Carta de Serviço: Integração e Interoperabilidade para Eficiência na Gestão Pública

Marcelo S. Santos[1], **Karen Cristina Braga** [1], **Pedro A. S. Borges**[2], **Rodrigo Brum Westphalen**[3]

[1]Universidade Lasalle (Unilassale)
Av. Victor Barreto, 2288, Centro - Canoas RS, 92010-000.

[2]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
CEP 91.501-970 – Porto Alegre – RS – Brazil

[3]Universidade FeeVale – Av. Dr. Maurício Cardoso, 510
– Bairro Hamburgo Velho, Novo Hamburgo – RS – CEP 93510-235

marcelo.santos@unilassale.edu.br, kbraga@novohamburgo.rs.gov.br,

pasborges79@gmail.com, rodrigobrum@novohamburgo.rs.gov.br

Abstract. *This article discusses systems integration and process automation in public management, focusing on the use of Drupal as a solution. Drupal's flexibility in building websites through modules and manipulating data in formats such as YAML and JSON stands out. Its simplified migrations and integration between modularity and content management make it an effective option for modernizing and streamlining processes in public administration. As a result, an integrated and automated solution was developed for publishing services and related documents, providing efficiency and agility.*

Resumo. *Este artigo discute a integração de sistemas e a automação de processos na gestão pública, com foco na utilização do Drupal como solução. Destaca-se a flexibilidade do Drupal na construção de sites por meio de módulos e na manipulação de dados em formatos como YAML e JSON. Suas migrações simplificadas e integração entre modularidade e gerenciamento de conteúdo o tornam uma opção eficaz para modernizar e agilizar processos na administração pública. Como resultado, foi desenvolvida uma solução integrada e automatizada para a publicação de serviços e documentos relacionados, proporcionando eficiência e agilidade.*

1. Introdução

Na gestão pública contemporânea, a interoperabilidade entre sistemas desempenha um papel crucial na eficiência dos processos, facilitando a integração de diferentes ferramentas e promovendo uma administração mais ágil e transparente. Reconhecendo essa importância, a Administração Pública Municipal de Novo Hamburgo iniciou um projeto para modernizar a divulgação dos serviços oferecidos, integrando uma ferramenta interna desenvolvida para gestão da Carta de Serviços, conforme definida pela Lei de Proteção e Defesa dos Usuários de Serviços Públicos, nº 13.460 de 2017 [BRASIL 2017], com o Portal Institucional.

Essa pesquisa resultou na implementação de um sistema automático de criação e atualização de conteúdo no website Drupal, com o objetivo de reduzir o trabalho manual,

melhorar a eficiência na atualização de informações e fortalecer o conhecimento técnico institucional. O artigo aborda conceitos fundamentais e descreve a solução tecnológica desenvolvida, oferecendo uma análise descritiva do projeto que pode servir como modelo inicial para outras instituições interessadas em integrar sistemas internos com portais Drupal.

2. Contextualização

Anteriormente, a publicação da Carta de Serviços era realizada manualmente em diversos locais, incluindo o Portal Institucional, a Intranet (filtrando apenas os serviços voltados para o servidor público) e o sistema da Ouvidoria (com uma versão resumida para facilitar o atendimento ao cidadão). Esse processo era conduzido por um pequeno grupo de servidores, responsáveis por analisar os processos e fluxos de operação, e redigir os textos em linguagem padronizada e simples, visando proporcionar a melhor experiência possível aos usuários. Até o momento, mais de 300 serviços diversos foram registrados e disponibilizados para a comunidade.

Até o ano de 2023, toda a edição era feita manualmente, utilizando editores de texto e planilhas eletrônicas para controle de numeração e edição. No entanto, esse processo moroso foi substituído por um sistema desenvolvido pelas equipes de Tecnologia da Informação da Instituição, denominado OCTO. O OCTO gerencia todo o procedimento, oferecendo um ambiente para edição dos textos, mapeamento dos processos, integração, categorização e controle de documentos relacionados, como modelos de formulários. Além disso, o sistema permite identificar e tratar dados sensíveis ou que necessitam de consentimento, segundo a Lei Geral de Proteção de Dados. Isso trouxe agilidade à gestão, unificando-a em um único local.

