MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21d/2023/01.04.23.36-TDI

STORM: PLATAFORMA PARA SUPORTE AO DESENVOLVIMENTO COLABORATIVO DE PESQUISAS REPRODUTÍVEIS EM APLICAÇÕES GEOESPACIAIS

Felipe Menino Carlos

Dissertação de Mestrado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Gilberto Ribeiro de Queiroz, e Rafael Duarte Coelho dos Santos, aprovada em 16 de dezembro de 2022.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34T/48AR43L>

INPE
São José dos Campos
2022

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Coordenação de Ensino, Pesquisa e Extensão (COEPE)
Divisão de Biblioteca (DIBIB)
CEP 12.227-010
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/7348
E-mail: pubtc@inpe.br

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21d/2023/01.04.23.36-TDI

STORM: PLATAFORMA PARA SUPORTE AO DESENVOLVIMENTO COLABORATIVO DE PESQUISAS REPRODUTÍVEIS EM APLICAÇÕES GEOESPACIAIS

Felipe Menino Carlos

Dissertação de Mestrado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Gilberto Ribeiro de Queiroz, e Rafael Duarte Coelho dos Santos, aprovada em 16 de dezembro de 2022.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34T/48AR43L>

INPE
São José dos Campos
2022

Dados Internacionais de Catalogação na Publicação (CIP)

Carlos, Felipe Menino.

C195s Storm: plataforma para suporte ao desenvolvimento colaborativo de pesquisas reprodutíveis em aplicações geoespaciais / Felipe Menino Carlos. – São José dos Campos : INPE, 2022.

xx + 157 p. ; (sid.inpe.br/mtc-m21d/2023/01.04.23.36-TDI)

Dissertação (Mestrado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2022.

Orientadores : Drs. Gilberto Ribeiro de Queiroz, e Rafael Duarte Coelho dos Santos.

1. Reprodutibilidade. 2. Observação da Terra. 3. Trabalho colaborativo. 5. Plataforma Computacional. I.Título.

CDU 004.7:528

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÕES

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

DEFESA FINAL DE DISSERTAÇÃO FELIPE MENINO CARLOS
BANCA N°337/2022, REG. 945820/2020.

No dia 16 de dezembro de 2022, as 09h, por teleconferência, o(a) aluno(a) mencionado(a) acima defendeu seu trabalho final (apresentação oral seguida de arguição) perante uma Banca Examinadora, cujos membros estão listados abaixo. O(A) aluno(a) foi APROVADO(A) pela Banca Examinadora, por unanimidade, em cumprimento ao requisito exigido para obtenção do Título de Mestre em Computação Aplicada. O trabalho precisa da incorporação das correções sugeridas pela Banca e revisão final pelo(s) orientador(es).

Título: "STORM: PLATAFORMA PARA SUPORTE AO DESENVOLVIMENTO COLABORATIVO DE PESQUISAS REPRODUTÍVEIS EM APLICAÇÕES GEOESPACIAIS".

Membros da Banca:

Dra. Karine Reis Ferreira Gomes – Presidente – INPE

Dr. Gilberto Ribeiro de Queiroz – Orientador – INPE

Dr. Rafael Duarte Coelho dos Santos – Orientador – INPE

Dr. Fabiano Morelli – Membro Interno – INPE

Dr. Clodoveu Augusto Davis Junior – Membro Externo – UFMG

Documento assinado eletronicamente por **Karine Reis Ferreira Gomes**, **Tecnologista**, em 20/12/2022, às 15:35 (horário oficial de Brasília), com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Rafael Duarte Coelho dos Santos**, **Coordenador de Pesquisa Aplicada e Desenvolvimento Tecnológico**, em 21/12/2022, às 09:23 (horário oficial de Brasília), com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Gilberto Ribeiro de Queiroz**, **Tecnologista**, em 21/12/2022, às 09:28 (horário oficial de Brasília), com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

"Quem não ouve a melodia, acha maluco quem dança".

Oswaldo Montenegro
em *"Mudar Dói, Não Mudar Dói Muito"*

AGRADECIMENTOS

Primeiramente, gostaria de agradecer a meus pais, Jaqueline e Silvano, por todo o apoio, amor, carinho e paciência que sempre tiveram comigo.

Agradeço aos meus orientadores, Dr. Gilberto Queiroz e Dr. Rafael Santos, pelas ideias, discussões, revisões e por todo apoio do início ao fim deste trabalho.

Agradeço aos pesquisadores e desenvolvedores do projeto *Brazil Data Cube* pelas sugestões, experiências compartilhadas e por disponibilizarem toda a infraestrutura computacional utilizada no desenvolvimento deste trabalho. Obrigado!

Aos colaboradores deste trabalho, Felipe Carvalho, Amita Muralikrishna, Vitor Gomes, Rennan Marujo, Rolf Simões, Rogério Flores e Alber Sanchez que me ajudaram com ideias, revisões e sugestões. Muito obrigado, pessoal!

Aos meus irmãos, Luiz, Pedro e Diogo por todo o apoio, amizade e conversas descontraídas.

Aos amigos, Fernanda, Lucas, Carlos, Weslei, Igor e Hiago, por estarem sempre me acompanhando em minhas jornadas.

Um enorme agradecimento a meu grande amigo Felipe Carvalho, que sempre me incentivou a fazer um bom trabalho. Obrigado por toda sua ajuda.

Agradeço também a amiga, Carolline Tressmann, por sempre me incentivar e me ajudar nos momentos de necessidade.

À todas as pessoas que me ajudaram de alguma forma durante essa jornada do mestrado. Muito obrigado!

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio financeiro.

RESUMO

O conhecimento científico, produzido através de estudos e pesquisas, é uma das bases para o avanço tecnológico e social que faz do mundo um lugar melhor. A aplicação desse conhecimento, nos últimos anos, possibilitou, por exemplo, a melhoria nas formas de comunicação humana, sendo também utilizado para o desenvolvimento de remédios e vacinas. Na área de Observação da Terra, os pesquisadores realizam estudos que produzem informações sobre a superfície do planeta, suas mudanças, fenômenos e os impactos causados por atividades antrópicas. Tais informações podem ser utilizadas para a formulação de políticas públicas que melhoram o planejamento de instituições no gerenciamento dos recursos e fenômenos naturais e no aumento da resiliência das sociedades a desastres naturais. Para que essas informações sejam confiáveis, os pesquisadores, aplicando o método científico, produzem resultados e os compartilham com a comunidade científica. Assim, caso resultados com bases teóricas e práticas frágeis sejam detectados, esses são corrigidos, evitando a disseminação de conhecimentos inválidos. Esse processo, na comunidade científica, cria um mecanismo de autocorreção que a mantém consistente. A reprodutibilidade é o elemento central desse processo para dar a credibilidade ou para contestar resultados. Entretanto, trabalhos recentes indicam uma grande dificuldade da comunidade científica em reproduzir pesquisas, inclusive as baseadas em métodos computacionais. Por isso, vários pesquisadores têm trabalhado na criação de soluções que auxiliam no desenvolvimento de trabalhos reprodutíveis. Entre as soluções propostas, destacam-se as plataformas computacionais, capazes de disponibilizar ferramentas de alto nível que auxiliam a produção de resultados reprodutíveis. No entanto, no âmbito da comunidade de Observação da Terra, há necessidades específicas, como o uso de grandes volumes de dados e o trabalho multidisciplinar e colaborativo. Essas propriedades não são garantidas nas plataformas já propostas. Nesse contexto, este trabalho propõe a Storm, uma plataforma para facilitar a produção de pesquisas reprodutíveis. Para validar os conceitos e ideias introduzidas nessa plataforma, foi realizada uma implementação demonstrativa e diversos experimentos aplicados na área de Observação da Terra. Os resultados indicam que a ferramenta produzida é capaz de auxiliar e facilitar a gestão dos artefatos de pesquisa para garantir seu compartilhamento de maneira reprodutível.

Palavras-chave: Reprodutibilidade. Observação da Terra. Trabalho colaborativo. Plataforma Computacional.

STORM: PLATFORM TO SUPPORT THE DEVELOPMENT OF REPRODUCIBLE AND COLLABORATIVE GEOSPATIAL APPLICATIONS

ABSTRACT

Scientific knowledge, produced through studies and research, is one of the bases for the technological and social advancement that makes the world a better place to live. In recent years, the application of this knowledge has made possible, for example, the improvement in human communication, and is also used for developing medicines and vaccines. In Earth Observation, researchers carry out studies that produce information about the planet, its changes, phenomena, and the impacts caused by human activities. Such information can be used to develop public policies that improve the planning of institutions in managing resources and natural phenomena and to increase the resilience of societies to natural disasters. For this information to be reliable, researchers apply the scientific method to produce the results and share them with the scientific community. Thus, if results with weak theoretical and practical bases are detected, they are corrected, avoiding the dissemination of invalid knowledge. This process in the scientific community creates a self-correcting mechanism that keeps it consistent. Reproducibility is the central element of this process to give credit or contest results. However, recent work indicates that the scientific community has problems reproducing research, including research based on computational methods. For this reason, several researchers have been working to create solutions that assist in developing reproducible work. Among the proposed solutions are computational platforms capable of providing high-level tools to produce reproducible results. However, specific needs exist within the Earth Observation community, such as using large volumes of data and multidisciplinary and collaborative work. These properties are not guaranteed in the already proposed platforms. In this context, this work proposes Storm, a platform to facilitate the production of reproducible research. To validate the concepts and ideas introduced in this platform, a demonstrative implementation and several experiments applied to Earth Observation were carried out. The results indicate that the tool produced can assist and facilitate the management of research artifacts to ensure their sharing in a reproducible way.

Keywords: Reproducible Research. Earth Observation. Computational Platform. Collaborative Research.

LISTA DE FIGURAS

Pág.

2.1 Espectro de reprodutibilidade. 8
2.2 Ciclo de vida da Pesquisa Científica. 11
2.3 Exemplo de um `Jupyter Notebook` mesclando narrativa, descrições das operações, código fonte e visualização dos resultados. 14
2.4 Comparação de resultados de mapas de LULC: (a) Versão antiga do *Software*; (b) Versão nova do *Software*; (c) Diferença dos resultados. . . . 15
2.5 Exemplo de `Dockerfile` utilizado para a criação de uma *Image*. 18
2.6 Componentes de uso da CWL. 21
2.7 Exemplo de utilização de um RC por meio de estrutura de diretórios. . . 24
2.8 Fluxo de funcionamento do `ReproZip`. 29
2.9 Exemplo de *Packing* (A) e *Unpacking* (B) com a CLI do ReproZip para uma tarefa de visão computacional. 31
2.10 Visão geral da arquitetura do `Reana`. 32
2.11 Ciclo de vida de um ERC na plataforma `o2r`. 34
2.12 Visão geral da arquitetura do `o2r`. 35
2.13 Visão geral da arquitetura do `Gigantum`. 36
2.14 Fluxo de operação da plataforma `Whole Tale`. 38
2.15 Visão geral dos componentes da plataforma Whole Tale. 38
2.16 Diferença entre coleção de imagens e cubos de dados. 40
2.17 Fluxo geral de uso dos EODC. 41
2.18 Exemplo de aplicação de EODCs para a geração de mapas de LULC. . . 42

3.1 Arquitetura resumida da `Plataforma BDC`. 44
3.2 Relacionamento das entidades da `Plataforma Storm`. 49
3.3 Fluxo geral de funcionamento da `Plataforma Storm`. 50
3.4 Fluxo de utilização da `Plataforma Storm` na produção de pesquisa reprodutível. 51
3.5 Exemplo de *Research Workflow* com o fluxo de classificação de desmatamento apresentado por Sanchez et al. (2019). 53
3.6 Fluxo de utilização da `Plataforma Storm` para a reprodução/replicação de pesquisa. 54
3.7 Arquitetura conceitual da `Plataforma Storm`. 57

