Revista Eletrônica em Gestão e Tecnologia
Print ISSN 2316-4972
ONLINE 2447-0422
FAQI
FACULDADE QI BRASIL

Publicação automática de documentos públicos como páginas web: criação e atualização programática de conteúdo via JSON no Drupal 9

Rodrigo Brum Westphalen

RESUMO
Este artigo descreve a solução técnica desenvolvida para a importação, publicação e atualização automática e periódica de informações da Carta de Serviços da Prefeitura Municipal de Novo Hamburgo no website institucional. O portal utiliza a ferramenta Drupal para gestão do conteúdo e a Carta de Serviços é gerenciada por um software interno. São discutidos conceitos de Software Livre, WCMS, Conteúdo e Modularidade. O código-fonte da solução é apresentado, o uso é demonstrado e os requisitos levantados são comparados com a solução desenvolvida. A solução se baseia na criação de um módulo *contrib*, uma configuração de *Migration* e a vinculação de uma função PHP ao *Cron* do Drupal para buscar os dados em um web service. É apresentado, ainda, a configuração de *plugin* personalizado para processamento de dados da *Migration*.

Palavras-chave: Site institucional. Automatização. PHP. Migração Drupal. JSON.

ABSTRACT
This article describes the technical solution developed for the automatic and periodic import, publication and update of information from the Carta de Serviços (Charter of Service) of the Municipality of Novo Hamburgo on its institutional website. The portal uses the Drupal tool for content management and the Charter of Service is managed by an internal software. Free Software, WCMS, Content and Modularity concepts are discussed. The source code of the solution is presented, it's use is demonstrated, and the identified software requirements are compared with the developed solution. The solution is based on creating a *contrib* module, a Migration configuration, and linking a PHP function to Drupal's *Cron* for fetching data at a web service. The configuration of a custom plugin for processing Migration data is also presented.

Keywords: Institutional Website. Automation. PHP. Drupal Migration. JSON.

1 INTRODUÇÃO

Para automatizar o processo de publicação dos serviços prestados, a Prefeitura Municipal de Novo Hamburgo, por meio da Diretoria de Inclusão Digital, iniciou um projeto de integração entre uma ferramenta interna de gestão da Carta de Serviços e o portal institucional (https://novohamburgo.rs.gov.br/). O conteúdo "Serviços" no website era, até então, publicado com preenchimento individual — o que, considerando a implementação da ferramenta de gestão interna da Carta de Serviços, passou a gerar trabalho duplicado, com risco de divergência

entre as informações, tanto por erro humano quanto por atraso na atualização. Por essa razão, por outro lado, era um trabalho possível de ser padronizado e automatizado.

Em uma primeira etapa deste projeto, estudou-se módulos do Drupal para identificar possibilidades de comunicação de dados entre sistemas; discutiu-se as possibilidades junto à Diretoria de Sistemas da Informação, equipe criadora do software de gestão da Carta de Serviços; iniciou-se a criação de um módulo personalizado (custom) para a leitura e processamento de dados de forma periódica e programática (sem ação de um usuário) a partir de um arquivo no formato JSON inicialmente fornecido em pasta local do servidor; então, com a solução parcial validada, a equipe da Diretoria de Sistemas da Informação forneceu um serviço web para buscar o arquivo JSON via HTTP. Por fim, documentou-se a solução em detalhes para consulta futura.

A presente pesquisa tem teor experimental e objetivou o estudo e implementação de um sistema de criação e atualização automático de conteúdo do website Drupal como forma de economizar trabalho humano, tornar o acesso à informação atualizada mais eficiente e desenvolver o conhecimento técnico institucional sobre as ferramentas de tecnologia da informação utilizadas. O artigo resultante discute conceitos fundamentais e descreve a solução tecnológica desenvolvida. A análise descritiva do projeto pode servir como modelo inicial para outras instituições que queiram integrar sistemas internos com portais Drupal.

2 FUNDAMENTAÇÃO TEÓRICA

Antes de apresentar a tecnologia desenvolvida, este artigo abordará alguns conceitos, como *software livre, código aberto, Web Content Management System, conteúdo e metadados, modularidade, migrações, YAML e JSON, Web Service*, além de descrever um pouco do funcionamento do software *Drupal.*

2.1 SOFTWARE LIVRE, CÓDIGO ABERTO E (WEB) CMS

Instituições com grande quantidade de informações para gerenciar e disponibilizar a público podem aproveitar estruturas de software criadas por comunidades de desenvolvedores que atuam em coletivos e redes voluntárias para gerar soluções a problemas comuns por meio do código aberto. Softwares de código aberto permitem acesso ao código-fonte da aplicação, tornando seus mecanismos de funcionamento transparentes para profissionais desenvolvedores.

Tanto o código aberto (open-source) quanto o software livre (free software) compartilham esse modelo de negócios que contrapõe o do software proprietário, garantindo "qualidade, estabilidade, desempenho e segurança, baseada em 'revisão pelos pares'" (FREITAS, 2006, p. 43).

Entre os softwares open-source disponíveis para a gestão da informação em websites, existem os *Web Content Management Systems* (WCMS) de segunda-geração, baseados em na linguagem de programação PHP. Esses sistemas facilitam a tarefa de gestão de um ou mais portais, fornecendo cargos (*roles*, níveis de acesso) à usuários e tornando possível a criação e edição de informações por pessoas sem conhecimento em desenvolvimento web (CANO et al., 2018). Antonio Tiboni (2014), em estudo sobre o Software Livre no setor público, destaca os benefícios:

> Dentre as razões para adoção de Software Livre pelo governo brasileiro, destacam-se a garantia da independência de fornecedores, o não aprisionamento a tecnologias, a possibilidade de desenvolver tecnologia própria e o fomento de iniciativas de inovação (TIBONI, 2014, p.16).

A Prefeitura Municipal de Novo Hamburgo optou pelo uso da ferramenta Drupal no portal institucional[1]. Drupal é um software livre de gestão de conteúdo (autoidentificado como "CMS") com licença GPL (GNU General Public License), permitindo o uso e modificação do seu código-fonte sem qualquer restrição ou necessidade de permissão, desde que a versão modificada mantenha as mesmas liberdades[2]. A licença GPL é mais restrita se comparada às licenças open-source (BSD, por exemplo), por condicionar a apropriação do código-fonte (distribuição, modificação e criação de trabalhos derivados) a softwares livres (que utilizem a licença GPL), impedindo, portanto, a criação de patentes — softwares proprietários — que se utilizem do software livre (FREITAS, 2006). Cano et al. (2018) utilizam o conceito de WCMS para tratar das tecnologias de gestão de conteúdo voltadas à construção e administração de páginas web — em contraste aos CMSs internos ou offline, de origem anterior e muito utilizados por empresas de mídia (MAUTHE; THOMAS, 2004) —, porém o nome CMS é popularmente usado para ambas.

Segundo o site da organização mantenedora do Drupal[3], a comunidade em torno da ferramenta é uma das maiores do mundo, com mais de um milhão de profissionais envolvidos.

1 https://novohamburgo.rs.gov.br/. Acesso em: 19 maio 2023.
2 https://www.gnu.org/licenses/gpl-3.0.html. Acesso em: 19 maio 2023.
3 https://drupal.org/. Acesso em: 19 maio 2023.

Na seção “Case Studies”[4], são informadas instituições que utilizam a tecnologia. Agências de governo dos Estados Unidos, de Londres e da França, empresas de mídia, como a BBC e NBC, e organizações internacionais e universidades, como a Anistia Internacional e a Universidade de Oxford são algumas citadas[5].

Cano et al. (2018) apontam o Drupal como terceiro WCMS mais utilizado, com 4,6% de participação de mercado (*market share*), atrás do *Joomla!* e *Wordpress*, com 6,5% e 60% respectivamente. No estudo, o Drupal foi considerado o mais complexo dos três para administrar, mas, em compensação, demonstrou ser o mais “robusto” em questão de segurança (p. 12). A versão mais atual do Drupal, no momento de escrita do artigo, é a 10.0.

