0 none;
        background: transparent;
        box-shadow: none;
        display: block;
    }

    .search-form .form-group input.form-control::-webkit-input-placeholder
{
        display: none;
    }

    .search-form .form-group input.form-control:-moz-placeholder {
        /* Firefox 18- */
        display: none;
    }

    .search-form .form-group input.form-control::-moz-placeholder {
        /* Firefox 19+ */
        display: none;
    }

    .search-form .form-group input.form-control:-ms-input-placeholder {
        display: none;
    }

    .search-form .form-group:hover,
    .search-form .form-group.hover {
        width: 100%;
        border-radius: 4px 25px 25px 4px;
    }

    .search-form .form-group span.form-control-feedback {
        position: absolute;
        top: -1px;
        right: -2px;
        z-index: 2;
        display: block;
        width: 34px;
        height: 34px;
        line-height: 34px;
        text-align: center;
        color: #3596e0;
        left: initial;
        font-size: 14px;
    }
</style>
```

template3.html

ANEXO I – APLICAÇÕES UTILIZADAS

Este anexo apresenta uma breve descrição dos sistemas operacionais, programas, ferramentas, utilitários e outros tipos de software que foram utilizados no desenvolvimento e implementação deste trabalho.

1. Windows 10

Sistema Operacional (SO) desenvolvido pela Microsoft, o Windows 10 foi o sistema operacional base sobre o qual o projeto foi desenvolvido. O sistema operacional Linux é uma alternativa para o desenvolvimento da aplicação considerando que todas as demais ferramentas se encontram também disponíveis para este SO. Uma replicação de todo este processo de implementação para uma plataforma Linux é não somente viável, mas recomendando devido a diversas facilidades para desenvolvimento deste tipo de aplicações.

Ref. em março de 2019: https://www.microsoft.com/pt-br/windows

2. Cygwin

É um conjunto de ferramentas de software livre desenvolvido pela Cygnus Solutions que permite que o sistema operacional Microsoft Windows possa executar aplicações construídas para as plataformas Unix. Na prática, o Cygwin simula um terminal shell Linux e permite que diversas aplicações ou comandos específicos construídos em shell script sejam executados. No desenvolvimento deste projeto, o Cygwin foi utilizado como base para o processamento em lote dos arquivos PDF que utiliza a aplicação Pdftotext desenvolvida para Linux.

Ref. em março de 2019: https://www.cygwin.com/

3. pdftotext

É um utilitário de código aberto desenvolvido para realizar a conversão de arquivos PDF para arquivos de texto puro. Isto é realizado pelo pdftotext extraindo-se a camada textual dos arquivos PDF mas não realiza nenhum tipo de

reconhecimento ótico de caracteres para conversão das camadas de imagem dos PDFs em texto puro.

O pdftotext foi utilizado sobre o cygwin durante a primeira fase de tratamento do corpus realizando a conversão de PDF para TXT. Uma alternativa para os desenvolvedores que estão utilizando Windows pode ser a versão do pdftotext para este sistema chamada Xpdf. No caso do desenvolvimento do SRI Conlegis o conjunto Cygwin + pdftotext foi escolhido devido a facilidade de criação de *shell scripts* que foram utilizados para a iniciação automática do comando para o processamento dos milhares de documentos contidos no corpus.

Ref. em março de 2019: https://linuxappfinder.com/package/poppler-utils ; e http://www.xpdfreader.com/index.html

4. Python

É uma linguagem de programação de alto nível criada em 1991 por Guido von Rossum. Os interpretadores python são disponíveis para uma larga escala de sistemas operacionais e o seu uso mundialmente difundido, em conjunto com sua facilidade de modularização, permite um desenvolvimento simples e rápido de aplicações que trabalham sobre processamento de texto, como os necessários para o desenvolvimento do SRI Conlegis.

No desenvolvimento deste projeto, todos os códigos desenvolvidos para os processos de conversão dos arquivos TXT para o formato JSON, assim como, os combinadores de arquivos TXT e JSON para um terceiro JSON foram desenvolvidos na linguagem python.

Ref. em março de 2019: https://www.python.org/

5. Elasticsearch

Baseado no Apache Lucene, a ferramenta Elasticsearch é um poderoso motor de busca e análise de texto que executa como um servidor de aplicação em cluster. Foi desenvolvido para ser de simples e rápida configuração e pode ser escalável sendo utilizado para uma amplitude de tipos e portes de soluções. O

Elasticsearch implementa um vasto conjunto de algoritmos e técnicas de recuperação da informação, permitindo uma infinidade de estratégias diferentes de implementação praticando o estado da arte na Recuperação de Informação. Uma alternativa ao Elasticsearch, atualmente, é o Apache Solr.

Foi utilizado neste projeto para realizar a indexação de todo o conteúdo textual do corpus e para compor o motor de busca com as características específicas necessárias para o SRI Conlegis.

Ref. em março de 2019: https://www.elastic.co/

6. Kibana

O Kibana proporciona uma interface gráfica para monitoramento, administração e desenvolvimento de aplicações sobre o cluster Elasticsearch possuindo uma vasta gama de funcionalidades para descoberta e visualização de dados.

Foi utilizado no desenvolvimento do projeto para o teste e parametrização das estratégias de indexação e do algoritmo de busca, assim como para a criação de todos os índices utilizados no Elasticsearch.

Ref. em março de 2019: https://www.elastic.co/,

7. Node.js