4.1 Arquitetura da implementação demonstrativa da `Plataforma Storm`. . . 64

4.2 Ciclo de vida de um *Execution Compendium* definido no `Storm Compendium`. 71
4.3 Exemplo de ligação de um *Research Workflow* implementado no `Storm Workflow`. 73
4.4 Modelo de operação assíncrona adotada no `Storm WS`. 76
4.5 Modelo lógico do banco de dados do `Storm WS`. 78
4.6 Infraestrutura de dados do `Storm WS`. 80
4.7 Esquema de integração entre `Storm WS` e serviços externos. 81
4.8 Exemplo de interface do `Storm Portal`. 83
4.9 Fluxo de utilização do `Storm Workbench`. 85
4.10 Fluxo de inicialização do espaço de trabalho com o `Storm Workbench`. . . 86
4.11 Fluxo de execução de um *script* via `Storm Workbench`. 88
4.12 Exemplo de utilização do `Stormfile` via `Storm Workbench`. 89
4.13 Exemplo de execução reprodutível do fluxo de classificação apresentado por Simoes et al. (2020) com execução de *script* em paralelo. 90
4.14 Fluxo de reprodução de um *workflow* via `Storm Workbench`. 92
4.15 Fluxo de utilização do `Storm Environment`. 94
4.16 Exemplo da interface de busca do `Storm Searcher`. 96
4.17 Exemplo da interface do `Storm Viewer`. 97

5.1 Metodologia de desenvolvimento dos experimentos deste trabalho. 99
5.2 Área de estudo localizada no Oeste da Bahia utilizada no experimento 1. 103
5.3 Metodologia de desenvolvimento do experimento 1. 105
5.4 Cenário de pesquisa do experimento 1. 106
5.5 Fluxo de operação do experimento 1. 107
5.6 Comparação dos mapas de LULC gerados. 108
5.7 Esquerda: Área de estudo. Direita: Cena da área de estudo extraída do cubo de dados Landsat-8/OLI (07/2019). 112
5.8 Exemplo de imagens fração para a área de estudo. 113
5.9 Fluxo de definição dos *endmembers*. 114
5.10 Metodologia de desenvolvimento do experimento 2. 115
5.11 Cenário de pesquisa do experimento 2. 116
5.12 Fluxo de execução utilizado no experimento 2. 117
5.13 Fluxo de reprodução do experimento 2. 118
5.14 Comparação de igualdade entre os arquivos originais e reproduzidos. . . . 120
5.15 Comparação dos resultados (Original X Reprodução). 121
5.16 Comparação de igualdade entre os arquivos originais e reproduzidos quando controle de semente de valores pseudo-aleatórios não é definido. . 122

LISTA DE TABELAS

Pág.

3.1 Tabela comparativa entre as plataformas de reprodutibilidade e os requisitos da `Plataforma Storm`. 47

4.1 Tabela de descrição dos campos de metadados utilizados na definição de um *Research Project*. 68
4.2 Tabela de descrição dos campos de metadados utilizados na definição de um *Execution Compendium*. 69
4.3 Tabela de descrição dos campos de metadados utilizados na definição de um *Research Workflow*. 72

5.1 Tabela de comparação entre dados originais e dados da reprodução. . . . 119

LISTA DE ABREVIATURAS E SIGLAS

API	–	*Application Programming Interface*
APS	–	*American Physical Society*
BDC	–	Brazil Data Cube
CAP	–	Computação Aplicada
CLI	–	*Command Line Interface*
CWL	–	*Common Workflow Language*
DAG	–	*Directed Acyclic Graph*
EO	–	*Earth Observation*
EODC	–	*Earth Observation Data Cube*
ERC	–	*Executable Research Compendium*
FAIR	–	*Findable, Accessible, Interoperable, Reusable*
HTML	–	*HyperText Markup Language*
HTTP	–	*HyperText Transfer Protocol*
INPE	–	Instituto Nacional de Pesquisas Espaciais
LULC	–	*Land Use and Land Cover*
ML	–	*Machine Learning*
MLME	–	Modelo Linear de Mistura Espectral
o2r	–	*Opening Reproducible Research*
OCI	–	*Open Container Initiative*
PDF	–	*Portable Document Format*
RC	–	*Research Compendium*
SR	–	Sensoriamento Remoto
SO	–	Sistema Operacional
Storm	–	SpatioTemporal Open Research Manager
SWMS	–	*Scientific Workflow Management System*

SUMÁRIO

Pág.

1 INTRODUÇÃO 1
1.1 Motivação 3
1.2 Objetivos 4
1.3 Contribuições 5

2 REVISÃO DA LITERATURA 7
2.1 Terminologias de Pesquisa Reprodutível 7
2.2 Pesquisa Aberta e Reprodutível 8
2.3 Ferramentas e técnicas para suporte à reprodutibilidade 10
2.3.1 *Computational Notebooks* 12
2.3.2 Gerenciamento de ambiente computacional 14
2.3.3 Ferramentas de *workflow* científico 19
2.3.4 *Research Compendium* 22
2.3.5 Repositórios digitais de dados de pesquisa 25
2.4 Plataformas computacionais para pesquisa reprodutível 28
2.5 Cubos de dados de Observação da Terra 39

3 SPATIOTEMPORAL OPEN RESEARCH MANAGER 43
3.1 Requisitos 45
3.2 Entidades 48
3.3 Fluxo de utilização 50
3.4 Arquitetura 55

4 IMPLEMENTAÇÃO DEMONSTRATIVA DA PLATAFORMA PROPOSTA 61
4.1 Princípios de design 61
4.2 Arquitetura da plataforma 63
4.3 Componentes *Server Side* 64
4.3.1 Storm WS 65
4.3.1.1 Modelagem de dados 76
4.3.1.2 Infraestrutura de dados 79
4.3.2 Serviços externos 80
4.4 Componentes *Client Side* 83

4.4.1 Storm Workbench . . . 84
4.4.2 Storm Environment . . . 93
4.4.3 Outras aplicações cliente . . . 95
4.5 Disponibilidade de código . . . 97

5 EXPERIMENTOS . . . 99
5.1 Produção de mapas de Uso e Cobertura de Terra com cubos de dados de Observação da Terra e algoritmos de Aprendizado de Máquina . . . 101
5.1.1 Contextualização . . . 102
5.1.2 Materiais e métodos . . . 103
5.1.3 Metodologia . . . 104
5.1.4 Cenário de pesquisa . . . 105
5.1.5 Resultados e discussões . . . 106
5.2 Detecção de áreas desmatadas utilizando modelo de mistura e técnicas de Aprendizado de Máquina . . . 109
5.2.1 Contextualização . . . 110
5.2.2 Materiais e métodos . . . 111
5.2.3 Metodologia . . . 115
5.2.4 Cenário de pesquisa . . . 116
5.2.5 Resultados e discussões . . . 117

6 CONSIDERAÇÕES FINAIS . . . 125
6.1 Trabalhos futuros . . . 126

REFERÊNCIAS BIBLIOGRÁFICAS . . . 127

APÊNDICE A - USER SCENARIO: PRODUÇÃO DE PESQUISA REPRODUTÍVEL . . . 149

APÊNDICE B - USER SCENARIO: NECESSIDADE DE REPLICAÇÃO DE PESQUISA REPRODUTÍVEL. . . . 155

1 INTRODUÇÃO

O avanço da ciência transforma a sociedade e sua economia através de um processo contínuo de investigação, que obtém e melhora os conhecimentos sobre o mundo e seus fenômenos (NATURE, 2019; NATIONAL ACADEMIES OF SCIENCES, 2019; LOPES et al., 2020). Diante disso, o conhecimento científico serve como base para a produção de tecnologias e métodos que garantem a melhora na qualidade de vida, gestão de recursos naturais e conservação da biodiversidade do planeta (RULL, 2014; MURALIKRISHNA; MANICKAM, 2017).

A produção do conhecimento e o avanço da ciência são realizados pelos cientistas através do desenvolvimento de pesquisas. Após finalizadas, essas são compartilhadas em publicações científicas (NüST; EGLEN, 2021). O propósito desse compartilhamento é apresentar para outros cientistas os resultados obtidos e provar que os mesmos são válidos. Com isso, as ideias e os resultados de uma publicação podem ser colocados à prova por outros investigadores, permitindo que ambos sejam corroborados e melhorados ou rejeitados e corrigidos. Isto forma, na comunidade científica, um mecanismo de autocorreção, que identifica e corrige problemas e afirmações inválidas (ALLISON et al., 2016).

Para esse processo científico, a reprodutibilidade é um requisito central (FREIRE et al., 2014; IVIE; THAIN, 2018). Com ela, é possível garantir que a autocorreção seja realizada e a validade das afirmações de uma publicação e seus resultados possam ser avaliados. Por conseguinte, a reprodutibilidade está relacionada à transparência e qualidade da publicação, tendo esta que descrever e especificar todos os materiais e passos necessários para a obtenção dos resultados (MUNAFÒ et al., 2017; BRUNSDON; COMBER, 2020).

Entretanto, garantir a reprodutibilidade em pesquisas que utilizam métodos computacionais é um desafio (PENG, 2011; EASTERBROOK, 2014; BHATT et al., 2020). A descrição e captura detalhada dos elementos computacionais utilizados, como o ambiente computacional subjacente, os métodos implementados e os conjuntos de dados utilizados, podem ser uma tarefa árdua e complexa, que exige tempo, atenção e conhecimentos especializados dos cientistas para serem realizadas (NATURE, 2012; KORNFELD; TITUS, 2016). Isto dificulta a geração de descrições completas, tornando limitada a reprodutibilidade das publicações científicas. Konkol et al. (2018), por exemplo, apresentam a tentativa de reprodução de 41 artigos que utilizam métodos computacionais, selecionados de duas revistas científicas. Os autores mostram que apenas dois artigos puderam ser reproduzidos com o material subjacente disponibi-

lizado, outros quatro artigos foram considerados irreprodutíveis por apresentarem problemas que não puderam ser resolvidos. Nos demais artigos, a reprodução foi feita mediante modificações que exigiram conhecimento especializado e ajuda dos autores originais, para adequações no código e ambiente de execução.

A falta ou dificuldade de reprodutibilidade reflete negativamente na ciência e sua credibilidade (RUSSELL, 2013). Por vezes, pode-se associar aos trabalhos científicos desenvolvidos, principalmente quando concebidos com dinheiro público, como tempo e investimento gastos sem retorno, tanto para comunidade científica, como para a sociedade no geral (FREEDMAN et al., 2015; EISNER, 2018). Para Stark (2018), a irreprodutibilidade descaracteriza a ciência, tirando-a do padrão "mostre-me o que foi feito" e substituindo por "confie em mim, funciona", fazendo com que o trabalho publicado passe a ser apenas uma propaganda do estudo realizado (CLAERBOUT; KARRENBACH, 1992).

Para entender as experiências da própria comunidade científica com o problema, Baker (2016) realizou um levantamento *online* com 1.576 cientistas. Nos resultados apresentados, mais de 70% dos participantes afirmaram não conseguir reproduzir experimentos de outros cientistas, ao mesmo tempo que mais da metade dos entrevistados informou não conseguir reproduzir os próprios experimentos. Outra revelação feita pela autora é que, para 52% dos entrevistados, o estado de reprodução das publicações representa uma "crise de reprodutibilidade".