O portal institucional da Prefeitura Municipal de Novo Hamburgo utiliza a versão 9 do Drupal e contém informações em diversos formatos e categorias, entre notícias, campanhas, perguntas frequentes, esquemas de estrutura organizacional, contatos, ferramentas de transparência ativa e apresentações de serviços prestados pelo Poder Executivo municipal por meio da Carta de Serviços. A criação do portal foi um esforço iniciado em 2017 para a unificação de 41 sites diferentes. O projeto foi realizado pela Diretoria de Inclusão Digital, ligado à Secretaria Municipal de Administração (Semad), e pela Diretoria de Comunicação Social, ligada ao Gabinete da Prefeita. Mais de 150 profissionais, distribuídos por diversas secretarias, já estiveram envolvidos na atualização do conteúdo[6].

A gestão dos níveis de acesso e das estruturas de software para a criação e edição de conteúdos por esses profissionais demonstra como a opção por utilizar estruturas WCMS de código aberto é estratégica para garantir maior segurança e eficiência na entrega de produtos e serviços digitais. Considerando que *conteúdo* é parte fundamental dos requisitos e funcionalidades — e que a ferramenta utilizada como alicerce é categorizada como um *sistema gestor de conteúdo* — é interessante aprofundar a definição do termo.

2.2 CONTEÚDO E METADADOS

Segundo Mauthe e Thomas (2004), o conceito de *conteúdo* não possui uma definição consensual. Os autores o definem como uma informação que pode ser acessada de forma permanente, estando sempre disponível a partir de uma requisição ao sistema. “O conteúdo

4 https://www.drupal.org/case-studies. Acesso em: 19 maio 2023.
5 https://www.drupal.org/about. Acesso em: 19 maio 2023.
6 https://www.novohamburgo.rs.gov.br/projeto. Acesso em: 19 maio 2023.

pode ser produzido, alterado, transmitido, consumido e trocado em partes ou por inteiro"[7] (p. 4). É citada a definição da Society of Motion Picture and Television Engineers (SMPTE) e da European Broadcasting Union (EBU) no contexto da indústria da mídia, onde *conteúdo* é um conjunto de informações que pode ser separada em "essência" e "metadados". A *essência* é a informação crua, que carrega a mensagem, de interesse do público, em geral, apresentada em tela: o texto, a imagem, o vídeo e o áudio. Os *metadados*, por outro lado, são informações que descrevem a essência e participam do processamento dos dados pela aplicação; são utilizados para organização, armazenamento, recuperação e exibição da informação essencial do conteúdo. Na definição dos autores, um sistema que gerencie essência e metadados é chamado de *Content Management System* (CMS). Um CMS deve prover ferramentas para criar e modificar metadados de conteúdos, além de prover metadados automáticos, reduzindo o trabalho de catalogação manual. O objetivo dos metadados é maximizar o acesso aos conteúdos existentes, feito principalmente por meio dos sistemas de busca (MORVILLE, 1998), cuja "qualidade dos resultados dependem fortemente da qualidade dos metadados com os quais os materiais foram descritos"[8] (MAUTHE; THOMAS, 2004, p. 30).

Uma ambição, expressa pelos autores Mauthe e Thomas (ibidem) sobre os CMSs, é a garantia de uma presença "permanente" do que está arquivado para o reuso. Uma das razões para o arquivamento digital é "criar uma *coleção de dados eterna*, que retenha sua integridade através do tempo, independente das mudanças nas tecnologias de armazenamento e mídia" (p. 229). Devido a isso, um paradigma inerente ao desenho de um CMS é o da *migração*. A migração deve ser uma parte integral da operação de um sistema de arquivamento digital, sendo uma função embutida, contínua e automatizada de um CMS.

Essa presença eterna dos conjuntos de dados na mídia digital é analisada de forma crítica por Chun (2008, p. 6-7):

> A característica de *sempre estar lá* da mídia digital era tornar as coisas mais estáveis, mais duradouras. A mídia digital, por meio da memória no seu núcleo, era para solucionar, senão dissolver, problemas de arquivamento como a degradação do celulóide ou um vinil riscado, e não criar problemas próprios.[9]

A autora aponta a repetição e a regeneração como características da mídia digital. Se

7 Tradução nossa. Original: *"Content can be produced, altered, transmitted, consumed, and traded in parts or in its entirety"* (p. 4).
8 Tradução nossa. Original: *"[...] the quality of search results strongly depends on the quality of the metadata with which the material has been described."* (p. 30).
9 Tradução nossa. Original: *"The always-thereness of digital media was to make things more stable, more lasting. Digital media, through the memory at its core, was supposed to solve, if not dissolve, archival problems such as degrading celluloid or scratched vinyl, not create archival problems of its own."* (p. 6-7).

algo permanece na memória digital é devido à característica de constante degeneração dessa mesma memória. É, portanto, a atualização constante que faz com que o efêmero permaneça. Esse "efêmero durante" é também uma característica do *conteúdo* (ibidem, p. 20). Nesse sentido, a *coleção de dados eterna* proposta por Mauthe e Thomas (2004) depende da permanente migração dos conteúdos.

2.3 MODULARIDADE E MIGRAÇÕES NO DRUPAL

O Drupal é um alicerce — *framework* — para construção de sites projetado para ter flexibilidade por meio de *módulos*. O conceito de modularidade, segundo Manovich (2005), representa também um princípio tecnocultural das mídias computadorizadas, junto com a *remixabilidade*. Não por isso é algo novo: a própria produção em massa das indústrias modernas se baseia na padronização de partes e como elas se compõem entre si — uma definição de *modularidade*. Porém, ainda que a modularidade fizesse parte das produções culturais de alguma maneira, a entrada do computador "modulariza a cultura num nível estrutural"[10] (p. 11), e os efeitos disso são vistos nas apropriações e *remixes* que usuários conseguem fazer de conteúdos online — e só são possíveis porque a informação é modularizada em dados. Especificamente no caso da programação, os sistemas são desenvolvidos, em boa parte, pelo que está disponível em módulos de software. "Os únicos campos onde o *sampling* e o *remixing* são feitos abertamente são na música e na programação de computador, onde desenvolvedores contam com bibliotecas de software ao escrever um novo software"[11] (p.3). Ou como a documentação do Drupal descreve: "Então, queira um criador de sites fazer um portal de notícias, loja online, rede social, blog, wiki, ou qualquer outra coisa, será apenas uma questão de combinar os módulos certos."[12] (DRUPAL, 2023a).

Uma forma de padronização de dados na mídia computadorizada é o tipo de arquivo, demarcado por sua extensão. No caso deste projeto, nos interessam os formatos PHP (PHP: Hypertext Preprocessor), YAML (YAML Ain't Markup Language) e JSON (JavaScript Object Notation). Como já apresentado, o Drupal é um software escrito em PHP, uma linguagem de programação de código aberto que compõe páginas web a partir de um processamento no lado

10 Tradução nossa. Original: *"[...] computerization modularizes culture on a structural level."* (p. 11).
11 Tradução nossa. Original: *"The only fields where sampling and remixing are done openly are music and computer programming, where developers rely on software libraries in writing new software."* (p. 3).
12 Tradução nossa. Original: *"So, whether a site builder is looking to create a news site, online store, social network, blog, wiki, or anything else, it's just a matter of combining the right modules. The only limitation is the creator's imagination."*

do servidor (server-side) e depende de um software interpretador (PHP, 2023). A modularidade do Drupal, em parte, se apresenta na forma como ele grava, lê e processa dados de configuração por meio de arquivos YAML. O YAML e o JSON são formatos para serialização de dados estruturados — e a serialização é um processamento dos dados para leitura em série, normalmente strings binárias, frequentemente utilizado na transmissão de dados (ERIKSSON; HALLBERG, 2011). No caso do YAML e JSON, ambos formatos são arquivos de texto simples, facilmente legíveis por pessoas, que levam a extensão ".yml" e ".json", respectivamente. Descrições das sintaxes de ambos estão disponíveis no trabalho de Eriksson e Hallberg (2011), e demonstrações de cada um estão no item 4.