Apesar disso, o desafio de atualização e publicação nos diversos locais persistia. Uma vez prontos no sistema, os serviços precisavam ser copiados manualmente para cada portal, uma prática que consumia tempo e aumentava o risco de erros.

Com exceção do sistema utilizado pela ouvidoria, todos os outros portais onde os serviços eram disponibilizados utilizavam uma framework de gerenciamento de conteúdo denominada Drupal. Após uma análise minuciosa dessa ferramenta, foi desenvolvido um módulo personalizado para comunicação de dados entre os diversos sistemas envolvidos, no formato JSON. Além disso, foi implementado um serviço web (*webservice*) para buscar os dados via HTTP e adaptado o sistema OCTO para gerenciar o momento em que essas comunicações seriam realizadas.

Essa pesquisa culminou na implementação de um sistema automático de criação e atualização de conteúdo no website Drupal. A partir do próprio sistema OCTO, o operador pode publicar ou despublicar cada serviço durante uma atualização, além de direcioná-los para os portais apropriados. Esse avanço foi realizado com o objetivo de reduzir a carga de trabalho manual, melhorar a eficiência na atualização de informações e fortalecer o conhecimento técnico institucional.

3. Fundamentação teórica

Nesta seção, serão abordados diversos conceitos, tais como conteúdo e metadados, modularidade e migrações de dados e *Webservices*.

3.1. Conteúdo e metadados

Mauthe e Thomas [MAUTHE and Thomas 2004] discutem o conceito de conteúdo, destacando sua natureza multifacetada como informações acessíveis e passíveis de serem produzidas, alteradas, transmitidas, consumidas e trocadas. Na indústria midiática, esse conteúdo é dividido em ”essência”e ”metadados”, sendo a essência a informação primária composta por texto, imagem, vídeo e áudio, enquanto os metadados descrevem e auxiliam no processamento desses dados, sendo vitais para sua organização, armazenamento, recuperação e exibição.

O CMS, segundo [MAUTHE and Thomas 2004], são sistemas projetados para gerir tanto a essência quanto os metadados dos conteúdos, oferecendo ferramentas para criação, modificação e fornecimento automático de metadados. A qualidade desses metadados é fundamental para ampliar o acesso aos conteúdos, especialmente por meio de sistemas de busca eficazes. A preservação do conteúdo arquivado para reutilização é uma meta destacada pelos autores, enfatizando a importância do arquivamento digital e da migração contínua como parte intrínseca do design dos CMSs para garantir a perpetuidade dos conteúdos na era digital.

3.2. Gerenciadores de conteúdo e migrações no Drupal

Os *Content Management Systems* (CMS) são softwares utilizados para a administração de informações em websites. Como exemplo de aplicação na gestão pública, o governo federal há bastante tempo adota CMS como Joomla, Plone e Drupal. Esta estratégia tem por objetivo promover a independência de fornecedores, liberdade de escolha tecnológica e incentivo à inovação [TIBONI 2014], além da agilidade na entrega dos produtos digitais.

Dentre essas opções, o Drupal [DRUPAL 023ab] destaca-se por sua flexibilidade na construção de sites através de módulos. Desenvolvido em PHP [PHP 2023], uma linguagem de código aberto amplamente utilizada para a construção de páginas web no lado do servidor, o Drupal demonstra sua modularidade na manipulação de dados de configuração em arquivos YAML e JSON, formatos essenciais para a serialização de dados e a transmissão eficiente de informações.

As migrações no Drupal são simplificadas pelo módulo ”*Migrate*”do core, permitindo a extensão das funcionalidades através de módulos contribuídos pela comunidade. Essas migrações podem abranger diferentes tipos de arquivos, como planilhas, XML e JSON, contando com suporte adicional oferecido pelos módulos ”*Migrate Plus*”e ”*Migrate Tools*”.