Em resposta a esses problemas, a comunidade científica tem trabalhado na criação de soluções que auxiliam os cientistas na produção de trabalhos reprodutíveis considerando diversas abordagens. Por exemplo, Sandve et al. (2013), Stodden e Miguez (2014) e Rule et al. (2019) criam listas de boas práticas para a produção de pesquisa computacional e utilização de ferramentas que auxiliam a reprodutibilidade. Outro exemplo, são as abordagens que permitem melhor verificação e garantia da reprodutibilidade das pesquisas, como as apresentadas por Munafò e Smith (2018) e Raphael et al. (2020). Periódicos, como a *Nature* e *Science*, também tem auxiliado no processo, criando diretrizes[1] para a garantia de reprodutibilidade das publicações realizadas. Outro tipo de solução que tem sido desenvolvida são as plataformas computacionais (KONKOL et al., 2020). Essas, unificam ferramentas que auxiliam a reprodutibilidade, entregando aos pesquisadores ambientes e funcionalidades de alto nível, que evitam a sobrecarga de gerenciamento e controle nas pesquisas.

[1]Na política editorial da *Science*, estão presentes exemplos de diretrizes, como o acesso ao código e dados: <https://www.sciencemag.org/authors/science-journals-editorial-policies>. Acesso em: 04/12/2022

1.1 Motivação

A produção de pesquisa colaborativa é cada vez mais comum. Essas são criadas com equipes multidisciplinares, que realizam o desenvolvimento de novas ideias e abordagens para a solução de problemas (FORTUNATO et al., 2018). À vista disso, tem-se pesquisas com quantidades crescentes de etapas, com relações e interpretações complexas (YANIV et al., 2017). Para a reprodutibilidade, isso representa um desafio, uma vez que a coordenação dessas pesquisas e seu fluxo de processamento deve considerar o mapeamento e controle das ações de todos os membros envolvidos. Quando as transformações e premissas aplicadas por qualquer um dos membros deixa de ser considerada, cria-se a impossibilidade de reprodução completa dos resultados (KEDRON et al., 2020). Além disso, sem o uso de ambientes padronizados entre todos os usuários nas etapas de processamento, há o aumento na complexidade das descrições dos elementos computacionais utilizados.

Ao mesmo tempo, na comunidade de Observação da Terra (EO, do inglês *Earth Observation*), a crescente adoção de políticas públicas de dados abertos, tornou disponível uma grande quantidade de imagens de diferentes satélites, como CBERS-4, Landsat-8 e Sentinel-2 (SOILLE et al., 2018). Essas têm sido utilizadas pelos cientistas da comunidade de EO para a produção de aplicações inovadoras (GIULIANI et al., 2019). Contudo, esse grande volume de dados também agrava as discussões em torno da reprodutibilidade, já que podem aumentar a quantidade de trabalhos que não podem ser reproduzidos (CRAGLIA; NATIVI, 2018).

Embora hajam diferentes plataformas para o auxílio à produção de pesquisa reprodutível, são poucas as soluções que permitem o desenvolvimento colaborativo considerando grandes volumes de dados. Em uma análise da literatura, foram identificadas cinco plataformas potenciais. A plataforma Gigantum (BURNS et al., 2018) tem capacidades de trabalho colaborativo, bem como provisionamento e padronização de ambientes de trabalho entre vários usuários. Entretanto, a coordenação do fluxo de execução fica a cargo dos pesquisadores e o compartilhamento de dados é limitado por uma cota, removida mediante uso de serviços pagos. A WholeTale (BRINCKMAN et al., 2019), possui funcionamento semelhante ao Gigantum, não tendo custos de utilização e funcionalidades de compartilhamento associados. Outras plataformas exploradas como ReproZip (CHIRIGATI et al., 2016), Reana (ŠIMKO et al., 2019) e o2r (NüST et al., 2017) auxiliam os pesquisadores no controle do fluxo de processamento e uso de grandes volumes de dados, mas não oferecem capacidades de trabalho colaborativo.

No contexto brasileiro, o Instituto Nacional de Pesquisas Espaciais (INPE), possui projetos que podem se beneficiar de iniciativas que facilitem a reprodutibilidade, como os subprojetos do Monitoramento Ambiental dos Biomas Brasileiros (FUNDO AMAZÔNIA, 2021). Em específico, o subprojeto Brazil Data Cube (BDC), que desenvolve pesquisas e tecnologias para a extração de informações desse grande volume de dados de imagens de satélite. Essas atividades são feitas através do trabalho colaborativo e multidisciplinar entre pesquisadores, alunos e desenvolvedores.

1.2 Objetivos

A hipótese utilizada para o desenvolvimento deste trabalho é que a reprodutibilidade de pesquisas colaborativas e baseadas em grandes dados de EO pode ser facilitada e melhor assegurada por meio de uma plataforma computacional enriquecida com elementos que suportem as práticas necessárias à Ciência Aberta.

Considerando a hipótese definida, a relevância das práticas reprodutíveis, e as demandas da comunidade de EO no desenvolvimento científico e tecnológico neste trabalho, é definido como objetivo geral projetar uma plataforma que forneça aos pesquisadores funcionalidades que auxiliem o desenvolvimento de pesquisas baseadas em grandes dados de EO de forma reprodutível e colaborativa.

Também são definidos os seguintes objetivos específicos:

- Propor a arquitetura de uma plataforma de suporte a reprodutibilidade, que forneça capacidades e operações para a produção e o gerenciamento de pesquisas em EO em cenários de desenvolvimento colaborativo e multidisciplinar;

- Desenvolver uma implementação demonstrativa da arquitetura da plataforma proposta, de modo a evidenciar suas principais características que auxiliam à solução das questões relacionadas à reprodutibilidade em cenários colaborativos;

- Validar as abordagens empregadas na arquitetura da plataforma proposta através da realização de experimentos de produção e reprodução de pesquisas com base nas funcionalidades disponíveis na implementação demonstrativa.

1.3 Contribuições

Durante o curso da pesquisa foram realizadas diversas atividades acadêmicas e tecnológicas. Sendo assim, tem-se que as contribuições deste trabalho abrangem ambos os temas. Abaixo, faz-se a apresentação das contribuições associadas a cada um desses tópicos:

a) Especificação e implementação da `Plataforma Storm`[2], junto a um ecossistema de ferramentas reutilizáveis, que auxiliam na criação de sistemas, ferramentas e experimentos reprodutíveis;

b) Especificação e implementação da biblioteca de código `sitscwl`[3], que permite a criação de *workflows* distribuídos e paralelos de classificação reprodutível de cubos de dados de EO para a geração de mapas de Uso e Cobertura da Terra. A ferramenta faz a criação do *workflow* utilizando a linguagem de *workflow* CWL;

c) Especificação e implementação da ferramenta `JupyterHub Docker`[4], a qual permite a configuração do `Jupyter Hub` como uma ferramenta de provisionamento de ambientes reprodutíveis. Essa ferramenta é utilizada no âmbito do Projeto Brazil Data Cube;

d) Produção de *Research Compendium* e co-autor dos seguintes artigos

- *A platform for land use and land cover data integration and trajectory analysis* (ZIOTI et al., 2022);
- *A reproducible and replicable approach for harmonizing Landsat-8 and Sentinel-2 images.*

e) Participação na produção de artigos como autor ou co-autor. Os artigos nos quais se teve participação são listados abaixo:

- *Diffuse Attenuation of Clear Water Tropical Reservoir: A Remote Sensing Semi-Analytical Approach* (CURTARELLI et al., 2020);
- *Integração dos ambientes Brazil Data Cube e Open Data Cube* (CARLOS et al., 2020);

[2]Disponível em: <`https://github.com/storm-platform/`>. Acesso em: 29/11/2022
[3]Disponível em: <`https://github.com/sitscwl/sitscwl`>. Acesso em: 29/11/2022
[4]Disponível em: <`https://github.com/brazil-data-cube/jupyterhub-docker`>. Acesso em: 29/11/2022

- *Integrating Open Data Cube and Brazil Data Cube Platforms for Land Use and Cover Classifications* (CARLOS et al., 2021);
- *Accessing and processing Brazilian Earth Observation Data Cubes with the Open Data Cube Platform* (GOMES et al., 2021);
- *Hybrid Semi-Analytical Algorithm for Estimating Chlorophyll-A Concentration in Lower Amazon Floodplain Waters* (FLORES JÚNIOR et al., 2022).

2 REVISÃO DA LITERATURA

Neste Capítulo são apresentados os conceitos necessários para o entendimento deste trabalho. Na Seção 2.1, são apresentadas as terminologias da Pesquisa Reprodutível. Em seguida, na Seção 2.2, são apresentados os conceitos de Pesquisa Aberta e Reprodutível. As ferramentas utilizadas para auxílio à reprodutibilidade são apresentadas na Seção 2.3. Por fim, na Seção 2.4, são apresentadas as plataformas computacionais para pesquisa reprodutível.

2.1 Terminologias de Pesquisa Reprodutível

A realização de uma Pesquisa Reprodutível enfrenta diversas barreiras (LEVEQUE et al., 2012). Um problema fundamental está no uso das terminologias de reprodução e replicação (HEROUX et al., 2018). Diferentes grupos de pesquisa, nas diversas áreas do conhecimento, utilizam as mesmas terminologias para determinar conceitos que são contraditórios entre si (BARBA, 2018).

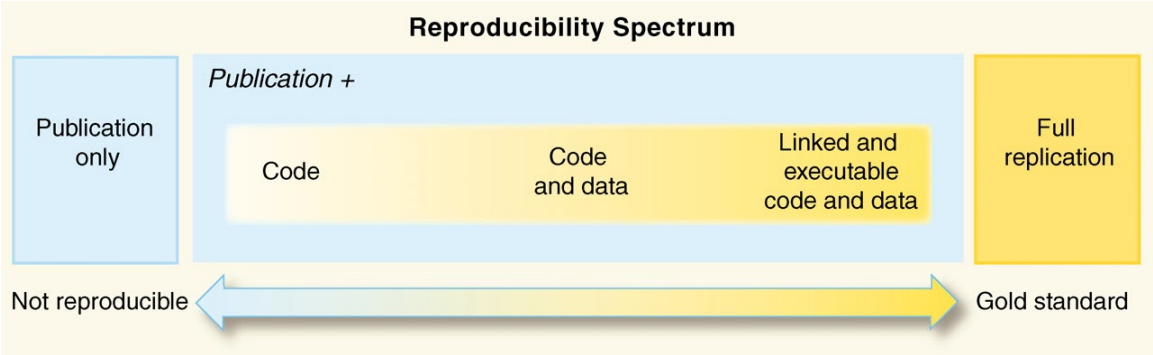
Neste trabalho, com base nos estudos apresentados por Barba (2018) e posteriormente expandidos pela National Academies of Sciences (2019) é feita a utilização da terminologia apresentada por Claerbout e Karrenbach (1992). Esta, com base nos artigos citados, é a terminologia mais amplamente utilizada em diversas áreas, como a Computação Científica, Econometria, entre outras.

Na terminologia utilizada, as palavras reprodução e replicação são definidas de formas distintas. A reprodução significa a geração de resultados idênticos aos apresentados em um artigo de publicação científica, mediante a utilização dos mesmos códigos e dados empregados pelos autores originais. Neste caso, nenhuma nova descoberta é realizada ou confirmada, apenas se tem a possibilidade de chegar nos mesmos resultados. Na replicação, porém, faz-se o uso de diferentes dados, códigos e métodos para a geração de resultados que possam responder a mesma pergunta científica do artigo original. Nesta, tem-se a confirmação das reivindicações de um artigo com base em novos testes e a possibilidade de geração de novos conhecimentos.

Essas definições indicam que, no momento da publicação de um artigo, os autores devem se preocupar com a quantidade e qualidade das informações que são publicadas. Neste contexto, Peng (2011) introduz a ideia de que atingir a reprodução e a replicação, possui requisitos que podem ser representados através de um "espectro de reprodutibilidade". No espectro, ilustrado na Figura 2.1, estão presentes dois extremos. De um lado, estão os trabalhos irreprodutíveis, enquanto do outro estão os

replicáveis, considerados pelo autor como o "padrão ouro" de pesquisa. Entre esses, estão os diversos níveis de reprodução que um artigo pode assumir.

Figura 2.1 - Espectro de reprodutibilidade.



