

**MINISTÉRIO DA DEFESA  
EXÉRCITO BRASILEIRO  
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA  
INSTITUTO MILITAR DE ENGENHARIA  
CURSO DE GRADUAÇÃO EM ENGENHARIA CARTOGRÁFICA**

**ARTHUR COSTA CAVALCANTE  
MARCIO MATHEUS DOS SANTOS  
PEDRO HENRIQUE DIAS KOVALCZUK**

**DESENVOLVIMENTO DE UMA APLICAÇÃO WEB PARA A CONSTRUÇÃO  
DE METADADOS DOS PRODUTOS CARTOGRÁFICOS DA DSG**

**RIO DE JANEIRO  
2024**

**ARTHUR COSTA CAVALCANTE  
MARCIO MATHEUS DOS SANTOS  
PEDRO HENRIQUE DIAS KOVALCZUK**

**Desenvolvimento de uma aplicação web para a construção  
de metadados dos produtos cartográficos da DSG**

Projeto de Final de Curso apresentado ao Curso de Graduação em Engenharia Cartográfica do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Bacharel em Engenharia Cartográfica.

Orientador(es): TC Marcos de Meneses Rocha e Maj Maurício Carvalho Mathias de Paulo.

Aprovada em 10 de Outubro de 2024, pela seguinte banca examinadora: TC Ivanildo Barbosa e Maj Felipe Ferrari.

---

**Maj Maurício Carvalho Mathias de Paulo - D.Sc do IME**

---

**Cel Marcos de Meneses Rocha - D.Sc do IME**

---

**Maj Felipe Ferrari - D.E do IME**

---

**TC Ivanildo Barbosa - D.Sc do IME**

Rio de Janeiro

2024

## AGRADECIMENTOS

Primeiramente, gostaríamos de agradecer às nossas famílias, que sempre estiveram ao nosso lado, oferecendo amor, incentivo e apoio incondicional. Aos nossos pais, por serem fontes inesgotáveis de inspiração e por nos ensinarem a importância da perseverança. Vocês são nossa base e nossa força. Ao nossos orientadores Maj Maurício e Cel Marcos, pela orientação meticulosa, paciência e conhecimento compartilhado ao longo deste processo. Seus conselhos e críticas construtivas foram fundamentais para moldar este trabalho. Aos amigos que estiveram ao nosso lado, nos motivaram e entenderam nossa ausência em tantos momentos importantes. Seus sorrisos e palavras de encorajamento foram o combustível que nos impulsionou. Agradecemos também a todos os professores e funcionários da instituição, que contribuíram indiretamente para a nossa formação acadêmica. Por último, mas não menos importante, agradecemos a todos aqueles que, de uma forma ou de outra, nos apoiaram e encorajaram durante este percurso. Cada gesto de carinho, cada palavra amiga e cada ato de gentileza foram fundamentais para nossa motivação e bem-estar.

Agradecimentos especiais são direcionados aos alunos da Engenharia Cartográfica 2024 que compartilharam conosco muitas noites de estudo, debates estimulantes e momentos de desafio e que contribuíram e que sempre irão contribuir para a manutenção do conhecimento.

## RESUMO

Este trabalho desenvolveu uma nova aplicação web para otimizar o processo de cadastramento e validação de metadados dos produtos cartográficos da Diretoria de Serviço Geográfico do Exército Brasileiro (DSG), superando as deficiências técnicas do sistema de formulário atual do BDGEx. A nova solução adota uma arquitetura modular e desacoplada, com comunicação via APIs entre cliente e servidor, o que facilita a integração com sistemas externos e flexibiliza o fluxo de preenchimento de metadados. Entre as melhorias implementadas, destaca-se a extração automática de dados essenciais dos produtos geoespaciais, como INOM, MI e extensão espacial, eliminando erros manuais recorrentes no sistema anterior. Além disso, a aplicação permite configurar dinamicamente os campos de metadados para cada tipo de produto, garantindo conformidade com as normas da ET-PCDG e NT-CMet. O sistema também valida automaticamente os dados submetidos com base em regras de negócios predefinidas, centralizando a administração de usuários e produtos. Em comparação com o formulário anterior, que era acoplado rigidamente ao front-end e dificultava a automação e integração com outros sistemas, a nova versão é mais flexível e eficiente, suportando operações descentralizadas e simultâneas. A complexidade de múltiplas tabelas foi substituída por uma estrutura simplificada, em que cada arquivo de metadado descreve diretamente um produto geoespacial. Isso facilita a interoperabilidade com o servidor CSW do Open Geospatial Consortium (OGC), melhorando a escalabilidade e eficiência na produção cartográfica da DSG.

**Palavras-chave:** BDGEx. catálogo de metadados. geoinformação.

## ABSTRACT

This work developed a new web application to optimize the process of registering and validating the metadata of the cartographic products of the Brazilian Army's Geographic Service Directorate (DSG), overcoming the technical deficiencies of the current BDGEx form system. The new solution adopts a modular and decoupled architecture, with communication via APIs between client and server, which facilitates integration with external systems and makes the flow of filling in metadata more flexible. Among the improvements implemented is the automatic extraction of essential data from geospatial products, such as INOM, MI and spatial extension, eliminating recurring manual errors in the previous system. In addition, the application makes it possible to dynamically configure the metadata fields for each type of product, ensuring compliance with the ET-PCDG and NT-CMet standards. The system also automatically validates submitted data based on predefined business rules, centralizing user and product administration. Compared to the previous form, which was rigidly coupled to the front-end and made automation and integration with other systems difficult, the new version is more flexible and efficient, supporting decentralized and simultaneous operations. The complexity of multiple tables has been replaced by a simplified structure in which each metadata file directly describes a geospatial product. This facilitates interoperability with the Open Geospatial Consortium (OGC) CSW server, improving scalability and efficiency in DSG's cartographic production.

**Keywords:** BDGEx. metadata catalog. geoinformation.

## LISTA DE ABREVIATURAS E SIGLAS

CSW	Catalogue Service for the Web
HTTP	Hypertext Transfer Protocol
OGC	Open Geospatial Consortium
WCS	Web Coverage Service
WFS	Web Feature Service
WMS	Web Map Service
WMTS	Web Map Tile Service
MGB	Perfil de Metadados Geoespaciais do Brasil
REST	Representational State Transfer
JSON	JavaScript Object Notation

# SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO</b>	<b>8</b>
1.1	JUSTIFICATIVA DO PROJETO	8
1.2	OBJETIVO DO PROJETO	10
<b>2</b>	<b>FUNDAMENTAÇÃO TEÓRICA</b>	<b>11</b>
2.1	ESTRUTURA DE SOFTWARE PARA APLICATIVOS WEB	11
2.2	PADRÕES DE METADADOS DE GEOINFORMAÇÃO	14
2.2.1	ISO 19115:2003	14
2.2.2	ISO 19139:2007 E ESQUEMAS XML	14
2.2.3	PERFIL DE METADADOS GEOESPACIAIS DO BRASIL	15
2.2.4	ET-PCDG	15
2.3	PROCESSO DE CADASTRO DE METADADOS DO BDGEX	16
2.4	CSW E OGC API RECORDS	17
2.4.1	CSW	18
2.4.2	OGC API RECORDS	18
<b>3</b>	<b>MATERIAIS E MÉTODOS</b>	<b>20</b>
3.1	GESTÃO DO PROJETO	20
3.1.1	PARTES INTERESSADAS	20
3.1.2	REQUISITOS	20
3.1.3	REQUISITOS FUNCIONAIS	20
3.1.4	REQUISITOS NÃO FUNCIONAIS	21
3.2	ESCOPO	22
3.2.1	ESTRUTURA ANALÍTICA DO PROJETO (EAP)	22
3.2.2	DICIONÁRIO DA EAP	22
3.2.3	EXCLUSÕES DO ESCOPO	24
3.2.4	ENTREGAS	24
3.3	PROCESSO DE DESENVOLVIMENTO DO SOFTWARE	25
3.3.1	ESTRUTURA DE GESTÃO COM KANBAN	25
3.3.2	EXEMPLO DE IMPLEMENTAÇÃO	26
3.4	VISÃO GERAL DO FORMULÁRIO DE CADASTRO DE METADADOS	26
3.5	COMPONENTES	28
3.5.1	<i>FRONT-END</i>	29
3.5.1.1	COMPONENTES DO FORMULÁRIO DE METADADOS	29
3.5.1.2	GERENCIAMENTO DAS CHAMADAS AO <i>BACK-END</i>	29
3.5.2	<i>BACK-END</i>	30

3.5.2.1	APIS . . . . .	31
3.5.2.1.1	API DE UPLOAD DE GEOPRODUTO . . . . .	31
3.5.2.1.2	API DE CONSTRUÇÃO DE XML . . . . .	32
3.5.2.1.3	API DE VALIDAÇÃO DE XML . . . . .	32
3.5.2.1.4	API DE INTERFACE DO CADASTRO GERAL . . . . .	32
3.5.2.2	SISTEMA DE ADMINISTRAÇÃO . . . . .	33
3.5.2.3	COMPONENTE DE SERVIDOR DE METADADOS (CSW/API RECORDS) . .	33
3.5.3	ALGORITMO DE CÁLCULO DE INOM E MI . . . . .	33
<b>4</b>	<b>RESULTADOS E DISCUSSÃO . . . . .</b>	<b>36</b>
4.1	FLUXO PRINCIPAL DE CADASTRO DE METADADOS . . . . .	36
4.2	FLUXO SECUNDÁRIO DE CADASTRO DE METADADOS . . . . .	42
4.3	APIS DA APLICAÇÃO . . . . .	44
4.3.1	API DE UPLOAD . . . . .	44
4.3.2	API DE INTERFACE DO CADASTRO GERAL . . . . .	46
4.4	VALIDAÇÃO DE METADADOS . . . . .	48
4.5	SISTEMA DE ADMINISTRAÇÃO . . . . .	50
4.5.1	PUBLICAÇÃO NO SERVIDOR CSW . . . . .	53
<b>5</b>	<b>CONCLUSÃO . . . . .</b>	<b>55</b>
5.1	CONSIDERAÇÕES FINAIS . . . . .	55
5.2	TRABALHOS FUTUROS . . . . .	56
	<b>REFERÊNCIAS . . . . .</b>	<b>57</b>
	<b>ANEXO A – DICIONÁRIO DE DADOS . . . . .</b>	<b>58</b>

# 1 INTRODUÇÃO

## 1.1 Justificativa do projeto

A crescente demanda por dados geoespaciais e a expansão do uso de tecnologias geográficas têm impulsionado a criação de ferramentas que facilitam a organização, acesso e distribuição de informações cartográficas.

Nesse contexto, a Diretoria de Serviço Geográfico do Exército Brasileiro (DSG), um órgão de apoio técnico-normativo do Departamento de Ciência e Tecnologia (DCT), é responsável por coordenar as atividades cartográficas relativas à elaboração e ao suprimento de produtos geoespaciais, bem como gerenciar os convênios decorrentes com outros órgãos da administração pública nacional, sendo também responsável pelo desenvolvimento e manutenção do Banco de Dados Geográfico do Exército (BDGEx).

O BDGEx é um sistema que unifica os armazenamentos dos produtos das suas diversas unidades de produção, que são disponibilizados via Internet ou EBnet e representa a materialização da disponibilização dos dados geoespaciais por parte da DSG (XAVIER; MEYER; LUNARDI, 2014). Uma parte crítica da disponibilização dos produtos é a criação e gestão dos metadados geoespaciais, que descrevem de forma abrangente cada produto, facilitando sua catalogação, busca e utilização por diversos usuários e sistemas. Eles fornecem informações detalhadas sobre a origem, propósito, escala, precisão e outras características dos produtos cartográficos.

O sistema de metadados do BDGEx busca atender as Especificações Técnicas para os Produtos de Conjuntos de Dados Geoespaciais (ET-PCDG), padrões de serviços de dados do *Open Geospatial Consortium* (OGC) e ao servidor *Catalog Services for the Web* (CSW) (NEBERT; VOGES; BIGAGLI, 2016), responsável pela comunicação do catálogo de metadados em padrões abertos.

No caso do BDGEx, a necessidade de um sistema eficiente para o registro e manutenção desses metadados levou ao desenvolvimento do SIGWeb Fonte e SIGWeb Mediador. O SIGWeb Fonte permite catalogar produtos geoespaciais, que são posteriormente publicados no SIGWeb Mediador. Enquanto apenas operadores da DSG cadastram produtos no SIGWeb Fonte, o SIGWeb Mediador também é acessado por usuários externos para localizar e acessar as informações geográficas necessárias. No entanto, o processo atual de cadastro de metadados está associado a uma arquitetura de software defasada (Figura 1), de difícil manutenção e com pouca escalabilidade de dados, pelo uso da linguagem *PHP* em conjunto com o *jQuery* em versões de mais de dez anos atrás.

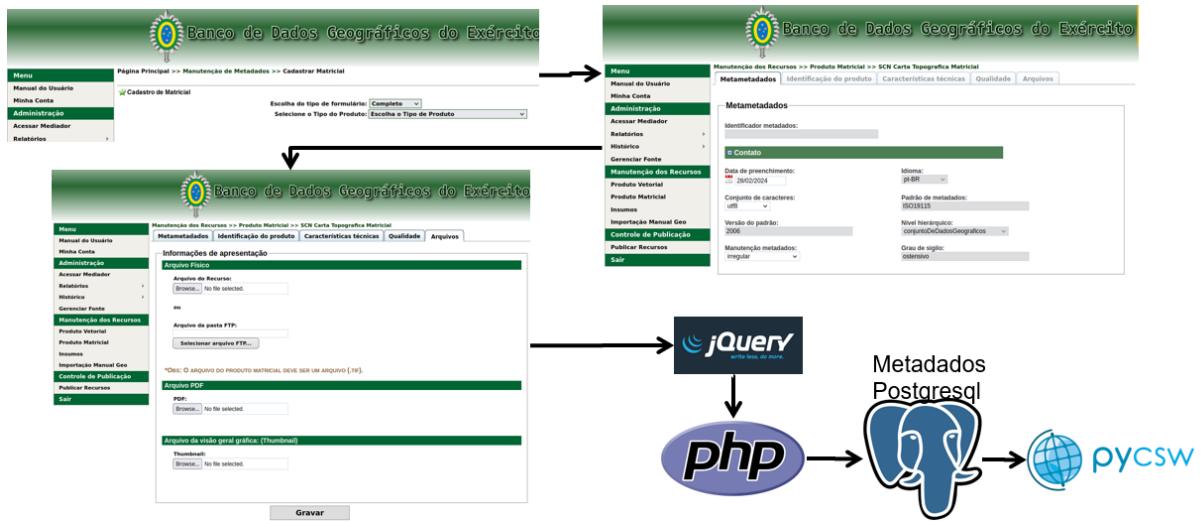


Figura 1 – Fluxo atual do formulário de Cadastro de Metadados do SIGWeb Fonte.

O SIGWeb Fonte foi modificado ao longo dos anos, com regras de preenchimento e validação dos metadados que foram inseridas conforme as normas e os padrões iam sendo documentados. Algumas das regras foram implementadas em PHP, algumas na interface em *HTML* com *jQuery*, algumas no importador de dados geoespaciais, que armazena os produtos no banco de dados, e a construção do arquivos de metadados é feita posteriormente no PostgreSQL. Desta forma, carregar um produto no BDGEx atualmente só pode ser feito seguindo todo o fluxo do formulário de Cadastro de Metadados, pela interface web, limitando a implementação de sistemas de carregamento em lote ou por sistemas externos.

