

FUNDAÇÃO UNIVERSIDADE FEDERAL DE RONDÔNIA
NÚCLEO DE TECNOLOGIA
DEPARTAMENTO ACADÊMICO DE CIÊNCIA DA COMPUTAÇÃO

GABRIEL MORAIS RUSSO

**DESENVOLVIMENTO DE UMA BIBLIOTECA PYTHON PARA BUSCA E
PROCESSAMENTO DE DADOS DO SATÉLITE SINO-BRASILEIRO CBERS-04A**

Porto Velho – RO
2023

FUNDAÇÃO UNIVERSIDADE FEDERAL DE RONDÔNIA
NÚCLEO DE TECNOLOGIA
DEPARTAMENTO ACADÊMICO DE CIÊNCIA DA COMPUTAÇÃO

**DESENVOLVIMENTO DE UMA BIBLIOTECA PYTHON PARA BUSCA E
PROCESSAMENTO DE DADOS DO SATÉLITE SINO-BRASILEIRO CBERS-04A**

GABRIEL MORAIS RUSSO

Monografia submetida à Fundação Universidade Federal
de Rondônia como requisito para a conclusão no curso de
Bacharelado em Ciência da Computação.

Orientador: Prof. Dr. Lucas Marques da Cunha

Porto Velho - RO
2023

Catalogação da Publicação na Fonte
Fundação Universidade Federal de Rondônia - UNIR

R969d Russo, Gabriel Morais.

Desenvolvimento de uma biblioteca Python para busca e processamento de dados do Satélite Sino-Brasileiro CBERS-04A / Gabriel Morais Russo. - Porto Velho, 2023.

51 f.: il.

Orientador: Prof. Dr. Lucas Marques da Cunha.

Monografia (Graduação). Departamento de Ciência da Computação. Núcleo de Tecnologia. Fundação Universidade Federal de Rondônia.

1. CBERS-04A. 2. Sensoriamento Remoto. 3. Biblioteca Python. 4. Automação. I. Cunha, Lucas Marques da. II. Título.

Biblioteca Central

CDU 004:021



MINISTÉRIO DA EDUCAÇÃO
FUNDAÇÃO UNIVERSIDADE FEDERAL DE RONDÔNIA
DEPARTAMENTO ACADÊMICO DE CIÊNCIAS DA COMPUTAÇÃO - PORTO VELHO

ATA DE DEFESA DE CONCLUSÃO DE CURSO

As **15:00** horas do dia 02 do mês de junho de 2023, realizou-se no(a) **google meet**: <https://meet.google.com/ycrc-mkoe-zru>, a Sessão de Apresentação e Defesa do Trabalho de Conclusão de Curso intitulado “**DESENVOLVIMENTO DE UMA BIBLIOTECA PYTHON PARA ACESSO E PROCESSAMENTO DE DADOS DOS SATÉLITES SINO-BRASILEIROS CBERS-04A**”, apresentado pelo(a) acadêmico(a) **GABRIEL MORAIS RUSSO** do Departamento Acadêmico de Ciências da Computação - Porto Velho. O trabalho foi julgado **APROVADO** pelos docentes Dr. Lucas Marques da Cunha (Orientador), Dr. Thiago Amaral Guarnieri (Membro) e Dr. Jonathan da Silva Ramos (Membro), todos do DEPARTAMENTO ACADÊMICO DE CIÊNCIAS DA COMPUTAÇÃO - PORTO VELHO, da Fundação Universidade Federal de Rondônia - UNIR, *Campus de Porto Velho*, com nota **10,0** como requisito parcial para obtenção do título de BACHARELADO em CIÊNCIA DA COMPUTAÇÃO, sem ressalvas para correções a serem feitas pelo aluno antes de submeter a versão definitiva para o fechamento da nota na disciplina: Trabalho de Conclusão de Curso.

Porto Velho, 02 de junho de 2023.

APROVADO pela Banca Examinadora constituída pelos seguintes professores:

ORIENTADOR: Dr. Lucas Marques da Cunha

Aprovado (x) Reprovado ()

AVALIADOR 1: Dr. Thiago Amaral Guarnieri

Aprovado (x) Reprovado ()

AVALIADOR 2: Dr. Jonathan da Silva Ramos

Aprovado (x) Reprovado ()

Reaberta a sessão pública o orientador proclamou os resultados e encerrou a sessão, da qual foi lavrada a presente ata que vai por mim assinada.

Dr. Lucas Marques da Cunha

Professor orientador



Documento assinado eletronicamente por **LUCAS MARQUES DA CUNHA, Docente**, em 02/06/2023, às 16:53, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



Documento assinado eletronicamente por **THIAGO AMARAL GUARNIERI, Docente**, em 02/06/2023, às 16:54, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



Documento assinado eletronicamente por **JONATHAN DA SILVA RAMOS, Membro da Comissão**, em 02/06/2023, às 17:58, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



A autenticidade deste documento pode ser conferida no site http://sei.unir.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1366328** e o código CRC **0D58164A**.

AGRADECIMENTOS

Aos meus pais, pelo apoio e incentivo.

Aos amigos do CENSIPAM, por todo os ensinamentos passados durante o estágio.

Aos meus amigos da SEDAM, pelo incentivo a ser um profissional de ponta.

Aos professores, pela ajuda e troca de conhecimentos.

Ao meu orientador pelos conhecimentos e dicas valiosas.

Dedico esta pesquisa ao Henrique Bernini, que me incentivou a não abandonar a área de sensoriamento remoto, na qual me identifico tanto.

A todos os amigos que fiz durante a faculdade, pelos momentos difíceis que superamos, pelas risadas e histórias inesquecíveis em que passamos. Em especial a: Bruno Bordin, Filipi Ricardo, Marcus Vinícius e Ana Isabel.

*“O sábio é aquele que procura a sabedoria, não
aquele que acredita já tê-la encontrado.”*

Sêneca

RESUMO

A utilização de imagens de satélite é amplamente utilizada em diversas áreas de estudo, pois, em conjunto com técnicas de sensoriamento remoto, permite que se obtenha informações sobre um objeto na superfície sem tocá-los fisicamente. Em 2019, o Brasil lançou o sexto satélite da família CBERS (China-Brazil Earth Resources Satellite), o CBERS-04A, disponibilizando gratuitamente imagens para o estudo da superfície terrestre. Logo, é necessário ferramentas que façam a busca e o processamento destas imagens. Foi observado que apenas haviam duas ferramentas disponíveis e, nenhuma delas, oferecia ao usuário a possibilidade de automatizar as tarefas com linguagens de programação. Portanto, o objetivo deste trabalho é criar uma biblioteca para a linguagem Python, que faça o consumo e processamento destes dados, aproveitando-se dos recursos da linguagem. A metodologia aplicada foi o desenvolvimento de uma classe com base no padrão de projeto estrutural *Facade* para abstrair a estrutura do *Spatial Temporal Asset Catalog* (STAC), e permitir a busca na base de dados sem preocupar-se com o entendimento da especificação STAC. Além disso, foi desenvolvido algoritmos de processamento de imagens de satélite para criar composições coloridas e realizar o *pansharpening* das bandas do CBERS-04A. Os resultados mostram os métodos da classe *Facade* chamada de *Cbers4aAPI*, apresentando os parâmetros necessários para realizar: busca, download e conversão para *GeoDataFrame*, além de exibir os resultados do processamento da composição colorida e *pansharpening*. O trabalho conclui que o software criado apresenta ser eficiente e intuitivo, realizando as tarefas mais comuns da área e, além disso, permitindo a automação de tarefas utilizando recursos da linguagem e integração com outras bibliotecas de terceiros.

Palavras-chave: CBERS-04A, Python, Sensoriamento Remoto, STAC, Automação

ABSTRACT

The use of satellite images is widely employed in various areas of study because, in conjunction with remote sensing techniques, it allows for obtaining information about an object on the surface without physically touching it. In 2019, Brazil launched the sixth satellite of the CBERS (China-Brazil Earth Resources Satellite) family, the CBERS-04A, providing free images for the study of the Earth's surface. Consequently, tools are needed to search for and process these images. It was observed that there were only two available tools, and neither of them offered users the possibility to automate tasks using programming languages. Therefore, the objective of this work is to create a library for the Python language that consumes and processes this data, taking advantage of the language's resources. The methodology applied was the development of a class based on the Facade structural design pattern to abstract the structure of the Spatial Temporal Asset Catalog (STAC) and enable database searches without worrying about understanding the STAC specification. Additionally, satellite image processing algorithms were developed to create color compositions and perform pansharpening of the CBERS-04A bands. The results demonstrate the methods of the Facade class called Cbers4aAPI, presenting the necessary parameters to perform search, download, and conversion to GeoDataFrame, as well as displaying the results of color composition and pansharpening processing. The work concludes that the created software is efficient and intuitive, performing the most common tasks in the field and, moreover, allowing for task automation using language resources and integration with other third-party libraries.

Keywords: CBERS-04A, Python, Remote Sensing, STAC, Automation

LISTA DE FIGURAS

Figura 1 - Foto aérea de Paris, 1858.....	14
Figura 2 - Imagem de satélite de Moscou, 1970.....	15
Figura 3 - Componentes de um sistema de sensoriamento remoto.....	16
Figura 4 - Radiação eletromagnética.....	17
Figura 5 - Espectro Eletromagnético.....	18
Figura 6 - As 7 bandas do satélite Landsat 5, contendo os seus respectivos intervalos de comprimento de onda e nomes.....	19
Figura 7 - Diferentes resoluções espaciais de uma imagem de satélite sobre uma residência.....	20
Figura 8 - Classificação dos sensores com relação a sua resolução espectral.....	21
Figura 9 - Satélite CBERS 04A.....	22
Figura 10 - Imagem do CBERS-04A da cidade de Porto Velho utilizando a composição falsa cor.	24
Figura 11 - Fluxograma do sistema proposto.....	26
Figura 12 - Diagrama UML dos componentes STAC.....	27
Figura 13 - Estrutura de classes utilizadas no software.....	28
Figura 14 - Classe Facade - Cbers4aAPI.....	30
Figura 15 - Código exemplo para requisitar um Item utilizando a biblioteca <i>requests</i>	31
Figura 16 - Resultado da requisição em sua forma recolhida.....	32
Figura 17 - Exemplo do parâmetro <i>location</i> na função <i>query</i> , aplicando a tipagem com a biblioteca <i>Typing</i>	33
Figura 18 - Exemplo de um GeoDataFrame.....	34
Figura 19 - Informações de um Item STAC no GeoDataFrame, incluindo o atributo <i>geometry</i>	34
Figura 20 - Plote da geometria ilustrada na figura 20 sobre a geometria do Estado de Rondônia, ilustrando sua localização e extensão.....	34
Figura 21 - Função <i>pansharpening</i> da biblioteca GDAL e seus 17 parâmetros de ajustes do algoritmo de <i>pansharpening</i>	36
Figura 22 - Fluxograma do algoritmo de <i>pansharpening</i>	37
Figura 23 - Exemplo de bloco de código que busca uma imagem de satélite.....	40
Figura 24 - Buscando uma banda pelo seu identificador.....	40
Figura 25 - Realizando uma busca e baixando as bandas.....	41
Figura 26 - Convertendo os Itens no formato JSON para dados tabulares, isto é, o GeoDataFrame.....	42
Figura 27 - Chamada para a função de composição colorida.....	42
Figura 28 - Cidade de Vilhena na composição cor verdadeira, resultado do processamento da função <i>rgbn_composite</i>	43
Figura 29 - Chamada para a função de <i>pansharpening</i> com os parâmetros preenchidos.....	43
Figura 30 - Resultado do <i>pansharpening</i>	44

LISTA DE GRÁFICOS

Gráfico 1 - Gráfico em barra com o número de downloads diários da biblioteca cbers4asat no PyPi num intervalo de 60 dias.....	45
-------------------------------------------------------------------------------------------------------------------------------	----

LISTA DE QUADROS

Quadro 1 - Características da câmera WFI, MUX e WPM.....	23
----------------------------------------------------------	----