Fonte: Peng (2011).

A produção de trabalhos mais reprodutíveis, seguindo o espectro, depende dos autores realizarem a publicação de mais elementos que ajudem no entendimento do trabalho realizado. Por exemplo, um autor pode publicar junto ao artigo científico, apenas o código fonte. Isto representa um artigo mais reprodutível que aquele onde tem-se apenas o documento estático, uma vez que, mesmo sem permitir a execução do código, oferece mais detalhes que podem ser utilizados no entendimento e validação do trabalho realizado. Esta necessidade de adição de mais materiais segue até o final do espectro à direita, em que são fornecidos elementos suficientes para a execução do código, geração dos resultados e testes considerando variações nos métodos e entradas utilizadas.

2.2 Pesquisa Aberta e Reprodutível

Keith (1986), em um artigo pioneiro, faz um apelo à comunidade de Visão Computacional de sua época, o autor questionava a falta de acesso aos códigos com a implementação das técnicas apresentadas nos artigos publicados. Segundo Keith (1986), esta falta de acesso aos materiais, em longo prazo, poderia retardar a velocidade dos avanços da área. Isto devido à duplicação de esforços para a aplicação de técnicas já apresentadas na literatura e implementadas em trabalhos anteriores.

Segundo a American Physical Society (APS), ciência pode ser definida como "o empreendimento sistemático de reunir conhecimento sobre o universo, organizando e condensando estes em leis e teorias testáveis" (SOCIETY, 1999). Ainda segundo a APS, o sucesso e a credibilidade da ciência tem raízes na exposição de ideias e a autocorreção dos conceitos e princípios com base em estudos que apresentem mais e melhores evidências sobre um fato.

O processo de autocorreção, feito na comunidade científica, é possível graças à comunicação realizada entre os pesquisadores. Assim, quem produz um novo resultado científico, o coloca à disposição da comunidade, que então realiza testes de reprodução e replicação, de forma independente ao pesquisador que o criou (PENG, 2011; POPPER, 2013). Isto permite que as evidências apresentadas possam ser fortalecidas ou mesmo corrigidas, se necessário (COMUNIDADE TURING WAY et al., 2019).

Comumente a comunicação científica é realizada através da publicação de artigos científicos (NüST; EGLEN, 2021). Esses realizam a descrição do trabalho desenvolvido, apresentam as etapas metodológicas e os resultados obtidos em um documento estático. Porém, como pode ser observado por Keith (1986), para o contexto em que há a utilização de métodos computacionais, tal forma de comunicação não é o suficiente para permitir que outros utilizem e validem as afirmações realizadas. Gentleman e Lang (2007) afirmam que a representação de um código de computador em uma linguagem natural raramente é satisfatória para quem a produz ou consome. Segundo os autores, a linguagem natural não é sucinta e exata o suficiente, fazendo com que certos detalhes tenham de ser deixados de lado durante a criação das descrições.

Atualmente, tal comportamento é agravado por conta do aumento das capacidades computacionais e a possibilidade do desenvolvimento de pesquisas em larga escala, baseadas em métodos computacionais (PENG, 2011; LEVEQUE et al., 2012; NÜST; PEBESMA, 2020). Assim, além do problema apresentado por Gentleman e Lang (2007), há também questões com a complexidade dos ambientes criados para o desenvolvimento das pesquisas. Nestes estão presentes diversos documentos, configurações de *hardware* e *softwares* específicos, além das estruturas de organização e formato dos dados. Quando há a necessidade da sumarização de todo esse ambiente em um documento estático, perde-se toda a riqueza de conhecimento gerado para sua produção e utilização (NüST; EGLEN, 2021).

Neste contexto, Jon Claerbout, o primeiro cientista a utilizar a frase "Pesquisa Reprodutível" em uma publicação científica (BARBA, 2016), especificou que, "um artigo sobre Ciência da Computação em uma publicação científica não representa a pes-

quisa em si, é uma mera propaganda. O atual conhecimento da pesquisa científica, que utiliza de métodos computacionais, está no ambiente completo de desenvolvimento e nas instruções utilizadas para a geração das figuras presentes no artigo científico" (CLAERBOUT; KARRENBACH, 1992).

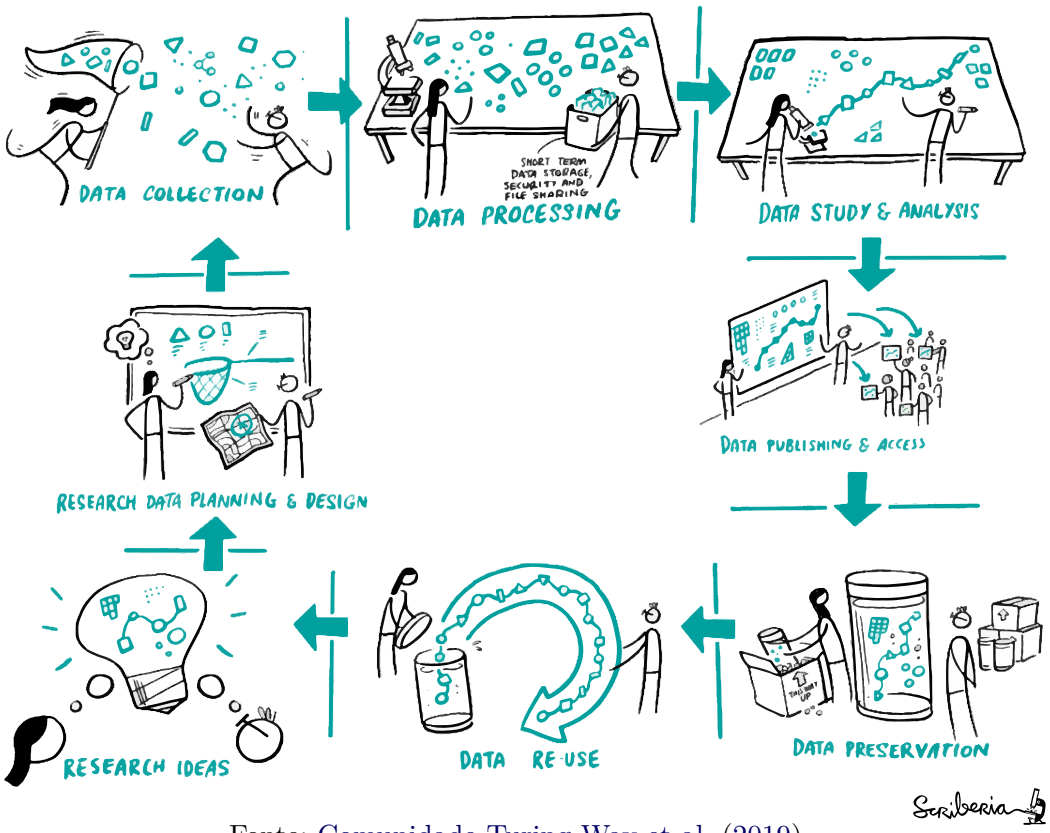
Desta forma, corroborando com Claerbout e Karrenbach (1992), diversos autores sugerem que a solução para este problema é a aplicação das práticas de Ciência Aberta (STODDEN et al., 2016; KONKOL et al., 2018; NÜST; PEBESMA, 2020; KEDRON et al., 2020). Nesta abordagem, é enfatizado o rigor para o desenvolvimento das pesquisas e a responsabilidade dos pesquisadores em fazer com que os ativos científicos, gerados durante o desenvolvimento de uma pesquisa, possam ser utilizados por terceiros (BEZJAK et al., 2018; COMUNIDADE TURING WAY et al., 2019). Munafò et al. (2017) apresentam alguns exemplos de iniciativas abertas utilizadas para auxiliar a reprodutibilidade, como as práticas de dados e materiais abertos e o compartilhamento do código fonte utilizado para a geração dos resultados. Há também autores como Nüst et al. (2017), que incentivam e oferecem mecanismos para a publicação do ambiente computacional subjacente ao código utilizado para a geração dos resultados.

Essas práticas formam a chamada Pesquisa Aberta e Reprodutível (PONTIKA et al., 2015; KONKOL et al., 2018) e tem ganhado cada vez mais aderência em diferentes áreas. Por exemplo, na Ciência da Computação (PENG, 2011; STODDEN, 2012), Geociências (OSTERMANN et al., 2020) e Biologia (LEWIS et al., 2018).

2.3 Ferramentas e técnicas para suporte à reprodutibilidade

O processo de desenvolvimento de um projeto de pesquisa, conforme sumarizado na Figura 2.2, é composto por várias etapas de diferentes níveis organizacionais. Primeiro, tem-se a necessidade da definição do objeto de estudo, bem como as perguntas científicas que poderão ser respondidas com o trabalho realizado. Em seguida, com base na revisão da literatura e experiência dos pesquisadores, faz-se o planejamento e a definição da metodologia que será utilizada. Após essas etapas iniciais de definição, tem-se um ciclo em que dados são coletados, processados e analisados. Então, após esse ciclo iterativo, os pesquisadores realizam o compartilhamento dos conhecimentos gerados. Para isso, os resultados são postos em artigos, ao mesmo tempo, em que os dados são compartilhados e preservados para usos futuros (COMUNIDADE TURING WAY et al., 2019).

Figura 2.2 - Ciclo de vida da Pesquisa Científica.



Fonte: Comunidade Turing Way et al. (2019).

A implementação e aplicação das práticas de Pesquisa Aberta e Reprodutível no fluxo da pesquisa, conforme ilustrado na Figura 2.2, dependem do pesquisador que as desenvolve. Isto, em um primeiro momento, pode representar a necessidade de mudanças no projeto da pesquisa ou mesmo nos hábitos de trabalho do pesquisador (DONOHO, 2010; STODDEN; MIGUEZ, 2014). Também, em etapas específicas da pesquisa, como as que compreendem o processamento, análise e preservação dos dados, pode ser necessária a adoção de técnicas e ferramentas que façam com que a pesquisa produzida possa ser utilizada por outros pesquisadores (BLISCHAK et al., 2019; VUORRE; CRUMP, 2020). Tais técnicas e ferramentas podem também ser utilizadas para reduzir possíveis sobrecargas de trabalho no dia-a-dia do pesquisador que desenvolve práticas reprodutíveis. Neste contexto, Piccolo e Frampton (2016) realizaram uma revisão das principais técnicas e ferramentas que podem ser utilizadas. Complementarmente, Marwick et al. (2018) apresentam técnicas para o comparti-

lhamento de pesquisas reprodutíveis.

Utilizando como base os artigos de revisão citados, foram selecionados quatro tipos de ferramentas e técnicas relevantes para este trabalho: (i) *Computational Notebooks*; (ii) Ambientes computacionais; (iii) Sistema de gerenciamento de *workflow*; e (iv) Unidades para compartilhamento de objetos de pesquisa.

O uso conjunto dessas ferramentas e técnicas, auxilia os pesquisadores na produção de pesquisas "padrão ouro" (Figura 2.1). Nelas, são fornecidas funcionalidades e formas de trabalho que podem ser utilizadas para a organização e definição do fluxo de processamento. Também é possível criar ambientes computacionais reprodutíveis e códigos com descrições completas das decisões e premissas utilizadas em sua construção.

Nas Subseções abaixo, cada um dos tipos selecionados são apresentados em detalhes, considerando para cada um deles as características relevantes para o desenvolvimento desta Dissertação.

2.3.1 *Computational Notebooks*

Rule et al. (2018) afirmam que durante o desenvolvimento de uma pesquisa envolvendo análise de dados, as conclusões geradas por dois pesquisadores podem ser completamente diferentes, e mesmo assim, estarem corretas. Segundo os autores, isto ocorre por conta das etapas de interpretação presentes na análise dos dados, que pode variar de acordo com cada pesquisador e seus objetivos.