As migrações, no Drupal, ocorrem por meio de um módulo *core* (módulos que fazem parte do Drupal), "Migrate", sendo possível estender as funcionalidades por meio de módulos *contrib* (módulos de contribuição da comunidade Drupal, não incorporados ao WCMS) (HUFFSTEDTLER, 2023). A migração pode ser feita a partir de um banco de dados, de um site de versão anterior do Drupal e de outros WCMSs, de modo a criar objetos próprios do Drupal, como *nodes, users, files, terms* e *comments* (RYAN, Mike et al. 2022). Alguns módulos *contrib* fornecem suporte para a importação de arquivos de planilha (ods, csv), XML (eXtensible Markup Language) e JSON. Desses, o módulo "Migrate Plus" estende as funcionalidades principais do framework de migração nativo, permitindo que migrações sejam criadas a partir de arquivos de configuração — *configuration entities* — em YAML e fornecendo funcionalidades de leitura e processamento de dados por meio de *plugins* (RYAN; HEDDING; DOROSHENKO, 2023a), enquanto o "Migrate Tools" fornece comandos de gestão de migrações, além de uma interface de usuário, dentro da área administrativa do site Drupal, para executá-los de forma simplificada (RYAN; HEDDING; DOROSHENKO, 2023b).

2.3.1 Terminologia Drupal: Node, Content Type, Term, Hook e Plugin

No Drupal, todo conteúdo é tratado internamente como um *Node*. Na documentação, são definidos quatro tipos de dados: *Content*, *Configuration*, *State* e *Session* (HODGDON, 2023). Para nosso projeto, interessa entender *Content* e *Configuration*. *Content* são *Nodes* também chamados de *Content Entity*. As diferentes entidades são agrupadas por tipos e subtipos, para permitirem maior variedade de estrutura (HODGDON; DUNHAM, 2023). O conteúdo que nos referimos aqui tem como subtipo o valor de nome de uma configuração chamada *Content Type*.

A gerência dos conteúdos e seus tipos pode ser feita pelo painel administrativo do site Drupal. As *Configurations* são armazenadas internamente como arquivos YAML, já mencionados anteriormente. Módulos do Drupal são implementados por meio de arquivos de *Configuration* e código PHP vinculados à *hooks* do Drupal. Os *hooks* são funções PHP com nomenclatura específica, que são vasculhadas pelo Drupal para execução conforme a funcionalidade de cada *hook* (SHINDELAR, 2022). Enquanto os *plugins* são funções PHP chamadas por módulos por meio de arquivos de configuração. Exemplos de usos serão apresentados em "Resultados".

Cada *Node* de *Content* pode conter *Fields* (campos). Os campos — que atuam similarmente a chaves de um objeto — são definidos também por arquivos de configuração (HODGDON; DUNHAM, 2023). Um conteúdo do site, por exemplo, contém um *Field* "*title"* obrigatório, e um *Field* "*body"*, onde se insere um texto HTML (*HyperText Markup Language*). Podem haver campos para números, listas, ou que apontem para outras entidades do Drupal. Os *Nodes* do tipo *Term*, por exemplo, são entidades que podem ser hierarquizadas por meio da Taxonomia (cuja funcionalidade é regida por um módulo *core* do Drupal), servindo para classificar conteúdos, agrupá-los e filtrá-los em buscas. Diferente de um valor de texto ou numeral, um *Field* pode conter o *id* de uma outra entidade — como um *Term* — e ser, assim, do tipo *Entity Reference.*

2.4 WEB SERVICE

Os serviços web (*web services*) são formas de distribuir sistemas em locais diferentes, permitindo-lhes o compartilhamento de dados via uma conexão de rede por meio do HTTP (*HyperText Transfer Protocol*) e sua extensão criptografada, o HTTPS (*Secure*). Diferentemente de documentos HTML e recursos de mídia que são servidos para a composição de um website no navegador de um usuário, o web service é voltado para compartilhar dados estruturados entre aplicações. Outro termo utilizado para se referir aos serviços web que entregam dados estruturados com JSON ou XML é Web API (*Web Application Programming Interface*), geralmente relacionados a *web services* que seguem os princípios arquiteturais RESTful (SALVADORI, 2015).

2.5 CARTA DE SERVIÇOS AO USUÁRIO

Conforme a Lei Federal nº 13.460/2017, que "dispõe sobre participação, proteção e defesa dos

direitos do usuário dos serviços públicos da administração pública" (BRASIL, 2017), os três Poderes — Executivo, Legislativo e Judiciário — nas três esferas de Governo — Federal, Estadual e Municipal — são obrigados por lei a divulgarem os serviços prestados, apontando os órgão ou entidades responsáveis pela realização e a autoridade administrativa a quem estão subordinados ou vinculados. O Art. 7º desta lei define a Carta de Serviços ao Usuário, documento com o objetivo de informar formas de acesso, requisitos, etapas, prazos, padrões de qualidade e compromissos do órgão público e locais e formas para o usuário apresentar manifestações sobre serviços realizados. Cabe a cada Poder regulamentar a operacionalização da própria Carta de Serviços, que deve, conforme o § 4º ser "objeto de atualização periódica e de permanente divulgação mediante publicação em sítio eletrônico do órgão ou entidade na internet".

A Prefeitura Municipal de Novo Hamburgo dispõe da Carta de Serviços ao Usuário em formato PDF para download[13] e em páginas individuais no site institucional[14]. No momento de escrita deste projeto, existem 215 serviços disponíveis em páginas do site. A escrita e publicação desses materiais eram, até então, realizados manualmente. O projeto interno de criação de um sistema de gestão da Carta de Serviços ao Usuário pela Diretoria de Sistemas da Informação levou à discussão da possibilidade de integrar os sistemas, prezando pela eficiência da manutenção e disponibilização atualizada de informações à população.

3 METODOLOGIA

O presente artigo descreve uma pesquisa experimental com objetivo de automatização do processo de criação de conteúdo no portal institucional da Prefeitura Municipal de Novo Hamburgo. O experimento consistiu na análise de requisitos do sistema, no estudo dos softwares vinculados ao Drupal, no desenvolvimento de uma solução técnica e na documentação da solução.

Os requisitos foram levantados a partir de *group work* (REHMAN et al. 2013), reunindo as equipes para discutir as necessidades e possibilidades. O estudo dos softwares vinculados ao Drupal foi uma pesquisa exploratória, necessária ao desenvolvimento da solução e cujas referências consultadas embasaram parte da fundamentação teórica. A solução técnica foi

13 Disponível em: https://novohamburgo.rs.gov.br/sites/pmnh/files/servicos/Carta_servicos_usuario_01_21.pdf. Acesso em: 26/04/2023.
14 Disponível em: https://www.novohamburgo.rs.gov.br/servicos. Acesso em: 26/04/2023.

desenvolvida de modo incremental, testando manualmente os efeitos da execução em um servidor de desenvolvimento. A documentação foi escrita em paralelo ao software, sendo atualizada conforme a melhoria da solução. Os detalhes estão no item 4.

4 RESULTADOS

A solução desenvolvida utiliza-se do sistema de migrações do Drupal para ler e processar os dados de um arquivo JSON cuja estrutura foi criada em comum acordo entre as equipes das Diretorias. Para realizar a verificação periódica deste arquivo por modificações, utilizou-se do "hook_cron", que vincula uma função à execução do Cron do servidor — composto de diversas funções PHP, normalmente relacionadas a segurança e manutenção, com periodicidade configurada pelo painel administrativo do site (DRUPAL, 2023b). A partir da validação inicial da solução com um arquivo JSON local, um web service foi criado pela Diretoria de Sistemas da Informação para servir o arquivo sempre atualizado na mesma URL. A solução, então, foi adaptada para buscar o arquivo via HTTPS.

Com nome de "Migrando JSON", o software desenvolvido funciona como módulo *contrib* do Drupal, podendo ter seus arquivos de configuração modificados para atender às necessidades de outras instituições e, então, ser instalado via painel administrativo. O código-fonte será disponibilizado para a comunidade de desenvolvedores após a consolidação da solução no servidor de produção. A documentação em detalhes do software encontra-se no arquivo "README.md", que será disponibilizado junto, via repositório online aberto.