SUMÁRIO

1	INTRODUÇÃO.....	11
1.1	JUSTIFICATIVA.....	12
1.2	OBJETIVO.....	12
1.3	GERAL.....	12
1.4	ESPECÍFICOS.....	13
2	FUNDAMENTAÇÃO TEÓRICA.....	14
2.1	SENSORIAMENTO REMOTO: CONTEXTO HISTÓRICO E DEFINIÇÕES.....	14
2.2	FORMAÇÃO DE UMA IMAGEM DE SATÉLITE.....	16
2.2.1	Radiação eletromagnética.....	16
2.2.2	Sistema sensor.....	18
2.3	RESOLUÇÕES DA IMAGEM: ESPACIAL, ESPECTRAL E TEMPORAL.....	19
2.4	O SATÉLITES IMAGEADOR CBERS-04A.....	21
2.5	PROCESSAMENTO DE IMAGEM DE SATÉLITE.....	23
2.5.1	Composição colorida.....	23
2.5.2	<i>Pansharpening</i>.....	24
2.6	SPATIO TEMPORAL ASSET CATALOG.....	25
3	METODOLOGIA.....	26
3.1	DEFINIÇÃO DAS TECNOLOGIAS UTILIZADAS.....	26
3.2	ESCOLHA DA BASE DE DADOS.....	27
3.3	MODELAGEM DOS OBJETOS STAC.....	27
3.4	ABSTRAINDO A COMPLEXIDADE COM <i>FACADE</i>	29
3.5	PROCESSO DE DESENVOLVIMENTO.....	31
3.6	ALGORITMOS DE PROCESSAMENTO DIGITAL DE IMAGENS.....	35
3.7	VALIDAÇÃO E TESTES.....	37
3.8	DISPONIBILIZAÇÃO DO SERVIDOR STAC E HOSPEDAGEM DO CÓDIGO.....	38
4	RESULTADOS E DISCUSSÕES.....	39
4.1	RESULTADOS DOS ALGORITMOS DE BUSCA.....	39
4.2	DOWNLOAD DE BANDAS.....	40
4.3	CONVERTER OBJETO JSON PARA GEODATAFRAME.....	41
4.4	OS ALGORITMOS DE PROCESSAMENTO DIGITAL DE IMAGENS.....	42
4.5	ESTATÍSTICAS DE USO DA BIBLIOTECA.....	44
5	CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS.....	46

6	REFERÊNCIAS BIBLIOGRÁFICAS.....	47
7	APÊNDICES.....	50

1 INTRODUÇÃO

Pálido ponto Azul, adjetivo como foi chamada a Terra, por Carl Sagan, famoso astrônomo e escritor a partir de uma perspectiva externa, portanto, a famosa fotografia tirada da Terra em 14 de fevereiro de 1990 pela sonda *Voyager* a seis bilhões de quilômetros do planeta Terra, demonstrou a pequenez humana, o poder da humanidade e suas ambições.

O objetivo de registrar a Terra a partir da perspectiva externa ou melhor, a partir do espaço, foi alimentada e evoluída ao longo dos anos conforme o progresso tecnológico avançava, expandindo também as fronteiras das possibilidades. Neste processo, novos campos científicos foram surgindo para cumprir este propósito de alimentar a curiosidade humana.

Neste contexto, podemos citar o sensoriamento remoto, ciência que pode ser traduzida como um conjunto de técnicas e ferramentas que busca obter informações físicas, químicas e biológicas de determinados alvos sem a necessidade em tocá-los fisicamente (ELACHI, 1987). Para obter tais dados, é preciso em termos superficiais, “apenas” estudar o comportamento da luz ao atingi-los.

O surgimento desta ciência ocorreu posteriormente à segunda guerra mundial, com o aperfeiçoamento da fotografia e de técnicas de fotogrametria (ciência de realizar medições a partir de fotografias), o termo sensoriamento remoto aparece, pela primeira vez, na literatura científica nos Estados Unidos, anos 50, pela Sra. Evelyn Pruitt do Escritório de Pesquisa Naval dos EUA (GRAHAM, 1999). Esse campo de conhecimento engloba diferentes áreas, como física, botânica, engenharia eletrônica e cartografia.

A somatória destes elementos resulta em ricas informações expresso na forma de imagens da superfície terrestre. Diante disto, uma infinidade de outras possibilidades de análise surgem e podem ser aplicadas. Por exemplo, o desenvolvimento de mapas para a localização de recursos minerais, propriedades, animais e plantas. Como também, seu uso no monitoramento de eventos climáticos muito utilizado na meteorologia, ou na identificação de queimadas/focos de calor.

Principalmente após a virada do século 21, os satélites de sensoriamento remoto se tornaram altamente disponíveis. Uma tecnologia que antes só servia a determinados grupos da sociedade, passou a ser massificada ao público por meio da internet.

O Brasil, no ano de 2019, realizou o lançamento do satélite CBERS-04A, um sistema sensor voltado para o imageamento da superfície terrestre e aberto ao público. Naturalmente, houve a crescente busca dos usuários para consumir esses dados. Todavia, pela escassez de desenvolvedores de software na área, apenas duas ferramentas foram criadas com este intuito:

o *cbers4a-downloader*, um *plugin* (programa que estende as funcionalidades de outro) para *QGIS* (Software SIG) que disponibiliza ferramentas para pesquisar, visualizar, baixar e processar imagens através da interface gráfica do software, todavia, se o usuário não possuir o *QGIS*, não há possibilidade de usá-lo. Há também, uma plataforma WEB chamada “Explore” da Divisão de Geração de Imagens do Instituto Nacional de Pesquisas Espaciais (DGI – INPE), que possibilita a busca e *download* das imagens somente. Apesar de ser multiplataforma, é dependente de um *browser* com suporte a *JavaScript* para utilizá-lo.

Portanto, o intuito deste trabalho é desenvolver uma biblioteca para a linguagem Python com finalidade de consumir os dados do CBERS-04A de forma automatizada, oferecendo à comunidade possibilidades de criação de algoritmos mais complexos, usando como base o produto desenvolvido nesta monografia.

1.1 JUSTIFICATIVA

Dada a crescente persistência dos dados e a necessidade de consumir e manipular Big Data, surge a demanda de criar ferramentas que auxiliam o desenvolvedor a manusear os dados para gerar novas informações. Segundo dados do Brazil Data Cube¹, a quantidade de dados produzidos no intervalo de dois anos (2017 e 2018) pelo satélite óptico China-Brazil Earth Resources Satellite (CBERS) já ultrapassa os 100 Terabytes (Ferreira et al, 2022). Com isso, esses conjuntos de dados são tão volumosos que os softwares tradicionais de consulta de dados não consegue gerenciá-los. Portanto, é necessário utilizar soluções, como o *Spatial Temporal Asset Catalog*, que faz o uso de estruturas de dados e técnicas modernas de indexação de dados e os disponibilizam através de *REST API*. Porém, a falta de ferramentas flexíveis que permitam a utilização desses dados através de linguagens de programação limita os analistas a apenas utilizar os softwares tradicionais com interface gráfica, ocasionando trabalhos repetitivos e ineficientes. Dessa forma, o trabalho proposto visa criar uma biblioteca para a linguagem de programação Python, com o foco no consumo e manipulação dos dados do satélite CBERS-04A.

1.2 OBJETIVO

1.3 GERAL

Desenvolver uma biblioteca na linguagem de programação Python para consultar e processar os dados dos satélites imageadores CBERS-04A.

¹ Brazil Data Cube: Disponível em: <http://brazildatacube.org/>

1.4 ESPECÍFICOS

- Definir base de dados dos produtos gerados pelo satélite CBERS 4A
- Implementar algoritmos de busca de imagens específicas com base nos parâmetros na base de dados.
- Implementar algoritmo para o download das imagens contidas na base de dados.
- Implementar técnicas de processamento de imagens de satélite.
- Implementar testes unitários para descobrir defeitos antecipadamente e obter um software mais robusto.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 SENSORIAMENTO REMOTO: CONTEXTO HISTÓRICO E DEFINIÇÕES

A busca por formas de mapear recursos naturais, cidades e propriedades com um maior nível de detalhamento tem um avanço significativo no início da segunda revolução industrial, com o advento da fotografia em 1839 por Daguerre e Niepce e sendo utilizada pelo Corpo de Engenharia da França em 1858 para tomada de fotografias a partir de balões (Figura 1) para o mapeamento topográfico de amplas áreas do território francês (NOVO, 2008).

Figura 1 - Foto aérea de Paris, 1858



Fonte: domínio público.

Posteriormente à segunda guerra mundial, com o aperfeiçoamento da fotografia e de técnicas de fotogrametria (ciência de realizar medições a partir de fotografias), o termo sensoriamento remoto aparece, pela primeira vez, na literatura científica nos Estados Unidos, anos 50, pela Sra. Evelyn Pruitt do Escritório de Pesquisa Naval dos EUA (GRAHAM, 1999).

Após o lançamento do primeiro satélite da história, o soviético Sputnik, houve a grande corrida espacial da guerra fria, ocasionando um avanço tecnológico significativo nos sistemas orbitais de sensoriamento remoto para fins de espionagem, como por exemplo, o programa CORONA do Serviço de Inteligência Americano (CIA), tendo como objetivo a obtenção de imagens a partir de satélites em órbita para o monitoramento de atividades da URSS (Figura 2).

Figura 2 - Imagem de satélite de Moscou, 1970



Fonte: National Archives Catalog. Disponível em: <https://catalog.archives.gov/id/595264>.

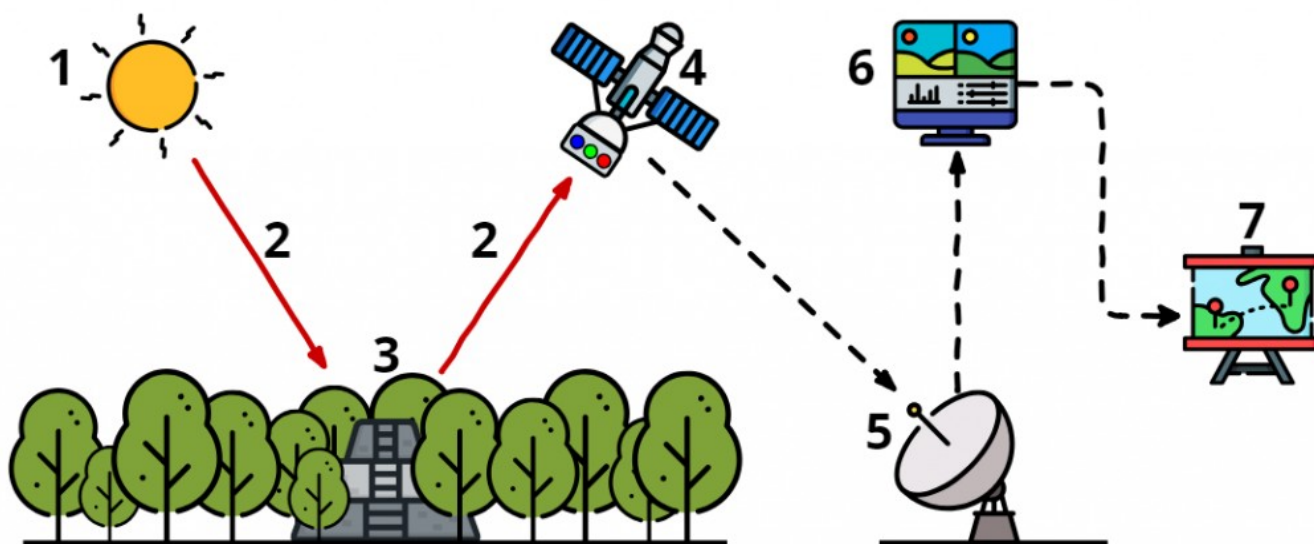
Consequentemente, assim como os computadores, esse tipo de tecnologia deixou de ser exclusivamente militar e passou a ser utilizado pelo público em geral, com a finalidade de estudar o planeta terra. Em virtude do advento da internet, as imagens de satélite são disponibilizados, em alguns casos, gratuitamente para o público através da *World Wide Web*. Como resultado, as imagens oriundas de satélites se tornaram o retrato de sensoriamento remoto, pois, fora os benefícios técnicos, a alta disponibilidade e facilidade de adquiri-las, tornam o principal objeto de análise para vários estudos.

A definição clássica do termo sensoriamento remoto (SR), segundo Elachi (1987), foi definida como um conjunto de técnicas destinado à obtenção de informação sobre objetos, sem que haja contato físico com eles. Porém, com o aprimoramento das técnicas e novas áreas de estudos sendo englobadas por esse campo, uma definição mais moderna é proposta por Novo (2008, v. 3, p. 4) em seu livro *Sensoriamento Remoto: Princípios e Aplicações*:

Podemos, então, a partir de agora, definir Sensoriamento Remoto como sendo a utilização conjunta de sensores, equipamentos para processamento de dados, equipamentos de transmissão de dados colocados a bordo de aeronaves, espaçonaves, ou outras plataformas, com o objetivo de estudar eventos, fenômenos e processos que ocorrem na superfície do planeta Terra a partir do registro e da análise das interações entre a radiação eletromagnética e as substâncias que o compõem em suas mais diversas manifestações.

Dessa forma, para que a captação da informação por esses sensores seja possível, essa informação precisa ser transportada pelo espaço sideral, o que muito se aproxima do vácuo. E isso ocorre porque ela é transportada por meio da radiação eletromagnética. Pode-se representar esse fluxo de aquisição e transporte dos dados com a Figura 3:

Figura 3 - Componentes de um sistema de sensoriamento remoto



Fonte: PAITITI Research. [Remote Sensing from Space]. 1 Ilustração. Disponível em: <https://paititi.info/research-technology/remote-sensing-from-space/>. Acesso em: 16 mar. 2023.