Uma abordagem para tornar claras as intenções dos pesquisadores é a aplicação das técnicas de *Literate Programming* (KNUTH, 1984). Nesta abordagem, tem-se como objetivo vincular descrições em linguagem natural com códigos de computador e seus resultados, formando assim, as narrativas computacionais. Com isso, os textos narrativos podem ser utilizados para deixar explícito o que é esperado que o computador faça, permitindo que humanos leiam e entendam o contexto e as motivações aplicadas na criação do código (STODDEN; MIGUEZ, 2014; PICCOLO; FRAMPTON, 2016). *Computational Notebooks* ou *notebooks* são documentos que permitem a aplicação do *Literate Programming* (RULE et al., 2018), fornecendo uma interface que permite mesclar descrições criadas em linguagem natural, códigos e resultados (WANG et al., 2019). Um exemplo de tecnologia que implementa o conceito de *notebook* é o `Jupyter Notebook` (KLUYVER et al., 2016).

Criado com base no uso de computação interativa (PÉREZ; GRANGER, 2007), o

`Jupyter Notebook` é uma ferramenta *open source* que fornece através de uma interface *web* a possibilidade de criar *notebooks* que suportam o uso conjunto de textos, imagens, vídeos e código fonte (KLUYVER et al., 2016). Um *notebook* criado pela ferramenta é composto por um conjunto de células. Cada célula pode conter elementos relacionados à narrativa ou análise computacional. Os elementos narrativos podem ser criados utilizando as linguagens de marcação HTML (do inglês, *HyperText Markup Language*), LaTeX ou Markdown. Para as análises, utilizam-se linguagens de programação. No início do `Jupyter Notebook`, os *notebooks* criados suportavam apenas a linguagem Python. Com a modularização do código da ferramenta e adesão da comunidade, atualmente várias linguagens podem ser utilizadas. Em 2016, a ferramenta suportava mais de 50 linguagens de programação (KLUYVER et al., 2016), incluindo Bash[1], C++[2], R[3] e entre outras.

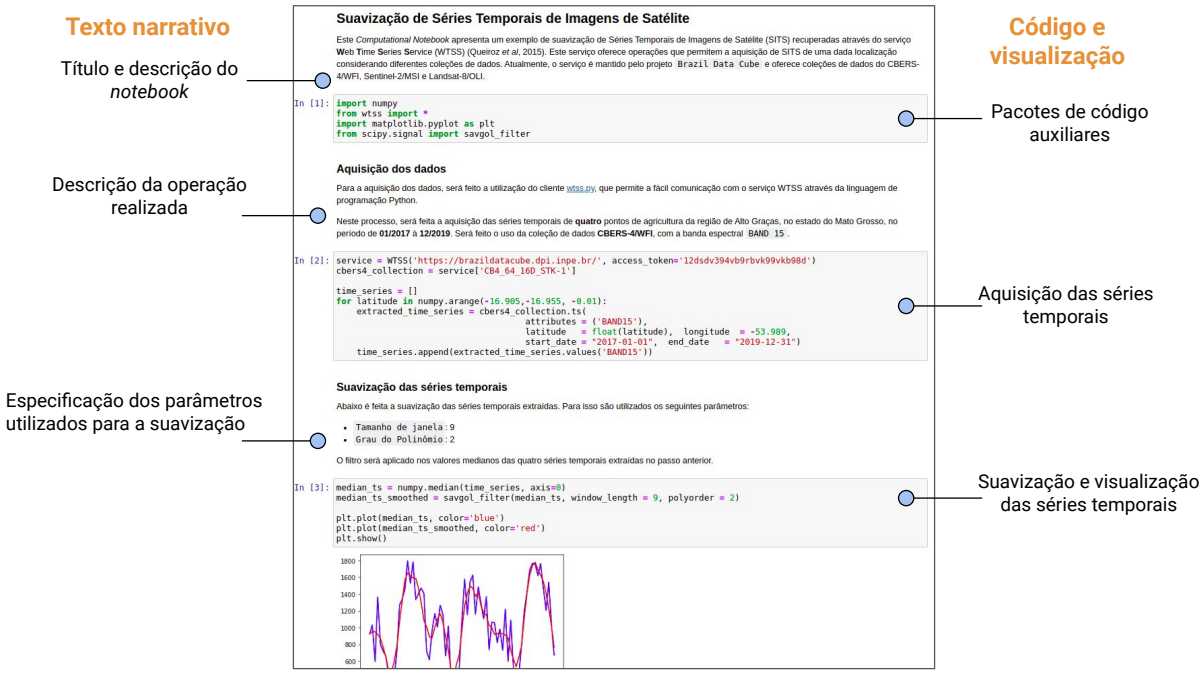
Na Figura 2.3, tem-se o exemplo de um *notebook* criado no `Jupyter Notebook`. Nessa Figura ilustram-se as etapas do processo de suavização de Séries Temporais de Imagens de Satélite. A utilização conjunta do código fonte, em Python, junto aos elementos narrativos, permitiu a descrição de cada uma das etapas, indo do contexto da aplicação aos parâmetros utilizados na etapa de suavização.

Além dos `Jupyter Notebooks`, existem outras tecnologias que permitem a construção de *Computational Notebooks* (WANG et al., 2019), como os `R Notebook` (XIE, 2019), `Apache Zeppelin`[4] e `Observable`[5]. Esta variedade de ferramentas permite aos pesquisadores a escolha das melhores formas de utilizar os *Computational Notebooks* em suas análises, tornando-as mais transparentes e reprodutíveis (BAUMER; UDWIN, 2015).

Com essas características o uso dos *notebooks*, auxilia a comunicação científica. Neste contexto, Lasser (2020) afirma que publicar um *notebook* no lugar de uma publicação convencional, em que apenas o PDF (do inglês, *Portable Document Format*) é disponibilizado, aumenta a compreensão do trabalho e evita ambiguidades de interpretação. Mesmo com os meios de publicação ainda não permitindo a utilização dos *notebooks* como publicação final, sua presença nos materiais complementares garante os benefícios citados (LASSER, 2020).

[1]Disponível em: <`https://github.com/takluyver/bash_kernel`>. Acesso em: 23/04/2021
[2]Disponível em: <`https://github.com/jupyter-xeus/xeus-cling`>. Acesso em: 23/04/2021
[3]Disponível em: <`https://github.com/IRkernel/IRkernel`>. Acesso em: 06/05/2021
[4]Disponível em: <`https://zeppelin.apache.org/`>. Acesso em: 23/04/2021
[5]Disponível em: <`https://observablehq.com/`>. Acesso em: 23/04/2021

Figura 2.3 - Exemplo de um `Jupyter Notebook` mesclando narrativa, descrições das operações, código fonte e visualização dos resultados.



Fonte: Adaptado de Rule et al. (2018).

2.3.2 Gerenciamento de ambiente computacional

Atualmente, a utilização de métodos computacionais na pesquisa não exige que todas as operações ou técnicas a serem utilizadas sejam implementadas do zero (PICCOLO; FRAMPTON, 2016). Estão disponíveis para instalação, bibliotecas de código *open source* que possuem funcionalidades já implementadas que podem ser utilizadas para acelerar o desenvolvimento da pesquisa (WICKHAM, 2014; LOWNDES et al., 2017; CARPENTER et al., 2021). Um exemplo de biblioteca é o próprio `Jupyter Notebook`, que é instalado no ambiente do pesquisador para ser utilizado.

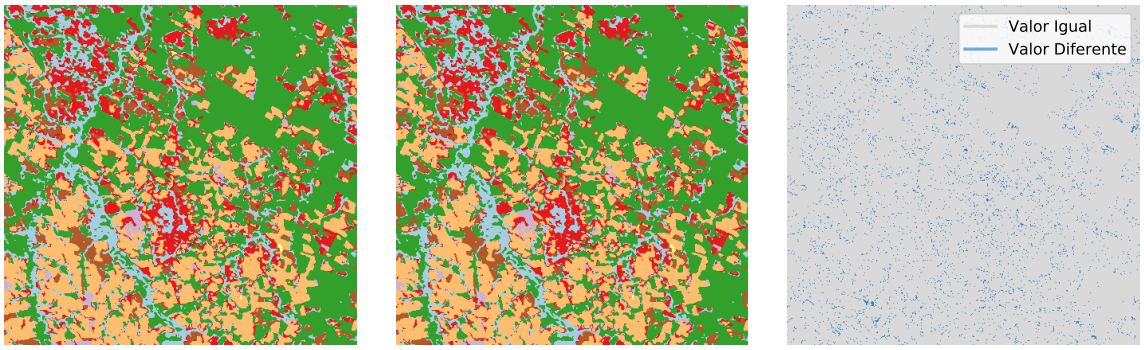
A utilização desses *softwares* terceiros, trazem consigo a necessidade do controle e gerenciamento das versões instaladas. Essa necessidade surge já que os pacotes de *software* evoluem e as diferentes versões podem produzir diferentes resultados (KONKOL et al., 2018). Fomel e Claerbout (2009), por exemplo, relatam que a simples mudança da versão do compilador trouxe problemas para uma de suas primeiras

tentativas na criação de uma pesquisa reprodutível.

Além disso, cabe citar que com o aumento da complexidade das operações realizadas atualmente, os pacotes de *software* passaram a ter dependências entre si, gerando assim, uma árvore de dependências (DECAN et al., 2016), que quando não controlada, torna a análise irreprodutível (BEAULIEU-JONES; GREENE, 2017). Desta forma, mesmo com o uso dos *Computational Notebooks*, não há garantia de reprodução dos resultados de uma pesquisa, uma vez que esses apenas guardam o conteúdo da análise, não considerando outros recursos, como a especificação do ambiente computacional e os dados (KEDRON et al., 2020).

Para exemplificar como os resultados podem ser influenciados pela mudança de versão do *software*, a Figura 2.4 apresenta dois mapas de Uso e Cobertura da Terra (LULC, do inglês *Land Use and Land Cover*), de uma região do município de Sinop, no estado do Mato Grosso[6]. Os mapas foram gerados utilizando os mesmos *scripts* e dados de entrada, variando apenas a versão do pacote de *software* utilizado para a classificação. Para evidenciar essa diferença gerou-se uma álgebra de mapas na qual foram comparados ambos os mapas (Figura 2.5a e Figura 2.5b) *pixel* a *pixel*. Na Figura 2.5c, essa diferença é evidenciada, tendo que valores que se diferem são apresentados em azul e os valores iguais com a cor cinza. Como é possível notar, os resultados diferem em diversos pontos.

Figura 2.4 - Comparação de resultados de mapas de LULC: (a) Versão antiga do *Software*; (b) Versão nova do *Software*; (c) Diferença dos resultados.



(a) (b) (c)

Fonte: Produção do autor.

[6]O *script* utilizado para as classificações, juntamente com o ambiente computacional de cada teste está disponível em: `https://tinyurl.com/storm-examples-lulcversions`

Considerando que o controle manual dessas dependências pode ser difícil ou impossível (BEAULIEU-JONES; GREENE, 2017). Piccolo e Frampton (2016) sugerem que uma primeira solução pode ser o uso de ferramentas de gerenciamento de dependências. Essas ferramentas conseguem mapear e controlar toda a árvore de dependências do ambiente do pesquisador. Um exemplo de ferramenta de gerenciamento de dependência é o `conda`[7], uma ferramenta *open source* que facilita o controle de versões, normalizando o processo de instalação de pacotes para linguagens de programação como Python, C++, Julia e R. Grüning et al. (2018) indicam que o `conda` pode ser aplicado para o auxílio à reprodutibilidade por realizar toda a construção e instalação dos pacotes em um ambiente isolado, evitando uso de dependências não rastreadas, que podem estar instaladas no Sistema Operacional (SO). O `conda` permite também que o ambiente de trabalho do pesquisador possa ser exportado em um arquivo com a descrição de todas as dependências e suas versões. Isto aumenta a portabilidade e facilita o compartilhamento do ambiente utilizado. Além do `conda`, existem outras ferramentas com o mesmo objetivo, por exemplo `renv`[8], para o controle de pacotes na linguagem R, `conan`[9] para a linguagem C++ e `pip`[10] para a linguagem Python.