Como forma de organizar a descrição de resultados, dividiu-se a seção nos subtítulos "Análise de requisitos", "Dependências e configurações iniciais", "Dados de Entrada", "Configuração da migration", "Configuração do Módulo Contrib" e "Plugins Personalizados" e "Demonstração de Conteúdo Migrado".

4.1 ANÁLISE DE REQUISITOS

A etapa de análise de requisitos foi realizada em reuniões entre as duas equipes, discutindo em conjunto as necessidades, limitações e possibilidades das tecnologias existentes. Duas reuniões foram realizadas, intercaladas com estudos das ferramentas, para começar o desenvolvimento da solução técnica. Os requisitos levantados foram:

I. Os dados de entrada devem ser independentes da plataforma Drupal;
II. A leitura e processamento dos dados pelo Drupal deve ocorrer de forma periódica e programática (sem intermediação humana);
III. Os dados de saída devem ser *Contents* reconhecidos e gerenciáveis pela interface Drupal;
IV. A modificação de um dado de entrada deve modificar o *Content* que o refere;
V. Os dados de entrada devem poder despublicar um *Content;*
VI. A solução deve ter documentação detalhada para consulta futura.

4.2 DEPENDÊNCIAS E CONFIGURAÇÕES INICIAIS

A versão do Drupal utilizada é 9.2. Os módulos Migrate Tools (v6.0) e Migrate Plus (v6.0) foram instalados a partir do download manual dos arquivos no site Drupal e inserção destes na pasta `/modules/contrib`. O módulo *core* Migrate foi instalado junto a estes pelo painel administrativo do site.

Um *Migration Group* será criado automaticamente pelo módulo Migrando JSON. As configurações do grupo estão descritas no arquivo `migrando_json/config/install/migrate_plus.migration_group.servicos_migration_json.yml`. Também é possível criar um novo grupo por meio da página de administração, na URL `/admin/structure/migrate/`, e copiar, então, o nome de máquina do grupo para a chave "migration_group" do arquivo de configuração da *Migration,* apresentado no item 4.4.

Foi criado manualmente um usuário apenas para a publicação dos conteúdos da integração, chamado *integrador-octo*. O número de *id* do usuário foi copiado a partir do acesso ao perfil deste, encontrando-se, por padrão, na URL `/user/{uid}`. Esse valor é incluído então na configuração da *Migration*, no campo `uid`, para atribuir corretamente a autoria dos conteúdos criados por via programática.

Para a execução de migrações, é preciso que o *Content Type* dos conteúdos que serão importados já exista, prevendo os campos (*fields*) necessários para exibição e pesquisa nas páginas do site. Abordar a criação de campos e da estrutura de páginas está fora do escopo desse artigo.

No nosso caso, temos o *Content Type* "Serviço", com o id `servico`. O quadro a seguir representa a tabela disponível no painel de administrador do site.

Quadro 1 - Campos do Content Type “Serviço”

LABEL	MACHINE NAME	TIPO DO CAMPO
Corpo	body	Texto (formatado, longo, com resumo)
Público alvo	field_publico_alvo	Listagem (texto)
Secretaria	field_secretaria	Referência de entidade
Contatos	field_contatos	Texto (formatado, longo)
Documentos	field_documentos	Texto (formatado, longo)
Tags	field_tags	Referência de entidade

Fonte: elaborado pelos autores.

4.3 DADOS DE ENTRADA

Para cumprir com o requisito I, as equipes definiram o formato do arquivo com os dados de entrada da migração como JSON.

Quadro 2 - Exemplo de Estrutura de Dados de Entrada (JSON)

```
{
	"servicos": [
	{
		"uuid": "",
		"titulo": "",
		"status": 1,
		"body": "",
		"publico_alvo": "",
		"secretaria": "",
		"contatos": [
		{
				"lotacao": "",
				"tel": "",
				"email": ""
		},
		{
				"lotacao": "",
				"tel": "",
				"email": ""
		}
		],
		"tags": "",
```

```
            "documentos": [
            {
                        "label": "",
                        "filename": ""
            },
            {
                        "label": "",
                        "filename": ""
            }
            ]
      }
      ]
}
```

Fonte: elaborado pelos autores.

4.4 CONFIGURAÇÃO DA MIGRATION

Ao instalar o módulo, a configuração modelo (Quadro 3), disponível na pasta `migrando_json/config/install` será automaticamente instalada junto. As configurações de *Migration* também podem ser importadas via painel administrativo do Drupal, pela URL `/admin/config/development/configuration/single/import`. Por meio da importação, é possível atualizar configurações de uma *Migration* sem precisar reinstalar o módulo. A chave de API do web service que nossa Migration consome foi omitida por segurança.

Quadro 3 - Exemplo de Migration (YAML)

```
uuid: 712745db-b2fb-4d02-9cb2-bc9816b4e1d6
langcode: en
status: true
dependencies:
  enforced:
       module:
       - migrando_json
migration_dependencies: null
id: carta_de_servicos_migration
class: null
field_plugin_method: null
cck_plugin_method: null
migration_tags:
  - "json"
  - "file"
migration_group: servicos_migration_json
label: "Importando Serviços via JSON"
source:
  plugin: url
  data_fetcher_plugin: http
  data_parser_plugin: json
  track_changes: true
  urls:
       - "https://ws.novohamburgo.rs.gov.br/index.php/api/Octo/buscaServico"
  item_selector: servicos
  fields:
```

```
      - name: src_uuid
        label: "UUID - Identificador Unico do Serviço"
        selector: uuid
      - name: src_titulo
        label: "Título do Serviço"
        selector: titulo
      - name: src_status
        label: "Serviço está público no site? (0 | 1)"
        selector: status
      - name: src_body
        label: "Corpo HTML do Artigo para o Portal"
        selector: body
      - name: src_publico_alvo
        label: "Público Alvo (Cidadão, Empresa, Servidor)"
        selector: publico_alvo
      - name: src_secretaria
        label: "Secretaria Principal do Serviço"
        selector: secretaria
      - name: src_contatos
        label: "Contatos Responsáveis por Tirar Dúvidas"
        selector: contatos
      - name: src_tags
        label: "Tags úteis para pesquisa"
        selector: tags
      - name: src_documentos
        label: "Documentos anexos"
        selector: documentos
  ids:
      src_uuid:
      type: string
process:
  type:
      plugin: default_value
      default_value: servico
  title: src_titulo
  status: src_status
  uuid: src_uuid
  body/value: src_body
  body/format:
      - plugin: default_value
      default_value: full_html
  field_publico_alvo:
      - plugin: skip_on_empty
      source: src_publico_alvo
      method: process
      message: "Público Alvo não encontrado no JSON original"
      - plugin: explode
      delimiter: ", "
  field_secretaria:
      - plugin: skip_on_empty
      source: src_secretaria
      method: process
      message: "Termo de Taxonomia de 'Secretaria' não encontrada no JSON
original"
      - plugin: entity_lookup
      entity_type: taxonomy_term
      value_key: name
      bundle_key: vid
      bundle: secretaria
      ignore_case: true
      single: true
  field_tags:
      - plugin: skip_on_empty
      source: src_tags
      method: process
```

```
      message: "Tags não encontradas no JSON original"
      - plugin: explode
      delimiter: ", "
      - plugin: entity_generate
      entity_type: taxonomy_term
      value_key: name
      bundle_key: vid
      bundle: tag
      ignore_case: true
      single: false
  field_contatos/value:
      - plugin: skip_on_empty
      source: src_contatos
      method: process
      message: "Contatos não encontrados no JSON original"
      - plugin: contatos_para_html
  field_contatos/format:
      - plugin: default_value
      default_value: full_html
  field_documentos/value:
      - plugin: skip_on_empty
      source: src_documentos
      method: process
      message: "Documentos não encontrados no JSON original"
      - plugin: documentos_para_html
      public_url: /sites/pmnh/files/servicos/
  field_documentos/format:
      - plugin: default_value
      default_value: full_html
  uid:
      plugin: default_value
      default_value: 4872
destination:
  plugin: "entity:node"
  default_bundle: servicos
```

Fonte: elaborado pelos autores.