Inicia-se com uma fonte de radiação eletromagnética, como o sol, que emite radiação para a superfície e incide sobre os objetos, no qual interagirá de diferentes formas em cada objeto alvo, ou seja, podendo absorver, refletir e transmitir a radiação eletromagnética diversificadamente. Então, essa reflectância (quantidade de energia eletromagnética refletida de um alvo) é captada pelos sensores a bordo do satélite que posteriormente são enviados para uma antena receptora, e por fim, os dados brutos são processados pelo centro de processamento de dados, gerando uma imagem da superfície terrestre.

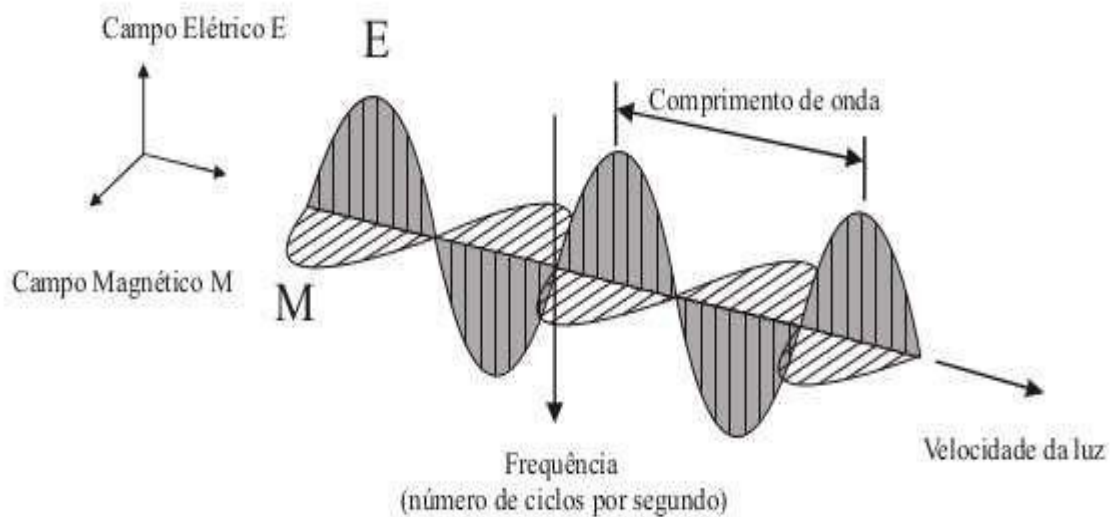
2.2 FORMAÇÃO DE UMA IMAGEM DE SATÉLITE

2.2.1 Radiação eletromagnética

A radiação eletromagnética é o principal meio de transporte das informações de reflectância dos objetos para os sensores passivos (não emitem energia eletromagnética). Em suma, a radiação eletromagnética consiste em um campo elétrico (E) que varia senoidalmente e em direção perpendicular a um campo magnético (M) que é orientado à direita do campo

elétrico. Ambos os campos viajam a velocidade da luz, ou seja, a 300.000km/s. Isso pode ser mais bem compreendido por meio da Figura 4, que apresenta os dois campos e a trajetória de deslocamento da onda.

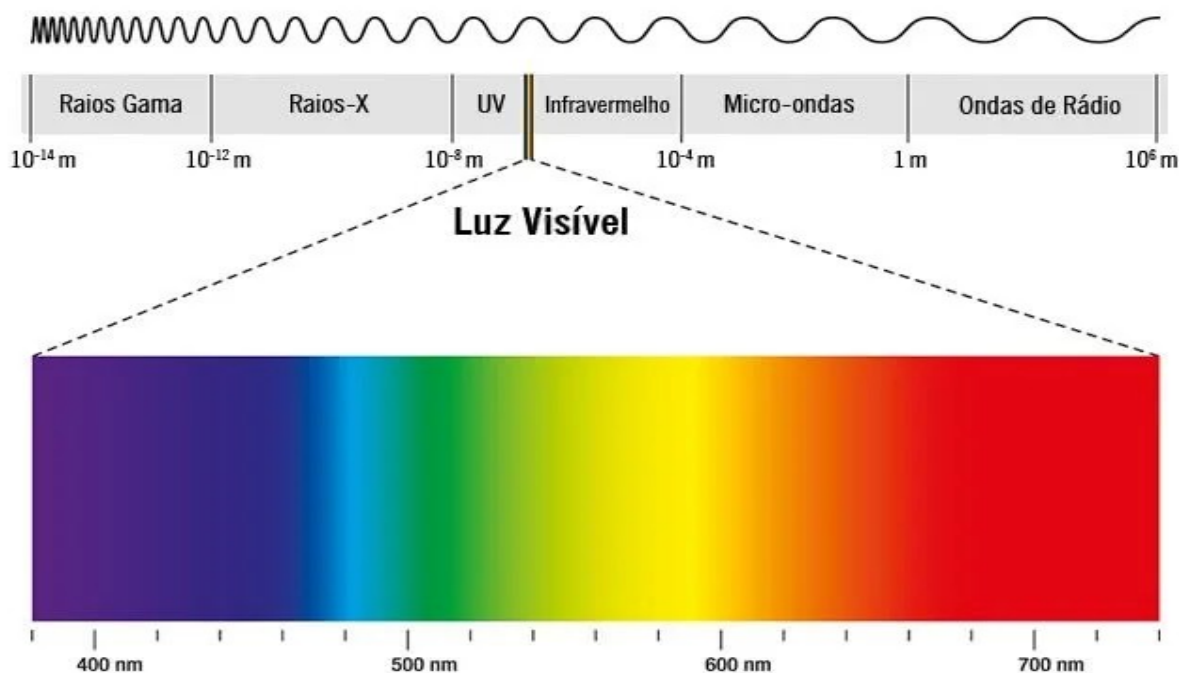
Figura 4 - Radiação eletromagnética



Fonte: Meneses et al. (2018)

Por consequência, o modelo ondulatório da radiação eletromagnética é caracterizado pelo comprimento de onda, no qual representa a distância entre dois pontos quaisquer de igual intensidade no campo elétrico e magnético, sendo esses dois pontos comumente da crista de uma onda a sua subsequente. O conjunto de comprimentos de ondas é conhecido como Espectro Eletromagnético (Figura 5), sendo uma forma de representar ordenadamente as principais regiões de radiação em função do comprimento de onda.

Figura 5 - Espectro Eletromagnético



Fonte: toda matéria. [Entenda Porque o Céu é Azul]. 2 Ilustração. Disponível em: <https://www.todamateria.com.br/por-que-o-ceu-e-azul/>. Acesso em: 21 dez. 2022.

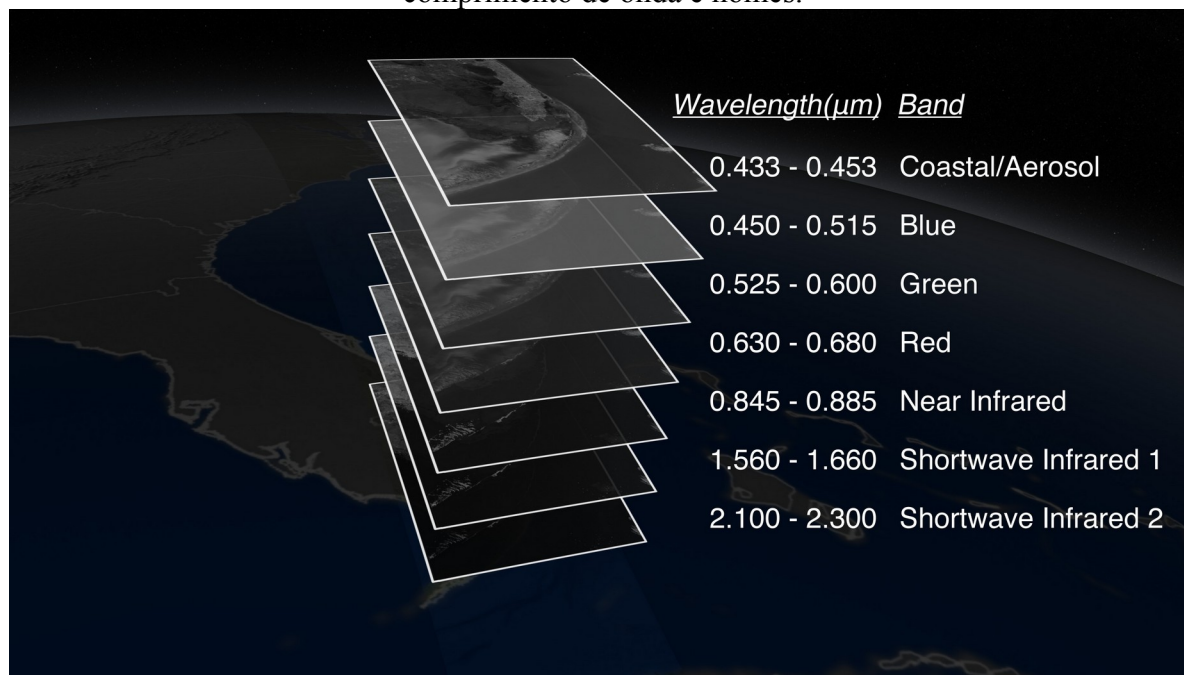
Dessa forma, o diagrama do espectro eletromagnético mostra de maneira ordenada, da esquerda para direita, comprimentos de onda da ordem dos picômetros, como os Raios Gama, até a ordem dos quilômetros, como as Ondas de Rádio.

2.2.2 Sistema sensor

Diferente de uma câmera digital doméstica, os sistemas sensores embarcados nos satélites possuem outras finalidades além de capturar imagens para fins visuais e artísticos. Como explica Kuchkorov (2021), um dos objetivos dos sistemas sensores é medir a quantidade de luz refletida ou emitida por objetos na superfície da terra em múltiplos comprimentos de ondas, e converter essas informações analógicas para um elemento pictorial.

Essa energia coletada, com comprimento de onda específica, é convertida e interpretada para um valor numérico, conhecido como número digital. Quartoli, Vicente e Araújo (2014, p. 67) explicam que esse número indica o nível de cinza, ou seja, a quantidade de luz refletida ou emitida da superfície terrestre, podendo variar do preto (nenhuma luz) ao branco (muita luz). Portanto, uma coleção de números digitais de um determinado intervalo de comprimento de onda, organizados em matriz, é conhecida como “banda”. A Figura 6 mostra as bandas do satélite *Landsat 5*.

Figura 6 - As 7 bandas do satélite Landsat 5, contendo os seus respectivos intervalos de comprimento de onda e nomes.



Fonte: Nasa. [Florida Everglades LDCM Band Remix]. 2 Ilustração. Disponível em:

<https://svs.gsfc.nasa.gov/4040>. Acesso em: 08 maio 2023.

2.3 RESOLUÇÕES DA IMAGEM: ESPACIAL, ESPECTRAL E TEMPORAL

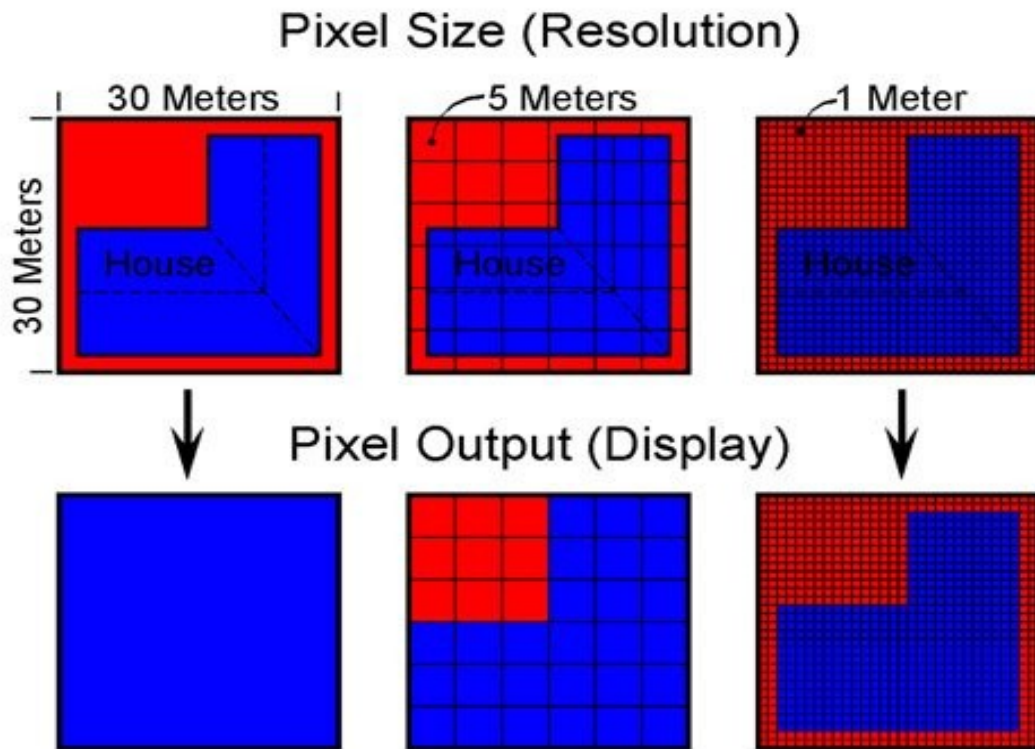
O objetivo de um sistema sensor define diretamente nas propriedades de suas imagens. Portanto, pode possuir diferentes características, como: variadas resoluções espaciais, inúmeros intervalos de comprimento de onda, capacidade de medir a radiância do alvo e diversificados intervalos de data em que a imagem é tomada no mesmo local. Essas propriedades são, respectivamente, resolução espacial, espectral, radiométrica e temporal. Para esse trabalho, é apenas necessário as definições das resoluções: espacial, espectral e temporal.