Neste trabalho, foi desenvolvida uma nova aplicação web para a construção de metadados dos produtos cartográficos da DSG, com o objetivo de otimizar o processo de cadastro, reduzir a margem de erro dos operadores e melhorar a acessibilidade das informações. A aplicação web foi projetada para fornecer uma interface intuitiva e funcionalidades avançadas que permitam a usuários de diferentes níveis de habilidade interagir de maneira eficiente com o sistema de metadados geoespaciais.

Ao longo deste trabalho, será explorada a importância dos metadados para a gestão de produtos cartográficos, analisaremos os requisitos técnicos para a construção da aplicação web que sacramentará uma reformulação no sistema de cadastro de metadados dentro da estrutura do BDGEx e será descrita a metodologia utilizada para seu desenvolvimento. A conclusão abordará os resultados esperados, incluindo benefícios para a eficiência do BDGEx e para os usuários e operadores de dados geoespaciais.

## 1.2 Objetivo do projeto

O objetivo deste trabalho é o desenvolvimento de uma nova aplicação web para o cadastro de metadados geoespaciais no BDGEx dos produtos cartográficos da Diretoria de Serviço Geográfico do Exército Brasileiro (DSG).

## 2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção serão abordados os conceitos que são utilizados no contexto de criação de um sistema de metadados, será desenvolvido a lógica por trás dos componentes do software do sistema, o *framework* do processo de cadastro de metadados atual do BDGEx que servirá de base para a aplicação e por fim os diferentes padrões que serão incorporados em diferentes níveis das escolhas que foram tomadas ao longo do desenvolvimento da solução.

### 2.1 Estrutura de Software para Aplicativos Web

A estrutura de um *software* existe com o intuito de planejar a construção de sistemas robustos e escaláveis. No contexto da criação de um formulário de cadastro de informações, introduz-se um conceito de desacoplamento entre a estrutura que interage diretamente com o usuário por meio de um navegador (*front-end*) e a estrutura que gere o fluxo e armazenamento de dados (*back-end* e banco de dados). Para isso, o planejamento de *software* tem o papel de desenhar como o sistema irá funcionar com a finalidade de garantir a interoperabilidade entre os componentes e a eficácia do sistema em cumprir sua função a todo momento.

O React é uma das bibliotecas JavaScript mais populares para a criação de interfaces de usuário interativas, ou seja, o *front-end*. Sua abordagem baseada em componentes facilita a reutilização de código e a construção de interfaces responsivas (META, 2024).

A existência de um *front-end* oferece vantagens como:

- **Componentização:** Permite a criação de componentes isolados, que podem ser reutilizados em diferentes partes do aplicativo, promovendo a consistência e a manutenção do código.
- **Fluxo de Dados Unidirecional:** Adota uma abordagem de fluxo de dados unidirecional, tornando a lógica do *front-end* mais previsível e facilitando a depuração de erros.
- **Ecossistema Amplo:** A vasta gama de bibliotecas e ferramentas disponíveis para React, como *Redux* para gerenciamento de estado e *React Router* para roteamento, oferece recursos para a criação de aplicativos complexos.

O Django por sua vez é um framework em Python, que se baseia na simplicidade e robustez para o desenvolvimento de diversas soluções. Ele possui características que

podem ser utilizadas para o *front-end*, porém utilizando ele somente no *back-end*, o sistema desenvolvido se beneficia de (*Django Software Foundation, 2019*):

- **Estrutura MVC:** Django segue o padrão *Model-View-Controller* (MVC), o que promove uma separação clara entre a lógica de negócios, a apresentação e o controle do sistema.
- **Interface Administrativa:** Django possui uma interface administrativa (`django-admin`) que permite o gerenciamento intuitivo de dados, facilitando a administração do sistema.
- **ORM Robusto:** O Object-Relational Mapping (ORM) do Django simplifica a interação com bancos de dados relacionais, tornando mais fácil trabalhar com os dados persistentes do aplicativo.
- **Segurança e Escalabilidade:** Django inclui mecanismos integrados de segurança e é conhecido por sua capacidade de escalar para grandes volumes de dados e usuários.

A comunicação entre o *front-end* em React e o *back-end* em Django pode ser realizada por meio de APIs RESTful, utilizando especificamente o *Django REST Framework*.

Interfaces de programação de aplicativos (APIs) que seguem os princípios do estilo arquitetônico Representational State Transfer (REST) são denominadas APIs RESTful. Essas interfaces facilitam a comunicação entre sistemas ou aplicações, utilizando verbos HTTP padronizados (GET, POST, PUT, DELETE, etc.) para executar operações sobre recursos.

O objetivo primordial das APIs RESTful consiste em oferecer um mecanismo simples e uniforme para acessar e manipular recursos (dados ou serviços) de maneira independente de plataforma, promovendo interoperabilidade e escalabilidade. A abordagem REST fundamenta-se em conceitos-chave, como a identificação de recursos através de Uniform Resource Locators (URLs), a natureza sem estado das operações, e a transferência de representações desses recursos em formatos padronizados, tipicamente JSON ou XML.

A característica sem estado das APIs RESTful implica que cada requisição contém todas as informações necessárias para ser processada, sem depender de contexto ou estado armazenado no servidor entre as requisições. Essa abordagem contribui para a escalabilidade, pois permite que múltiplos servidores atendam às requisições de forma independente, sem a necessidade de compartilhar estado (FIELDING, 2000).

Dentro desse contexto, o *Django REST Framework* é um *framework* de desenvolvimento *web*, construído utilizando a componentização do *Django*, que permite a criação de APIs REST abordando uma programação orientada a componentes para a construção de

APIs, tornando tanto o processo de desenvolvimento mais estruturado e eficiente, quanto a comunicação entre o *front-end* e *back-end* (VAINIKKA, 2018).

Por fim, o banco de dados é parte essencial da arquitetura de uma aplicação, especialmente para um aplicativo de cadastro de informações, pois se comunica diretamente com o *back-end* e armazenando as informações sinalizadas por ele. O Django suporta uma variedade de bancos de dados relacionais, como PostgreSQL e MySQL, permitindo que o sistema escolha a melhor solução para suas necessidades. A arquitetura do banco de dados deve ser projetada para garantir a normalização dos dados, a consistência e a escalabilidade, aspectos essenciais para o armazenamento seguro de metadados.

A arquitetura da Figura 2 exemplifica os papéis principais de cada um dos componentes operando em conjunto em uma aplicação.

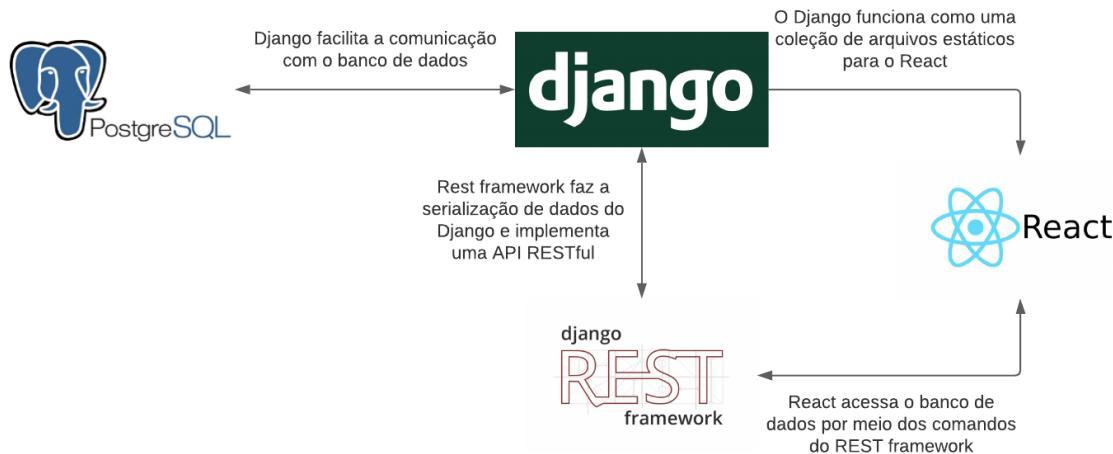


Figura 2 – Exemplo de Arquitetura utilizando as ferramentas descritas. (Github Violeta, 2020)

## 2.2 Padrões de metadados de geoinformação

Os metadados são o conjunto de informações descritivas sobre os dados a que se referem, incluindo as características de produção, qualidade, armazenamento. Esses “dados sobre dados” são essenciais para identificar, documentar, integrar e disponibilizar os produtos geoespaciais descritos. Desta forma, os metadados permitem descrever as características particulares da informação geográfica tais como: escala, autor, data, projeção, nome do mapa, localização etc., permitindo assim que o usuário conheça as características técnicas de um determinado produto. (CONCAR, 2020)

Nesse contexto, os padrões de metadados geoespaciais são utilizados para descrever, documentar e intercambiar informações sobre produtos geoespaciais de forma padronizada, permitindo a interoperabilidade entre sistemas e a utilização consistente de dados. Entre os principais padrões abordados estão a ISO 19115, o esquema XML da ISO 19139, o Perfil de Metadados Geoespaciais do Brasil (MGB) e a ET-PCDG. Esses padrões e esquemas definem estruturas e diretrizes para o correto preenchimento e validação de metadados.

### 2.2.1 ISO 19115:2003

A ISO 19115:2003 é uma norma que especifica o modelo conceitual para a descrição de metadados de informações geoespaciais. A norma fornece uma estrutura hierárquica composta por entidades, atributos e relacionamentos, permitindo a descrição detalhada e precisa das características dos dados. Sendo assim, ela estabelece os elementos fundamentais necessários para documentar aspectos como a identificação do dado geoespacial, sua qualidade, sua cobertura geográfica e as condições de uso. A norma define, por exemplo, elementos de identificação como o título do dado, a data de criação e o responsável pela manutenção, além de incluir aspectos relacionados à precisão locacional e temática. O objetivo principal é garantir que os dados descritos possam ser reutilizados de maneira consistente e que a informação seja documentada de forma padronizada, facilitando a integração entre diferentes sistemas (ISO, 2014).

### 2.2.2 ISO 19139:2007 e esquemas XML

A ISO 19139:2007 (ISO, 2019) define um conjunto de esquemas XML para metadados definidos na ISO 19115:2003 (ISO, 2014). Seu objetivo é definir um formato de arquivos para metadados geoespaciais que seguem a norma ISO 19115:2003. Esses esquemas permitem estruturar e validar os arquivos XML dos metadados em conformidade com a norma.

Mantido desde 1998 pelo World Wide Consortium (W3C), o XML é uma linguagem de marcação flexível e simples. Apresenta como características principais ser uma linguagem baseada em texto, ter separação do conteúdo da formatação, ser simples e fácil de ser

interpretado e ter a possibilidade de criação de tags sem limitações, facilitando o intercâmbio de dados pela Internet (GOLDBERG, 2009). O formato XML também permite que as instâncias de metadados possam trafegar em serviços Web geoespaciais, como previsto na especificação do *Catalogue Services – Web* (NEBERT; VOGES; BIGAGLI, 2016).

Um documento XML pode ser considerado bem formado se estiver de acordo com o que prescreve as normas (BRAY et al., 1997). Este documento pode ser válido desde que obedeça a algumas normas descritas na sua gramática. O XML Schema Definition (XSD) é um formato de gramática para XML (Fallside e Walmsley, 2004). A ISO 19139 é descrita sob a forma de esquemas XML construídos sob a especificação de Thompson et al. (2004). O XSD da ISO 19139 é utilizado pelos softwares de serviços CSW para validar a estrutura dos metadados que são disponibilizados no esquema da ISO 19115.

### 2.2.3 Perfil de Metadados Geoespaciais do Brasil

O Perfil de Metadados Geoespaciais do Brasil (MGB) é um padrão desenvolvido com base na ISO 19115, adaptado às necessidades nacionais. Ele visa padronizar o uso de metadados em instituições brasileiras, estabelecendo uma estrutura comum para a criação e descrição de informações geoespaciais. O Perfil MGB oferece dois níveis de conformidade: o sumarizado, que abrange os elementos essenciais da ISO 19115, e o completo, que contempla uma parte maior da norma. A adoção desse perfil tem como objetivo principal garantir a consistência e a interoperabilidade na descrição de produtos geoespaciais no Brasil, facilitando o compartilhamento de informações entre diferentes instituições públicas e privadas (CONCAR, 2020).

### 2.2.4 ET-PCDG

A Especificação Técnica para Produtos de Conjuntos de Dados Geoespaciais (ET-PCDG), é um documento técnico desenvolvido pela DSG do Exército Brasileiro. Ele estabelece diretrizes e padrões para a produção de produtos cartográficos digitais pela DSG. Um de seus pilares é a adoção do padrão ISO 19115 para a documentação de metadados é umas das partes essenciais da ET-PCDG, proporcionando uma estrutura robusta para descrever e gerenciar informações geoespaciais. (DSG, 2016)

Na ET-PCDG, a DSG detalha o processo de documentação para cada tipo de produto cartográfico, garantindo que todos os elementos obrigatórios do ISO 19115 sejam atendidos. A documentação começa com a criação de metadados que incluem informações básicas, como título, autor, data de criação e palavras-chave, que ajudam a identificar e categorizar o produto. Em seguida, são adicionados detalhes mais específicos, como projeções cartográficas, sistema de coordenadas, escala e precisão. (DSG, 2016)

Uma parte crucial do processo de documentação é a associação de metadados com

os produtos cartográficos correspondentes, como por exemplo a definição das informações necessárias no cadastro de uma carta topográfica ou uma ortoimagem. A DSG utiliza ferramentas próprias para criar e validar os valores dos metadados de acordo com os padrões internacionais, facilitando também a integração com sistemas de catálogo geoespeciais, permitindo o compartilhamento dos dados registrados. (DSG, 2020)

Além disso, a ET-PCDG destaca a importância de manter os metadados atualizados ao longo do tempo. À medida que os produtos cartográficos são revisados ou atualizados, os metadados também devem ser ajustados para refletir as mudanças, garantindo informações precisas ao usuário final. (DSG, 2016)

## 2.3 Processo de cadastro de metadados do BDGEx

O sistema de cadastro de metadados implementado no SIGWeb Fonte segue a Nota Técnica de Cadastro de Metadados (NT-CMet) (DSG, 2020). Cada produto cartográfico, como mapas, ortofotos ou dados topográficos, é acompanhado por um conjunto de metadados que documentam suas características essenciais. Esses metadados devem conter informações sobre a origem, escala, precisão, data de produção, métodos de coleta e outros detalhes técnicos definidos pela NT-CMet para a compreensão e uso adequado de cada produto. (DSG, 2020)

Para os produtos de dados geoespaciais produzidos pela DSG e previstos na ET-PCDG (DSG, 2016), deverão ser seguidos os padrões da ISO 19115 (ISO, 2014) conforme os perfis definidos na própria ET-PCDG. Para os demais produtos, salvo orientação do contrário, será adotado o perfil sumarizado conforme a ISO 19115 e o Perfil de Metadados Geoespaciais do Brasil (Perfil MGB) (CONCAR, 2020).

O processo de cadastro envolve a entrada manual dessas informações em um sistema que armazena e organiza os metadados em XML. Este sistema permite que usuários internos e externos acessem os dados geográficos utilizando parâmetros de busca específicos para localizar produtos conforme suas necessidades (vide figura 3).