Para documentar a solução, o arquivo de configuração foi separado em partes e teve o código comentado no arquivo “README.md”. Da chave *uuid* até *label*, são configurados metadados sobre a migração; em *source*, detalhes sobre o arquivo de entrada, apontando a URL para busca do JSON via HTTP; em *item_selector*, define-se a chave raíz da lista de conteúdos que será lida pela *Migration*; em *fields*, o caminho da chave que será lida no arquivo de entrada é definido pelo *selector* e seu valor é atribuído à variável *name*; em *ids*, é informada a chave única dos itens da lista; em *process*, definem-se as chaves do objeto de saída — do *Content* — para receberem o retorno de operações gerenciadas por *plugins*; os campos que não precisam de processamento apenas recebem como valor o nome da variável criada em *fields*, enquanto os outros passam por uma sequência de um ou mais plugins fornecidos pelo Migrate e pelo Migrate Plus (SIPILÄ et al. 2018) — por exemplo, *field_tags* recebe ou não uma *string* de termos separados por vírgula, que são separados em um *array*, onde cada item passará por um *look-up* nos termos de taxonomia existentes e, caso não seja encontrado, novos termos serão

gerados; o campo *uid* atribui a autoria do conteúdo a um usuário a partir de seu *id*, como descrito no item 4.2; em *destination*, é informado o plugin “entity:node”, que define a estrutura dos dados de saída.

4.5 CONFIGURAÇÃO DE MÓDULO CONTRIB

O módulo Migrando JSON é definido no arquivo `migrando_json.info.yml` e tem sua funcionalidade descrita no arquivo `migrando_json.module`, que a vincula ao *hook_cron*. As dependências, como de arquivos de configuração de *Migrations*, são definidas no arquivo “info.yml”.

Quadro 4 - Arquivo “migrando_json.info.yml” (YAML)

```
type: module
name: Migrando JSON
description: "Migração de JSON para Content"
package: Migration
core_version_requirement: ">=9.1"
dependencies:
  - drupal:migrate
  - migrate_plus:migrate_plus
  - migrate_tools
config:
  - migrate_plus.migration.carta_de_servicos_migration
```

Fonte: elaborado pelos autores.

Quadro 5 - Arquivo “migrando_json.module” (PHP)

```
<?php

use Drupal\migrate\MigrateExecutable;
use Drupal\migrate\MigrateMessage;
use Drupal\migrate\MigrateSkipRowException;

/**
 * Implements hook_cron().
 */
function migrando_json_cron()
{
      migrando_json_migration_execute();
}

/**
 * Executa a Migration "carta_de_servicos_migration".
 */
function migrando_json_migration_execute()
{
      // Load the migration plugin.
```

```
      $migration_plugin_id = 'carta_de_servicos_migration';
      $migration =
\Drupal::service('plugin.manager.migration')->createInstance($migration_p
lugin_id);

      // Create a MigrateExecutable object.
      $executable = new MigrateExecutable($migration, new
MigrateMessage());

      try {
      // Execute the migration.
      $executable->import();
      } catch (MigrateSkipRowException $e) {
      // Handle skipped rows, if necessary.
      }
}
```

Fonte: elaborado pelos autores.

4.6 PLUGINS PERSONALIZADOS

Para exibir o conteúdo de cada Serviço em uma página da forma como a Diretoria de Inclusão Digital definiu, foi necessário criar dois *plugins* personalizados para o processamento dos dados na *Migration*, "contatos_para_html" e "documentos_para_html". Os *plugins* são classes derivadas de *ProcessPluginBase*, mapeadas a partir da *annotation* "@MigrateProcessPlugin()", que implementam o método *transform.* Por meio dele, recebe-se um valor do *pipeline* de processamento da *Migration* e retorna-se um valor para o *pipeline*. O *plugin* personalizado apresentado no quadro 7 transforma múltiplos valores em uma única *string* de HTML.

Quadro 7 - Trecho do arquivo "contatosParaHtml.php"

```
/* . . . */
/**
 * @MigrateProcessPlugin(
 *   id = "contatos_para_html",
 *   handle_multiples = TRUE
 * )
 */
class contatosParaHtml extends ProcessPluginBase
{
      public function transform($value, MigrateExecutableInterface
$migrate_executable, Row $row, $destination_property)
      {
            $body = array('');
            if (count($value) > 0) {
                  $body[0] = "<h3>Dúvidas?</h3>";
                  foreach ($value as $contato) {
                        $lotacao = $contato['lotacao'];
                        $tel = $contato['tel'];
                        $email = $contato['email'];
                        $html = "<strong>{$lotacao}</strong><br><i
class='fas fa-phone'></i> {$tel}<br><i class='fas fa-envelope'></i>
{$email}<br>";
                        $body[] = $html;
```

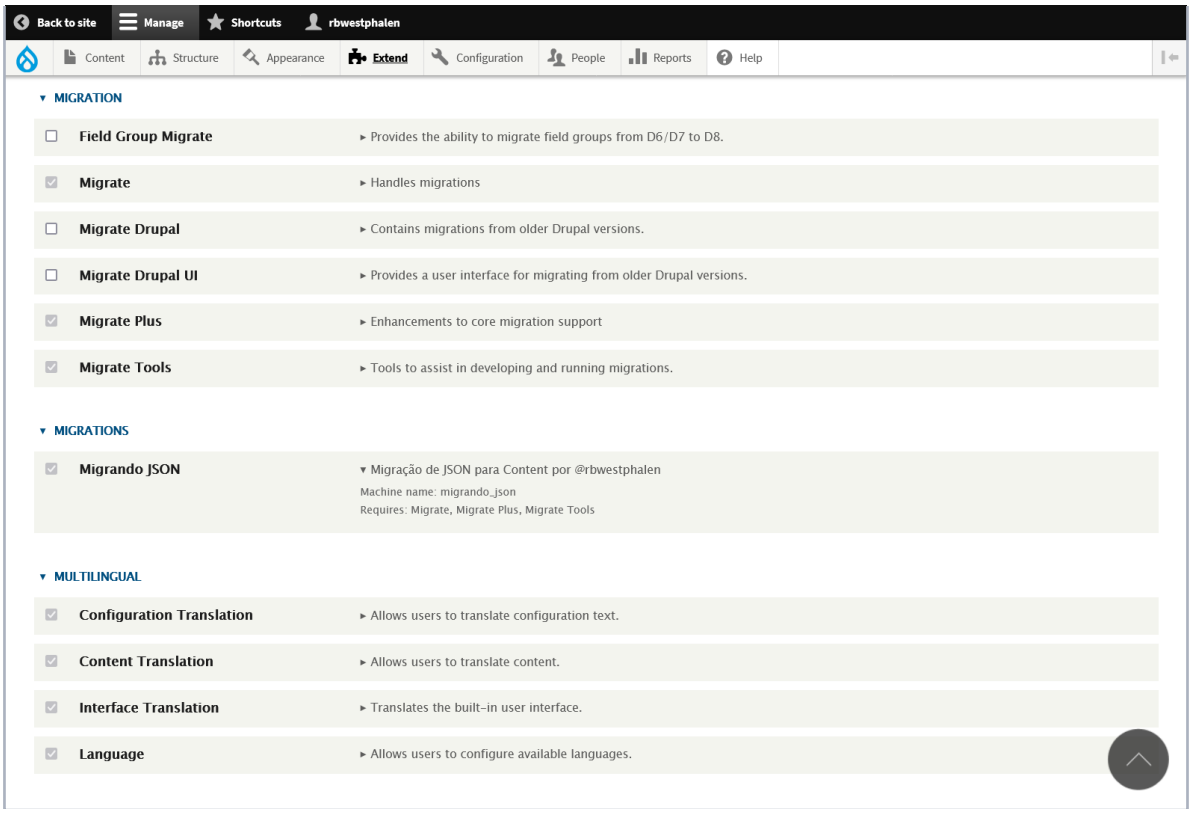
```
                    }
                }
                return implode("<br>", $body);
        }
}
```

Fonte: elaborado pelos autores.