Além das ferramentas de gerenciamento, para auxiliar no controle do ambiente computacional, os pesquisadores podem fazer o uso de ambientes virtuais (PICCOLO; FRAMPTON, 2016; GRüNING et al., 2018). Tais ambientes podem ser gerados através de virtualização, baseada em *containers* ou *hypervisors* (MORABITO et al., 2015). Nas técnicas baseadas em *hypervisors*, é feita a criação de uma máquina virtual, em que se tem todos os componentes virtualizados, incluindo o *software* com o sistema operacional e o *hardware* (FIRESMITH, 2017). Ademais, com o uso de *containers*, tem-se uma virtualização mais leve (MORABITO et al., 2015), em que é criado um ambiente em tempo de execução que emula um SO, através do uso de funcionalidades do *kernel* subjacente (FIRESMITH, 2017). Em ambos os casos, os ambientes gerados são isolados dos sistemas subjacentes. Considerando a facilidade de uso e a demanda menor por recursos computacionais, os *containers* são foco de aplicação deste trabalho. Mais informações sobre o uso de máquinas virtuais para reprodutibilidade podem ser encontradas na revisão apresentada por Howe (2012).

A utilização das técnicas de virtualização por *container* podem ser realizadas através das ferramentas de gerenciamento de *containers*. Essas são responsáveis por representar e gerenciar as principais abstrações dessa abordagem: *Images* e *Containers*.

[7]Disponível em: <`https://docs.conda.io/en/latest/`>. Acesso em: 23/04/2021
[8]Disponível em: <`https://rstudio.github.io/renv/`>. Acesso em: 23/04/2021
[9]Disponível em: <`https://conan.io/`>. Acesso em: 23/04/2021
[10]Disponível em: <`https://pypi.org/project/pip/`>. Acesso em: 23/04/2021

Uma *Image* representa um *template* "somente leitura" de um ambiente computacional. Neste *template*, estão registradas bibliotecas de código, dados e ferramentas de *software*, que ficarão disponíveis no ambiente criado com a execução da *Image*. Os *Containers* são *Images* em execução (TURNBULL, 2014; MIELL, 2016).

Diversas ferramentas utilizam esses conceitos em suas implementações. Para permitir o uso interoperável das *Images* e *Containers*, foi criada em Junho de 2015 a Open Container Initiative (OCI)[11], uma iniciativa de governança aberta com o propósito de criar padrões abertos em torno das abstrações e ferramentas de execução de virtualização por *containers*. Atualmente, a OCI possui duas especificações, `image-spec` e `runtime-spec`. Enquanto a primeira especificação trata sobre a organização e representação das *Images*, a segunda descreve como uma *Image* deve ser executada para a geração de um *Container* (OCI, 2020).

Atualmente, diferentes ferramentas que seguem estas especificações estão disponíveis, sendo que sua escolha depende das necessidades de cada usuário. Por exemplo, Priedhorsky e Randles (2017), Kurtzer et al. (2017) e Benedicic et al. (2019) apresentam gerenciadores de *containers* específicos para aplicações científicas que utilizam *High Performance Computing*. Outros gerenciadores disponíveis incluem `podman`[12] e `cri-o`[13].

No contexto da reprodutibilidade, frequentemente tem-se feito o uso do `Docker`[14] (BOETTIGER, 2015; NüST; HINZ, 2019), um gerenciador de *containers open source* que fornece aos pesquisadores meios simples e convenientes de criar *Images* e *Containers*. No `Docker`, toda a descrição de uma *Image* é feita através de um arquivo de configuração nomeado `Dockerfile`. Neste arquivo, pode-se especificar todos os pacotes de *software*, dados e demais elementos que precisam estar disponíveis no ambiente computacional.

A Figura 2.5 apresenta um exemplo de `Dockerfile` que especifica uma *Image* que será criada considerando como base o sistema operacional Ubuntu 18.04, junto ao interpretador da linguagem Python na versão 3.8. Utilizando o gerenciador de pacotes `pip`, durante a criação da imagem é realizada também a instalação do pacote para análise de dados `Pandas` (MCKINNEY, 2010) e todas as suas dependências.

[11]Disponível em: <`https://opencontainers.org/`>. Acesso em: 23/04/2021
[12]Disponível em: <`https://podman.io/`>. Acesso em: 23/04/2021
[13]Disponível em: <`https://cri-o.io/`>. Acesso em: 06/05/2021
[14]Disponível em: <`https://www.docker.com/`>. Acesso em: 23/04/2021

Figura 2.5 - Exemplo de `Dockerfile` utilizado para a criação de uma *Image*.

```
1      FROM ubuntu:18.04
2
3      RUN apt update -y \
4          && apt install python3.8 python3-pip \
5          && pip3 install pandas==1.2.4
```

Fonte: Produção do autor.

Com o ambiente especificado neste arquivo, o pesquisador pode gerar uma *Image* e a executar para gerar um *Container*. O ambiente criado no *Container*, isolado do SO subjacente, pode ser utilizado para o desenvolvimento da pesquisa. Esse mecanismo do `Docker` permite o aumento da portabilidade, por representar as configurações em um arquivo. Isso facilita o compartilhamento do ambiente utilizado pelo pesquisador para a obtenção dos resultados (BOETTIGER; EDDELBUETTEL, 2017).

A ferramenta *open source* `BinderHub` (JUPYTER et al., 2018) é um exemplo notável de uso da portabilidade do `Docker` em benefício da pesquisa reprodutível. Essa ferramenta permite que pesquisas disponibilizadas em repositórios de controle de versão `git`[15] que possuam um `Dockerfile` sejam facilmente reproduzidas, exploradas e replicadas. Para isso, `BinderHub` carrega o repositório, valida o `Dockerfile` e em seguida constrói o ambiente utilizando as instruções presentes neste arquivo. Sua única restrição é a necessidade da presença do `Jupyter Notebook` instalado no ambiente criado.

Por fim, Boettiger (2015) sumariza os benefícios para a reprodutibilidade que o uso do `Docker` pode trazer para um projeto de pesquisa reprodutível: (i) Controle de dependências; (ii) Solução da imprecisão de artigos em descrever o ambiente computacional; (iii) Remoção de problemas com dependências que não estão mais disponíveis, salvando *Containers* como novas *Images*; (iv) Redução de barreiras para adaptação e reuso, causadas pela dificuldade de configuração do ambiente original. Todas as características citadas são gerais e podem ser utilizadas para o aumento da reprodutibilidade em diversas áreas da ciência. Por exemplo, nas áreas de Geoinformática (ZARAGOZÍ et al., 2020), Sensoriamento Remoto (KNOTH; NüST, 2017), Bi-

[15]Disponível em: <`https://git-scm.com/`>. Acesso em: 26/04/2021

oinformática (MENEGIDIO et al., 2017) e Psicologia (CLYBURNE-SHERIN et al., 2019).

2.3.3 Ferramentas de *workflow* científico

O desenvolvimento e geração de resultados em pesquisas que envolvem o processamento e análise de dados não é linear. A natureza experimental dessa atividade torna necessária a mudança e reexecução dos códigos de processamento diversas vezes. Isto pode representar um desafio para a reprodutibilidade. As diversas modificações nos códigos e a geração de resultados da pesquisa em ambientes "contaminados" - com modificações interativas sem o registro das sementes de valores pseudo-aleatórios - podem tornar o resultado da pesquisa impossível de ser reproduzido (BRYAN; HESTER, 2018; BLISCHAK et al., 2019)[16].

Uma solução para este caso é o uso de *workflows* científicos (LUDÄSCHER et al., 2006; DUFFY et al., 2012; PERKEL, 2019). *Workflow* científico, ou simplesmente *workflow*, é uma descrição das etapas tomadas para a obtenção dos resultados de uma pesquisa, sendo cada etapa composta por uma ou mais tarefas. Em um *workflow*, as tarefas representam a execução de um processo computacional, por exemplo, a execução de um código ou ferramenta de linha de comando, chamada a um serviço *web* e o registro de informações em um banco de dados (GOBLE et al., 2020).

Com o uso de *workflows* em uma pesquisa, é possível separar a *definição* da *operação*. Na *definição*, tem-se a construção dos códigos e a especificação do *workflow*. Na *operação*, o *workflow* especificado é executado (GOBLE et al., 2020). Isto permite que os problemas citados anteriormente possam ser evitados, uma vez que cada execução é realizada de forma independente, evitando a interferência de mudanças interativas que alterem o estado da execução (BRYAN; HESTER, 2018).

Diferente dos *workflows* de negócios, o foco dos *workflows* científicos é a representação do fluxo de dados das operações (LUDÄSCHER et al., 2006). Desta forma, a determinação da ordem de execução de cada um dos passos é realizada através da análise de dependências dos dados (FILGUIERA et al., 2016; STROZZI et al., 2019). Como a distinção dos tipos de *workflows* não é o foco deste trabalho, caso seja de interesse do leitor, recomenda-se o trabalho apresentado por Barga e Gannon (2007), que analisa e distingue em detalhes cada um dos tipos citados.

Para a utilização dos *workflows*, estão disponíveis para os pesquisadores diversas

[16]Neste trabalho foi criado um exemplo que apresenta os impactos causados pelos ambientes "contaminados" na reprodução dos resultados. Exemplo disponível em: <`https://tinyurl.com/storm-examples-lulcexecutions`>

ferramentas, com diferentes variações de abordagens tecnológicas (LEIPZIG, 2016). Inicialmente, as execuções dos *workflows* podem ser feitas através de ferramentas como `GNU Make`[17] e *scripts* de linguagens como Python, Perl e Bash. Tais ferramentas são recomendadas para *workflows* simples, pois à medida que a complexidade da execução aumenta, é difícil manter o controle de longas execuções. Mecanismos como *cache* de etapas e controle de falhas podem não estar disponíveis facilmente nessas abordagens (LAMPA et al., 2019; STROZZI et al., 2019).

Como alternativa, pode-se fazer o uso dos Sistemas de Gerenciamento de *Workflow* Científicos (SWMS, do inglês *Scientific Workflow Management System*) infraestruturas especializadas para o gerenciamento e execução de *workflows* (GOBLE et al., 2020). Os SWMS oferecem uma interface comum entre os pesquisadores e os recursos computacionais (ATKINSON et al., 2017). Para o uso dessa interface, as descrições dos *workflows* são feitas em linguagens de alto nível, que focam no que precisa ser executado, deixando de lado detalhes de como a execução é feita. O SWMS interpreta essas descrições de *workflow* e realiza a execução. Dependendo do SWMS e sua implementação, podem ser adotadas diferentes estratégias e otimizações nas execuções, como *cache* de etapas, execuções paralelas e controle de falhas (GIL, 2009; TALIA, 2013; COHEN-BOULAKIA et al., 2017).

A escolha da linguagem para a descrição dos *workflows* pode depender do SWMS que está sendo utilizado e os formatos aceitos pelo mesmo. Por exemplo, Strozzi et al. (2019) apresentam o uso de SWMSs com quatro diferentes linguagens de definição de *workflow*. Em um esforço para garantir a interoperabilidade, portabilidade e reprodutibilidade das descrições dos *workflows* entre as diversas tecnologias disponíveis, foi criada pelo Common Workflow Language Working Group a *Common Workflow Language* (CWL) (AMSTUTZ et al., 2016). Esta é uma especificação de linguagem para a descrição de *workflows*.