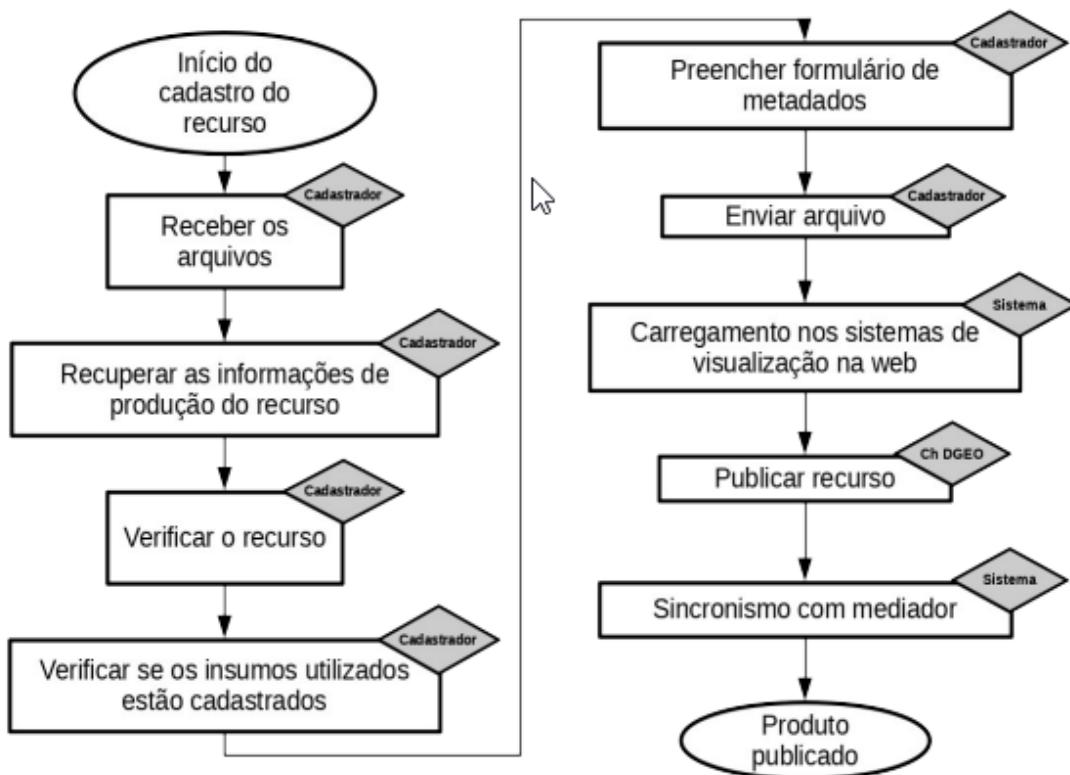


Figura 3 – Fluxograma de Cadastramento de acordo com NT-Cmet. Fonte: (DSG, 2020)

Sendo assim, a NT-CMet prevê como cada componente dentro de um sistema de cadastro de metadados deve funcionar, como cada um dos arquivos que caracterizam os produtos devem ser preparados, como cada campo deve ser preenchido e como devem ser realizadas as verificações de erros.

## 2.4 CSW e OGC API Records

O CSW (Catalogue Service for the Web) (NEBERT; VOGES; BIGAGLI, 2016) e o OGC API Records (OGC, 2024) são dois padrões desenvolvidos pelo OGC para a descoberta e acesso a informações sobre recursos geoespaciais. Cada um desses padrões tem sua própria abordagem para fornecer serviços de catálogo, mas ambos visam permitir a busca e recuperação de metadados geoespaciais de forma estruturada e padronizada, permitindo que metadados cadastrados no BDGEx sejam disponibilizados de forma pública e consumidos por diferentes softwares por meios da descentralização dos dados.

### 2.4.1 CSW

O CSW é um padrão estabelecido voltado à criação de serviços de catálogo na Web. Ele permite a descoberta, pesquisa e acesso a metadados geoespaciais por meio de uma interface baseada em HTTP. O CSW utiliza um formato XML para estruturar as solicitações e respostas, e suporta uma variedade de esquemas de metadados, como ISO 19115, Dublin Core e outros padrões de metadados geoespaciais. (NEBERT; VOGES; BIGAGLI, 2016)

Entre suas principais características estão (NEBERT; VOGES; BIGAGLI, 2016):

- **Interoperabilidade:** O CSW foi projetado para ser compatível com diversos tipos de metadados, facilitando a interoperabilidade entre sistemas diferentes.
- **Funcionalidades de busca:** Os serviços CSW oferecem recursos avançados de busca, como consultas por palavra-chave, filtros espaciais e temporais, e outros critérios de pesquisa.
- **Promover a Integração com Outros serviços:** O CSW pode ser integrado com outros serviços padrões do OGC, como WMS (*Web Map Service*) e WFS (*Web Feature Service*), para fornecer acesso a recursos geoespaciais completos.

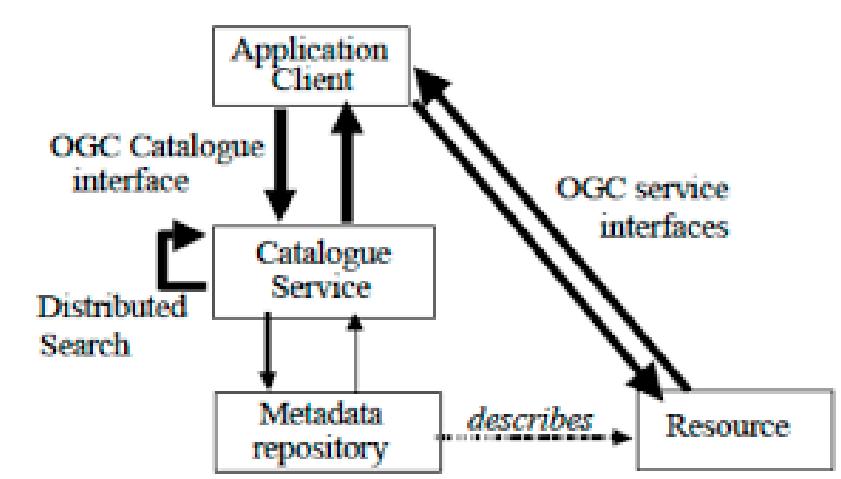


Figura 4 – Arquitetura do CSW em conjunto com outros serviços OGC. (OGC, 2018)

### 2.4.2 OGCAPI Records

O OGCAPI Records é um padrão desenvolvido em 2018 e faz parte da iniciativa do OGC para modernizar seus padrões de serviços, adotando uma abordagem mais orientada ao uso de APIs RESTful. O OGCAPI Records tem como objetivo fornecer serviços de catálogo que sejam mais leves, escaláveis e compatíveis com a infraestrutura moderna da web. Entre suas principais características estão (OGC, 2024):

- **Arquitetura RESTful:** O OGC API Records usa princípios REST para comunicação, facilitando a integração com aplicativos *web* modernos e simplificando as operações.
- **Uso de formatos de dados modernos:** Em vez de XML, o OGC API Records pode usar formatos como JSON para solicitações e respostas, tornando-o mais acessível para desenvolvedores *web*.
- **Escalabilidade e flexibilidade:** Devido ao uso de APIs REST, o OGC API Records pode ser mais escalável e flexível do que o CSW, permitindo uma integração com tecnologias *web* contemporâneas.
- **Compatibilidade com outras APIs do OGC:** O OGC API Records (OGC, 2024) é parte de um esforço maior do OGC para criar um conjunto de APIs interoperáveis para serviços geoespaciais, proporcionando maior coesão entre diferentes serviços e padrões.

O OGC API Records em conjunto com o CSW são padrões que devem ser utilizados para disponibilização ampla dos metadados gerados por um formulário de cadastro de metadados.

## 3 MATERIAIS E MÉTODOS

Este capítulo apresenta os materiais e métodos utilizados no desenvolvimento da aplicação. Serão abordados os requisitos, arquitetura da solução e ferramentas e tecnologias de desenvolvimento empregadas na solução final.

### 3.1 Gestão do projeto

#### 3.1.1 Partes interessadas

As partes interessadas nesta aplicação são:

- **DSG:** Espera um componente que tenha as mesmas funcionalidades do sistema atual, mas permita a evolução e integração com novos componentes.
- **Chefes de Divisão de Geoinformação:** Autorizam a publicação de produtos. Gostariam que o sistema verificasse diferentes aspectos de qualidade dos produtos e metadados cadastrados, para auxiliar na decisão de publicar ou não.
- **Cadastradores de metadados:** Vão inserir produtos no sistema. Desejam que o sistema tenha o máximo de automação para reduzir horas trabalhadas e aumentar a quantidade de produtos carregados por dia.
- **Consumidores dos produtos cartográficos:** Desejam acessar e pesquisar produtos cartográficos na aplicação e também pelo servidor OGC CSW.

#### 3.1.2 Requisitos

Os requisitos do sistema de cadastro de metadados foram organizados em duas categorias principais: requisitos funcionais, que descrevem as funcionalidades que a aplicação deve fornecer, e requisitos não funcionais, que especificam as qualidades que o sistema deve atender.

#### 3.1.3 Requisitos Funcionais

- **Cadastro otimizado de metadados:** Simplificar o processo de registro de informações pelos cadastradores de metadados, com a automatização da extração de dados dos arquivos geoespaciais e de metadados, bem como a possibilidade de realizar requisições diretamente ao servidor da aplicação.

- **Ferramentas para o gerenciamento dos metadados:** Prover ferramentas para a verificação da precisão e consistência dos metadados cadastrados, além de gerenciar todos os arquivos submetidos para assegurar a qualidade dos dados no BDGEx.
- **Criação de metadados no *Front-end*:** Implementar uma interface que permita a criação de metadados, seguindo os padrões da ET-PCDG, para cada tipo de produto geoespacial.
- **Integração com o sistema CadastroGeral:** Consumir os dados gerais da produção cartográfica de cada CGeo por meio de uma API Rest para preenchimento automático de informações no sistema.
- **Importação de XML de metadados no *Back-end*:** O sistema deve permitir a importação de arquivos XML contendo metadados já preenchidos, para simplificar o processo de cadastro.
- **Validação de XML de metadados no *Back-end*:** Implementar validações automáticas dos arquivos XML de metadados, assegurando conformidade com as normas da ET-PCDG e da NT-CMet, para garantir a integridade dos dados.

### 3.1.4 Requisitos Não Funcionais

- **Usabilidade:** O sistema deve proporcionar uma interface amigável e intuitiva para os operadores, facilitando o cadastro e a navegação entre os diferentes módulos da aplicação.
- **Desempenho:** O sistema deve ser capaz de processar grandes volumes de metadados (*gigabytes*) de forma eficiente, garantindo tempos de resposta rápidos nas consultas e operações de importação/exportação de dados.
- **Integração com outros sistemas geoespaciais:** Desenvolver a aplicação para facilitar a integração com outras plataformas geoespaciais, conforme os padrões da OGC e das ISOs, possibilitando o intercâmbio de dados entre entidades.
- **Conformidade com padrões internacionais:** Garantir que o sistema esteja de acordo com os padrões ISO 19115 e ISO 19139 para metadados geoespaciais, além de atender às normativas do Exército Brasileiro, como a ET-PCDG e o MGB.

## 3.2 Escopo

### 3.2.1 Estrutura analítica do projeto (EAP)

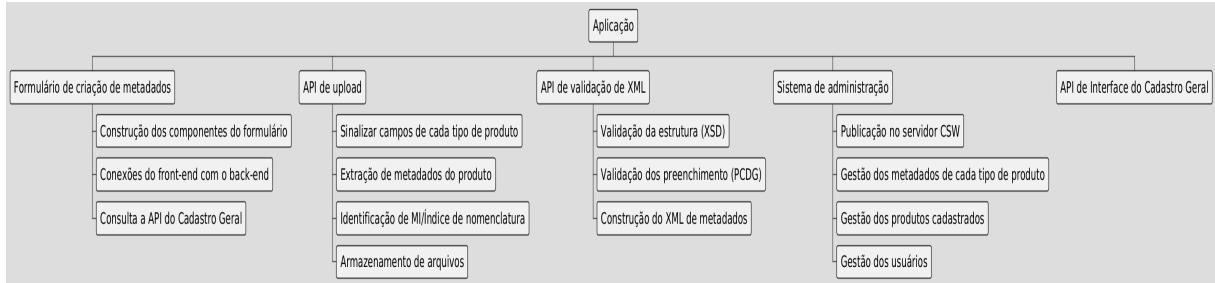


Figura 5 – Estrutura analítica do projeto.

### 3.2.2 Dicionário da EAP

#### 1. Aplicação

- **Descrição:** Projeto principal que abrange todas as atividades relacionadas à criação, desenvolvimento e administração da aplicação.

##### 1.1. Formulário de criação de metadados

- **Descrição:** Desenvolvimento do *front-end* responsável pela criação e gerenciamento do formulário de metadados.
- **Subcomponentes:**
  - **1.1.1. Construção dos componentes:** Desenvolvimento dos componentes em React que serão parte integrante do formulário, como os campos do formulário e a tela de upload de arquivo.
  - **1.1.2. Conexões do *front-end* com o *back-end*:** Conexões com as APIs do *back-end* de tal forma a transformar o preenchimento do formulário dinâmico.
  - **1.1.3. Consulta à API do Cadastro Geral:** Assegurar a responsividade às consultas a API REST que intermedia as informações para a obtenção de dados do sistema do Cadastro Geral.

##### 1.2. API de upload

- **Descrição:** API dedicada ao upload de metadados e produtos geoespaciais.
- **Subcomponentes:**

- **1.2.1. Sinalizar tipos de produtos:** Identificação dos tipos de produto a partir da extensão do arquivo.
- **1.2.2. Extração de metadados do produto:** Processo de extração de metadados a partir de um produto geoespacial.
- **1.2.3. Identificação de MI/Índice de nomenclatura:** Desenvolver um mecanismo de identificação e seleção automatizada do Índice de Nomenclatura a partir de produtos geoespaciais.
- **1.2.4. Armazenamento dos arquivos:** Estrutura de armazenamento para os arquivos de metadados e produtos geoespaciais.

### 1.3. API de validação de XML

- **Descrição:** API que valida a estrutura e o conteúdo dos arquivos XML de metadados nos padrões nacionais e internacionais.
- **Subcomponentes:**
  - **1.3.1. Validação da estrutura (XSD):** Verificação da conformidade dos XMLs com os esquemas XSD definidos para cada tipo de produto.
  - **1.3.2. Validação dos preenchimentos (ET-PCDG):** Validação dos preenchimentos conforme os padrões da ET-PCDG e NT-CMet.
  - **1.3.3. Construção do XML de metadados:** Geração do XML de metadados a partir dos preenchimentos dos campos fornecidos no formulário.

### 1.4. Sistema de administração

- **Descrição:** Sistema de gestão e administração dos formulários, produtos e usuários cadastrados.
- **Subcomponentes:**
  - **1.4.1. Publicação no servidor CSW:** Mecanismo de publicação dos metadados cadastrados no servidor de catálogo de serviços web.
  - **1.4.2. Gestão dos campos dos formulários de produtos:** Administração e configuração dos campos disponíveis para cada tipo de produto que devem ser renderizados pelo *front-end*.
  - **1.4.3. Gestão dos produtos cadastrados:** Gerenciamento e controle dos produtos cadastrados no sistema.
  - **1.4.4. Gestão dos usuários:** Administração e controle de permissões de usuários no sistema.

## 1.5. API de Interface do Cadastro Geral

- **Descrição:** Interface dedicada ao gerenciamento e intermediação do fluxo de informações com o Cadastro Geral para obtenção dos dados como: responsáveis técnicos dos CGeos, organizações responsáveis pelos produtos, etc.