Os sensores de sensoriamento remoto capturam informações sobre a superfície da Terra dividindo-a em pixels. Cada pixel é uma unidade de área na imagem e contém informações sobre as características presentes nessa área específica. Para definir o tamanho da área do pixel (resolução espacial), é necessário definir a taxa de amostragem no terreno, ou seja, a densidade com que os pontos de dados são coletados em uma determinada área na superfície da terra. Quanto maior a taxa de amostragem, mais informações da superfície.

Em síntese, resolução espacial define o tamanho do menor objeto que pode ser identificado em uma imagem, logo, um objeto na superfície terrestre só irá ser reconhecido se possuir, no mínimo, um tamanho igual ou maior que a resolução espacial (MENESES et al,

2012, p. 25). Pode-se ilustrar na Figura 7 o que foi descrito acima, a quantidade de amostragem de um pixel para resolver uma casa de 30 x 30 metros de tamanho.

Figura 7 - Diferentes resoluções espaciais de uma imagem de satélite sobre uma residência



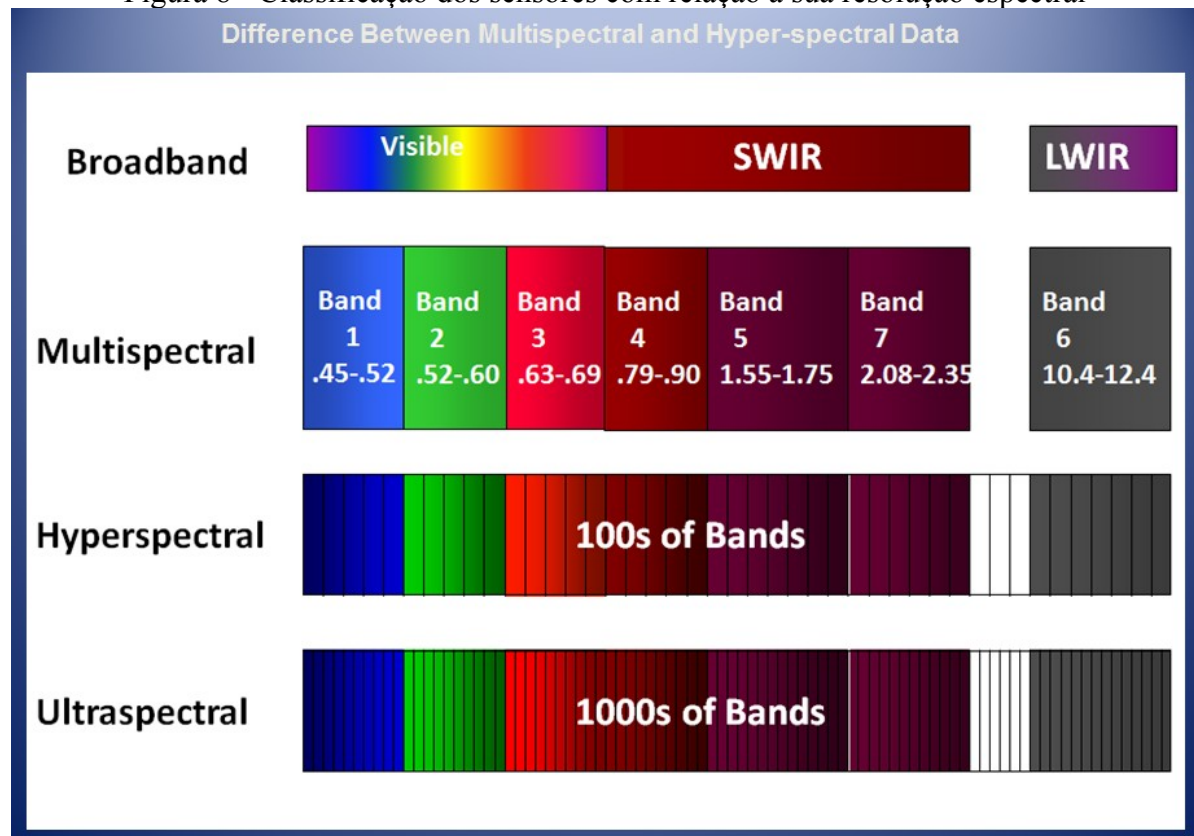
Fonte: SATIMAGINGCORP .[**Characterization of Satellite Remote Sensing Systems**]. Disponível em: <https://www.satimagingcorp.com/services/resources/characterization-of-satellite-remote-sensing-systems/>.

Acesso em: 13 maio 2023

Segundo Meneses et al (2012, p. 27), para o sensoriamento remoto, a obtenção simultânea de imagens em múltiplas bandas espectrais é, sem dúvida, a propriedade mais importante dos sensores imageadores. Com isso, o que define a resolução espectral de um sensor é sua quantidade de bandas em diferentes regiões do espectro eletromagnético.

Dessa maneira, os sistemas sensores são categorizados de acordo com a quantidade de bandas que podem processar, como mostra a Figura 8. Da menor capacidade de enxergar múltiplos comprimentos de ondas até a maior, são nomeados, respectivamente, como: *Broadband*, Multiespectral, Hiperespectral, Ultraespectral.

Figura 8 - Classificação dos sensores com relação a sua resolução espectral



Fonte: Altigator. [Multispectral and Hyperspectral drone imagery] .3 figura. Disponível em:

<https://altigator.com/en/multispectral-and-hyperspectral-drone-imagery/>. Acesso em: 16 maio 2023

Considerando que os sensores estão orbitando a Terra, os dados coletados de um mesmo ponto na superfície possuem um intervalo de tempo, esta propriedade é portanto denominado resolução temporal que se refere à frequência que o sensor revisita uma área e obtém imagens periódicas ao longo de sua vida útil (MENESES et al, 2012, p. 31).

2.4 O SATÉLITES IMAGEADOR CBERS-04A

O CBERS (*China-Brazil Earth Resources Satellite*, Satélite Sino-Brasileiro de Recursos Terrestres na tradução livre) é um programa de construção de satélites de sensoriamento originado de um acordo entre a China e Brasil em 6 de julho de 1988. Compartilhando recursos financeiros e humanos, os países uniram esforços para derrubar as barreiras que impedem o desenvolvimento e a transferência de tecnologias sensíveis impostas pelos países desenvolvidos. A parceria conjunta rompeu os padrões que restringiam os acordos internacionais à transferência de tecnologia e o intercâmbio entre pesquisadores de nacionalidades diferentes (INPE, 2018).

O CBERS-04A (Figura 9) é o sexto satélite da família CBERS, foi lançado e colocado em órbita com sucesso no início da madrugada do dia 20 de dezembro de 2019, pelo foguete Longa Marcha 4B, a partir do Centro de Lançamento de Satélites de Taiyuan (TSLC), na China (INPE, 2019).

Figura 9 - Satélite CBERS 04A



Fonte: Akaer. [Câmeras MUX e WFI, desenvolvidas pela OPTO S&D, integram o satélite CBERS 04A, a ser lançado dia 20 de dezembro]. 1 Figura. Disponível em: <https://www.akaer.com.br/2019/12/19/cameras-mux-e-wfi-desenvolvidas-pela-opto-sd-integram-o-satelite-cbers-04a-a-ser-lancado-dia-20-de-dezembro/>.

Acesso em: 17 maio 2023

O satélite CBERS 04A, assim como os anteriores, é equipado com câmeras para observações ópticas de todo o globo terrestre, além de um sistema de coleta de dados e monitoramento ambiental (INPE, 2019). A plataforma conta com 3 câmeras: Multiespectral (MUX), Campo Largo (WFI) e Multiespectral e Pancromática de Ampla Varredura (WPM), e suas configurações são especificadas no Quadro 1.

Quadro 1 - Características da câmera WFI, MUX e WPM

Característica	WFI	MUX	WPM
Resolução espectral	4 bandas	4 bandas	5 bandas
Bandas	B13: 0,45 - 0,52 μm B14: 0,52 - 0,59 μm B15: 0,63 - 0,69 μm B16: 0,77 - 0,89 μm	B05: 0,45 - 0,52 μm B06: 0,52 - 0,59 μm B07: 0,63 - 0,69 μm B08: 0,77 - 0,89 μm	P : 0,45 - 0,90 μm B1: 0,45 - 0,52 μm B2: 0,52 - 0,59 μm B3: 0,63 - 0,69 μm B4: 0,77 - 0,89 μm
Resolução Espacial	55 m	16,5 m	2m (pancromática) 8 m (multiespectral)
Resolução Temporal	5 dias	31 dias	31 dias

Fonte: INPE (2019)

Cada câmera apresenta vantagens em diferentes casos de uso, por exemplo, a WFI apresenta uma baixa resolução temporal, ideal para monitoramento e vigilância. A MUX e WPM apresentam uma alta resolução temporal, em compensação, uma baixa resolução espacial, ideal para análises que exijam o reconhecimento de objetos em solo de até 16,5 metros para a MUX e 8 ou 2 metros para a WPM.

2.5 PROCESSAMENTO DE IMAGEM DE SATÉLITE

2.5.1 Composição colorida

A cor é um poderoso descritor que muitas vezes simplifica a identificação do objeto e sua extração de uma cena, e, além disso, os seres humanos são capazes de discernir milhares de tons e intensidades de cor, diferente dos tons de cinza (GONZALEZ e WOODS, 2009, p. 259).

Dessa maneira, Costa (1998, p. 31) explica como uma imagem em escala de cinza passa a ser colorida:

Uma das maneiras mais diretas de se introduzir cores em uma imagem monocromática é através do fatiamento de níveis de cinza. Neste procedimento, também chamado de pseudocoloração, a escala de níveis de cinza é dividida em intervalos disjuntos, e para cada um deles é atribuída uma cor, definida por uma diferente combinação de vermelho, verde e azul.

Aplicando esses conceitos básicos de processamento digital de imagens ao sensoriamento remoto, podemos combinar as diferentes bandas no vermelho, verde e azul para visualizar detalhes que não são visíveis quando visualizadas isoladamente, por exemplo,

a composição falsa cor (Figura 10), no qual combina as bandas do infravermelho próximo, vermelho e verde (R4B3G2) para destacar as plantas, pois as mesmas possuem uma alta reflectância no infravermelho próximo, criando um maior destaque do canal vermelho na imagem.

Figura 10 - Imagem do CBERS-04A da cidade de Porto Velho utilizando a composição falsa cor



Fonte: Elaborado pelo Autor, 2023

2.5.2 *Pansharpening*

Alguns sistemas de sensoriamento remoto registram sinais de energia eletromagnética concentrada num único e largo intervalo de comprimento de onda, por exemplo, de 500 a 700 nm, é chamado de banda pancromática (JENSEN, 2015, p. 459).

Dessa forma, por possuir uma maior densidade de informações, essa banda, consequentemente, possui uma alta resolução espacial. Além disso, a imagem não apresenta uma precisão de informações espectrais pelo fato de englobar múltiplos comprimentos de onda.

Quando imagens de resolução espacial inferior são mesclados com imagens pancromáticas de maior resolução espacial, é comumente referido como *pansharpening*. O objetivo do *pansharpening* é criar uma imagem nítida incorporando todo o conteúdo de informação espacial da imagem pancromática sem perder nenhuma informação espectral da imagem multiespectral (JENSEN, 2015, p. 169).

2.6 SPATIO TEMPORAL ASSET CATALOG

O *SpatioTemporal Asset Catalog* (STAC) é uma especificação que visa padronizar a forma como os metadados de ativos geoespaciais são estruturados e consultados. Esta especificação também conta com uma implementação de serviço web de código aberto que permite a busca de imagens de satélites através de um *Representational State Transfer* (REST) API, no qual expõe todos os dados através de URIs e são requisitados por meio do protocolo HTTP.

Segundo o site oficial do STAC², o objetivo é criar um índice de todas as imagens (satélite, aéreas, drones, etc), produtos de dados derivados e capturas geoespaciais alternativas. Dessa forma, ao passar parâmetros de busca como uma geometria, ou intervalo de datas no padrão ISO 8601 (padrão para representar datas e tempo).