4.7 DEMONSTRAÇÃO DE CONTEÚDO MIGRADO

Para executar a *Migration*, colocamos o módulo Migrando JSON na pasta `/sites/all/modules/contrib` e o instalamos via painel de administrador (Figura 1).

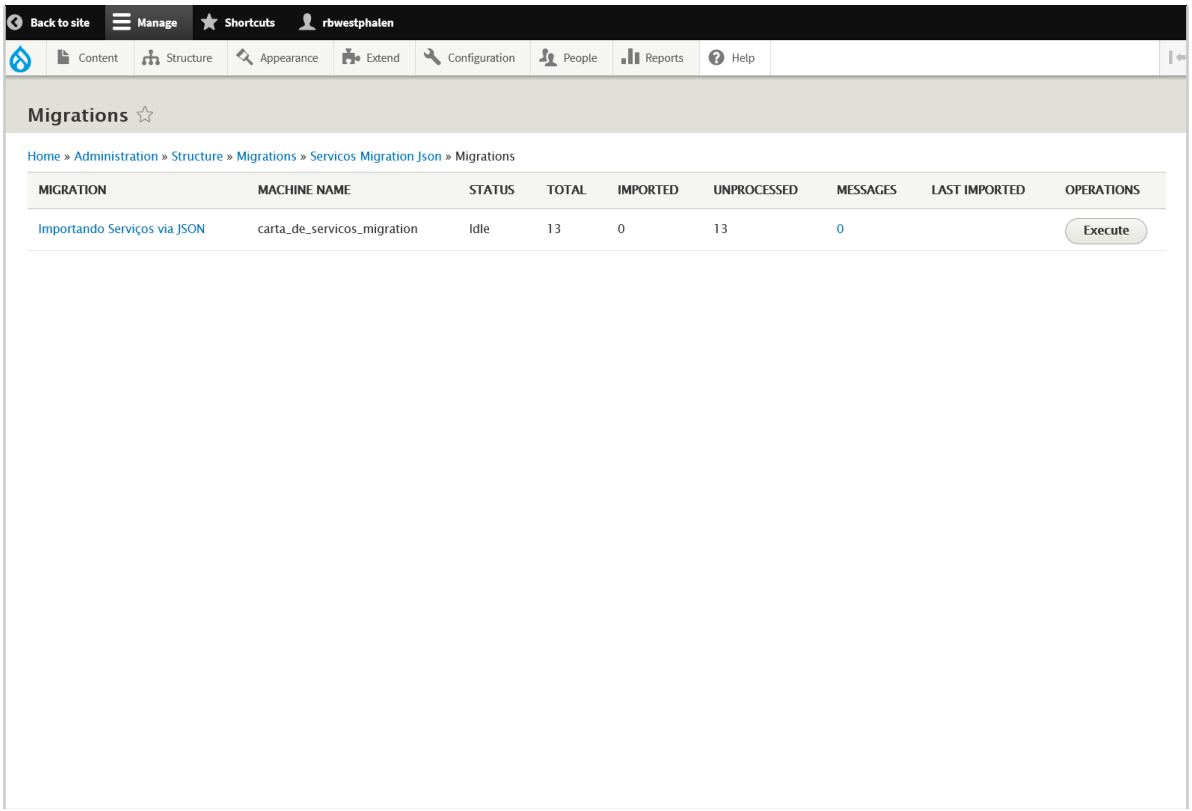
Figura 1 - Captura de Tela da Instalação do Módulo Contrib



Fonte: elaborado pelos autores.

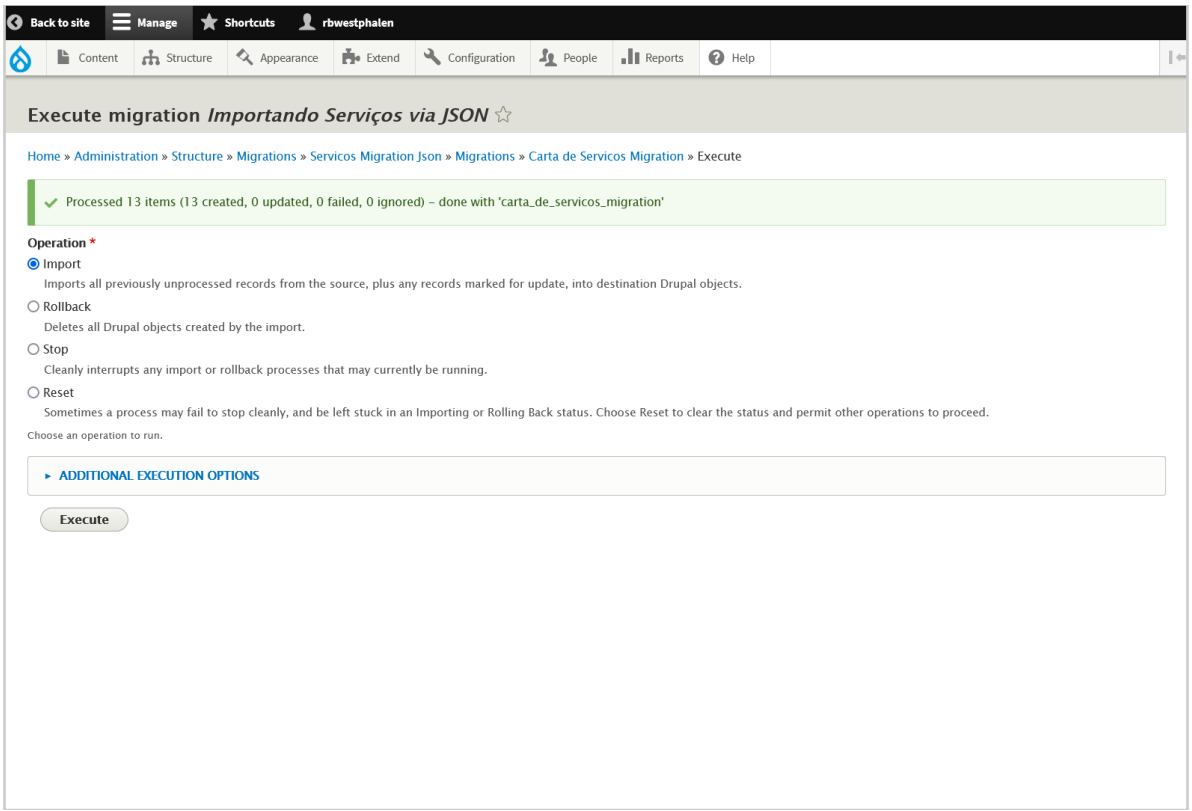
Identificamos a *Migration* via painel de administração e a executamos (Figuras 2 e 3).

Figura 2 - Captura de Tela da Identificação da Migration criada



Fonte: elaborado pelos autores.