### 3.2.3 Exclusões do Escopo

Devido à extensão das normas de cadastro de metadados da DSG, o escopo deste projeto contempla os procedimentos de cadastro de Cartas Topográficas Matriciais, sumarizadas e completas. O sistema foi projetado de forma a permitir futuras expansões para os demais tipos de produtos.

### 3.2.4 Entregas

#### Softwares:

- *front-end* - `geometadata_editor`: Foi proposto a entrega de um *front-end* desenvolvido em React, com componentes que em conjunto formam o formulário de cadastro de metadados que é visto pelos usuários da aplicação.
- *back-end* - `geometadata_creator`: Foi proposto a entrega de um *back-end* desenvolvido em Django, onde é realizado todas as manipulações de dados, armazenamento de arquivos e construção das APIs que possibilita as trocas de informações entre as informações introduzidas pelos usuários no *front-end* e a validação, processamento e armazenamento dessas informações.

#### Código-fonte:

- `geometadata_editor`: [https://github.com/pedro-kovalczuk/geometadata\\_editor](https://github.com/pedro-kovalczuk/geometadata_editor)
- `geometadata_creator`: [https://github.com/mauriciodev/geometadata\\_creator](https://github.com/mauriciodev/geometadata_creator)

#### Contêineres:

- Docker - Parte integrante do `geomedata_creator` que permite em apenas algumas linhas de código seja possível transformar todo o ambiente de produção e o servidor em funcionais.

#### Documentação:

- Swagger/Open API: Documentação de todas as APIs construídas no *back-end*, com suas entradas, saídas e também descrições de como são utilizadas pelo *front-end*;

- Github: Elaboração de arquivo Read.me contendo instruções de como operacionalizar o docker e a aplicação, além de outros detalhes de ambas as interfaces (*front-end* e *back-end*).

### 3.3 Processo de Desenvolvimento do Software

O desenvolvimento do software foi conduzido utilizando a metodologia ágil Kanban com ciclos de desenvolvimento (*sprints*) de 2 semanas, escolhida para proporcionar maior visibilidade do fluxo de trabalho e facilitar a adaptação a mudanças nos requisitos e priorizações ao longo do projeto. As tarefas foram subdivididas em componentes da EAP (1.1 a 1.4) para que o desenvolvimento do formulário fosse realizado de forma incremental e assíncrona entre as duas interfaces.

A equipe responsável pelo desenvolvimento era composta por quatro integrantes, com funções bem definidas:

- **Maj Maurício**, atuando como *Product Owner* (PO), foi o responsável por assegurar que as funcionalidades do software estivessem de acordo com as necessidades da DSG e com as normas previamente estabelecidas no formulário de cadastro de metadados do BDGEx.
- **Arthur**, na função de *Product Manager* (PM), coordenou as atividades da equipe, garantindo que o desenvolvimento fosse executado conforme os objetivos estratégicos do projeto.
- **Márcio e Pedro**, respectivamente desenvolvedores do *back-end* e do *front-end*, foram responsáveis pela implementação técnica das funcionalidades, que incluíram a construção dos formulários, integração com APIs e validação de dados.

#### 3.3.1 Estrutura de Gestão com Kanban

O Kanban foi utilizado como ferramenta de gestão, com o *software* online Trello, para organizar e monitorar as atividades da equipe, distribuídas em cinco colunas principais:

1. **Backlog**: lista de todas as tarefas a serem realizadas, priorizadas conforme as necessidades identificadas pelo PO e as diretrizes do projeto.
2. **A Fazer**: tarefas selecionadas pelo PM, prontas para serem iniciadas pelos desenvolvedores na próxima sprint.
3. **Em Progresso**: tarefas que estavam sendo ativamente desenvolvidas por Márcio e/ou Pedro, com descrições detalhadas e critérios de avaliação previamente definidos.

4. **Em Revisão:** tarefas concluídas pelos desenvolvedores, aguardando revisão e validação por parte do PO, conforme os requisitos especificados.
5. **Concluído:** tarefas que passaram pela revisão e foram integradas ao sistema, concluindo o ciclo de desenvolvimento.

### 3.3.2 Exemplo de Implementação

Uma das tarefas críticas desenvolvidas foi a construção do XML de metadados (item 1.1.4 da EAP). Esta tarefa consistiu na geração automatizada de arquivos XML a partir dos dados inseridos nos formulários de cadastro, assegurando a conformidade com os padrões XSD estabelecidos. O PO definiu os requisitos de estrutura e preenchimento do XML, enquanto o PM organizou as entregas dos desenvolvedores de forma que cada parte do XML fosse validada iterativamente.

Márcio e Pedro desenvolveram as rotinas necessárias em cada interface, do *back-end* e do *front-end*, para a geração do XML e realizaram a integração com APIs para consulta ao Cadastro Geral, permitindo que o sistema obtivesse e organizasse os dados corretamente, conforme os padrões exigidos.

A utilização do Kanban possibilitou uma organização eficiente das atividades e um acompanhamento claro do progresso de cada tarefa. Isso contribuiu para a adaptação ágil a novas demandas e alterações nos requisitos, assegurando a entrega contínua e progressiva de funcionalidades que atendiam tanto aos requisitos técnicos da NT-CMet quanto às normas vigentes do BDGEx.

## 3.4 Visão geral do formulário de cadastro de metadados

Foram desenvolvido dois fluxos para a construção e armazenamento de metadados de produtos cartográficos. O fluxograma principal de cadastro de metadados, com sua interação entre os agentes podendo ser visualizada na Figura 6, é o fluxo onde o operador possui o produto cartográfico e quer a partir dele gerar o XML de metadados.

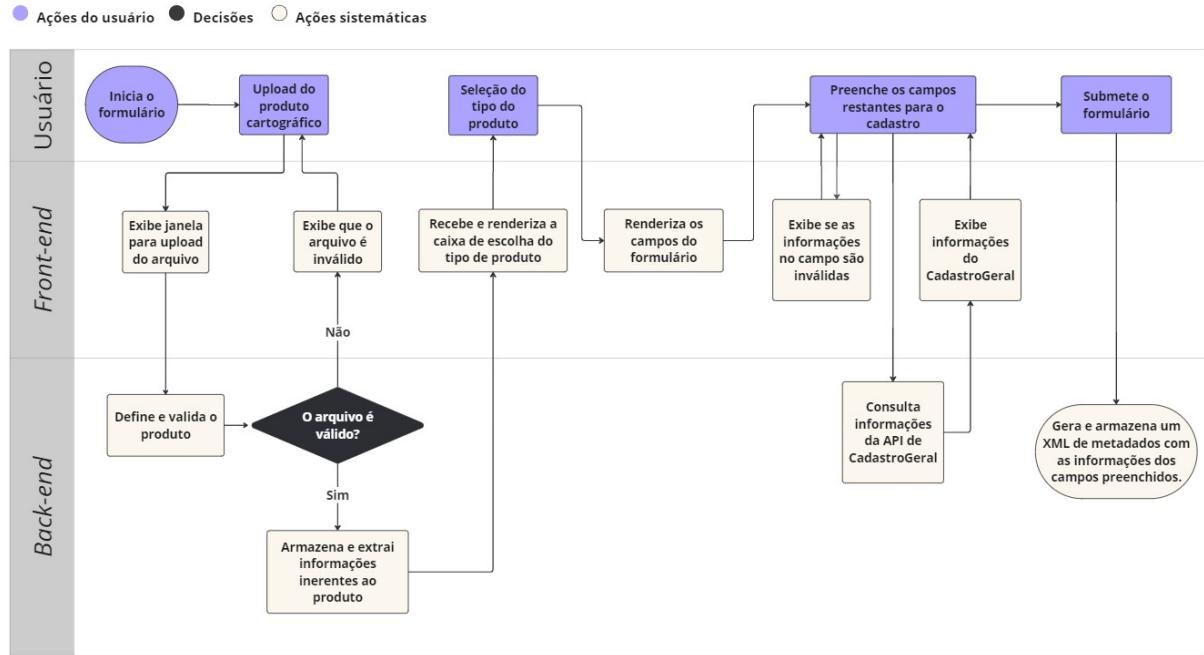


Figura 6 – Fluxograma principal do cadastro de metadados.

Neste fluxo o cadastrador de metadados inicia o formulário carregando um arquivo geoespacial (Cartas topográficas, ortoimagens, Modelos digitais de superfícies, etc) por meio do *front-end* que o envia para o *back-end*. O *back-end* valida se o arquivo é compatível com alguns tipos de produtos e define quais desses tipos de produto ele pode ser. Caso o arquivo seja inválido, o usuário recebe uma mensagem de erro e deve realizar novamente o upload de um produto, e caso o arquivo seja válido, o *back-end* armazena o produto no banco de dados e extraí algumas informações de metadados do produto de forma automatizada. Caso o processamento do *back-end* tenha sucesso, ele retorna ao *front-end* os tipos de produto que o usuário pode cadastrar, bem como os metadados que compõe cada tipo, a id do produto armazenado e os campos que foram extraídos de forma automática. O usuário então seleciona o tipo de produto que ele está cadastrando, e a partir disso o *front-end* renderiza os campos do formulário para o preenchimento. A medida que o usuário realiza o preenchimento é retornada a validação para cada campo, assim como as opções de preenchimento dos campos relacionados ao CadastroGeral diretamente da API que intermedia essa comunicação. Após o usuário finalizar o preenchimento e submeter o formulário, as informações são enviadas ao *back-end* que gera o XML de metadados e associa ao produto já armazenado.

A aplicação também possui um fluxo secundário de cadastro, que pode ser visualizado na Figura 7, onde o cadastrador de metadados já possui o XML de metadados a ser cadastrado e precisa armazená-lo junto a um produto cartográfico no sistema.

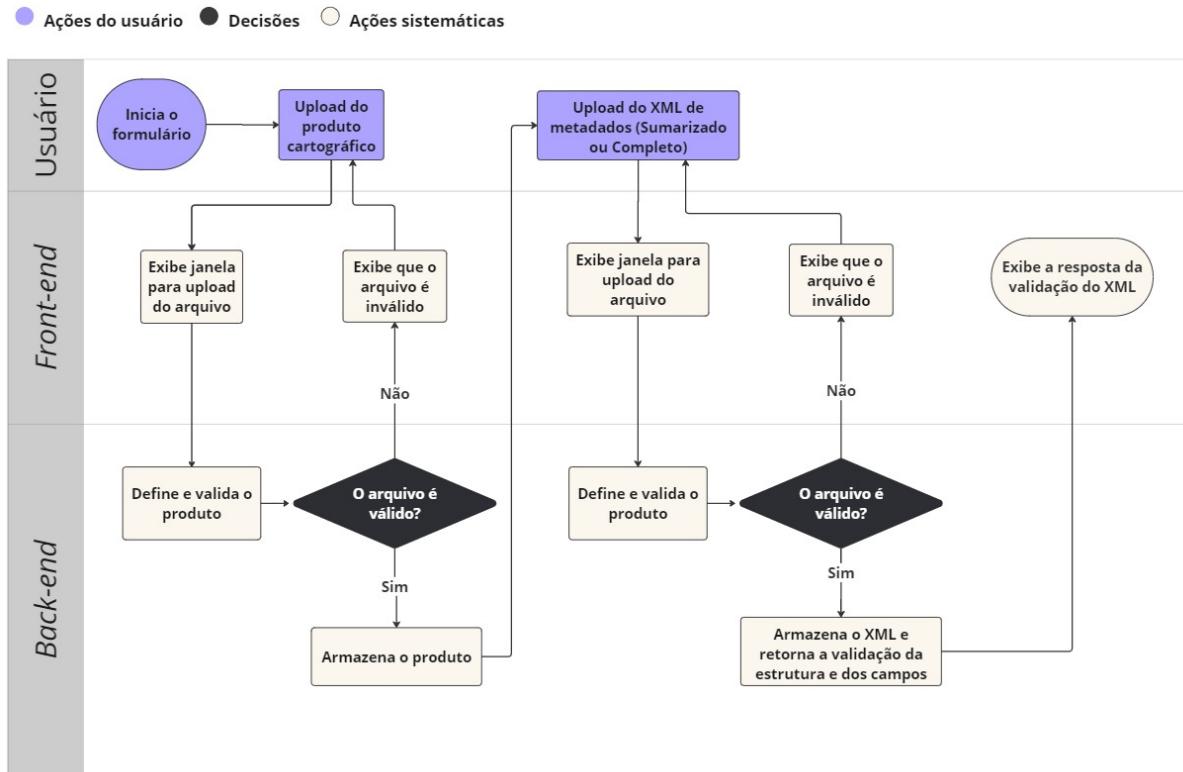


Figura 7 – Fluxograma secundário do cadastro de metadados.

Neste fluxo o cadastrador de metadados inicialmente realiza o upload do produto cartográfico, que é validado se o arquivo é compatível com alguns tipos de produtos (.shp, .tif, etc) e caso não seja válido, o usuário deve submeter um produto válido. Caso o arquivo seja válido, o cadastrador deve fazer o upload do XML de metadados e indica, caso seja um XML já cadastrado, o id no sistema. Após isso o formulário retorna se o XML é válido ou não, retornando caso não seja válido que o XML deve ser enviado novamente. No caso do XML sendo válido, o *back-end* define o tipo de produto, armazena o XML e envia a resposta completa da validação do XML. O *front-end*, a partir da resposta, renderiza a validação do XML para o usuário.

### 3.5 Componentes

Nessa seção será abordado os componentes que constituem a aplicação do formulário de metadados, sendo aprofundados quais seus objetivos, como se deram suas construções, seus funcionamentos e como ocorre a interação entre eles.

### 3.5.1 *Front-end*

O *front-end* tem a responsabilidade de promover uma visão responsiva, onde o usuário terá uma interface explicativa para preenchimento do formulário e indicações caso o preenchimento do campo esteja inválido com as normas e regras de validações definidas no *back-end*. Ele também possui a responsabilidade de possibilitar o preenchimento descentralizado entre os vários cadastradores de metadados que irão utilizar a aplicação.

Dessa forma, o *front-end*, desenvolvido em *React*, foi dividido entre os campos e *layouts* que constituem o formulário de metadados e as configurações de comunicação via APIs do *back-end*.

#### 3.5.1.1 Componentes do formulário de metadados

Os componentes em *React* foram criados de acordo com os requisitos funcionais que o *front-end* deve cumprir, assim como os padrões da NT-CMet que foram mantidos nessa aplicação.

Foram criados componentes para o *Header* que representa o título da aplicação na parte superior da tela, bem como para a *sidebar*, que possui as opções dinâmicas de acordo com o momento em que o usuário se encontra no preenchimento do formulário. Além desses, a aplicação possui o componente de Upload dos produtos ou XML de metadados, bem como o componente *MetadataEditor* que define e gere os tipos de campos a serem renderizados no formulário de acordo com as respostas do *back-end*. Por fim, temos o componente *MainContent* que gere o funcionamento de todos os componentes entre si para assegurar um fluxo contínuo e responsável de preenchimento do formulário de cadastro de metadados.

Também foram criados *layouts* para estilizar a interface da aplicação e os componentes citados anteriormente, visando transformá-la em acessível para deficientes visuais e responsiva.

#### 3.5.1.2 Gerenciamento das chamadas ao *back-end*

Foram criadas configurações utilizando o *Axios*, que é uma biblioteca do *React* para realizar requisições HTTP, permitindo realizar chamadas de API assíncronas ao *back-end*, como utilizar a API de upload do *back-end* para enviar um produto cartográfico carregado pelo cadastrador de metadados e receber sua resposta em tempo real, podendo renderizar a partir da resposta os tipos de produtos que devem ser preenchido. Além do exemplo da API de upload, também foi conectada a resposta dos campos que o *front-end* deve renderizar a partir do tipo de produto escolhido pelo cadastrador ao componente de *MetadataEditor*. Também foi configurado pelo *axios* a resposta dos valores dados pela API de CadastroGeral a cada um dos campos renderizados no formulário que são sinalizados como pertencentes

ao CadastroGeral em sua criação, com o intuito de oferecer um *dropdown* de opções para o usuário nestes campos.