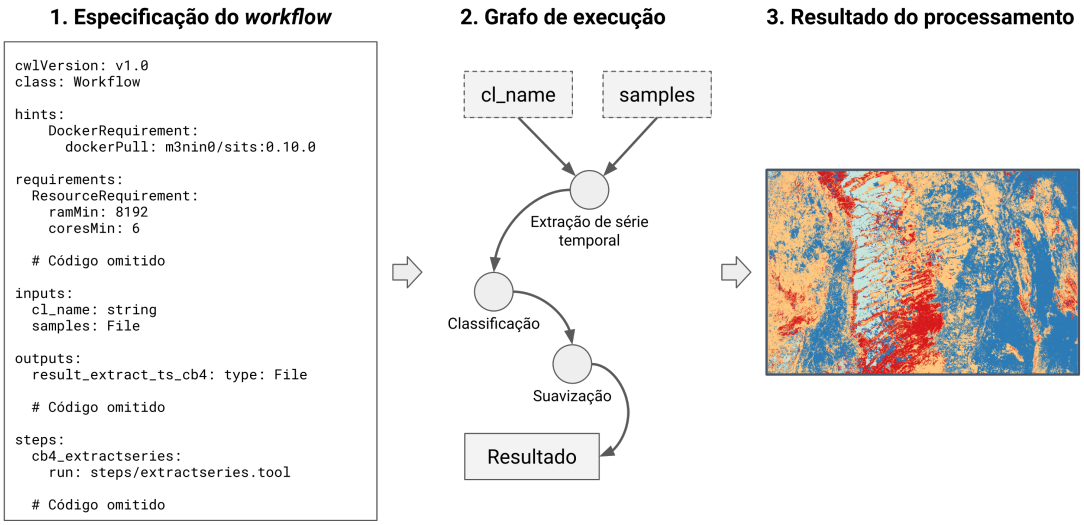
Na CWL, o *workflow* é representado por um Grafo Acíclico Direcionado (DAG, do inglês *Directed Acyclic Graph*) (AMSTUTZ et al., 2016). Neste, cada vértice representa um processo computacional, enquanto as arestas indicam o fluxo de entradas e saídas dos dados durante a execução (TALIA, 2013). A ordem de construção do DAG é determinada pelas dependências de dados (AMSTUTZ et al., 2016). A especificação desse DAG é feita com a CWL em um arquivo no formato `YAML` ou `JSON`. Nesse arquivo, define-se cada uma das tarefas a serem executadas, bem como suas entradas e saídas.

[17] Disponível em: <`https://www.gnu.org/software/make/`>. Acesso em: 19/11/2022

Adicionalmente, quando um *workflow* está sendo especificado na CWL, para cada uma das tarefas, além das entradas e saídas, é possível especificar as necessidades de sua execução. Por exemplo, pode-se definir o `Docker` *Image* do ambiente em que a tarefa deve ser executada, junto aos recursos computacionais mínimos para a execução, como a quantidade de *Cores* e Memória RAM.

Na Figura 2.6[18] é ilustrado em alto nível cada um dos elementos CWL mencionados. No bloco esquerdo da Figura, tem-se a especificação CWL do *workflow* no formato `YAML`. Nesse, são definidas cada uma das tarefas executadas, suas entradas, saídas e os requisitos para a execução. No bloco central, tem-se o DAG do *workflow* especificado, no qual, nota-se a presença de três tarefas. Por fim, no bloco à direita na Figura, tem-se um exemplo de resultado que pode ser obtido através da execução do *workflow* definido em um SWMS.

Figura 2.6 - Componentes de uso da CWL.



Fonte: Produção do autor.

Jackson et al. (2021) informam que uma vez definido o fluxo de processamento em CWL, têm-se unidades que podem ser reutilizadas. Então, no exemplo apresentado na Figura 2.6, o *workflow* criado passa a ser um componente que pode ser reutilizado

[18]O arquivo completo apresentado na Figura está disponível em: <https://tinyurl.com/storm-examples-cwlexample>

para a construção de outros *workflows* ou mesmo reutilizado para o processamento com outros dados de entrada. Esta característica auxilia a reprodução e replicação das pesquisas desenvolvidas utilizando CWL.

Por fim, para a execução de *workflows* CWL, diferentes SWMS podem ser utilizados. Na página da especificação[19], está disponível uma lista com as tecnologias que suportam por completo ou parcialmente sua utilização. Considerando os SWMS com suporte completo, estão listados `Toil` (VIVIAN et al., 2017), `Reana` (ŠIMKO et al., 2019), `CWL-Airflow` (KOTLIAR et al., 2019), `Arvados`[20] e a implementação de referência[21]. O *workflow* especificado com CWL pode ser executado em qualquer uma dessas ferramentas, sendo que a escolha depende apenas das vantagens oferecidas. Por exemplo, Vivian et al. (2017) destacam que `Toil` foi criado com o objetivo de processar grandes volumes de dados em plataformas distribuídas. Para isso, a ferramenta fornece otimizações que evitam a movimentação de dados entre múltiplos nós de processamento e permite o *cache* de resultados. Outro exemplo é o `CWL-Airflow`, que utiliza o arcabouço tecnológico do `Apache Airflow`[22] para as execuções, o que garante o controle de falhas e o acesso multiusuário.

2.3.4 *Research Compendium*

Como citado anteriormente, uma das bases para a Pesquisa Aberta e Reprodutível está na transparência das publicações. Para alcançar esse padrão, busca-se realizar a publicação de todos os objetos resultantes de uma pesquisa (BARBA, 2019). Isto inclui código, dados, ambiente computacional e *workflow* (STODDEN et al., 2016). A publicação desses elementos exige que eles estejam organizados e centralizados. Desta forma, evitam-se problemas com elementos faltantes na publicação, possibilitando que a pesquisa possa ser reprodutível e replicável.

No entanto, boa parte das pesquisas pode nem mesmo chegar nesse estágio de maturidade. A organização e disposição dos objetos criados em uma pesquisa devem ser consistentes do início ao fim, o que não ocorre normalmente (MARWICK et al., 2018). A não definição de uma estrutura organizacional no início do desenvolvimento da pesquisa, pode torná-la inconsistente e consequentemente difícil de ser reproduzida, mesmo aplicando as ferramentas tratadas nas Subseções anteriores (BLISCHAK et al., 2019).

[19]Disponível em: <`https://www.commonwl.org/`>. Acesso em: 25/04/2021
[20]Disponível em: <`https://arvados.org/`>. Acesso em: 25/04/2021
[21]Disponível em: <`https://github.com/common-workflow-language/cwltool`>. Acesso em: 25/04/2021
[22]Disponível em: <`https://airflow.apache.org/`>. Acesso em: 25/04/2021

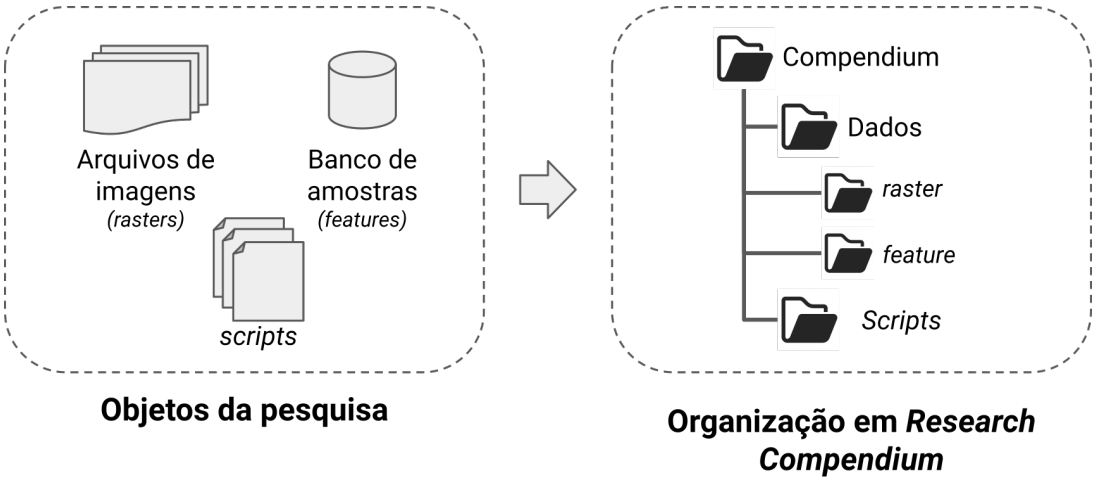
Introduzido por Gentleman e Lang (2004), o *Research Compendium* (RC) é uma unidade de pesquisa utilizada para centralizar todos os elementos gerados durante uma produção científica. Isto inclui dados, código fonte, documentação, ambiente computacional e *workflow* científico (COMUNIDADE TURING WAY et al., 2019). Os autores especificam que o RC deve ser adequado aos cenários práticos em que é aplicado. Assim, sua materialização é feita com base em convenções definidas pelos pesquisadores em acordo com suas respectivas comunidades científicas. Definindo-se, assim, quais os elementos obrigatórios de estarem no RC e sua organização interna (MARWICK et al., 2017).

Com isso, a materialização de um RC é feita de várias formas. Em uma primeira abordagem, pode-se fazer a organização dos elementos da pesquisa através de uma estrutura básica de diretórios (COMUNIDADE TURING WAY et al., 2019). Em casos mais complexos, pode-se utilizar ferramentas como `rrtools`[23] e `devtools`[24], que materializam o RC no formato de pacotes da linguagem de programação R (MARWICK et al., 2018). Na Figura 2.7 é apresentado um exemplo de uso de RC com a organização de diretórios. No bloco do lado esquerdo desta Figura é ilustrado um conjunto de elementos espalhados na máquina de um pesquisador, os quais podem ser traduzidos por arquivos em diretórios distintos. No lado direito desta Figura, ilustra-se o uso do RC por meio de uma estrutura simples de diretórios, em que se têm os elementos centralizados e organizados.

Desta forma, nota-se que a utilização do RC e os conceitos de centralização e organização com base em uma convenção, mesmo simples, quando aplicado, pode garantir que a publicação dos resultados seja feita junta com os outros objetos da pesquisa que foram utilizados para sua geração (D'ANDREA, 2020). Além disso, aqueles que vão reexecutar a análise para possíveis reproduções e replicações tem a consistência da organização, tornando o uso simples e evitando a necessidade de comunicação com os autores originais (MARWICK et al., 2018; JOSEPH, 2020).

[23]Disponível em <`https://github.com/benmarwick/rrtools`>. Acesso em: 10/04/2021
[24]Disponível em: <`https://github.com/r-lib/devtools`>. Acesso em: 10/04/2021

Figura 2.7 - Exemplo de utilização de um RC por meio de estrutura de diretórios.



Fonte: Produção do autor.

Sendo uma boa prática para o desenvolvimento do código de pesquisa (BRYAN et al., 2016; WILSON et al., 2017), a utilização de *softwares* para controle de versão também é relevante para a reprodutibilidade. Esses realizam a criação de um histórico de mudanças, que aumenta a transparência e permite que os passos tomados até a versão final dos materiais possam ser compreendidos (HERRERA, 2019). Dentre as ferramentas de controle de versão, destaca-se o `git-scm`[25], também chamado de `git`, uma ferramenta *open source*, que permite o versionamento de arquivos e diretórios. Auxilia também no desenvolvimento de trabalhos colaborativos (BARBA, 2019). A centralização feita pelo RC permite que o controle de versão seja aplicado nos materiais da pesquisa (COMUNIDADE TURING WAY et al., 2019), sendo possível ao pesquisador usufruir dos benefícios citados.

Por centralizar os elementos da pesquisa, o RC também auxilia na comunicação, sendo tratado como uma unidade de compartilhamento científico (GENTLEMAN; LANG, 2004; NüST et al., 2017). Isto o tornou relevante no processo de publicação científica há alguns anos (NÜST et al., 2018; NÜST; PEBESMA, 2020). Para esse contexto, além dos elementos da pesquisa, pode-se fazer a adição de arquivos de metadados e licença no RC, permitindo que outros pesquisadores possam encontrar e entender quais as regras de uso dos materiais da pesquisa (STODDEN, 2009; MARWICK et al., 2018). Alguns exemplos de publicações com RC podem ser encontrados em Boettiger

[25]Disponível em: <`https://git-scm.com/`>. Acesso em: 26/04/2021

et al. (2014), Marwick (2017) e Boettiger e Memarzadeh (2018).