2 Página oficial do STAC disponível em: <https://stacspec.org/>

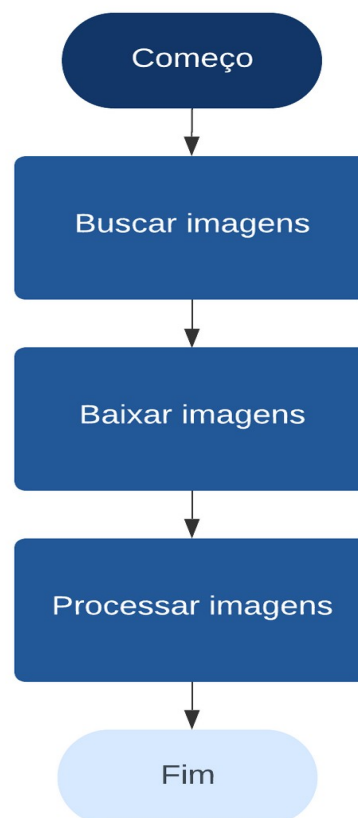
3 METODOLOGIA

3.1 DEFINIÇÃO DAS TECNOLOGIAS UTILIZADAS

Nesta seção, será descrita as etapas utilizadas no desenvolvimento da aplicação. O sistema proposto para buscar, baixar e processar as imagens do satélite CBERS-04A foi desenvolvido a partir da necessidade de automatizar essas atividades utilizando linguagens de programação, e, a partir disso, suas funcionalidades foram desenvolvidas com base nessa premissa.

O software foi desenvolvido de maneira gradual, primeiro sendo desenvolvido a busca, depois o download e em seguida os algoritmos de processamento de imagens, como mostra o fluxograma (Figura 11).

Figura 11 - Fluxograma do sistema proposto



Fonte: Elaborado pelo Autor, 2023

O programa foi implementado utilizando a linguagem de programação Python, em sua versão 3.10. Além disso, foi desenvolvido num ambiente Linux com a IDE *Pycharm*.

3.2 ESCOLHA DA BASE DE DADOS

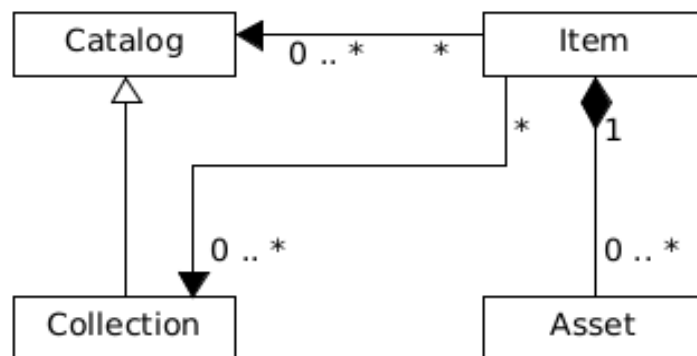
O primeiro passo foi determinar a base de dados que seria consumido pela aplicação para a busca das imagens. De certa forma, não há nenhuma página que especifique links e instruções de uso para acessar os dados do satélite CBERS-04A, então, foi necessário realizar o estudo do código fonte de um *plugin*³ para *QuantumGIS* (Software de SIG), no qual realiza as mesmas funções que o programa proposto neste trabalho, porém apenas via interface gráfica.

Após o link ter sido extraído do código fonte, foi feita uma conexão com o servidor STAC do Instituto nacional de Pesquisas Espaciais e feito um estudo para verificar a disponibilidade e qualidade dos dados, verificando a viabilidade de construir uma aplicação que será completamente dependente deste banco de dados.

3.3 MODELAGEM DOS OBJETOS STAC

A especificação STAC possui alguns componentes que formam a principal estrutura do projeto: A Collection, Catalog, Item e Asset (na tradução livre: Coleção, Catálogo, Item e Ativo). A estrutura no diagrama UML pode ser observada na Figura 12.

Figura 12 - Diagrama UML dos componentes STAC



Fonte: Elaborado pelo Autor, 2023

A estrutura proporciona uma organização aos dados, onde, de maneira hierárquica, cada componente é estritamente ligado ao outro. Antes de tudo, é necessário entender cada uma das entidades presentes no diagrama, iniciando pelo Catálogo. O catálogo fornece uma estrutura flexível para vincular vários itens e outros catálogos, possibilitando uma estrutura

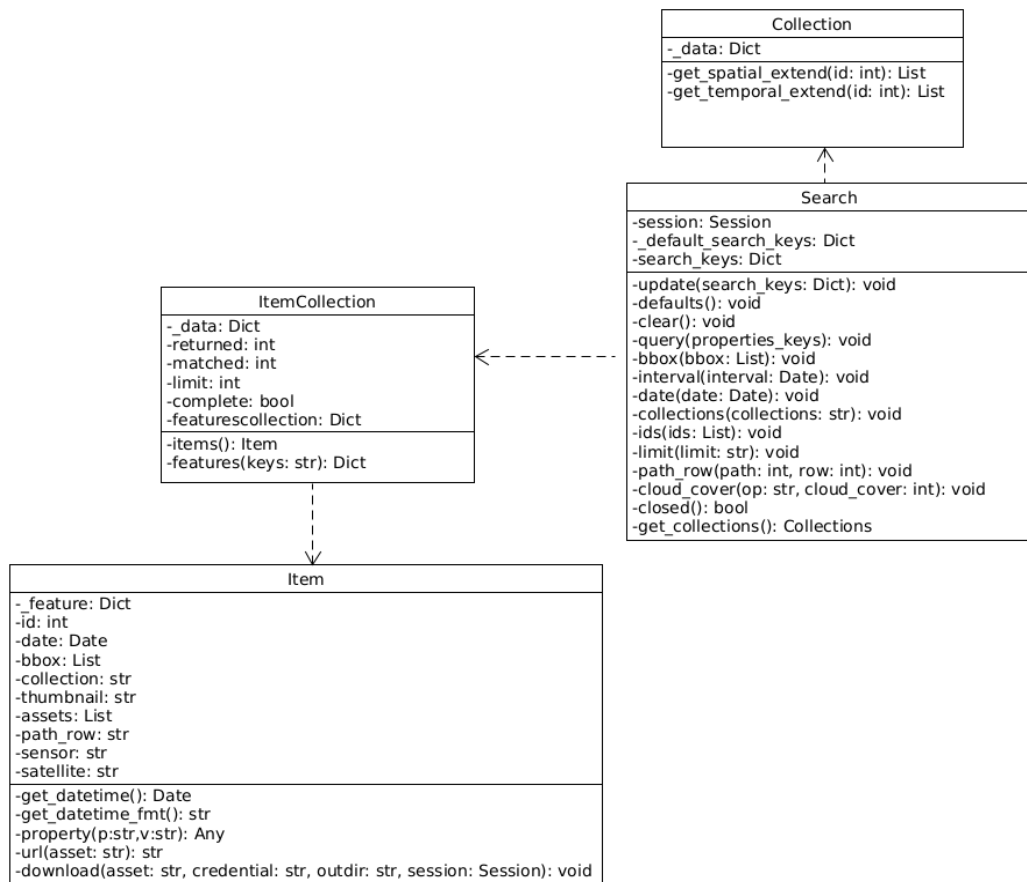
3 Código fonte disponível em: <https://github.com/sandroklippel/cbers4a>

semelhante a um dígrafo. A coleção é uma especialização do catálogo, possuindo mais campos de metadados, como licença, palavras-chaves, fontes e etc. Item é uma unidade atômica do STAC, portando metadados, referências para os ativos e *thumbnail*. Para finalizar, os ativos é onde as imagens se localizam, ou seja, um objeto que contém uma referência direta ao arquivo.

A estrutura da especificação STAC traduzida para classes foi reaproveitada do código do *plugin* de QGIS mencionado anteriormente, usufruindo dos direitos concebidos da licença de software livre do MIT. Dessa forma, com o código modificado e adaptado, foi criada uma estrutura que visa facilitar a manipulação dos objetos recebidos do serviço, porém, é apenas para a utilização interna do programa, pois, dessa forma, não irá expor ao usuário a complexidade da especificação STAC.

Observa-se na Figura 13 o diagrama das classes utilizadas internamente para abstrair as informações recebidas no formato JSON (*JavaScript Object Notation*).

Figura 13 - Estrutura de classes utilizadas no software



Fonte: Autoria própria

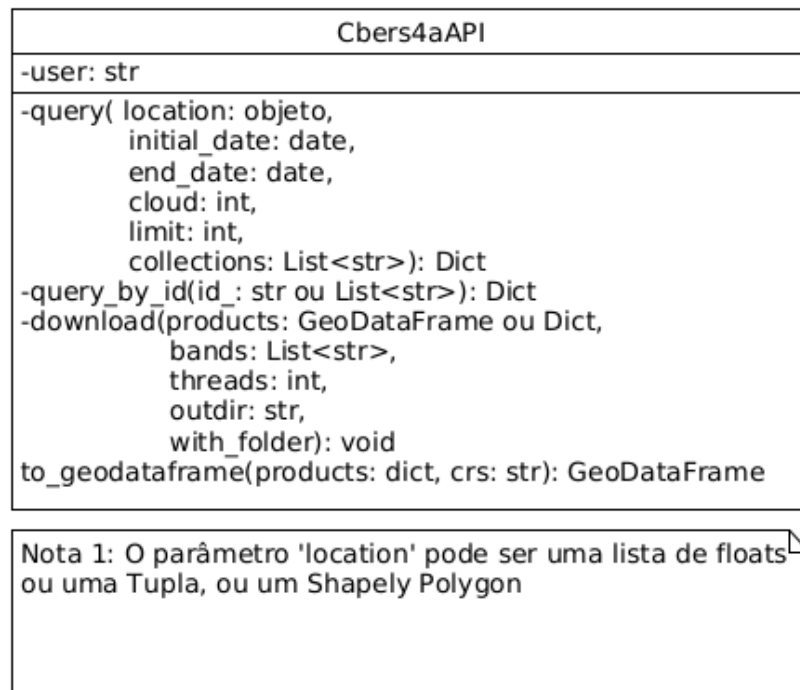
O diagrama acima apresenta os atributos e métodos necessários para o funcionamento do programa, os mesmos proporcionam a abstração necessária para facilitar o manuseio dos dados provenientes do STAC, além de facilitar o desenvolvimento.

A classe *Search* é responsável por buscar os dados e disponibilizá-lo para o usuário, o mesmo conta com diversos métodos para modificar e criar parâmetros de pesquisa, como por exemplo o “*cloud_cover*”, cujo adiciona um parâmetro de pesquisa para limitar a quantidade de nuvem que deve conter na imagem. Todos esses parâmetros serão organizados e enviados para o servidor STAC, que realizará a pesquisa na base de dados e devolver uma Coleção. Para manipular essa coleção é utilizado uma das duas classes disponíveis, dependendo do contexto. Para consultar os metadados da coleção, utiliza-se a classe *Collection*. Para extrair os itens da coleção, utiliza-se o *ItemCollection*, cujo devolverá uma lista de objetos da classe *Item*, permitindo a fácil manipulação de cada *Asset* presente no JSON recebido da classe *Search*.

3.4 ABSTRAINDO A COMPLEXIDADE COM *FACADE*

Como dito, toda a complexidade da especificação STAC foi ocultada do usuário, isto é, aplicou-se um padrão de projeto conhecido como *Facade* (Fachada na tradução livre). O *Facade* é um padrão de projeto estrutural que fornece uma interface simplificada para uma biblioteca, um *framework*, ou qualquer conjunto complexo de classes. Dessa forma, possibilita ao usuário da biblioteca que execute uma tarefa sem preocupar-se com a regra de negócio. A classe *Facade* é ilustrada na Figura 14.

Figura 14 - Classe Facade - Cbers4aAPI



Fonte: Elaborado pelo Autor, 2023

O método *query* é responsável por realizar a busca das bandas na base de dados. O parâmetro *location* especifica a localização geográfica da imagem, ou seja, toda e qualquer imagem na região especificada irá ser buscada. Os parâmetros *initial_date*, *end_date* e *cloud* são os filtros de metadado, isto é, toda e qualquer *Asset* que estiver dentro do intervalo de tempo e porcentagem de nuvem, irá ser selecionada. *Limit* e *collection* estão relacionados diretamente a metadados da Coleção, o primeiro especifica o limite de Itens que a coleção enviará ao solicitante, e o segundo indica para qual coleção esses filtros serão aplicados.

Query_by_id foi desenvolvido para caso um usuário já possua um identificador único da imagem contida na base de dados do serviço STAC, pois, dessa forma, o usuário poderá criar algoritmos para um conjunto de imagens específicas.

A função *download* é responsável por baixar as bandas da base de dados. A mesma recebe por parâmetro os produtos que o usuário adquiriu ao usar o método *query*, uma lista de quais bandas deseja, como por exemplo a vermelha, e mais parâmetros relacionados a *multithreading* e armazenamento.