### 3.5.2 Back-end

Para atingir os requisitos funcionais e não funcionais descritos anteriormente no projeto, o *back-end* foi desenvolvido em *Django*, e possui como algumas das suas responsabilidades:

- Otimizar o sistema atual do formulário de cadastro de metadados;
- Garantir a conformidade dos metadados com os padrões descritos na NT-CMet;
- Possibilitar a integração com outros sistemas, conforme os padrões da OGC, por meio de uma arquitetura REST;
- Fazer a hospedagem dos produtos e metadados no servidor OGC CSW;
- Gestão e armazenamento dos produtos, metadados e usuários cadastrados.

Além do código em *Django*, o *back-end* do formulário também compreende a base de dados em PostgreSQL. Essas duas partes do *back-end* são separadas em contêineres Docker, pois dessa forma a base de dados é protegida em caso de erros com a aplicação e possibilita a implantação do código em qualquer servidor disponível para hospedagem. O Django permite armazenar arquivos no sistema de arquivos do servidor, organizando-os em tabelas no banco de dados de acordo com as classes e suas interações com as chamadas via APIs. Desta forma, toda a integração entre os arquivos armazenados e os dados fornecidos durante o carregamento será feita por meio de classes do Django.

Contudo, vale destacar que, em alguns módulos da aplicação, não há a necessidade de persistência de dados por meio de um banco de dados tradicional. Nestes casos, a aplicação utiliza um modelo *stateless*, no qual os dados não são armazenados de forma duradoura, mas são *materializados* conforme as chamadas à API são feitas. Isso significa que os dados são processados e retornados dinamicamente durante as requisições, sem a necessidade de um armazenamento permanente. Assim, quando uma solicitação é feita para visualizar ou submeter metadados, a API se comunica com serviços externos ou fontes de dados (nesse nosso caso um PostgreSQL) para gerar as informações temporariamente, atendendo a requisição sem estado persistente.

Nesse contexto, foram desenvolvidos diferentes componentes para atingir os requisitos necessários da aplicação, podemos separá-los em: as APIs, o sistema de administração e o banco de dados.

### 3.5.2.1 APIs

O Django possibilita a criação, por meio do *Django REST Framework*, das APIs REST que serão criadas para comunicação com o *front-end* ou com outros sistemas no que se diz respeito ao *upload* de arquivos, validação do XML dos metadados e também a intermediação dos dados do sistema do Cadastro Geral. Isso permite um preenchimento descentralizado por parte de servidor e interface com o usuário, facilitando possíveis mudanças que podem ocorrer nas normas que padronizam os produtos a serem submetidos pelos cadastradores de metadados no formulário, e possibilitando que todos os CGeos com seus vários operadores possam realizar seus cadastros simultaneamente.

Nessa seção serão abordados como cada API foi construída e quais são seus *endpoints*, que são URLs específicas que representam uma operação no servidor.

#### 3.5.2.1.1 API de upload de geoproduto

A API de upload, chamada de *GeoresourceUploadAPI* foi criada com os seguintes objetivos:

- Armazenamento dos produtos e metadados enviados;
- Filtrar e retornar possíveis tipos de produtos que são compatíveis com o arquivo enviado (Ex: Carta Topográfica Matricial Sumarizada);
- Extrair e retornar campos de metadados dependentes do arquivo enviado (Ex.: extensão geográfica, sistema de referência, etc.);

Sendo assim, essa API possui o seguinte *endpoint*:

- **geoproduct/**

Esse caminho realiza o envio dos produtos ao *back-end*, retornando o id do produto criado, os tipos de produtos válidos no preenchimento e os campos extraídos do produto. Aqui será realizado as seguintes extrações por meio deste caminho: as extensões espaciais (Norte, Sul, Leste e Oeste), tipo de representação de dados, código EPSG, formato de distribuição do arquivo, escala, resolução espacial, MI e INOM. Foram criadas funções auxiliares para essas extrações, porém a que possui os maiores tratamentos do produto geoespacial é a função de extração do MI e do INOM, que possui a sua descrição na seção 3.5.3.

### 3.5.2.1.2 API de construção de XML

- **geoprodut/{id}/build\_metadata.**

Esse caminho é utilizado ao final do preenchimento do formulário de metadados, no fluxo principal, onde ele valida os campos enviados e cria um XML de metadados baseado no template que é armazenado no sistema (que foi enviado pelas partes interessadas), recebendo um conjunto estruturado dos caminhos ISO do XML de metadados e seus respectivos valores que foram preenchidos no *front-end* direto no body da requisição. A resposta dessa requisição é um arquivo XML de metadados, que é armazenado juntamente com o seu produto geoespacial.

### 3.5.2.1.3 API de validação de XML

- **geoprodut/{id}/send\_xml\_metadata.**

A API de validação de XML é responsável por receber, e validar o arquivo de metadados enviado pelo usuário. Ela se aplica ao fluxo secundário, onde após o envio do produto geoespacial pelo endpoint "geoprodut/" da API de Upload, é utilizado para envio do XML de metadados. O endpoint armazena o XML de metadados no mesmo objeto que o produto geoespacial, relacionados pela *id* do sistema enviada no nome da sua chamada, e valida a estrutura e os campos, retornando um JSON de validação do XML.

### 3.5.2.1.4 API de interface do Cadastro Geral

A API de interface do Cadastro Geral foi criada com o intuito de intermediar o acesso ao servidor do sistema Cadastro Geral, que possui o controle dos valores possíveis de alguns campos que são armazenados em um XML de metadados do BDGEx. Os seguintes endpoints abaixo são responsáveis por retornar ao *front-end* um dicionário de metadados associados as suas respectivas lista de valores possíveis.

- **cadastro\_geral/MD\_DataIdentification-extent-verticalExtent-verticalDatum**

É retornado os datums verticais possíveis utilizados para os projetos do BDGEx.

- **cadastro\_geral/MD\_Identification-citation-collectiveTitle**

É retornado o metadado que se diz respeito ao projeto, dentro do contexto do BDGEx, em que aquele produto se refere.

- **cadastro\_geral/MD\_Metadata-contact-individualName**

São retornado os metadados referentes ao responsável pelas informações cadastrados, como o nome, função do responsável, nome da sua organização, website da sua organização e função dentro da organização.

- **cadastro\_geral/MD\_Metadata-contact-organisationName**

São retornado os metadados referentes a organização responsável pelo produto, como nome, endereço da organização, telefone da organização, website da organização e função da organização.

### 3.5.2.2 Sistema de administração

O Django possui um recurso chamado Django-Admin que, após a configuração, produz uma interface para manipular dados. Os “modelos” implementados no Django são abstrações para estruturar e manipular os dados da aplicação. Quando um modelo é registrado no Django-Admin o *framework* cria todos os recursos necessários para listar, inserir e manipular os dados dos modelos, em uma interface *web*.

A interface de administração é utilizada para configurar tópicos do comportamento do sistema que são utilizados apenas pelos administradores do BDGEx ou dos Ch DGEo. Por exemplo, a relação de quais componentes dos formulários de metadados aparecem em cada produto é feita nesta interface. O controle de acesso é feito utilizando o próprio sistema de autenticação do Django.

### 3.5.2.3 Componente de servidor de metadados (CSW/API Records)

É também na interface de administração dos produtos cadastrados e por meio da OSWLib, implementado pela biblioteca PyCSW, que é disponibilizado uma ação de publicação no servidor CSW que realiza a comunicação do *back-end* com esse servidor para que os produtos sejam validados, cadastrados e disponibilizados pelo serviço CSW.

### 3.5.3 Algoritmo de cálculo de INOM e MI

Para simplificar o formulário atual e evitar os erros de cadastro de Índice de nomenclatura e MI, foi desenvolvido essa função para extrair automaticamente essas variáveis com o tratamento do arquivo do produto geoespacial utilizando a biblioteca *rasterio*.

O primeiro passo do algoritmo é criar uma relação de tamanhos (em graus) dos enquadramentos em cada escala, iniciando na escala de 1:1.000.000, seguindo para as

escalas seguintes (500.000, 250.000, 100.000, 50.000, 25.000, 10.000, 5.000, 2.000 e, por fim, 1.000). Cada enquadramento possui um tamanho fixo quando representados em latitude e longitude. Os enquadramentos se iniciam na escala 1:1.000.000, com 4° de latitude e 6° de longitude, começando a partir do equador e da latitude -180, respectivamente.

Os enquadramentos das escalas maiores são subdivisões sistemáticas do enquadramento de 1:1.000.000. Portanto, cada divisão é representada por um código. Por exemplo, na escala 1:500.000 os códigos V, X, Y ou Z são utilizados para identificar qual das quatro subdivisões do enquadramento de 1:1.000.000 são representadas no produto. O código obtido na subdivisão de cada escala será aqui designado por INOM parcial. O INOM de um produto é, portanto, composto pela concatenação de todos os INOM parciais obtidos pelas divisões inteiras, em cada escala, da latitude e longitude do ponto central do produto.

Os passos seguintes do algoritmo para determinar o valor do INOM podem ser descrito da seguinte forma:

1. A escala do produto é obtida comparando a extensão do produto com as extensões previstas em cada escala. Ou seja, a escala cuja extensão prevista for o maior retângulo que ainda caiba dentro da extensão do produto é definida como escala do produto;
2. São obtidas a latitude e longitude do ponto central do produto e posteriormente são calculadas as linhas ou colunas que o ponto central pertence, da grade do enquadramento sistemático de cada escala. Para representar as divisões do sistema UTM, as linhas e colunas são contadas à partir de 88° latitude (norte) e -180° longitude. É calculado o resto da divisão das linhas e colunas, pela quantidade de subdivisões de cada escala. A combinação do resto da longitude e da latitude é utilizada para identificar o INOM parcial de cada escala. Por exemplo, na escala 1:500.000 (V,X,Y,Z) um produto com resto 0 na latitude e resto 1 na longitude recebe o INOM parcial X. Este procedimento é repetido da escala 1:500.000 até a escala do produto.
3. Para cada escala, é definida uma regra para compor o índice de nomeclatura final:
  - a) Para a escala de 1.000.000: Utilizamos a Latitude para definir se o INOM começará por N ou por S, depois realizaremos uma divisão inteira da latitude e longitude para definir a linha e coluna da grade que o ponto central se encontra, e o restante do INOM é composto por: É adicionada uma letra do alfabeto da posição correspondente a divisão inteira da sua latitude (começando por A até a 22ª letra do alfabeto), e de acordo com a divisão inteira da longitude é acrescido ao nome o fuso UTM correspondente. Exemplo: O INOM SA-22.

- b) Para a escala de 1.500.000 em diante: A partir dessa escala, os retângulos definidos pela escala 1.000.000 são divididos em 4 ou 6 partes, e para cada uma dessas divisões são definidos os códigos de cada um dos "setores". Exemplo: Ao verificar a divisão inteira para o ponto central nas escalas 1:500.000 e 1:250.000, são acrescidas as letras V e B, compondo um INOM na escala 1:250.000 igual à SA-22-V-B.

O INOM é utilizado para calcular o MI de cada produto. Foi obtida a lista de MIs do mapeamento sistemático à partir do WFS do BDGEx (ms:F100\_WGS84 e ms:F250\_WGS84). Estas tabelas foram introduzidas no sistema e representadas um modelo do Django chamado de *IndexMap*. Desta forma, um método busca na tabela qual é o MI da escala 1:100.000 ou 1:250.000 que compõe a parte inicial do INOM do produto. No caso do produto ser de escalas maiores, o restante do INOM é adicionado ao MI. Por exemplo, um produto na escala 1:25.000 com INOM SD-21-X-B-III-2-NO utiliza o MI 1873, relativo à parte SD-21-X-B-III do INOM (da escala 1:100.000), e as subdivisões -2-NO. Assim, o MI calculado seria 1873-2-NO.

Como o INOM é obtido pelas extensões espaciais do arquivo do produto, este procedimento permite obter também o MI. Caso o produto tenha tamanho compatível com a escala, porém não cubra o produto completamente a região, será considerado não sistemático.

## 4 RESULTADOS E DISCUSSÃO

Neste capítulo será abordado os resultados da aplicação que foi desenvolvida a partir dos requisitos funcionais e não-funcionais descritos na seção 3.1.2, bem como as comparações com a aplicação atual do BDGEx.

Os principais objetivos das mudanças ao criar a nova aplicação foram aumentar a capacidade do sistema de extrair informações do arquivo geoespacial que será carregado no início do preenchimento, e facilitar a integração com outros componentes do BDGEx, como por exemplo o sistema de Cadastro Geral que armazena as informações gerais da produção cartográfica de cada CGeo cadastrados anteriormente no BDGEx, e assim modularizando as responsabilidades de cada componente do formulário, transformando a aplicação atual que funciona como um monólito, e permitindo uma arquitetura desacoplada entre *front-end* e *back-end*.

Porém, devido às restrições de escopo o formulário de cadastro de metadados foi disponibilizado apenas para dois produtos, que são as Cartas Topográficas Matriciais Sumarizadas e Completas, aceitando apenas arquivos raster para upload de produtos geoespaciais.

Nesse contexto, foram analisados os resultados do fluxo principal (Figura 6) e também do fluxo secundário (Figura 7) do formulário de cadastro de metadados, bem como os detalhes das respostas de cada API do *back-end*.

### 4.1 Fluxo principal de cadastro de metadados

No contexto do fluxo do sistema atual, o usuário seleciona o "Adicionar novo" (Figura 8) na primeira tela do formulário, e após isso escolhe o tipo de formulário a ser preenchido (sumarizado ou completo) e o tipo do produto a ser cadastrado (Figura 9).

Figura 8 – Tela inicial do atual formulário.

Figura 9 – Tela de cadastro de um novo produto do atual formulário.

Para o novo formulário, tem-se uma mudança na ordem de cadastro, onde o upload do produto cartográfico é realizado antes da seleção do tipo de formulário e tipo de produto, de tal forma que essas informações se tornam restritas aos tipos de produtos que são carregados, como por exemplo ao carregar um arquivo de extensão .TIF não é possível preencher um tipo de produto como "SCN Carta Topográfica Matricial". Dessa forma, o fluxo começa com a solicitação de upload de um produto no diretório local do usuário

(Figura 10), que faz o upload de um arquivo (Figura 11) e confirma o envio do produto ao *back-end* para a validação e extração dos metadados contidos no produto cartográfico (Figura 12). Após o envio, é retornado a mensagem se o produto foi validado e, com as informações enviadas pelo *back-end*, é gerado a lista de tipos de formulários disponíveis para preenchimento (Figura 13).