Além disso, no Drupal, todo conteúdo é tratado como um Node, sendo gerenciado através de configurações armazenadas em arquivos YAML e PHP vinculados a hooks específicos [DRUPAL 023aa], o que demonstra a integração entre modularidade e gerenciamento de conteúdo neste *framework*.

3.3. Webservice

Os serviços web (*webservices*) são meios de distribuir sistemas em diferentes locais, permitindo o compartilhamento de dados por meio do HTTP (*HyperText Transfer Protocol*) e do HTTPS (*Secure*). Ao contrário de documentos HTML e recursos de mídia destinados à visualização em navegadores, os *webservices* visam compartilhar dados estruturados

entre aplicações. Outra terminologia usada para se referir a serviços web que fornecem dados estruturados com JSON ou XML é Web API (*Web Application Programming Interface*), comumente associada a *webservices* que seguem os princípios arquiteturais RESTful [SALVADORI 2015].

Segundo a Lei Federal nº 13.460/2017, todos os Poderes (Executivo, Legislativo e Judiciário) em todas as esferas de Governo (Federal, Estadual e Municipal) devem divulgar os serviços oferecidos, indicando os órgãos responsáveis e a autoridade administrativa vinculada. A Carta de Serviços ao Usuário, conforme definida no Art. 7º, tem o objetivo de informar sobre acesso, requisitos, prazos, padrões de qualidade, compromissos do órgão público e canais de manifestação do usuário. Cada Poder deve regulamentar sua própria Carta de Serviços, mantendo-a atualizada e divulgada em um site institucional (BRASIL, 2017).

4. Metodologia

Este artigo descreve uma pesquisa experimental com foco na automação do processo de criação de conteúdo no portal institucional. O experimento incluiu a análise dos requisitos do sistema, o estudo dos *softwares* relacionados ao Drupal, o desenvolvimento de uma solução técnica e a documentação correspondente, que serão descritos nesta seção.

4.1. Análise de requisitos

Durante a fase de análise de requisitos, foram realizadas reuniões entre as duas equipes para discutir em conjunto as necessidades, limitações e possibilidades das tecnologias existentes, conforme descrito por [REHMAN and et al 2013]. Duas reuniões foram realizadas, intercaladas com estudos das ferramentas, para iniciar o desenvolvimento da solução técnica. Os requisitos levantados incluem a necessidade dos dados de entrada serem independentes da plataforma Drupal, a leitura e processamento periódicos e programáticos dos dados pelo Drupal (sem intervenção humana), a garantia de que os dados de saída sejam reconhecidos e gerenciáveis pela interface do Drupal, a modificação de um dado de entrada refletindo na modificação do conteúdo associado, a capacidade de despublicar um conteúdo a partir dos dados de entrada e a exigência de documentação detalhada para consultas futuras.

4.2. Dependências e configurações iniciais

Na implementação do Drupal 9.2, os módulos "*Migrate Tools*" (v6.0) e "*Migrate Plus*", ambos na v6.0, foram instalados manualmente, baixando os arquivos do site do Drupal e inseridos na pasta "/modules/contrib". O módulo "*Core Migrate*" foi instalado simultaneamente através do painel administrativo do site. Um "*Migration Group*" foi automaticamente criado pelo módulo "Migrando JSON", cujas configurações estão detalhadas no arquivo: "servicos_migration_json.yml". Alternativamente, foi possível criar um novo grupo acessando a página de administração em "/admin/structure/migrate/" e copiar o nome de máquina do grupo para a chave "migration_group" no arquivo de configuração da *Migration*.

Além disso, foi criado um usuário específico, chamado "integrador-octo", exclusivamente para a publicação de conteúdos da integração. O ID do usuário foi obtido acessando o perfil do mesmo, que por padrão está disponível em "/user/uid". Esse ID é

então incluído na configuração da *Migration*, no campo "uuid", para garantir a autoria correta dos conteúdos gerados de forma programática. Vale ressaltar que a estrutura de páginas e a criação de campos necessários para os conteúdos importados devem ser previamente configuradas no Content Type "Serviço"(Quadro 1),com o ID serviço, conforme representado no painel de administração do site.