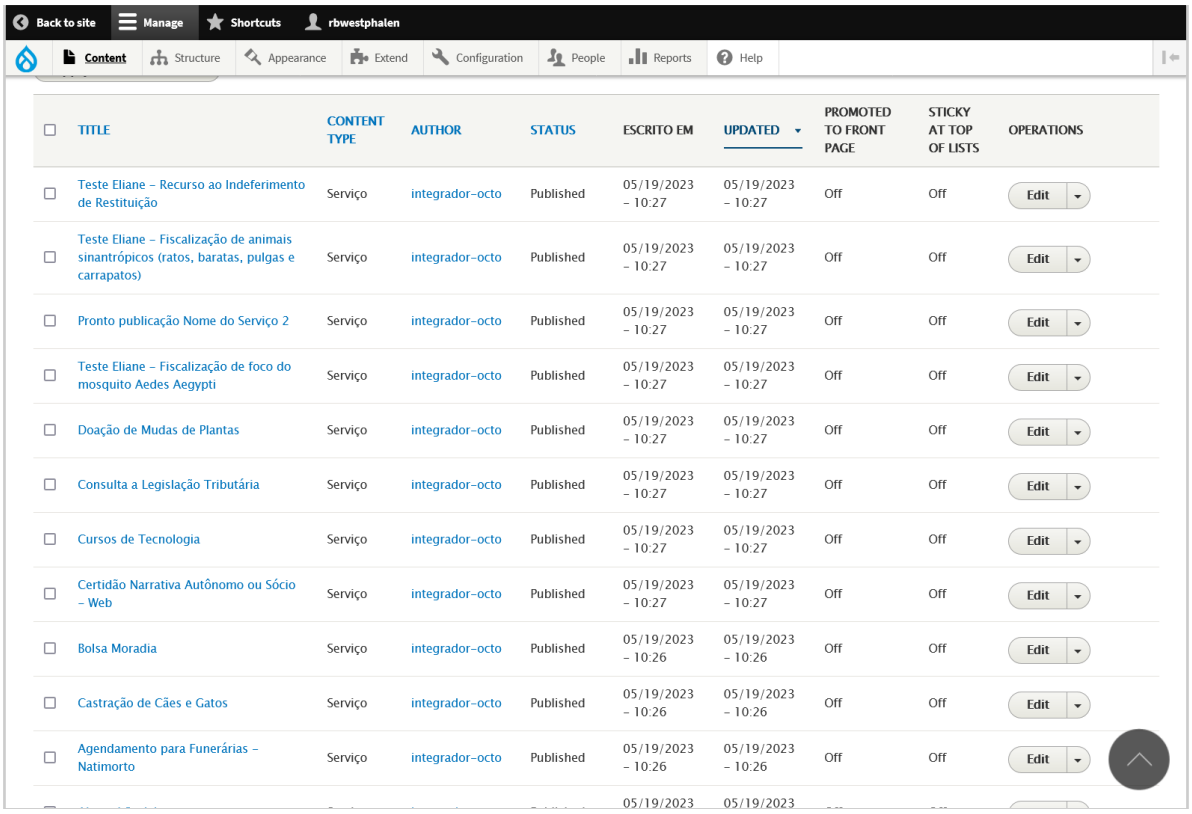
Figura 3 - Captura de Tela da Execução da Migration



Fonte: elaborado pelos autores.

Verificamos, então, a página de listagem de *Contents* (URL: `/admin/content/node`) para encontrar os itens migrados (Figura 4).

Figura 4 - Captura de Tela da Listagem de Contents

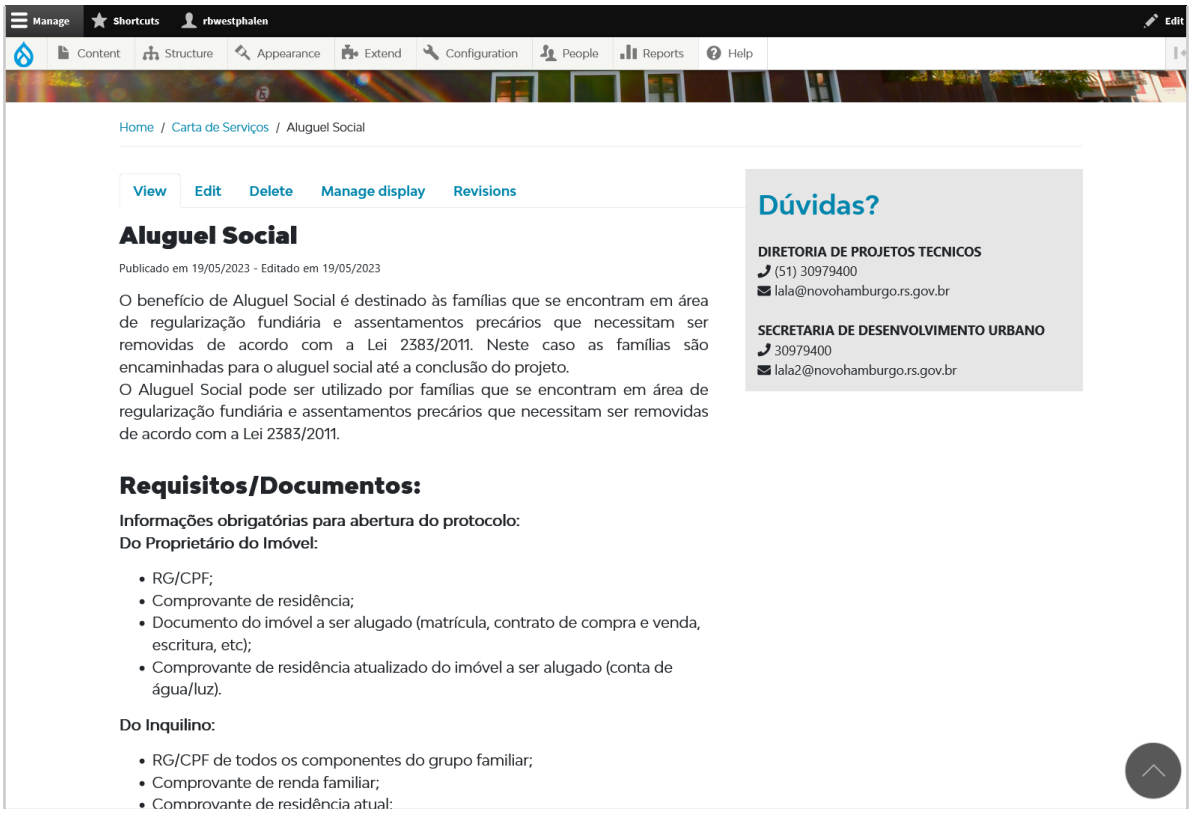


TITLE	CONTENT TYPE	AUTHOR	STATUS	ESCRITO EM	UPDATED	PROMOTED TO FRONT PAGE	STICKY AT TOP OF LISTS	OPERATIONS
Teste Eliane – Recurso ao Indeferimento de Restituição	Serviço	integrador-octo	Published	05/19/2023 – 10:27	05/19/2023 – 10:27	Off	Off	Edit
Teste Eliane – Fiscalização de animais sinantrópicos (ratos, baratas, pulgas e carrapatos)	Serviço	integrador-octo	Published	05/19/2023 – 10:27	05/19/2023 – 10:27	Off	Off	Edit
Pronto publicação Nome do Serviço 2	Serviço	integrador-octo	Published	05/19/2023 – 10:27	05/19/2023 – 10:27	Off	Off	Edit
Teste Eliane – Fiscalização de foco do mosquito Aedes Aegypti	Serviço	integrador-octo	Published	05/19/2023 – 10:27	05/19/2023 – 10:27	Off	Off	Edit
Doação de Mudas de Plantas	Serviço	integrador-octo	Published	05/19/2023 – 10:27	05/19/2023 – 10:27	Off	Off	Edit
Consulta a Legislação Tributária	Serviço	integrador-octo	Published	05/19/2023 – 10:27	05/19/2023 – 10:27	Off	Off	Edit
Cursos de Tecnologia	Serviço	integrador-octo	Published	05/19/2023 – 10:27	05/19/2023 – 10:27	Off	Off	Edit
Certidão Narrativa Autônomo ou Sócio – Web	Serviço	integrador-octo	Published	05/19/2023 – 10:27	05/19/2023 – 10:27	Off	Off	Edit
Bolsa Moradia	Serviço	integrador-octo	Published	05/19/2023 – 10:26	05/19/2023 – 10:26	Off	Off	Edit
Castração de Cães e Gatos	Serviço	integrador-octo	Published	05/19/2023 – 10:26	05/19/2023 – 10:26	Off	Off	Edit
Agendamento para Funerárias – Natimorto	Serviço	integrador-octo	Published	05/19/2023 – 10:26	05/19/2023 – 10:26	Off	Off	Edit

Fonte: elaborado pelos autores.

Os conteúdos foram criados com sucesso e as páginas de teste podem ser vistas no portal (Figuras 5 e 6). A Figura 5 apresenta na lateral direita o produto do plugin custom `contatos_para_html`.

Figura 5 - Captura de Tela de Página de Teste Criada pela Migração



Fonte: elaborado pelos autores.

A Figura 6 apresenta o elemento de tabela com links para download gerado pelo plugin custom `documentos_para_html` em “Arquivos” e as entidades de Taxonomia geradas e vinculadas automaticamente pela Migração em “Tags”.