O método *to_geodataframe* é uma alternativa ao usuário para converter a coleção de itens em formato JSON para GeoDataFrame. O GeoDataFrame é uma classe da biblioteca

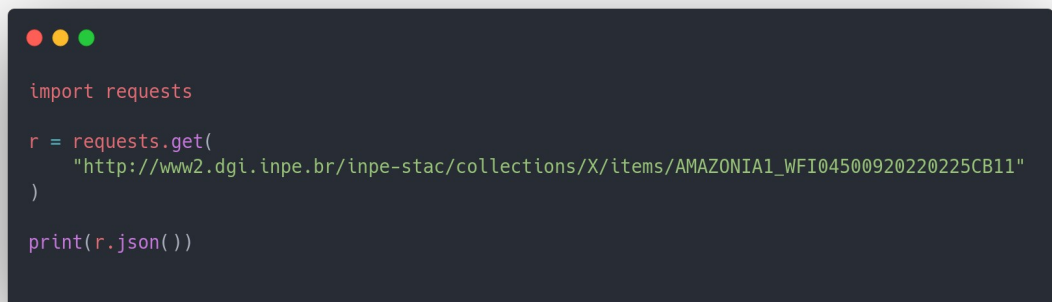
GeoPandas, que herda todas as funcionalidades da biblioteca Pandas, porém adicionando funcionalidades de manipulação geográfica.

3.5 PROCESSO DE DESENVOLVIMENTO

A linguagem Python conta com um sistema de distribuição de biblioteca de terceiros, o PyPi. A partir disso, o trabalho desenvolvido conta com uma gama de dependências de softwares de terceiros.

Para trabalhar com o protocolo HTTP, foi utilizado a biblioteca *Requests*. Ela possui uma gama de comandos que abstrai a complexidade de manusear a interface de rede dos computadores. Essa biblioteca foi utilizada para buscar o conjunto de dados, montando a URL na classe *Search* e executando-a com o método *get*. A Figura 15 mostra uma requisição de um Item ao serviço STAC, e a Figura 16 mostra o resultado em sua forma recolhida (não mostra o objeto em detalhes).

Figura 15 - Código exemplo para requisitar um Item utilizando a biblioteca *requests*



```
import requests

r = requests.get(
    "http://www2.dgi.inpe.br/inpe-stac/collections/X/items/AMAZONIA1_WFI04500920220225CB11"
)

print(r.json())
```

Fonte: Elaborado pelo Autor, 2023

Figura 16 - Resultado da requisição em sua forma recolhida

```

stac_version:      "0.9.0"
▶ stac_extensions:  [...]
type:              "Feature"
id:                "AMAZONIA1_WFI04500920220225CB11"
collection:        "AMAZONIA1_WFI_L2_DN"
▶ geometry:         {...}
▶ bbox:             [...]
▶ properties:       {...}
▶ assets:           {...}
▶ links:            [...]

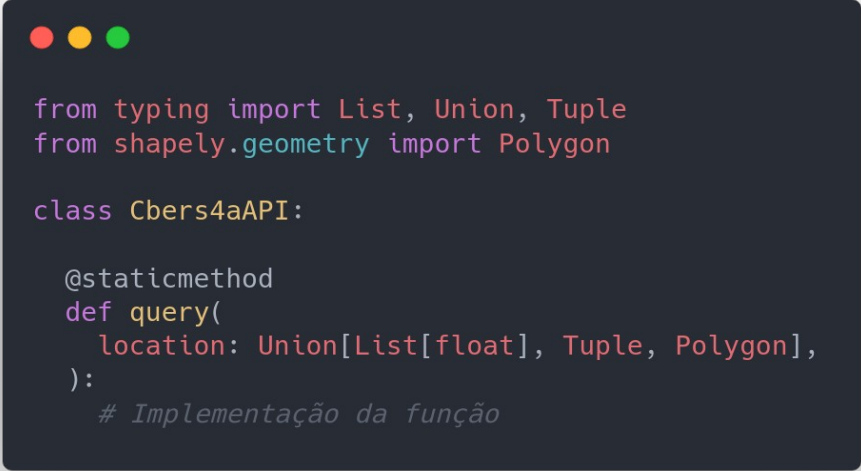
```

Fonte: Elaborado pelo Autor, 2023

Pelo fato da linguagem de programação Python ser de tipagem dinâmica, foi necessário a utilização de uma biblioteca para mudar este paradigma, o *Typing*. Dessa forma, previne com que o usuário erre os tipos dos parâmetros ao utilizar os métodos da classe *Facade*. Como por exemplo, é explícito os tipos de dados possíveis no parâmetro *location* na função *query* quando são utilizados os objetos do *Typing* (Figura 17), ou uma lista de floats, ou uma tupla ou um *Shapely Polygon*.

Shapely é uma biblioteca que traz a capacidade de manipular as representações vetoriais geográficas para a linguagem. Em síntese, a representação vetorial em sua essência é um objeto discreto, na qual possui uma localização específica, e podem ser representados graficamente como três formas básicas: pontos, linhas e polígonos. Dessa maneira, o usuário pode especificar uma área específica utilizando um objeto geométrico.

Figura 17 - Exemplo do parâmetro *location* na função *query*, aplicando a tipagem com a biblioteca *Typing*.



```
from typing import List, Union, Tuple
from shapely.geometry import Polygon

class Cbers4aAPI:

    @staticmethod
    def query(
        location: Union[List[float], Tuple, Polygon],
    ):
        # Implementação da função
```

Fonte: Elaborado pelo Autor, 2023

Para oferecer uma alternativa ao usuário que não deseja trabalhar com JSON, foi desenvolvido a função *to_geodataframe*, que permite a conversão para GeoDataFrame. Esse objeto provém da biblioteca Geopandas, também uma dependência da biblioteca desenvolvida neste trabalho.

O GeoPandas é um projeto de código aberto para facilitar o trabalho com dados geoespaciais em Python. O GeoPandas estende os tipos de dados usados pelos Pandas (biblioteca de análise dados) para permitir operações espaciais em tipos geométricos. (GEOPANDAS, 2023). Uma ilustração de um GeoDataFrame foi retirada da documentação oficial do GeoPandas e ilustrado na Figura 18.

Com isso, ao buscar a coleção de itens da base de dados, os mesmos possuem um atributo chamado “*geometry*” (Figura 19), isto é, um Polígono que contém as informações geográficas da banda, ou seja, sua localização e extensão. E transformando-a para uma informação tabular, processar e plotar múltiplas tuplas que contenham essa informação geográfica deixa de ser um desafio, pois desta maneira é possível ter uma prévia da localização dos dados antes de baixá-los, como mostra a Figura 20.

Figura 18 - Exemplo de um GeoDataFrame

BoroCode	BoroName	Shape_Leng	Shape_Area	geometry
5	Staten Island	330470.010332	1.623820e+09	MULTIPOLYGON (((970217.022 145643.332, 970227....
4	Queens	896344.047763	3.045213e+09	MULTIPOLYGON (((1029606.077 156073.814, 102957...
3	Brooklyn	741080.523166	1.937479e+09	MULTIPOLYGON (((1021176.479 151374.797, 102100...

Fonte: Geopandas. [Reading files]. 2 ilustração. Disponível em:

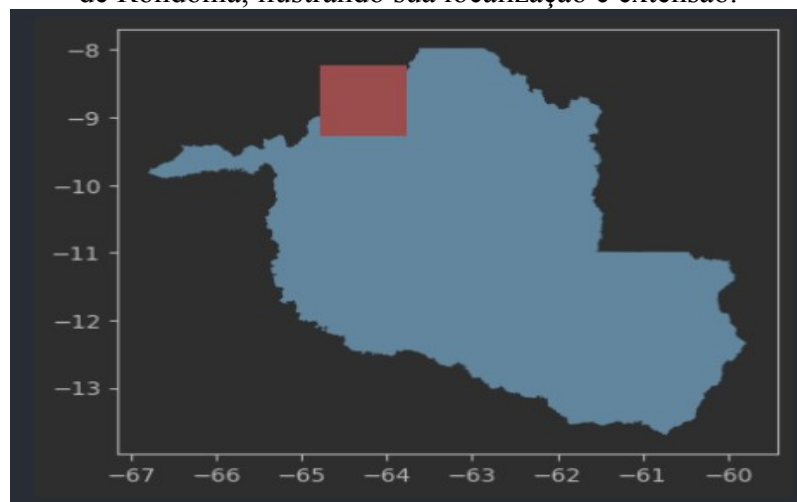
https://geopandas.org/en/stable/getting_started/introduction.html. Acesso em 24 maio 2023.

Figura 19 - Informações de um Item STAC no GeoDataFrame, incluindo o atributo geometry

1 rows x 10 columns pd.DataFrame	
	1
	CBERS4A_WPM22912420210529
geometry	POLYGON ((-64.79340 -8.22668, -64.79340 -9.27490, -63.76810 -9.27490, -63.76810 -8.22668, -64.79340 -8.22668))
datetime	2021-05-29T14:57:55
path	229
row	124
satellite	CBERS4A
sensor	WPM
cloud_cover	80.0
sync_loss	None
eo:gsd	-1
eo:bands	[{'name': 'pan', 'common_name': 'pan'}, {'name': 'blue', 'common_name': 'blue'}, {'name': 'green', 'common_name': 'green'}, {'name': 'red', 'common_name': 'red'}]

Fonte: Elaborado pelo Autor, 2023

Figura 20 - Plote da geometria ilustrada na figura 20 sobre a geometria do Estado de Rondônia, ilustrando sua localização e extensão.



Fonte: Elaborado pelo Autor, 2023

3.6 ALGORITMOS DE PROCESSAMENTO DIGITAL DE IMAGENS

Alguns algoritmos de processamento digital de imagens são usados com frequência ao trabalhar com imagens do CBERS-04A. E esses algoritmos são: Composição colorida e *Pansharpening*.

O algoritmo de composição colorida foi implementado utilizando as bibliotecas Rasterio e NumPy. O Rasterio é uma biblioteca que disponibiliza funções para leitura de arquivos com imagens de satélite. Dessa forma, é possível adquirir informações como metadados e dados matriciais.

Após ler o arquivo com Rasterio, os dados matriciais extraídos para realizar o empilhamento das matrizes são no formato *ndarray*, a matriz multidimensional do NumPy. Em síntese, NumPy é uma biblioteca que estende as funcionalidades do python, com uma sintaxe muito semelhante ao Matlab e, além disso, possui a velocidade de execução próxima a uma linguagem compilada.

Portanto, para empilhar as bandas foi necessário: ler as imagens com o Rasterio, extrair os dados matriciais e empilhá-los em uma única matriz com três dimensões utilizando o NumPy *stack*.

O *Pansharpening* é um algoritmo amplamente utilizado em diversos satélites que possuam a banda pancromática, portanto, o mesmo já foi implementado em outra biblioteca, o GDAL. Porém, há dois problemas com a utilização desse algoritmo: o excesso de parâmetros para ajustes finos (Figura 21) que força o usuário a ter conhecimentos avançados sobre processamento de imagens de satélite, visto que esta função foi desenvolvida com a premissa de poder ser utilizada em qualquer sistema sensor. E a complexidade de implementar um *wrapper* (função destinada a chamar uma ou mais funções) para ela, pois usuários podem possuir diferentes versões do GDAL, ocasionando problemas de incompatibilidade de versão e possibilidade de mudanças na estrutura dos parâmetros em lançamentos futuros, pois a mesma é desenvolvida para encaixar-se com outros satélites.

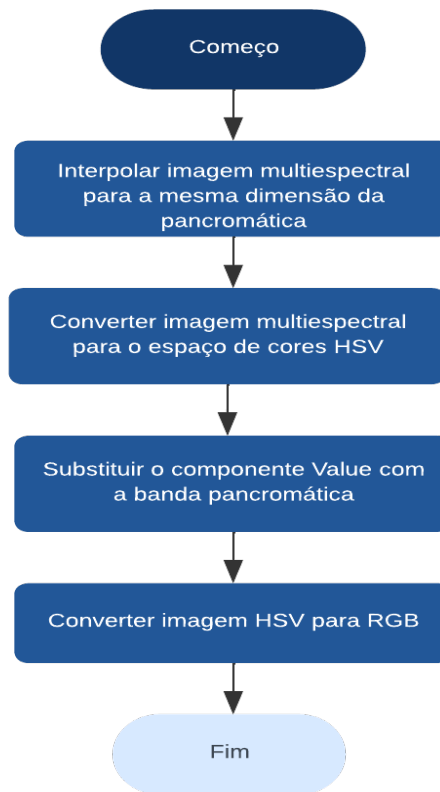
Figura 21 - Função pansharpening da biblioteca GDAL e seus 17 parâmetros de ajustes do algoritmo de pansharpening

```
def gdal_pansharpen(
    argv: Optional[Sequence[str]] = None,
    pan_name: Optional[str] = None,
    spectral_names: Optional[Sequence[str]] = None,
    spectral_ds: Optional[List[gdal.Dataset]] = None,
    spectral_bands: Optional[List[gdal.Band]] = None,
    band_nums: Optional[Sequence[int]] = None,
    weights: Optional[Sequence[float]] = None,
    dst_filename: Optional[str] = None,
    driver_name: Optional[str] = None,
    creation_options: Optional[Sequence[str]] = None,
    resampling: Optional[str] = None,
    spat_adjust: Optional[str] = None,
    num_threads: Optional[Union[int, str]] = None,
    bitdepth: Optional[Union[int, str]] = None,
    nodata_value: Optional[Union[Real, str]] = None,
    verbose_vrt: bool = False,
    progress_callback: Optional = gdal.TermProgress_nocb,
):
```

Fonte: Código fonte do *gdal_pansharpen*. Github.