O Node.js é um interpretador de códigos desenvolvidos na linguagem Javascript. É uma ferramenta de código aberto que tem o objetivo de migrar o Javascript que normalmente rodaria no lado do cliente, para os servidores. Além disso, permite o desenvolvimento de um conjunto enorme de aplicações sem a necessidade de um navegador com alta escalabilidade e performance.

Foi utilizado como plataforma base para executar as aplicações web que compõe a interface do SRI, uma vez que elas rodam sobre códigos desenvolvidos em Javascript.

Ref. em março de 2019: https://nodejs.org/en/

7.1 npm

É uma ferramenta gerenciadora de pacotes desenvolvidos em Javascript pela comunidade de forma livre e aberta. O próprio npm foi desenvolvido em Javascript e, além disso, é o gerenciador de pacotes padrão para o interpretador Node.js já vindo nele instalado. No npm é possível obter acesso livre a mais de 900.000 pacotes de software desenvolvidos e sustentados por desenvolvedores do mundo todo.

Ref. em março de 2019: https://www.npmjs.com/

7.2 express, body-parser, elasticsearch (connector)

Os pacotes express, body-parser e elasticsearch (este último sendo apenas um conector desenvolvido em Javascript) foram os pacotes utilizados sobre o Node.js e baixados pelo npm para serem utilizados no desenvolvimento da interface gráfica do SRI Conlegis.

O express é utilizado para a implementação de um servidor web extremamente leve e de configuração muito simples. O body-parser traz um conjunto de funcionalidades para analisar e encaminhar as requisições HTTP que serão enviadas ao express. E o elasticsearch é a biblioteca oficial do Node.js que realiza a conexão com o servidor Elasticsearch que já estará armazenando os atos normativos e executando o motor de busca do sistema.

Estas ferramentas, na implantação do SRI Conlegis, são executadas no lado do servidor.

Ref. em março de 2019: https://www.npmjs.com/package/express ; https://www.npmjs.com/package/body-parser ; https://www.npmjs.com/package/elasticsearch

7.3 axios, vue

O axios e o vue são pacotes Javascript utilizados para a implementação da interface gráfica do SRI Conlegis.

O axios funciona como um cliente HTTP intermediário entre o navegador web do usuário e o servidor da aplicação SRI para realizar tratamentos nos dados,

interceptando e transformando os dados das requisições e das respostas de forma a acelerar e facilitar a interoperabilidade dos dados que são trocados em formato JSON. Já o vue é um pacote que traz ao navegador um conjunto de funcionalidades para a construção de interfaces gráficas dinâmicas.

Tanto o axios quanto o vue são executados do lado do cliente na aplicação do SRI Conlegis.

Ref. em março de 2019: https://github.com/axios/axios ; https://vuejs.org/

8. Notepad++

O Notepad++ é um utilitário editor de texto com funcionalidades excelentes para desenvolvimento de códigos. Funciona com uma quantidade enorme de linguagens de programação, apresentando recursos de análise e marcação de sintaxe de forma a facilitar o processo de codificação.

Durante o desenvolvimento do SRI Conlegis todos os códigos foram desenvolvidos sobre o Notepad++ que foi utilizado para desenvolvimento de códigos em Shell Script, Python, JSON, HTML e SQL e Javascript.

Ref. em março de 2019: https://notepad-plus-plus.org/

9. Vim for Windows

O Vim (sigla de Vi Improved) é um editor de texto melhorado a partir do clássico editor Vi desenvolvido para os sistemas Unix. Como é uma ferramenta normalmente disponibilizada para as distribuições Linux, foi utilizada sua versão para o sistema da Microsoft, chamada Vim for Windows.

Neste trabalho o Vim foi utilizado apenas para carregar o cabeçalho do arquivo de extração (*dump*) da base de dados do Conlegis. Como a base de dados obtida não tem informação direta sobre de SGBD do qual ela foi extraída e o arquivo possui um tamanho muito grande (1.7GB) para ser expandido na memória de um editor de texto comum o Vim foi utilizado para que fossem lidas apenas as primeiras

linhas do arquivo, o seu cabeçalho, que continha as informações do tipo de SGBD utilizado.

10. MySQL Server

O MySQL é um sistema de gerenciamento de banco de dados (SGBD) da Oracle Corporation. É, atualmente, um dos SGBDs mais populares do mundo e são usados tanto para pequenas aplicações quanto por grandes empresas mundialmente conhecidas.

O MySQL foi utilizado, durante o desenvolvimento deste trabalho, para restaurar o banco oficial do Conlegis, uma vez que o sistema utiliza este SGBD como banco de dados oficial para armazenamento dos metadados necessários para a aplicação.

Ref. em março de 2019: https://www.mysql.com/

11. MySQL Workbench

O MySQL Workbench é uma ferramenta gráfica para administração, desenvolvimento, design criação e manutenção de bancos de dados MySQL. Permite, também, a extração e restauração de bases de dados de forma prática e fácil para o usuário.

Ref. em março de 2019: https://dev.mysql.com/downloads/workbench/8.0.html

Ref. em março de 2019: https://www.cygwin.com/

12. Chrome e Firefox

O Chrome e Firefox são dois renomados navegadores web muito populares desenvolvidos respectivamente pela Google e pela Mozilla Corporation.

A utilização de navegadores web foi essencial para o desenvolvimento e testes já que toda a interface gráfica do SRI Conlegis é implementada como uma aplicação web.

Ref. em março de 2019: https://www.google.com/chrome/ ; https://www.mozilla.org/pt-BR/firefox/new/

12.1 Javascript inspector/console

Os dois navegadores acima já possuem, de forma integrada, funcionalidades para desenvolvedores de aplicações web. Entre elas foi utilizada as funcionalidades de inspeção e console Javascript do lado do cliente para detecção, análise e correção de erros que aconteciam nas aplicações que são executadas no lado do cliente (principalmente referentes as bibliotecas axios e vue).