LABEL	MACHINE NAME	TIPO DO CAMPO
Corpo	body	Texto (formatado, longo)
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

Figura 1. Campos do Content Type "Serviço". Fonte elaborado pelos autores.

As equipes trabalharam na migração dos dados do software de gestão interna da Carta de Serviços ao Drupal, estabelecendo o formato do arquivo JSON conforme mostrado no Figura 2.

4.3. Configuração da *Migration*

Ao instalar o módulo, a configuração modelo será automaticamente instalada junto. As configurações de *Migration* também podem ser importadas através do painel administrativo do Drupal, acessando a URL "/admin/config/development/configuration/single/import". Esta importação permite atualizar as configurações de uma *Migration* sem a necessidade de reinstalar o módulo. Vale ressaltar que a chave de API do *webservice* que a *Migration* consome foi omitida por questões de segurança.

O processo de documentação da solução incluiu a divisão do arquivo de configuração em partes e a inserção de comentários no arquivo "README.md". Nessa documentação, da chave "*uuid*" até "*label*", são especificados os metadados relacionados à migração. A seção source detalha informações sobre o arquivo de entrada, indicando a URL para acessar o JSON via HTTP. Já o "item_selector" define a chave raiz da lista de conteúdos que será lida pela *Migration*. No campo "*fields*", é determinado o caminho da chave no arquivo de entrada através do seletor, e seu valor é atribuído à variável "*name*".

A seção "*ids*" informa a chave única dos itens da lista. Em "*process*", são definidas as chaves do objeto de saída do Content, que receberão o resultado de operações gerenciadas por plugins fornecidos pelo *Migrate* e pelo *Migrate Plus* [SIPILÄ and et al 2018]. Por exemplo, o campo "field_tags" pode receber uma string de termos separados por vírgula, que são transformados em um *array*. Cada item desse *array* passa por um processo de busca nos termos de taxonomia existentes; caso não sejam encontrados, novos termos são gerados. O campo "uid" atribui a autoria do conteúdo a um usuário com base em seu "id". Por fim, em "*destination*", é especificado o plugin "entity:node", que define a estrutura dos dados de saída conforme as configurações previamente mencionadas.

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


Figura 2. Exemplo de Estrutura de Dados de Entrada (JSON). Fonte: elaborado pelos autores.

4.4. Configuração de módulo contrib

O módulo Migrando JSON, cujas funcionalidades estão detalhadas no arquivo "migrando_json.module" (Figura 3) e são vinculadas ao hook_cron, é definido no arquivo "migrando_json.info.yml" (Figura 4). As dependências, incluindo os arquivos de configuração de *Migrations*, são especificadas no arquivo "info.yml".


```
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
```


Figura 3. Arquivo migrando_json.info.yml (YAML). Fonte: elaborado pelos autores.

4.5. *Plugins* personalizados

Para exibir o conteúdo de cada Serviço conforme definido, foram desenvolvidos dois *plugins* personalizados durante o processamento dos dados na *Migration*: "contatos_para_html"(Figura x) e "documentos_para_html". Esses plugins são classes derivadas

```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_plugin_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.
        }
}
```

Figura 4. Arquivo migrando_json.module (PHP). Fonte: elaborado pelos autores.

de “ProcessPluginBase” e são mapeados através da annotation ”@MigrateProcessPlugin()”. Eles implementam o método transform, no qual um valor é recebido do pipeline de processamento da *Migration* e um valor é retornado para o pipeline. O plugin personalizado mencionado transforma múltiplos valores em uma única *string* de HTML.

```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);
                }
        }
```

Figura 5. Trecho do arquivo “contatosParaHtml.php”. Fonte: elaborado pelos autores.

4.6. Demonstração de conteúdo migrado

Para iniciar a *Migration*, o procedimento envolve a inserção do módulo ”Migrando JSON”na pasta “/sites/all/modules/contrib” e sua ativação através do painel de controle do administrador, conforme demonstrado na Figura 6.