O pansharpening possui diversas técnicas de transformações, como: IHS (Intensidade, Hue e Saturação), Brovey, Principais Componentes, Gram-Schmidt e outros. O Escolhido para este trabalho foi a transformação IHS, pois, seguindo o autor Meneses et al (2012), diz que a transformação foi concebida como uma poderosa técnica para ser usada como um meio de combinar imagens de diferentes sensores, com diferentes resoluções espaciais. Além disso, a facilidade para implementar tal transformada, contribuiu para a escolha. O fluxograma do algoritmo é especificado abaixo (Figura 22).

Figura 22 - Fluxograma do algoritmo de pansharpening



Fonte: Elaborado pelo Autor, 2023

Na implementação, além das bibliotecas utilizadas no algoritmo de composição colorida, foi adicionado a biblioteca *Scikit-Image*, uma biblioteca de processamento digital de imagens que conta com diversos algoritmos comuns utilizados na área, por exemplo, a conversão do espaço de cores RGB para HSV e vice-versa.

3.7 VALIDAÇÃO E TESTES

Segundo Sommerville (2019, p. 203), testes pretendem mostrar que um programa faz o que foi destinado a fazer e descobrir defeitos antes que ele seja colocado em uso. Portanto, implementar testes unitários para a biblioteca faz com que o código se torne robusto e confiável.

Foi utilizado a biblioteca *Pytests*, pois a mesma possui extensões feitas pela comunidade, que atendem as necessidades de implementação para o caso de uso deste trabalho.

Para testar as funções que utilizam requisições HTTP, utiliza-se a técnica de *Mock*. *Mocks* são imitações ou unidades falsas que simulam o comportamento de unidades reais, isto é, para não efetuar uma conexão com a base de dados, é feita uma classe (Apêndice A) que contem, funções e atributos idênticos aos objetos do *Requests*.

Para os algoritmos de processamento de imagens, é utilizado uma imagem recortada de uma cena completa, então enviada ao algoritmo e comparada com uma imagem anteriormente processada.

3.8 DISPONIBILIZAÇÃO DO SERVIDOR STAC E HOSPEDAGEM DO CÓDIGO

O código fonte da biblioteca desenvolvida neste trabalho foi versionado e hospedado no GitHub <https://github.com/gabriel-russo/cbers4asat>, sob a licença de software livre do MIT.

Além disso, a biblioteca está sendo distribuída através do *Python Package Index* (PyPi), podendo ser instalada através do comando: *pip install cbers4asat*.

O serviço Spatial Temporal Asset Catalog do Instituto Nacional de Pesquisas Espaciais é hospedado na URI www2.dgi.inpe.br/inpe-stac/, e não há necessidade de autenticação ou autorização.

4 RESULTADOS E DISCUSSÕES

4.1 RESULTADOS DOS ALGORITMOS DE BUSCA

A busca é a funcionalidade com mais destaque, pois permite a pesquisa de bandas de maneira automatizada. Dessa forma, foi planejada para ser o mais objetivo possível, realizando buscas somente com os parâmetros necessários. Portanto, para realizar uma busca com a função, o usuário deve preencher os seguintes parâmetros:

- *location*: Uma referência geográfica do local, como, órbita ponto (cobertura sistemática e regular de todo o globo terrestre) ou *bounding box* (caixa delimitadora mínima) ou Polígono *Shapely*.
- *initial_date*: Início do intervalo da data de captura das imagens.
- *end_date*: Fim do intervalo de data de captura das imagens.
- *cloud*: Porcentagem de cobertura de nuvem máxima na imagem.
- *limit*: Quantidade máxima de itens que o STAC devolverá da busca.
- *collections*: As coleções para serem incluídas na busca, cuja listada na plataforma INPE *Explore*.

A Figura 23 mostra um exemplo de busca, podendo ser interpretado como: Buscar uma imagem que está na órbita 229 e no ponto 124 (Órbita ponto que está sobre Porto Velho), no intervalo de data entre 01/04/2021 e 01/07/2021, com no máximo 100% de nuvens na imagem. E que, na resposta, devolva apenas um Item da coleção “CBERS4A_WPM_L4_DN”.

Além disso, é possível buscar uma imagem pelo seu identificador no STAC, como mostra a figura 24. O resultado padrão de uma busca, pode ser visualizado no Apêndice B.

Figura 23 - Exemplo de bloco de código que busca uma imagem de satélite

```
from cbers4asat import Cbers4aAPI
from datetime import date

api = Cbers4aAPI('gabrielrusso@protonmail.com')

porto_velho = (229, 124)

produtos = api.query(location=porto_velho,
                     initial_date=date(2021, 4, 1),
                     end_date=date(2021, 7, 1),
                     cloud=100,
                     limit=1,
                     collections=['CBERS4A_WPM_L4_DN'])

print(produtos)
```

Fonte: Elaborado pelo Autor, 2023

Figura 24 - Buscando uma banda pelo seu identificador

```
from cbers4asat import Cbers4aAPI

api = Cbers4aAPI('gabrielrusso@protonmail.com')

produto = api.query_by_id("CBERS4A_WPM22912420210529")

print(produto)
```

Fonte: Elaborado pelo Autor, 2023

4.2 DOWNLOAD DE BANDAS

Para realizar o download das bandas é necessário possuir um cadastro na plataforma “Explore” do DGI – INPE e adicionar o e-mail registrado no construtor da classe. Após isso, o usuário deve realizar uma busca através do método *query*, assim como explicado anteriormente, e possuir pelo menos um Item no resultado, e então, salvar o resultado numa variável. Após isso, essa variável será utilizada para preencher o parâmetro *products*. Além deste parâmetro, o usuário deve especificar a banda e a pasta que irá ser armazenado, como mostra a Figura 25.

Figura 25 - Realizando uma busca e baixando as bandas

```

api = Cbers4aAPI('gabriel@protonmail.com')

porto_velho = (229, 124)

produtos = api.query(location=porto_velho,
                     initial_date=date(2021, 4, 1),
                     end_date=date(2021, 7, 1),
                     cloud=100,
                     limit=1,
                     collections=['CBERS4A_WPM_L4_DN'])

api.download(products=produtos,
             bands=['red', 'green', 'blue'],
             outdir="./imagens")

```

Executed at 2023.05.24 19:50:52 in 21s 80ms

Fonte: Elaborado pelo Autor, 2023

Pelo fato das bandas serem nomeadas com números, como banda 3, elas representam faixas do espectro que são, também, nomeadas. Dessa forma, especificar o nome da cor que a banda representa é a maneira mais segura para ser utilizada como valor possível de parâmetro, pois os números de banda variam de acordo com o sensor, mas sua representação no espectro eletromagnético não.

4.3 CONVERTER OBJETO JSON PARA GEODATAFRAME

Para realizar a conversão de um objeto JSON retornado da busca para um GeoDataFrame, é necessário apenas uma linha de código, a chamada da função *to_geodataframe*, como mostra a Figura 26. Dessa forma, todos os Itens STAC irão ser convertido para linhas com seus respectivos atributos em colunas, melhorando a visualização dos dados e, além disso, podendo se aproveitar das funções especializadas de análise de dados do GeoPandas.

Figura 26 - Convertendo os Itens no formato JSON para dados tabulares, isto é, o GeoDataFrame.

```
api = Cbers4aAPI('gabriel@protonmail.com')

porto_velho = (229, 124)

produtos = api.query(location=porto_velho,
                     initial_date=date(2021, 4, 1),
                     end_date=date(2021, 7, 1),
                     cloud=100,
                     limit=1,
                     collections=['CBERS4A_WPM_L4_DN'])

api.to_geodataframe(produtos)
Executed at 2023.05.24 20:10:01 in 31s 278ms
```

	geometry	datetime
CBERS4A_WPM22912420210529	POLYGON ((-64.79340 -8.22668, -64.79340 -9.27490, -63...	2021-05-29T14:57:55

Fonte: Elaborado pelo Autor, 2023

4.4 OS ALGORITMOS DE PROCESSAMENTO DIGITAL DE IMAGENS

A implementação destes algoritmos, foram planejados para serem o mais simples e eficientes possíveis. Portanto, com poucos parâmetros pode realizar as tarefas designadas, como por exemplo, a composição colorida (Figura 27) necessita apenas dos caminhos das bandas em cada canal de cor, além de especificar o nome do arquivo resultante.

Figura 27 - Chamada para a função de composição colorida

```
from cbers4asat.tools import rgbn_composite

rgbn_composite(red='BAND3.tif',
               green='BAND2.tif',
               blue='BAND1.tif',
               filename='multispectral.tif')
Executed at 2023.05.24 20:33:20 in 11ms
```

Fonte: Elaborado pelo Autor, 2023

Como resultado do processamento de empilhamento das bandas, vermelha, verde e azul em seus respectivos canais, resulta-se na composição cor verdadeira, como mostra a Figura 28.

Figura 28 - Cidade de Vilhena na composição cor verdadeira, resultado do processamento da função *rgbn_composite*



Fonte: Elaborado pelo Autor, 2023

Em conjunto com o algoritmo de composição colorida, foi desenvolvido o algoritmo de *pansharpening*, no qual exige um grande poder de processamento, por se tratar de um aumento de resolução na imagem. A Figura 28 mostra a função com os parâmetros preenchidos, cujo o parâmetro *multispectral* é o caminho para o arquivo da banda multiespectral, e o parâmetro *panchromatic* é o caminho da banda pancromática, além de especificar o nome do arquivo resultante.

Figura 29 - Chamada para a função de *pansharpening* com os parâmetros preenchidos

```
from cbers4sat.tools import pansharpening

pansharpening(multispectral='multiespectral.tif',
               panchromatic='BAND0.tif',
               filename='pansharp.tif')
Executed at 2023.05.24 20:48:03 in 604ms
```

Fonte: Elaborado pelo Autor, 2023

Pelo fato de utilizar o método IHS para o algoritmo do *pansharpening*, a imagem resultado apresenta anomalias espectrais, pois esta transformação não oferece pesos para que possa ajustar a influência das cores na imagem, como a transformação de Brovey (Meneses et al, 2012, p. 135). Como exemplo, foi utilizada a mesma imagem da Figura 28 como entrada

para o *pansharpening*, resultando na Figura 30. Observa-se uma tendência da cor azul na imagem.

Figura 30 - Resultado do pansharpening



Fonte: Elaborado pelo Autor, 2023

O resultado de uma imagem que foi processada pelo *pansharpening* é chamada de “imagem fusionada”. A imagem fusionada da Figura 28 é nada mais que, a multiespectral com mais informações espaciais, ou seja, houve um aumento da resolução espacial de 8 metros para 2 metros, tornando os objetos em solo mais fáceis de serem diferenciados.