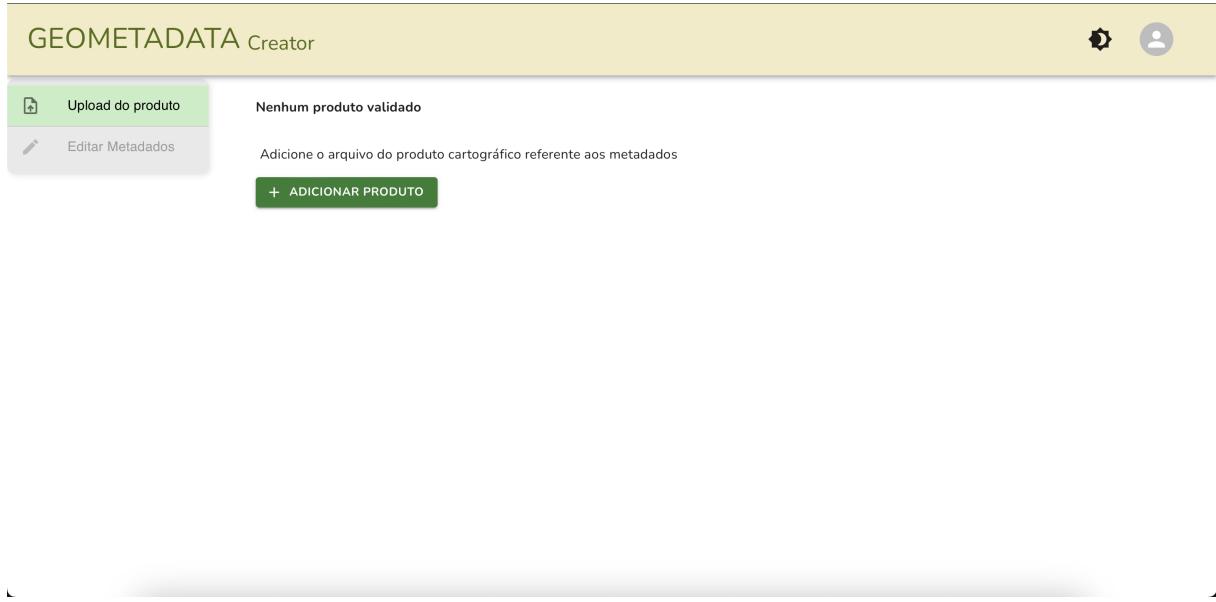


Figura 10 – Tela inicial de cadastro de um novo produto no novo formulário.

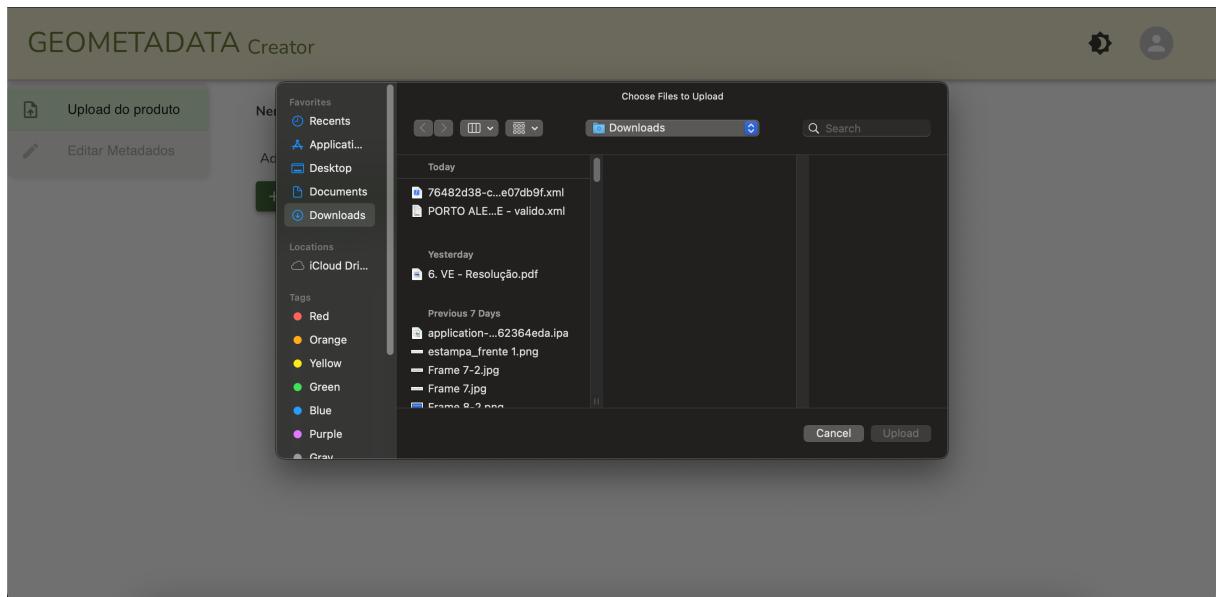


Figura 11 – Janela de upload do produto no novo formulário.

The screenshot shows the GEOMETADATA Creator interface. At the top, there's a green header bar with the title 'GEOMETADATA Creator'. Below it, a sidebar on the left has two buttons: 'Upload do produto' (highlighted in green) and 'Editar Metadados'. The main area displays a message 'Nenhum produto validado' (No validated product) and a placeholder 'Adicione o arquivo do produto cartográfico referente aos metadados'. A green button '+ ADICIONAR PRODUTO' is centered below. On the right, a file 'NDWIOUTUBRO2022.tif' (1.68 MB) is listed with a red 'EXCLUIR' button and a green 'ENVIAR' button.

Figura 12 – Confirmação para o envio do produto ao *back-end*.

This screenshot shows the same GEOMETADATA Creator interface as Figure 12, but with validation results. The file 'NDWIOUTUBRO2022.tif' now has a green checkmark next to it labeled 'Validado'. Below the file list, there's a note 'Adicione o arquivo XML de metadados do produto acima (Opcional)' and a green '+ ADICIONAR XML' button. At the bottom, a dropdown menu shows 'SCN Carta Topografica Matricial' and a green 'GERAR FORMULÁRIO' button.

Figura 13 – Retorno da validação do *back-end* e surgimento campo de tipo de formulário.

No atual formulário, ao selecionar o tipo de formulário e produto, é renderizado uma tela para seleção de índice de nomeclatura internacional (INOM), MI/MIR e a escala do produto (Figura 14). De acordo com a DSG, demandante do projeto, essas são as informações que geram mais retrabalho para os CGeos em relação a ajustes de erros, visto que não há nenhuma validação se o produto cadastrado possui os parâmetros indicados nessa etapa. Para eliminar essa problemática no novo formulário essas informações são

extraídas pelo *back-end* de forma automatizada de acordo com a seção 3.5.3, otimizando o cadastro dessas informações.

Escala	Índice	MI/MIR	Valores possíveis
250.000	SD-23-Y-C	398	
100.000	SD-23-Y-C-IV	2215	
50.000	SD-23-Y-C-IV-2	2215-2	1 a 4
25.000	SD-23-Y-C-IV-2-NO	2215-2-NO	NO, NE, SO, SE
10.000	SD-23-Y-C-IV-2-NO-A	2215-2-NO-A	A, B, C, D, E, F
5.000	SD-23-Y-C-IV-2-NO-A-	2215-2-NO-A-IV	I a IV em romanos
2.000	SD-23-Y-C-IV-2-NO-A-IV-3	2215-2-NO-A-IV-3	1 a 6
1.000	SD-23-Y-C-IV-2-NO-A-IV-3-B	2215-2-NO-A-IV-3-B	A, B, C, D

Figura 14 – Tela de seleção INOM/MI e escala do antigo formulário.

Por fim, no atual formulário, ao selecionar o INOM, MI/MIR e escala, é disponibilizado os campos estáticos (Figura 15), dividido entre diferentes seções, e somente com a última seção de preenchimento sendo o upload do produto geoespacial (Figura 16).

Figura 15 – Tela de preenchimento dos campos de metadados do formulário atual.

Figura 16 – Tela de upload do produto geoespacial do formulário atual.

Já no novo formulário, após o envio do tipo de produto escolhido, o *front-end* recebe todos os campos que deverão ser renderizados, bem como os valores dos campos que foram extraídos do produto geoespacial que são preenchidos automaticamente, seguindo os padrões da NT-CMet e divididos em seções em uma mesma tela (Figura 17). Após o preenchimento de todos os metadados obrigatórios e a submissão do formulário, é gerado o XML de metadados que é exposto em tela como confirmação ao usuário (Figura 18).

GEOMETADATA Creator

Upload do produto: NDWIOUTUBRO2022.tif / SCN Carta Topográfica Matricial

Editar Metadados

**Metametadados**

Identificador metadados: 15f8c106-5abe-45b7-93f2-3f3c1db41ba9

**Contato**

Responsável pelas informações de metadados/CREA	Função do responsável	Nome da organização
Endereço eletrônico da organização	Função da organização	

Data de preenchimento: 30/09/2024 | Idioma: Pt-BR | Conjunto de caracteres: utf8

Padrão de metadados: ISO19115 | Versão do padrão: 2003 | Nível hierárquico: conjuntoDeDadosGeográficos

Mantenção metadados: irregular | Grau de sigilo: ostensivo

Figura 17 – Tela de preenchimento dos metadados faltantes no novo formulário.