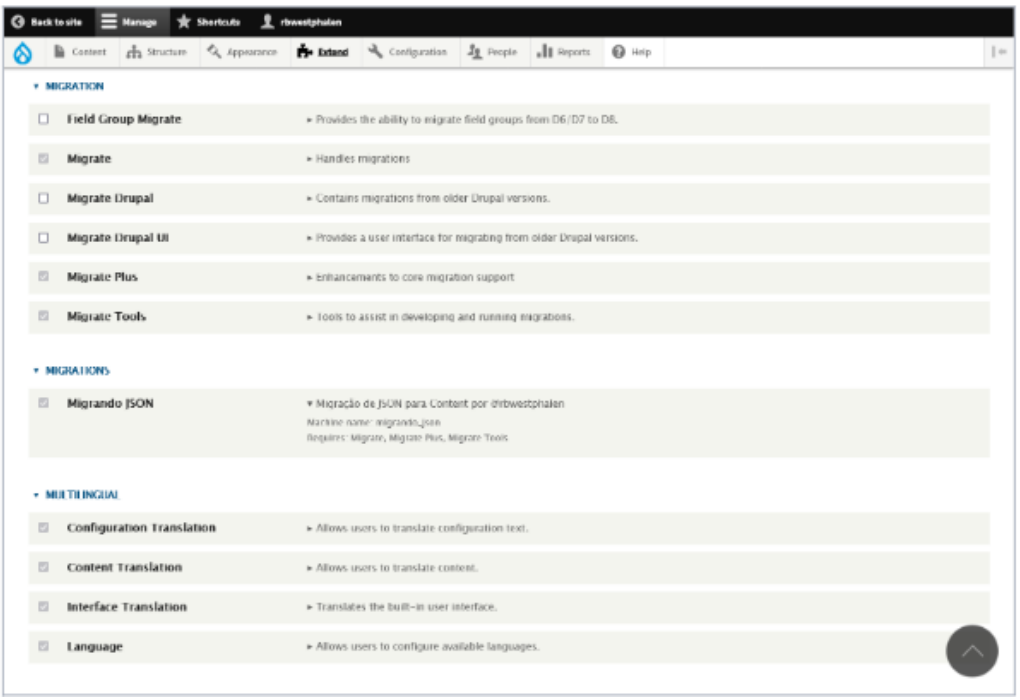

Figura 6. Captura de Tela da Instalação do Módulo Contrib. Fonte: elaborado pelos autores.

A *Migration* é identificada através do painel de administração do Drupal, sendo executada por ele. Então, a página de listagem de Contents (URL: /admin/content/node) é verificada para encontrar os itens migrados. A Figura 7 apresenta na lateral direita o produto do plugin custom "contatos_para_html".

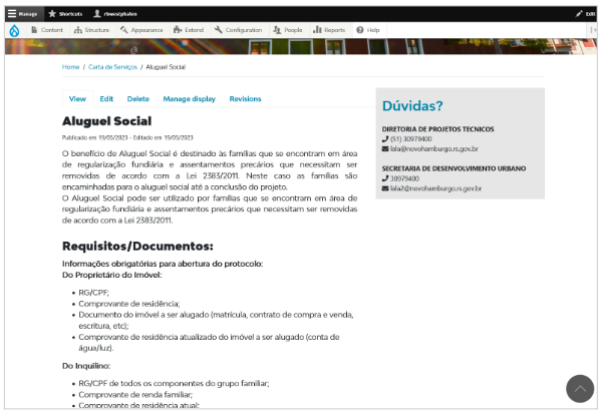

Figura 7. Captura de Tela de Página de Teste Criada pela Migração. Fonte: elaborado pelos autores.