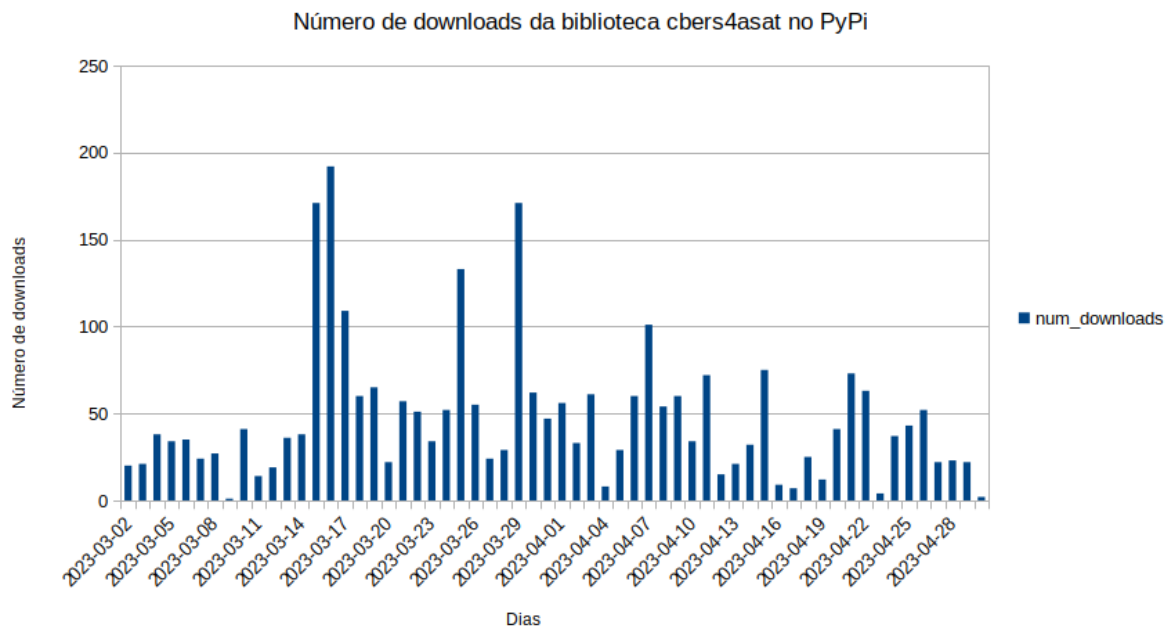
4.5 ESTATÍSTICAS DE USO DA BIBLIOTECA

A biblioteca desenvolvida neste trabalho foi publicada e nomeada de *cbers4sat*. Ela foi publicada oficialmente no dia 12 de agosto de 2022, em sua versão 0.5, visto que a mesma foi versionada localmente até a versão 0.4.

Após o lançamento, a biblioteca foi divulgada no LinkedIn, Instagram e Whatsapp, recebendo diversas visitas ao repositório no GitHub e *downloads* no Pypi. Dessa maneira, foi coletado da base de dados aberta do *Python Package Index*, disponível no Google BigQuery⁴, a quantidade de downloads diários em um prazo de 60 dias a partir do dia dois de março de 2023 e gerado um gráfico em barra, como mostra o Gráfico 1.

4 Dados do Pypi no Google BigQuery disponível em: <https://console.cloud.google.com/bigquery?p=bigquery-public-data&d=pypi&page=dataset>

Gráfico 1 - Gráfico em barra com o número de downloads diários da biblioteca cbers4asat no PyPi num intervalo de 60 dias.



Fonte: Elaborado pelo Autor, 2023

No intervalo de 2 de março até 30 de abril de 2023 houve uma média diária de 47 downloads e um total de 2828 *downloads*. Pode-se concluir que há um interesse pela biblioteca por parte da comunidade de sensoriamento remoto. Portanto, os métodos de engenharia de software aplicados neste trabalho, foram de extrema importância para que os usuários mantivessem a confiança pelo software desenvolvido, além do reconhecimento da necessidade de haver uma ferramenta que consumisse os dados do satélite CBERS-04A utilizando uma linguagem de programação.

5 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

A democratização das imagens de satélite permitiu que as técnicas e aplicações de sensoriamento remoto se integrassem a diversas áreas e, conseqüentemente, aumento na demanda de ferramentas de tecnologia da informação que auxiliem os cientistas a manusear este dado de forma eficiente e simplificado.

O presente trabalho teve como objetivo desenvolver uma biblioteca para a linguagem de programação Python para buscar, baixar e processar imagens do satélite sino-brasileiro CBERS-04A. O software criado mostrou ser eficiente e intuitivo, realizando as tarefas mais comuns da área e, além disso, permitindo a automação de tarefas utilizando recursos da linguagem e integração com outras bibliotecas de terceiros.

Os resultados obtidos indicam que um software especializado para determinada tarefa, permite chegar a um resultado mais facilmente, evitando o estudo de documentações extensas com códigos genéricos e poucos exemplos de utilização.

Dessa maneira, aplicando os conhecimentos de ciência da computação para facilitar o consumo dos dados oriundos de satélites aos especialistas de sensoriamento remoto, contribui para o avanço de estudos da área de geociências, proporcionando novas descobertas para inúmeras áreas, ou melhor, para a ciência.

Para trabalhos futuros, será implementado novas funcionalidades para consultar informações das bandas na base de dados, como, por exemplo, verificar o tamanho de espaço em disco necessário para cada banda, verificar o *checksum* do produto baixado e entre outros. Além disso, poderá ser adicionado a coleção de algoritmos de processamento de digital de imagens algoritmos de correção atmosférica e classificação de solo utilizando redes neurais.

6 REFERÊNCIAS BIBLIOGRÁFICAS

CÂMARA, Gilberto; DAVIS, Clodoveu; MONTEIRO, Antônio M. Vieira. **INTRODUÇÃO À CIÊNCIA DA GEOINFORMAÇÃO**. INPE, Introdução, 2001.

NEEDHAM, Joseph, and Colin A. Ronan. **The Shorter Science and Civilisation in China: Volume 2**. Vol. 2. Cambridge University Press, 1978.

CÂMARA, Gilberto et al. **ANATOMIA DE SISTEMAS DE INFORMAÇÃO GEOGRÁFICA**. INPE, Prefácio, 1996.

SANT'ANA, R. C. G. **Ciclo de vida dos dados: uma perspectiva a partir da ciência da informação**. Informação & Informação, v. 21, n. 2, 2016. Disponível em: <http://www.uel.br/seer/index.php/informacao/article/download/27940/20124>>. Acesso em: 19 dez. 2022.

ELACHI, Charles. *Introduction to the Physics and Techniques of Remote Sensing*, New York, Wiley, 1987.

NOVO, Evlyn Márcia Leão de Moraes. **INTRODUÇÃO AO SENSORIAMENTO REMOTO**. 3. ed. São Paulo: INPE, 2008, 68 p.

GRAHAM, Steve. Remote Sensing: Introduction and History. **Earth Observatory**, Estados Unidos, 17 set. 1999. Disponível em: <https://earthobservatory.nasa.gov/features/RemoteSensing>. Acesso em: 19 dez. 2022>.

NGA in History - Defining Moments: CORONA Program. **National Geospatial - Intelligence Agency**, Estados Unidos. Disponível em: https://www.nga.mil/defining-moments/CORONA_Program.html. Acesso em: 19 dez. 2022>.

BAPTISTA, Gustavo. **Fundamento de Sensoriamento Remoto**. 1. ed. Brasília: Laboratório de Propulsão Digital, 2021, p. 66, 2019.

KUCHKOROV, Temurbek et al. **Satellite image formation and preprocessing methods**. In: 2020 International Conference on Information Science and Communications Technologies (ICISCT). IEEE, 2020. p. 1-4.

JENSEN, J. R. **INTRODUCTORY DIGITAL IMAGE PROCESSING: A REMOTE SENSING PERSPECTIVE**. 4. ed. Carolina do Sul: Pearson, 2015, p. 37.

Ferreira, K. R., Queiroz, G. R., Marujo, R. F. B., and Costa, R. W.: **BUILDING EARTH OBSERVATION DATA CUBES ON AWS**, Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., XLIII-B3-2022, 597–602. Disponível em: <https://doi.org/10.5194/isprs-archives-XLIII-B3-2022-597-2022>, 2022>. Acesso em: 13 de outubro de 2022.

Armono, Haryo; Gema, Billy; Zikra, Muhammad. **The Usage of Geographical Information System in the Selection of Floating Cages Location for Aquaculture at Prigi Bay, Trenggalek Regency, East Java**. IOP Conference Series: Earth and Environmental Science. 2018. Disponível em: https://www.researchgate.net/publication/324049146_The_Usage_of_Geographical_Information_System_in_the_Selection_of_Floating_Cages_Location_for_Aquaculture_at_Prigi_Bay_Trenggalek_Regency_East_Java>. Acesso em: 19 dez. 2022

SOMMERVILLE, Ian. **Engenharia de Software**. Tradução: Luiz Claudio Queiroz. 10. ed. São Paulo: Pearson, 2019, p. 753.

MENESES, P. R. et al. **Introdução ao Processamento de Imagens de Sensoriamento Remoto**. [S.l.]: Universidade Federal de Brasília, 2012. p. 276.

O que é uma API?. Disponível em: <https://aws.amazon.com/pt/what-is/api/>>. Acesso em: 21 dez. 2022.

MOODY, M. **GraphQL vs. REST API architecture - the Startup - medium**. Disponível em: <https://medium.com/swlh/graphql-vs-rest-api-architecture-3b95a77512f5>>. Acesso em: 21 dez. 2022.

CÂMARA, G. **Modelos, linguagens e arquiteturas para bancos de dados geográficos**. São José dos Campos: Instituto Nacional de Pesquisa Espacial, Nov. 1995.

MENG, M.; STEINHARDT, S.; SCHUBERT, A. Application programming interface documentation: What do software developers want? **Journal of technical writing and communication**, v. 48, n. 3, p. 295–330, 2018.

FIELDING, R. T.; TAYLOR, R. N. Principled design of the modern Web architecture. **ACM transactions on Internet technology**, v. 2, n. 2, p. 115–150, 2002.

História do Programa CBERS, 2018. Instituto Nacional de Pesquisas Espaciais (INPE). Disponível em: <http://www.cbers.inpe.br/sobre/historia.php>. Acesso em: 17 maio 2023.

Câmeras Imageadoras CBERS 04A, 2019. Instituto Nacional de Pesquisas Espaciais (INPE). Disponível em: <http://www.cbers.inpe.br/sobre/cameras/cbers04a.php>. Acesso em: 17 maio 2023.

CBERS 04A, 2019. Instituto Nacional de Pesquisas Espaciais (INPE). Disponível em: <http://www.cbers.inpe.br/sobre/cbers04a.php>. Acesso em: 17 maio 2023.

GONZALEZ, Rafael C.; WOODS, Richard C.. **Processamento digital de imagens**. 3. ed. São Paulo: Pearson Education do Brasil Ltda., 2009. 259 p.

COSTA, Carlos A. R. Introdução a processamento digital de imagens: uma abordagem voltada para sensoriamento remoto e funcionalidades do sistema SPRING. Campinas: EMBRAPA-CNPTIA, 1998. 45p. (EMBRAPA-CNPTIA. Relatório Técnico, 4).

SUBRAMANIAN, Harihara. RAJ, Pethuru. **Hands-On: RESTful API Design Patterns and Best Practices**. Burmingham: Packt Publishing, 2019.

GEPANDAS. GeoPandas developers. 2023. Disponível em: <https://geopandas.org/en/stable/>. Acesso em: 24 de maio de 2023.

7 APÊNDICES

Apêndice A - Mock test de um objeto do Requests

```
class MockStacFeatureCollectionResponse:
    # Gabriel Russo
    def __init__(self):
        self.status_code = 200

    # Gabriel Russo
    def raise_for_status(self):
        pass

    # Gabriel Russo
    @staticmethod
    def iter_content(chunk_size):
        yield b"dummydata"

    # Gabriel Russo
    @staticmethod
    def json():
        return {
            "type": "FeatureCollection",
            "features": [
                {
                    "type": "Feature",
                    "id": "ABC123",
                    "collection": "y",
                    "geometry": {...},
                    "bbox": [-48.3106, -16.4178, -47.2492, -15.3637],
                    "properties": {"x": "XYZ"},
                    "assets": {"blue": {"href": "http://test.dev/image.tif"}},
                    "links": [{"x": "y"}],
                }
            ],
        }
```

Apêndice B - Resultado da busca de uma imagem através do método query. O resultado desta figura está incompleto por ser muito longo.

```
{'type': 'FeatureCollection',
  'features': [{'type': 'Feature',
    'id': 'CBERS4A_WPM22912420210529',
    'collection': 'CBERS4A_WPM_L4_DN',
    'geometry': {'type': 'Polygon',
      'coordinates': [[[[-64.7934, -8.22668],
        [-64.7934, -9.2749],
        [-63.7681, -9.2749],
        [-63.7681, -8.22668],
        [-64.7934, -8.22668]]]]},
    'bbox': [-64.7934, -9.2749, -63.7681, -8.22668],
    'properties': {'datetime': '2021-05-29T14:57:55',
      'path': 229,
      'row': 124,
      'satellite': 'CBERS4A',
      'sensor': 'WPM',
      'cloud_cover': 80.0,
      'sync_loss': None,
      'eo:gsd': -1,
      'eo:bands': [{'name': 'pan', 'common_name': 'pan'},
        {'name': 'blue', 'common_name': 'blue'},
        {'name': 'green', 'common_name': 'green'},
        {'name': 'red', 'common_name': 'red'},
        {'name': 'nir', 'common_name': 'nir'}]},
    'assets': {'pan': {'href': 'http://www2.dgi.inpe.br/api/download/TIFF/CBERS4A/2021_05/CBERS_4A_WPM_RAW_2021_05_29.14_53_30_ETC2/229_124_0/4_BC_UTM_WGS84/CBERS_4A_WPM_20210529_229_124_L4_BAND0.tif',
      'type': 'image/tiff; application=geotiff',
      'eo:bands': [0]},
      'pan_xml': {'href': 'http://www2.dgi.inpe
```