GEOMETADATA Creator

Upload do produto: Nenhum produto validado

Editar Metadados

+ ADICIONAR NOVO

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <gmd:MD_Metadata
3   xmlns:gco="http://www.isotc211.org/2005/gco/"
4   xmlns:gmd="http://www.isotc211.org/2005/gmd"
5   xmlns:gml="http://www.opengis.net/gml/"
6   xmlns:gts="http://www.isotc211.org/2005/gts"
7   xmlns:ows="http://www.opengis.net/ows/"
8   xmlns:xlink="http://www.w3.org/1999/xlink"
9   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
10  <gmd:fileIdentifier>
11    <gco:CharacterString>cfa08cc1-e710-4872-a139-bb6a57c4f1a1</gco:CharacterString>
12  </gmd:fileIdentifier>
13  <gmd:language>
14    <gco:CharacterString>pt-BR</gco:CharacterString>
15  </gmd:language>
16  <gmd:characterSet>
17    <gmd:MD_CharacterSetCode
18      codeList="http://resources/codeList.xml#MD_CharacterSetCode"
19      codeListValue="utf8">utf8</gmd:MD_CharacterSetCode>
20  </gmd:characterSet>
21  <gmd:parentIdentifier gco:nilReason="missing"/>
22  <gmd:hierarchyLevel>
23    <gmd:MD_ScopeCode
24      codeList="http://resources/codeList.xml#MD_ScopeCode"
25      codeListValue="dataset">dataset</gmd:MD_ScopeCode>
26  </gmd:hierarchyLevel>
27  <gmd:hierarchyLevelName gco:nilReason="missing"/>
28  <gmd:contact>
29    <gmd:CI_Responsibility>
30      <gmd:individualName>
31        <gco:CharacterString>GABRIEL THOMÉ BROCHADO /</gco:CharacterString>
32        <gmd:individualName>
33          <gmd:organisationName>
34            <gco:CharacterString>1º Centro de

```

Figura 18 – Tela de sucesso da submissão do novo formulário com o XML criado.

## 4.2 Fluxo secundário de cadastro de metadados

O fluxo secundário de cadastro de metadados possui o mesmo objetivo que o botão de importação nova presente no formulário atual (Figura 8), que é importar um XML de metadados já construído pelo cadastrador no BDGEx. Dessa forma, o novo fluxo além de realizar a importação do produto geoespacial juntamente com o XML pré-preenchido, ele realiza a validação do XML de metadados, mostrando ao usuário ao final do fluxo o

resultado da validação do XML, destacando os problemas estruturais e de preenchimento de valores de metadados inválidos. Dessa forma, o usuário consegue realizar os ajustes e submeter novamente os metadados atualizados, sobrescrevendo o arquivo atual no sistema, salvando todo o histórico de edições no sistema de administração.

Podemos visualizar nas figuras 19, 20 e 21 o funcionamento desse fluxo no *front-end*.

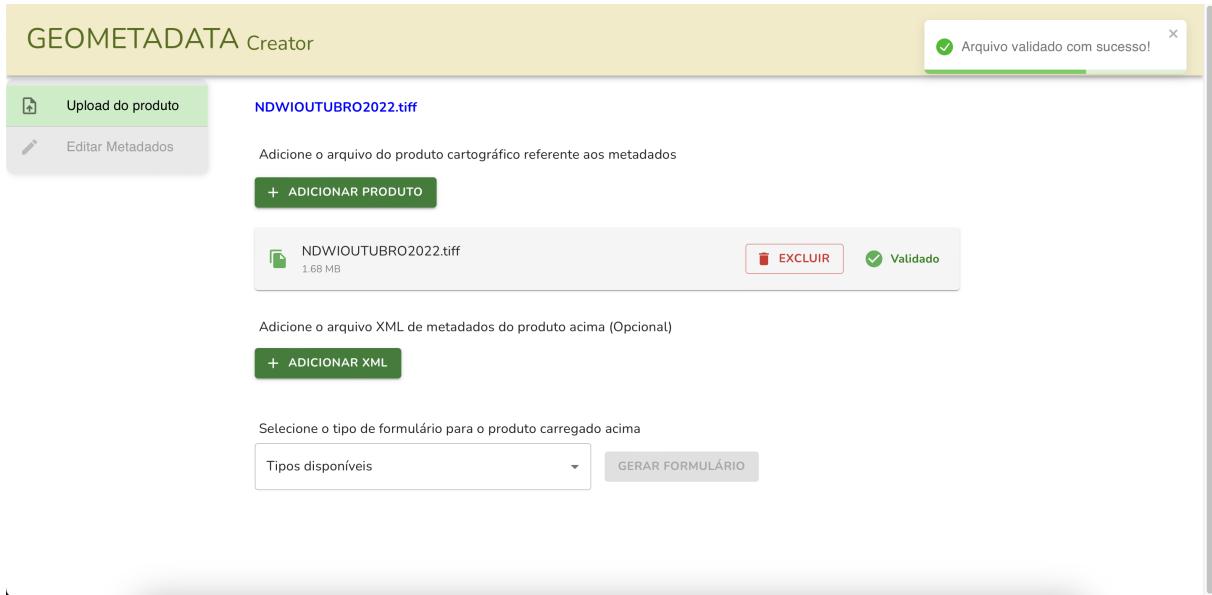


Figura 19 – Interface de upload dos produtos e XML de metadados.

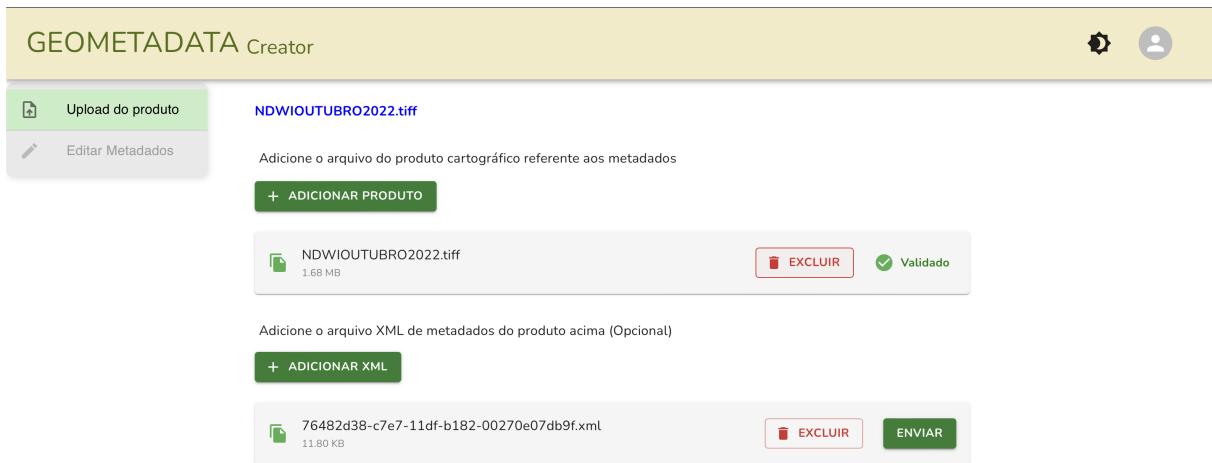


Figura 20 – XML carregado com indicação do botão de envio ao *back-end* para validação

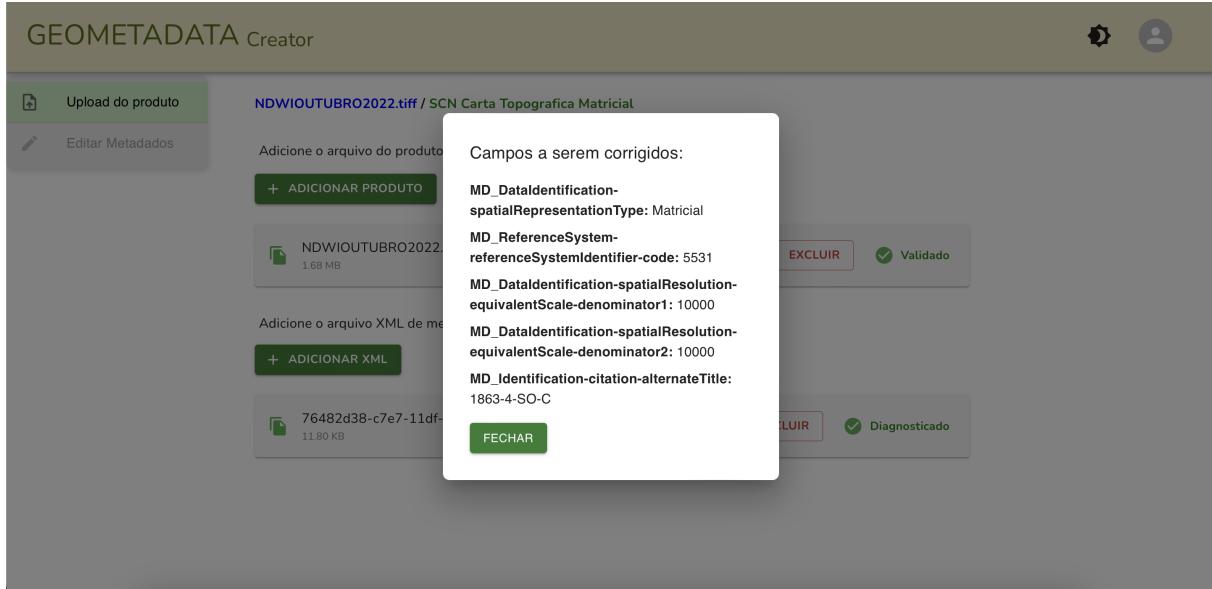


Figura 21 – Resposta da validação do XML de metadados submetido pelo cadastrador.

O ganho dessa mudança, frente ao que foi implementado no formulário atual, não é só a possibilidade de validação do XML na submissão, mas também a realização dessas submissões via requisições as APIs do *back-end*, assim os cadastradores de metadados do CGeo conseguem realizar submissões de metadados criados por eles de forma direta ao servidor, otimizando assim o tempo que eles tomam ao utilizar o formulário atual.

## 4.3 APIs da aplicação

Nessa seção é desenvolvido em detalhes as respostas das APIs para o funcionamento do formulário de cadastro de metadados e explicações de como suas estruturações permitem uma interoperabilidade que não está presente no formulário atual.

### 4.3.1 API de upload

O envio dos produtos geoespaciais é feito por meio do endpoint "geoproduct/", podendo ser visualizado suas entradas e saídas da comunicação com o *front-end* pela Figura 22, que representa uma submissão válida, e 23, que representa a submissão de um arquivo inválido. Na primeira figura é enviado um arquivo geoespaciais, com as extensões possíveis definidas pelo *back-end*.

geoprodut

POST /geoprodut/

Parameters

No parameters

Request body

multipart/form-data

geodata\_file  
string(\$binary)  
Choose File autazes1.tif  
 Send empty value

Execute Clear

Responses

Code	Details
201	Response body <pre>{ "file_id": 5, "product_types": [ { "id": 1, "name": "SCN Carta Topográfica Matricial" }, { "id": 2, "name": "SCN Carta Topográfica Matricial (sumarizada)" } ], "file_fields": { "MD_DataIdentification-extent-geographic_Element1-northBoundLatitude": -3.566277541, "MD_DataIdentification-extent-geographic_Element1-southBoundLatitude": -3.607695379, "MD_DataIdentification-extent-geographic_Element1-eastBoundLongitude": -59.09309490459226, "MD_DataIdentification-extent-geographic_Element1-westBoundLongitude": -59.15618793140774, "MD_DataIdentification-spatialRepresentationType": "Matricial", "MD_ReferenceSystem-referenceSystemIdentifier-code": 4326, "MD_Distribution-distributionFormat-name": "Tiff", "MD_DataIdentification-spatialResolution-equivalentScale-denominator1": 1000, "MD_DataIdentification-spatialResolution-equivalentScale-denominator2": 1000, "MD_Identification-citation-alternateTitle1": "", "MD_Identification-citation-alternateTitle2": "", "MD_Identification-citation-alternateTitle3": "" } }</pre> <p> </p>

Figura 22 – Representação no Swagger da requisição do envio de um arquivo válido.

The screenshot shows the Swagger UI interface for a POST request to the endpoint `/geoprodut`. The request body is set to `multipart/form-data` and contains a file named `RelatorioExecucao.xlsx` under the parameter `geodata_file`. The response section shows a 400 Bad Request error with the message `"Arquivo não é do tipo geoespacial"`.

Figura 23 – Representação no Swagger da resposta do envio de um arquivo que não é geoespacial e, portanto, é considerado inválido.

#### 4.3.2 API de interface do cadastro geral

A API de interface do cadastro geral deve intermediar a comunicação com o servidor do cadastro geral. Nesse contexto, não foi possível realizar as autenticações no servidor, sendo adaptada a dados locais. A figura 24 representa a visão geral dos *endpoints* da API, mostrando que as requisições são todas do tipo GET, onde são enviados apenas requisições ao servidor de retornar dados pré-definidos. Sendo assim, a figura 25 representa o retorno da requisição de um dos *endpoints* da API.

**cadastro\_geral**

^

<b>GET</b>	/cadastro_geral/MD_DataIdentification-extent-verticalExtent-verticalDatum		
<b>GET</b>	/cadastro_geral/MD_Identification-citation-collectiveTitle		
<b>GET</b>	/cadastro_geral/MD_Metadata-contact-individualName		
<b>GET</b>	/cadastro_geral/MD_Metadata-contact-organisationName		

Figura 24 – Visão geral das APIs do cadastro geral.

**cadastro\_geral**

^

**GET** /cadastro\_geral/MD\_DataIdentification-extent-verticalExtent-verticalDatum

Can be accessed read only.

**Parameters**

No parameters Cancel

Execute Clear

**Responses**

Server response

Code	Details
200	Response body <pre>{   "1": {     "MD_DataIdentification-extent-verticalExtent-verticalDatum": "Datum de Imbituba - SC"   } }</pre> <span style="float: right;"> </span>

Figura 25 – Visão de entrada e saída do endpoint de Datum Vertical.

## 4.4 Validação de metadados

A API de validação e construção de metadados verifica se há campos ausentes, se existem campos não previstos no tipo de produto selecionado e se os campos relacionados ao arquivo estão em conformidade com as informações extraídas do próprio arquivo. Após essa verificação, os campos são inseridos no template XML e anexados ao objeto correspondente do arquivo, garantindo a integridade e consistência dos metadados. A figura 26 apresenta a entrada com todos os campos faltando e saída da requisição feita ao *endpoint*, onde a resposta apresenta a razão do erro, também com os campos que faltam serem recebidos na estrutura do XML para aquele tipo de produto. Dessa forma, a API garante a integridade dos arquivos XML de metadados criados a partir do formulário de cadastro e de forma sistemática.

The screenshot shows a POST request to the endpoint `/geoproduct/{id}/build_metadata/`. The request body contains the JSON object `{ "metadata_fields": [], "product_type": 1 }`. The response code is 400 (Bad Request), and the response body is a JSON object with the error message: `{"error": "There are fields required for the product_type selected that are missing.", "missing_fields": ["MD_Metadata-characterSet", "MD_Identification-citation-edition-date-dateType", "MD_Identification-citation-alternateTitle3", "MD_Metadata-fileIdentifier", "MD_DataIdentification-spatialResolution-equivalentScale-denominator2", "MD_DataIdentification-extent-geographicElement1-northBoundLatitude", "MD_Keywords-type", "MD_Identification-pointOfContact-organisationName", "MD_DataIdentification-spatialResolution-distance", "MD_Identification-citation-editionDate", "MD_Distributor-distributorContact-organisationName", "MD_Distributor-distributorContact-contactInfo-address-deliveryPoint", "MD_Identification-citation-title", "MD_Metadata-contact-organisationName", "MD_ReferenceSystem-referenceSystemIdentifier-code", "MD_Distributor-distributorContact-role", "MD_Identification-credit", "MD_DataIdentification-resourceMaintenance-maintenanceAndUpdateFrequency", "MD_Metadata-contact-individualName", "MD_DataIdentification-spatialRepresentationType"]}`.

Figura 26 – Entrada e saída do *endpoint* de construção do XML de metadados.

Pode ser visualizado também na figura 27 a entrada com valores inválidos e a saída do *endpoint* de validação dos campos, apresentando um erro em sua saída, assim como os

erros de cada um dos campos, para que o cadastrador realize os ajustes necessários nos dados de cada campo.

**POST** /geoprodut/{id}/build\_metadata/

Pass the metadata fields with its respective values: - Validate the them against the file's information; - Validate the contact information agaist the logged databases; - Create the XML metadata file;

**Parameters**

Name	Description
<b>id</b> <small>required</small>	A unique integer value identifying this geospatial resource.
integer <i>(path)</i>	1

**Request body** required

application/json

```
{
  "metadata_fields": [
    {"label": "MD_Metadata-fileIdentifier", "value": "123"},
    {"label": "MD_Metadata-contact-individualName", "value": "123"},
    {"label": "MD_Metadata-contact-positionName", "value": "123"},
    {"label": "MD_Metadata-contact-organisationName", "value": "123"},
    {
      "label": "MD_Metadata-contact-contactInfo-onlineResource-linkage",
      "value": "123"
    },
    {"label": "MD_Metadata-contact-role", "value": "123"},
    {"label": "MD_Metadata-dateStamp", "value": "123"},
    {"label": "MD_Metadata-language", "value": "123"},
    {"label": "MD_Metadata-characterSet", "value": "123"},
    {"label": "MD_Metadata-metadatastandardName", "value": "123"},
    {"label": "MD_Metadata-metadatastandardVersion", "value": "123"},
    {"label": "MD_Metadata-hierarchyLevel", "value": "123"},
    {
      "label": "MD_Metadata-metadatamaintenance-maintenanceAndUpdateFrequency",
      "value": "123"
    }
  ]
}
```

**Server response**

Code	Details
400 <i>Undocumented</i>	Error: Bad Request

**Response body**