Figura 6 - Captura de Tela de Elementos Gerados Pela Migração

Fonte: elaborado pelos autores.

5. CONCLUSÃO

A solução desenvolvida atende aos requisitos levantados, ao definir o arquivo de entrada como JSON (I); ao vincular a leitura e o processamento dos dados ao *Cron* do servidor (II); ao verificar a possibilidade de edição do conteúdo via painel administrativo (III); ao definir *uuids* nos dados de entrada e o atributo "track_changes" na *Migration* (IV); ao receber e processar o atributo *status* nos dados de entrada (V); ao documentar o módulo *contrib* em artigo, externamente, e em tutorial completo no arquivo "README.md", à ser disponibilizado no futuro (VI).

O projeto é bem sucedido em avançar o desenvolvimento de tecnologias de código-aberto por — e para — o Poder Público, inserindo-se no contexto do software livre para sistemas de gestão de conteúdo na web (WCMS) e prezando pelos princípios da legalidade (no atendimento à norma jurídica referente à Carta de Serviços), eficiência (na redução do trabalho humano) e publicidade (na agilidade de publicação e manutenção da Carta de Serviços e na divulgação, em contexto científico, da solução técnica produzida), expressos no art. 37 da Constituição (BRASIL, 1988).

A solução, até o momento de escrita deste artigo, não foi executada no servidor de produção por ainda ser necessário a inclusão de todos os serviços atualmente disponíveis no portal institucional no software de gestão interna da Carta de Serviços. A partir da conclusão desta etapa, os conteúdos atuais do site serão excluídos e a migração importará todos os serviços no formato mais atualizado. Com a efetiva consolidação da integração no servidor de produção,

o módulo *contrib* será disponibilizado para a comunidade de desenvolvedores com código-aberto e licença GPL-3.

REFERÊNCIAS

BRASIL. **Constituição da República Federativa do Brasil.** 1988. Disponível em: https://www.planalto.gov.br/ccivil_03/Constituicao/Constituicao.htm. Acesso em: 02 maio 2023.

BRASIL. **Lei n. 13.640, de 26 de junho de 2017.** Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2017/lei/l13460.htm. Acesso em: 19 maio 2023.

CANO, Maria-Dolores et al. **A Comparative Study of Web Content Management Systems.** Information, vol. 9, ed. 2. 2018. p. 27. Disponível em: https://www.mdpi.com/2078-2489/9/2/27. Acesso em: 24 abr. 2023.

CHUN, Wendy. **The Enduring Ephemeral, or the Future Is a Memory.** In: Critical Inquiry, vol 35, 2008, p. 148-171. Disponível em: https://summit.sfu.ca/_flysystem/fedora/sfu_migrate/18196/Hui_Kyong_Chun--the_enduring_ephemeral-2008.pdf. Acesso em: 19 maio 2023.

DRUPAL. Drupal, © 2000-2023. **Overview of Drupal.** 2023a. Disponível em: https://www.drupal.org/docs/understanding-drupal/overview-of-drupal. Acesso em: 25 abr. 2023.

DRUPAL. Drupal, © 2000-2023. **function hook_cron**. 2023b. Disponível em: https://api.drupal.org/api/drupal/core%21core.api.php/function/hook_cron/9. Acesso em: 27/04/2023.

ERIKSSON, Malin; HALLBERG, Victor. **Comparison between JSON and YAML for data serialization.** Trabalho de conclusão de curso (Bacharelado em Ciência da Computação). School of Computer Science and Engineering, Royal Institute of Technology, 2011. Disponível em: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=636a2b04d98c0af8e9d6f59148352dd63af4f0c1. Acesso em: 25/04/2023.

FREITAS, Allan E. S. **Software Livre e Código Aberto.** In: Relatório de Tendências em Tecnologia da Informação e Comunicação. SECTI, 2006. p. 39-48. Disponível em: https://www.researchgate.net/publication/274375291_SOFTWARE_LIVRE_E_CODIGO_ABERTO. Acesso em: 24 abr. 2023.

HODGDON, Jennifer. Drupal, © 2000-2023. **1.5 Concept: Types of Data**. 2023. Disponível em: https://www.drupal.org/docs/user_guide/en/understanding-data.html. Acesso em: 27 abr. 2023.

_____; DUNHAM, Grant. Drupal, © 2000-2023. **2.3. Concept: Content Entities and Fields.** 2023. Disponível em: https://www.drupal.org/docs/user_guide/en/planning-data-types.html. Acesso em: 27 abr. 2023.

HUFFSTEDTLER, Rob. Drupal, © 2000-2023. **Extending Drupal.** 2023. Disponível em: https://www.drupal.org/docs/extending-drupal. Acesso em: 25 abr. 2023.

MANOVICH, Lev. **Remixability and Modularity.** manovich, 2005. Disponível em: http://manovich.net/content/04-projects/046-remixability-and-modularity/43_article_2005.pdf. Acesso em: 25/04/2023.

MAUTHE, Andreas; THOMAS, Peter. **Professional Content Management Systems: Handling Digital Media Assets**. 2004. Disponível em: http://ndl.ethernet.edu.et/bitstream/123456789/12037/1/5.pdf.pdf. Acesso em: 24 abr. 2023.

MORVILLE, Peter. **Information Architecture on the World Wide Web.** O'Reilly Media, ed. 1. 1998. Disponível em: https://skat.ihmc.us/rid=1KR7TNX24-19KZ6GL-5SVD/O'Reilly%20-%20Information%20Architecture%20For%20The%20World%20Wide%20Web.pdf. Acesso em 25 abr. 2023.

PHP. **O que é o PHP?** 2023. Disponível em: https://www.php.net/manual/pt_BR/intro-whatis.php. Acesso em: 25 abr. 2023.

REHMAN, Rousif et al. **Analysis of Requirement Engineering Processes, Tools/Techniques and Methodologies.** In: I.J. Information Technology and Computer Science, vol 3, Fevereiro de 2013, p.40-48. Disponível em: https://www.mecs-press.org/ijitcs/ijitcs-v5-n3/IJITCS-V5-N3-5.pdf. Acesso em 27/04/2023.

RYAN, Mike et al. Drupal, © 2000-2023. **Modules: Migrate.** 10 mar. 2022. Disponível em: https://www.drupal.org/project/migrate. Acesso em: 25/04/2023.

_____; HEDDING, Lucas; DOROSHENKO, Ivan. Drupal, © 2000-2023. **Modules: Migrate Plus.** 2023a. Disponível em: https://www.drupal.org/project/migrate_plus. Acesso em: 19 maio 2023.

_____; _____; _____. Drupal, © 2000-2023. **Modules: Migrate Plus.** 2023b. Disponível em: https://www.drupal.org/project/migrate_plus. Acesso em: 19 maio 2023.

SALVADORI, Ivan Luiz. **Desenvolvimento de Web APIs RESTful Semânticas Baseadas em JSON.** Dissertação (Mestrado em Ciência da Computação) — Programa de Pós-Graduação em Ciência da Computação, Departamento de Informática e Estatística, Universidade Federal de Santa Catarina (UFSC). Florianópolis, p. 158. 2015. Disponível em: https://repositorio.ufsc.br/bitstream/handle/123456789/132469/333102.pdf. Acesso em: 17 maio 2023.

SHINDELAR, Joe. Drupal, © 2000-2023. **Creating Modules: Understanding Hooks.** 2022. Disponível em: https://www.drupal.org/docs/creating-modules/understanding-hooks. Acesso em: 27 abr. 2023.

SIPILÄ, Markus et al. Drupal, © 2000-2023. **Migrate process plugins.** 2018. Disponível em: https://www.drupal.org/docs/8/api/migrate-api/migrate-process-plugins. Acesso em: 02 maio 2023.

TIBONI, Antonio. **Software livre como política de governo**. Trabalho de conclusão

(Especialização em Gestão Pública), Escola de Administração, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2014. p. 67. Disponível em: http://www.lume.ufrgs.br/handle/10183/127438. Acesso em: 28 abr. 2023.