5. Resultados

Foi desenvolvida uma solução que utiliza o sistema de migrações do Drupal para processar dados de um arquivo JSON com estrutura acordada entre as equipes das Diretorias. Para verificar periodicamente o arquivo em busca de modificações, foi empregado o "hook_cron", vinculando uma função à execução do Cron do servidor. Posteriormente,

um web service foi criado para fornecer o arquivo atualizado na mesma URL, adaptando a solução para buscar o arquivo via HTTPS.

O software, denominado “Migrando JSON”, opera como módulo contrib do Drupal, permitindo ajustes nos arquivos de configuração para atender a outras instituições e ser instalado por meio do painel administrativo. Após a consolidação da solução no servidor de produção, o código-fonte será disponibilizado para a comunidade de desenvolvedores, juntamente com a documentação detalhada no arquivo “README.md” via repositório online aberto. A descrição dos resultados foi organizada em subtítulos como “Análise de requisitos”, “Dependências e configurações iniciais”, “Dados de Entrada”, “Configuração da *Migration*”, “Configuração do Módulo Contrib” 6, “Plugins Personalizados” e “Demonstração de Conteúdo Migrado” 7.

6. Conclusão

A solução desenvolvida atendeu aos requisitos levantados ao definir o arquivo de entrada como JSON, vinculando a leitura e o processamento dos dados ao Cron do servidor, verificando a possibilidade de edição do conteúdo via painel administrativo, definindo UUIDs nos dados de entrada e o atributo ”track_changes”na *Migration*, recebendo e processando o atributo status nos dados de entrada, e documentando o módulo contrib em artigo externo e tutorial completo no arquivo ”README.md”, a ser disponibilizado no futuro. Este projeto representa um avanço no desenvolvimento de tecnologias de código-aberto para a Administração Pública, especialmente no contexto do software livre para sistemas de gestão de conteúdo na web (CMS). O mesmo está alinhado aos princípios da legalidade, eficiência e publicidade, conforme estabelecido no artigo 37 da Constituição brasileira de 1988 [BRASIL 1988].

No momento desta redação, a solução ainda não foi implementada no servidor de produção, pois é necessário incluir todos os serviços atualmente disponíveis no portal institucional no software de gestão interna da Carta de Serviços. Após essa etapa, os conteúdos atuais do site serão excluídos e a migração importará todos os serviços no formato mais atualizado. Assim que a integração for consolidada no servidor de produção, o módulo contribuirá para a comunidade de desenvolvedores com código aberto e licença GPL-3.

Referências

BRASIL (1988). Constituição da república federativa do brasil. `https://www.planalto.gov.br/ccivil_03/Constituicao/Constituicao.htm`. Online; Acesso em 02 Abr 2024.

BRASIL (2017). Lei n. 13.640, lei de proteção e defesa dos usuários de serviços públicos. `https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2017/lei/l13460.htm`. Online; Acesso em 02 Abr 2024.

DRUPAL (2023aa). Function hook$_cron$.. Online; Acesso em 02 Abr 2024.

DRUPAL (2023ab). Overview of drupal. `https://www.drupal.org/docs/understanding-drupal/overview-of-drupal`. Online; Acesso em 02 Abr 2024.

MAUTHE, A. and Thomas, P. (2004). *Professional Content Management Systems: Handling Digital Media Assets*. wiley, 1th edition.

PHP (2023). O que é o php? `https://www.php.net/manual/pt_BR/intro-whatis.php`. Online; Acesso em 02 Abr 2024.

REHMAN, R. and et al (2013). Analysis of requirement engineering processes, tools/techniques and methodologies. In PRESS, editor, *I.J. Information Technology and Computer Science*. PRESS.

SALVADORI, I. (2015). *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).

SIPILÄ, M. and et al (2018). Migrate process plugins. `https://www.drupal.org/docs/8/api/migrate-api/migrate-process-plugins`. Online; Acesso em 02 Abr 2024.

TIBONI, A. (2014). Software livre como política de governo. trabalho de conclusão (especialização em gestão pública), escola de administração, universidade federal do rio grande do sul. `http://www.lume.ufrgs.br/handle/10183/127438`. Online; Acesso em 02 Abr 2024.