```
{
  "error": "There were errors with fields when comparing the values found in file.",
  "mismatched_file_fields": {
    "MD_DataIdentification-extent-geographicElement1-northBoundLatitude": "-3.566277541 != 1.0",
    "MD_DataIdentification-extent-geographicElement1-southBoundLatitude": "-3.607695379 != 1.0",
    "MD_DataIdentification-extent-geographicElement1-eastBoundLongitude": ".59.09309490459226 != 1.0",
    "MD_DataIdentification-extent-geographicElement1-westBoundLongitude": "-59.15618793140774 != 1.0",
    "MD_DataIdentification-spatialRepresentationType": "Matricial != Vetorial",
    "MD_ReferenceSystem-referenceSystemIdentifier-code": "4326 != 2343",
    "MD_Distribution-distributionFormat-name": "GTiff != Not IT",
    "MD_DataIdentification-spatialResolution-equivalentScale-denominator1": "1000 != 1",
    "MD_DataIdentification-spatialResolution-equivalentScale-denominator2": "1000 != 1",
    "MD_Identification-citation-alternateTitle2": " != not_it",
    "MD_Identification-citation-alternateTitle3": " != not_it"
  }
}
```

**Download**

Figura 27 – Entrada e saída do *endpoint* de validação dos valores do XML de metadados.

## 4.5 Sistema de administração

Um dos problemas mapeados do atual formulário do BDGEx é a forma de gerir as funcionalidades de cada um dos tipos de produtos e as regras de preenchimento de metadados. Várias dessas atividades eram descentralizadas, com parte implementada no *front-end*, parte no *back-end*, parte no sistema de importação de dados e parte no próprio banco de dados. Isso dificulta aos desenvolvedores a identificação do caminho para a resolução dos problemas e atualização de regras de preenchimento, bem como a dificuldade de realizar alterações na validação de algum tipo específico.

Com o sistema do módulo de administração do Django implementado (Figura 28), é possível realizar a gestão dos metadados cadastrados pelo formulário pelo módulo "Geospatial resources" (Figura 29). Neste módulo é possível acessar os arquivos XML de metadados e geoespaciais associados a cada produto cadastrado, bem como a indicação se o produto está ou não publicado em servidor CSW. Também é possível realizar três ações em massa nos arquivos armazenados: deletar os produtos selecionados, publicação dos produtos em servidor CSW ou retirada dos produtos do servidor CSW.

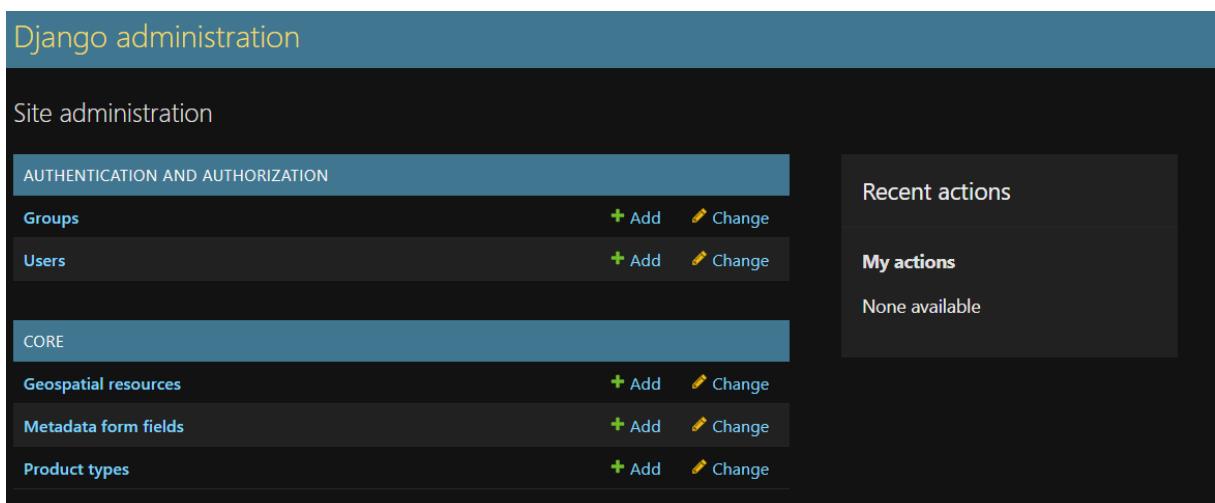


Figura 28 – Interface do *Django Admin*.

The screenshot shows the Django admin interface for the 'Geospatial resources' model. At the top, a green banner displays the message: "The geospatial resource 'GeospatialResource object (1)' was added successfully." Below this, the main area shows a table with one row of data. The columns are: METADATA ID, TITLE, HAS METADATA, HAS GEODATA, HAS PDF, and PUBLISHED ON. The single row contains the value 'f54f7cdf-d44e-472e-ac1d-8a7fb079d4bf' for METADATA ID, 'teste - porto alegre' for TITLE, and green checkmarks for HAS METADATA, HAS GEODATA, and HAS PDF, while PUBLISHED ON has a red cross. To the left of the table is a search bar and a 'Go' button. Above the table, there's an 'Action:' dropdown set to '-----' and a 'Go' button with '0 of 1 selected'. On the far right, there's a 'FILTER' sidebar with several dropdown menus and checkboxes for filtering by published status on csw, Geospatial metadata XML, and Geospatial data file.

Figura 29 – Interface de gerenciamento do módulo *Geospatial resources*.

Também teremos dois módulos que estão conectados entre si, que são os módulos "*product types*" e "*metadata form fields*". O módulo "*product types*"(Figura 30) define quais são os tipos de produtos a serem aceitos pelo formulário, o template que servirá como padrão para a construção do XML de metadados de cada produto cadastrado a partir das informações enviadas pelo preenchimento do usuário, bem como quais são os campos do formulário que irão compor o cadastro de metadados desse tipo de produto. Para a definição desses campos, é utilizado o módulo "*metadata form fields*"(Figura 31, que armazena as informações de todos os campos de todos os tipos de produto, os quais são selecionados de forma dinâmica no módulo "*product type*" para cada tipo de produto (como vemos na Figura 30 no campo chamado "*metadata fields*").

Figura 30 – Interface de gerenciamento do módulo *product types*.

Figura 31 – Interface de gerenciamento do módulo *metadata form fields*.

Por fim, é disponibilizado uma interface para gerenciamento dos usuários (Figura 32), facilitando os ajustes de acesso para cada um deles.

Action:	USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
<input type="checkbox"/>	arthur				<input checked="" type="checkbox"/>
<input type="checkbox"/>	arthurcc				<input checked="" type="checkbox"/>
<input type="checkbox"/>	arthurcosta				<input checked="" type="checkbox"/>

3 users

Figura 32 – Interface de gerenciamento dos usuários da aplicação.

#### 4.5.1 Publicação no servidor CSW

No formulário atual, temos a publicação dos produtos nos servidores CSW é feita por meio do PostgresSQL e qualquer alteração em seus parâmetros são feitos diretamente no banco de dados e não na aplicação, assim a publicação dos produtos fica estreitamente atrelado ao modelo do banco de dados. Sendo assim, a publicação ou retirada no servidor CSW (Figura 33) foi introduzida como uma ação no sistema de administração, sendo configurada diretamente no *back-end*. Essa mudança facilita o gerenciamento dos produtos no sistema de administração por parte dos cadastrado de metadados, consolidando essa publicação junto com as outras ações que podem ser realizadas aos produtos.

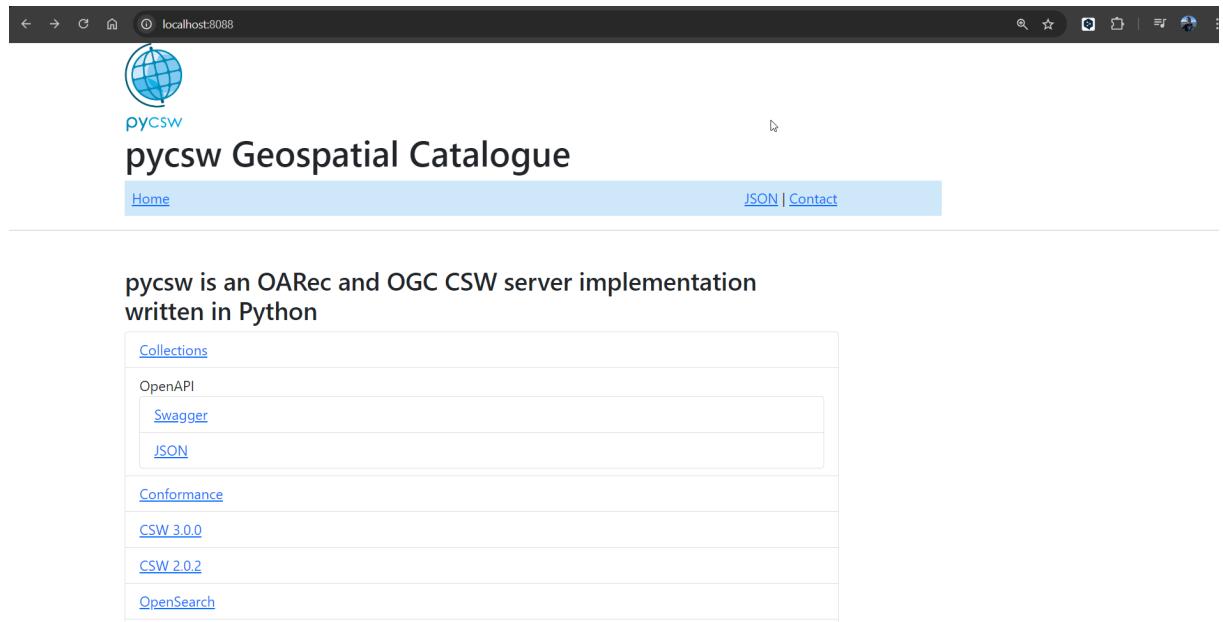


Figura 33 – Servidor CSW conectado a aplicação o que permite a publicação dos produtos.

## 5 CONCLUSÃO

### 5.1 Considerações finais

Neste trabalho, foi desenvolvida uma nova aplicação web para o cadastro e validação de metadados dos produtos cartográficos da Diretoria de Serviço Geográfico do Exército Brasileiro (DSG), com o objetivo de modernizar e otimizar o processo existente no BDGEx. A solução buscou superar limitações identificadas na estrutura anterior, especialmente no que tange à integração com sistemas externos, à automação de tarefas manuais e à conformidade com normas técnicas. O projeto focou na criação de uma arquitetura desacoplada, na automação do preenchimento de dados geoespaciais, na validação eficiente de metadados e na geração automática de arquivos XML padronizados.

No que se refere à interface do usuário, a aplicação implementou um fluxo de trabalho mais intuitivo e eficiente. A separação entre o cliente e o servidor permitiu um preenchimento mais flexível e descentralizado, algo que era uma limitação no sistema anterior. A nova abordagem eliminou a dependência de operações rígidas e possibilitou uma experiência mais responsiva, além de preparar o sistema para futuras expansões e adaptações a outros produtos geoespaciais.

A automação no preenchimento dos metadados, diretamente a partir dos arquivos geoespaciais carregados, foi um dos avanços mais relevantes. Antes, operadores tinham que inserir manualmente informações como INOM e MI, o que gerava retrabalho e margem para erros. Agora, o sistema faz essa extração automaticamente, o que torna o processo mais confiável e eficiente, além de reduzir significativamente o tempo de cadastro.

A validação dos metadados também foi aprimorada, com a implementação de um sistema automatizado que verifica a conformidade dos dados com as normas da ET-PCDG e outras regras definidas. Isso evita inconsistências e agiliza o processo de validação, que no sistema anterior era mais dependente de verificações manuais, além de fornecer feedback imediato sobre o preenchimento.

Por fim, a aplicação introduziu a geração automática do XML de metadados, o que representa um grande salto em relação ao sistema antigo. Esse processo, agora totalmente automatizado, elimina erros manuais e garante que os arquivos gerados estejam em conformidade com padrões internacionais, facilitando a interoperabilidade com outros sistemas e plataformas cartográficas.

## 5.2 Trabalhos futuros

A aplicação desenvolvida abre espaço para diversas melhorias e expansões. Uma das principais evoluções futuras seria a inclusão de outros tipos de produtos cartográficos além das cartas topográficas matriciais, permitindo que o sistema abranja um espectro maior de produtos geoespaciais gerenciados pela DSG. Isso incluiria a adaptação e a configuração dos campos para novos tipos de produtos, que irão ser refletidos pela interface do formulário.

Outra área de evolução é a integração mais profunda com outros sistemas do Exército e órgãos governamentais. Isso incluiria a criação de APIs mais robustas para interligação com bancos de dados externos e sistemas de gestão de geoinformação, além de permitir a comunicação direta com o Cadastro Geral e automatizar ainda mais o preenchimento dos metadados.

Além disso, há oportunidades para melhorar a interface do usuário, traduzindo componentes para o português e adaptando a padrões mais próximos aos outros sistemas do BDGEx, e possibilitar uma validação de dados mais precisa na própria interface do formulário. Por fim, outros sistemas podem utilizar as APIs criadas para efetuar o carregamento em lote de produtos geoespaciais e metadados, permitindo que múltiplos arquivos sejam carregados e validados simultaneamente, o que traria um ganho de eficiência significativo para grandes volumes de dados.

## REFERÊNCIAS

- BRAY, T.; PAOLI, J.; SPERBERG-MCQUEEN, C. M.; MALER, E.; YERGEAU, F. Extensible markup language (XML). *World Wide Web Journal*, v. 2, n. 4, p. 27–66, 1997. Publisher: Citeseer.
- CONCAR. *Perfil de Metadados Geoespaciais do Brasil*. [S.l.]: Comissão Nacional de Cartografia/Ministério do Planejamento, 2020.
- Django Software Foundation. *Django*. 2019. Disponível em: <[https://django-project.com](https://.djangoproject.com)>.
- DSG. *Especificação técnica para produtos de conjuntos de dados geoespaciais*. 2<sup>a</sup> edição. ed. Diretoria de Serviço Geográfico/Ministério da Defesa, 2016. Disponível em: <[http://www.geoportal.eb.mil.br/portal/images/PDF/ET\\_PCDG\\_2016\\_2aEdicao\\_Aprovada\\_Publicada\\_BE\\_7\\_16.pdf](http://www.geoportal.eb.mil.br/portal/images/PDF/ET_PCDG_2016_2aEdicao_Aprovada_Publicada_BE_7_16.pdf)>.
- DSG. *Nota técnica para cadastro de metadados*. 0.6.6. ed. [S.l.]: Diretoria de Serviço Geográfico/Ministério da Defesa, 2020.
- FIELDING, R. T. Architectural styles and the design of network-based software architectures. *University of California Journal*, 2000.
- GOLDBERG, K. H. *XML-Guia prático visual*. [S.l.]: Rio de Janeiro: Alta Books, 2009.
- ISO. *ISO 19115-1:2014 Geographic information - Metadata*. [S.l.]: International Standards Organization, 2014.
- ISO. *ISO 19139-1:2019 Geographic information - XML schema implementation*. [S.l.]: International Standards Organization, 2019.
- META. *React*. 2024. Disponível em: <<https://react.dev/>>.
- NEBERT, D.; VOGES, U.; BIGAGLI, L. Implementation Standard, *OGC® Catalogue Services 3.0 - General Model*. Open Geospatial Consortium, 2016. Disponível em: <<https://docs.ogc.org/is/12-168r6/12-168r6.html>>.
- OGC. *OGC API - Records - Part 1: Core (Draft)*. Open Geospatial Consortium, 2024. Disponível em: <<https://docs.ogc.org/DRAFTS/20-004.html>>.
- VAINIKKA, J. *Full-stack web development using Django REST framework and React*. Tese (Bachelor's Thesis), 2018. Publisher: Metropolia Ammattikorkeakoulu.
- XAVIER, E.; MEYER, W.; LUNARDI, O. *Banco de dados geográficos do Exército Brasileiro: arquitetura e resultados*. [S.l.: s.n.], 2014.

## ANEXO A – DICIONÁRIO DE DADOS

- *Geospatial resources*

A tabela *Geospatial resources* representa a tabela que armazenará os metadados e arquivos resultante do cadastro do formulário, e para isso ela possui os seguintes campos:

- **metadata\_id:**

- Tipo: UUIDField
- Descrição: Um identificador único gerado automaticamente para cada instância do modelo.
- Padrão: `uuid.uuid4`
- Editável: `False`

- **title:**

- Tipo: CharField
- Descrição: O título do recurso geoespacial.

- **metadata\_file:**

- Tipo: FileField
- Descrição: Arquivo XML contendo metadados geoespaciais.
- Local de upload: "repository"

- **geodata\_file:**

- Tipo: FileField
- Descrição: Arquivo de dados geoespaciais.
- Local de upload: "repository"

- **pdf\_file:**

- Tipo: FileField
- Descrição: Arquivo PDF para impressão.
- Local de upload: "repository"

- **published\_on\_csw:**

- Tipo: BooleanField

- Descrição: Indica se o recurso está publicado no catálogo de serviços web (CSW).
- Valor padrão: `False`

- **history:**

- Tipo: `HistoricalRecords`
- Descrição: Mantém o histórico de alterações em cada campo, para um determinado `metadata_id`.

- *Metadata Form Field*

A tabela *Metadata Form Field* consolida as descrições de todos os campos que farão parte do XML resultante do cadastro de um determinado produto no formulário. É a tabela que irá gerar as informações necessárias para que o *front-end* renderize corretamente cada um dos campos. A criação dessa tabela facilita futuras mudanças na estrutura do formulário, visando possibilitar a gestão dos campos por meio do *Django Admin*, removendo a necessidade que um administrador mais técnico tenha que realizar modificações diretamente no código. Ela possui os seguintes campos:

- **id:**

- Tipo: `BigAutoField`
- Descrição: Identificação do campo gerada automaticamente após sua adição.

- **label:**

- Tipo: `CharField`
- Descrição: O nome do campo do formulário.

- **iso\_xml\_path:**

- Tipo: `CharField`
- Descrição: O caminho descrito na ISO para um campo no arquivo XML de metadados.

- **field\_type:**

- Tipo: `CharField`
- Descrição: O tipo do campo de formulário.
- Valores possíveis: lista, caixa suspensa, data e texto.

- **is\_static:**

- Tipo: BooleanField
  - Descrição: Indica se o campo é estático.
  - Padrão: `False`
  - **possible\_values:**
    - Tipo: TextField
    - Descrição: Valores possíveis para o campo de formulário.
  - **default\_value:**
    - Tipo: CharField
    - Descrição: Valor padrão do campo de formulário.
  - **comments:**
    - Tipo: TextField
    - Descrição: Comentários adicionais sobre o campo de formulário.
  - **old\_path:**
    - Tipo: TextField
    - Descrição: O caminho antigo do campo no arquivo XML utilizado anteriormente pelo formulário. Esse campo é utilizado para facilitar a migração de campos para a estrutura descrita nesse modelo.
- A tabela *ProductType* define a relação entre os tipos de produto (Ex: Carta Topográfica Matricial Sumariada) e os campos presentes nos metadados desse produto. Ela possui os seguintes campos:
- **id:**
    - Tipo: BigAutoField
    - Descrição: Um identificador único gerado automaticamente para cada tipo de produto adicionado.
  - **name:**
    - Tipo: CharField
    - Descrição: O nome do tipo de produto.
  - **metadata\_fields:**
    - Tipo: ManyToManyField

- Descrição: É um campo relacional que armazena todos os campos que são relacionados ao tipo de produto em questão.

A tabela *IndexMap* define a relação entre os INOM e os MIs para diferentes escalas. Ela possui os seguintes campos:

- **id:**
  - Tipo: BigAutoField
  - Descrição: Um identificador único gerado automaticamente para cada relacionamento de MI e INOM.
- **INOM:**
  - Tipo: CharField
  - Descrição: Determinado INOM.
- **MI:**
  - Tipo: CharField
  - Descrição: Define o MI relacionado ao INOM.
- **scale\_denominator:**
  - Tipo: CharField
  - Descrição: O campo que define em qual escala está a relação de MI e INOM.