Além do RC, existem outras unidades de pesquisa que podem ser consideradas. Por exemplo, o *Executable Research Compendium* (ERC) é a extensão de um RC, que torna obrigatória a definição do ambiente computacional através de um arquivo `Dockerfile` ou `Docker` *Image* (NüST et al., 2017). Outro exemplo é o *Knowledge Package*, o qual permite o empacotamento de elementos de pesquisa e o vínculo de materiais associados através do uso de identificadores persistentes.

2.3.5 Repositórios digitais de dados de pesquisa

O desenvolvimento de pesquisas utilizando práticas reprodutíveis aumenta o grau de atenção que os pesquisadores devem possuir nos detalhes de suas pesquisas. Isso, com base nos conteúdos apresentados nas seções anteriores, pode representar mudanças em seus fluxos de trabalho, com a adoção de novas ferramentas ou mesmo práticas.

Todo esse esforço, quando aplicado corretamente, faz com que outros pesquisadores consigam validar e explorar as pesquisas realizadas e seus resultados. No entanto, para que todo o material desenvolvido durante uma pesquisa seja acessível por outros pesquisadores, é necessário também cuidados especiais aos aspectos de compartilhamento e preservação desses materiais (COMUNIDADE TURING WAY et al., 2019).

No contexto da Pesquisa Aberta e Reprodutível, esses aspectos podem ser tratados por repositórios digitais de dados de pesquisa (ou simplesmente chamados aqui de repositórios digitais). Tais repositórios representam um lugar onde os recursos produzidos durante o curso da pesquisa, sejam eles dados, *software*, publicações ou qualquer outro tipo de dados de pesquisa possam ser armazenados e preservados por um longo período de tempo (SAYÃO; SALES, 2016). Esses repositórios ainda auxiliam os pesquisadores na contextualização de suas pesquisas, fornecendo a possibilidade de descrição e especificação dos materiais compartilhados através de metadados. Fornecem também métodos de busca que permitem que, com base nas informações apresentadas nos metadados, outros pesquisadores encontrem os trabalhos e materiais de pesquisa compartilhados (WITTER; BAINBRIDGE, 2003). Com isso, ao fazerem uso desses repositórios, os pesquisadores são capazes de potencializar a disseminação de suas pesquisas, ao mesmo tempo que mantém os materiais associados a elas preservados (BONTEMPS; OROZCO, 2020).

Existem diversos tipos de repositórios digitais, podendo esses serem criados para atender comunidades específicas ou mesmo para atender um público multidiscipli-

nar (SAYÃO; SALES, 2016). Um exemplo de repositório de dados de pesquisa é o CERN Analysis Preservation, um serviço criado especialmente para os fluxos de trabalho da comunidade de *High-Energy Physics* do CERN (CHEN et al., 2016). O serviço é utilizado para integrar todas as informações utilizadas nos experimentos realizados pelos pesquisadores. Com a centralização das informações e a possibilidade de registro de todos os objetos digitais da pesquisa, segundo os autores, torna-se mais fácil a reprodução e replicação dos experimentos. Outro exemplo está no Terrabrasilis Research Data, um repositório digital criado especialmente para o compartilhamento e preservação de dados científicos geoespaciais (SANSIGOLO, 2020). De modo a fornecer recursos específicos à comunidade geoespacial, fornece métodos especializados para a preservação e recuperação dos dados. Um exemplo desses métodos é a busca espaço-temporal.

Um aspecto importante na seleção e uso de repositórios digitais está nas tecnologias empregadas na composição de suas funcionalidades. Nesse contexto, de modo a fazer uma apresentação breve sobre tais características, tem-se no tópico abaixo a apresentação de ferramentas que podem ser utilizadas para a construção desses repositórios.

Ferramentas para construção de repositórios digitais de dados de pesquisa

Os aspectos de preservação e compartilhamento dos materiais associados a uma pesquisa são de grande valia para toda a comunidade científica. No entanto, deve-se notar que nem sempre esses aspectos possuem respostas técnicas simples, exigindo soluções especializadas, que garantam a consistência das buscas pelos materiais, bem como sua preservação.

De modo a tratar os aspectos técnicos envolvidos na construção de um repositório digital, muitas soluções técnicas foram desenvolvidas. Nesse contexto, Amorim et al. (2017) apresentam a análise de 15 soluções tecnológicas que podem ser utilizadas para a construção de repositórios digitais de dados de pesquisa. De forma geral, as soluções apresentadas podem ser categorizadas em dois grupos. O primeiro deles inclui ferramentas que podem ser empregadas na construção de repositórios digitais, sem a necessidade de etapas de desenvolvimento para a customização ou definição das funcionalidades do repositório. A esse primeiro grupo estão associados ferramentas como `CKAN`[26] e `Zenodo`[27]. No segundo grupo, por outro lado, têm-se ferramentas onde há a necessidade de implementação das funcionalidades que estarão disponíveis

[26]Disponível em <`https://ckan.org/`>. Acesso em: 28/11/2022
[27]Disponível em <`https://zenodo.org/`>. Acesso em: 28/11/2022

no repositório.

Com esses dois grupos de repositórios, os autores supracitados apresentam as diferentes abordagens que podem ser utilizadas na construção dos repositórios digitais. Nesse contexto, nota-se que as ferramentas do segundo grupo também podem ser empregadas para o desenvolvimento de soluções que vão além de repositórios digitais, uma vez que fornecem capacidades que tange o gerenciamento de dados e metadados. Soluções para o suporte a reprodutibilidade, por exemplo, podem ser desenvolvidas com o uso dessas tecnologias.

Para esse tipo de desenvolvimento, dentre as ferramentas mencionadas por Amorim et al. (2017), tem-se como destaque o `Invenio Framework`[28], um *framework* para a linguagem de programação Python que fornece recursos e funcionalidades para a construção de repositórios digitais de larga escala, capazes de gerir grandes volumes de dados e metadados. As funcionalidades do `Invenio Framework` em abordagem de módulos, onde cada funcionalidade está implementada em um módulo que pode ser utilizado separadamente, permitindo que eles sejam empregados em diferentes contextos. Dentre os principais módulos fornecidos pelo *framework*, estão os seguintes:

`Invenio DB`: Módulo que fornece capacidades para o gerenciamento de entidades de dados em bancos de dados relacionais;

`Invenio Access`: Módulo que fornece capacidades para o controle de acesso e usuários;

`Invenio Records`: Módulo extensível para o gerenciamento de metadados. Consegue gerir diferentes padrões de metadados, possuindo funcionalidades para a validação e controle de consistência dos metadados registrados;

`Invenio PIDStore`: Módulo que fornece funcionalidades para a criação e o gerenciamento de identificadores persistentes. Os identificadores podem ser gerados de forma local, ou utilizando serviços externos como o DataCite Fabrica[29], para a geração de DOIs.

`Invenio Files Rest`: Módulo para o gerenciamento de arquivos. Possui suporte a integração com diferentes tecnologias de sistemas de arquivos, como `MinIO`[30] e

[28]Disponível em <`https://inveniosoftware.org/`>. Acesso em: 28/11/2022
[29]Disponível em <`https://doi.datacite.org/`>. Acesso em: 28/11/2022
[30]Disponível em <`https://min.io/product/overview`>. Acesso em: 28/11/2022

Gluster[31].

Por conta dessas características, o *framework* é empregado no desenvolvimento de diversas soluções, incluindo o CERN Analysis Preservation e Zenodo do CERN. Outro exemplo de uso do Invenio Framework está na iniciativa GEO Knowledge Hub[32], criada pelo GEO - Group on Earth Observations[33], que, com base nas capacidades do *framework* desenvolve um repositório de pesquisa especializado no compartilhamento de aplicações de EO reprodutíveis e reutilizáveis. Nesse repositório, são fornecidos métodos especializados como busca espaço-temporal e buscas temáticas.

Por fim, outro exemplo de utilização do *framework* está no anúncio[34] feito pelo CERN em 2019, sobre a criação do InvenioRDM, uma solução de repositório digital pronta para uso, criada com base no Invenio Framework e suas funcionalidades.

2.4 Plataformas computacionais para pesquisa reprodutível

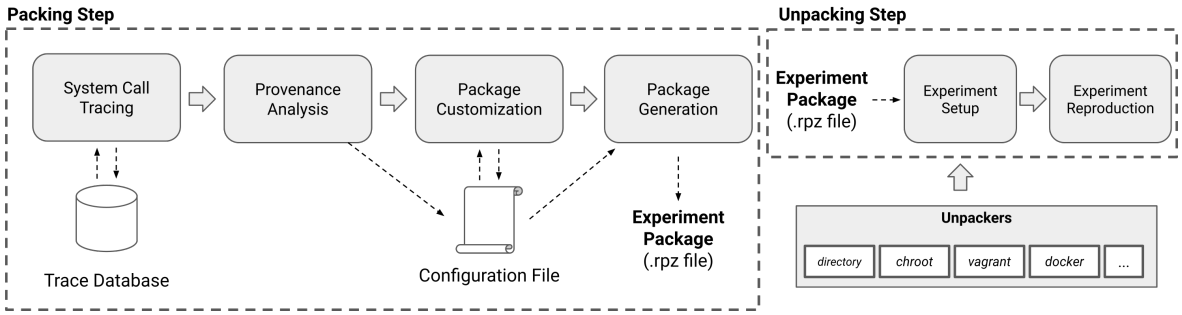
A utilização individual das ferramentas e técnicas apresentadas anteriormente pode não garantir a reprodutibilidade de uma pesquisa. Além disso, seu uso integrado e conjunto pode exigir conhecimentos especializados dos pesquisadores, criando assim barreiras técnicas que inviabilizam sua ampla adoção. Nesta Seção, serão apresentadas plataformas computacionais que integram tais ferramentas para a composição de funcionalidades de alto nível, que auxiliam os pesquisadores na produção de pesquisa reprodutível.

Chirigati et al. (2016) propõem o ReproZip, uma ferramenta *open source*, que auxilia a reprodutibilidade das pesquisas através da criação automática de pacotes reprodutíveis. Nestes, são armazenados todos os elementos de *software* utilizados para a execução dos *scripts* de processamento de uma pesquisa. Esses elementos incluem: variáveis de ambiente, arquivos de bibliotecas de SO, arquivos de *scripts* e os arquivos de dados de entrada e saída. O ReproZip determina o que deve ser inserido no pacote através da análise de chamadas do sistema. Nesta, todos os elementos utilizados em qualquer parte da execução do *script* de processamento são copiados para o pacote. Após sua criação, o pacote pode ser compartilhado e executado em ambientes independentes do original.

[31] Disponível em <https://www.gluster.org/>. Acesso em: 28/11/2022
[32] Disponível em <https://gkhub.earthobservations.org/>. Acesso em: 28/11/2022
[33] Disponível em <https://www.earthobservations.org/index.php>. Acesso em: 28/11/2022
[34] Disponível em <https://inveniosoftware.org/blog/2019-04-29-rdm/>. Acesso em: 28/11/2022

Para a realização das etapas citadas, o `ReproZip` implementa as operações de *Packing* e *Unpacking*, que, respectivamente, são responsáveis pela criação e execução de um pacote reprodutível. O uso dessas operações é feito por uma interface de linha de comando (CLI, do inglês, *Command-Line Interface*), disponibilizada pela ferramenta. Na Figura 2.8 é ilustrado o fluxo de funcionamento do `ReproZip` em cada uma das operações citadas.

Figura 2.8 - Fluxo de funcionamento do `ReproZip`.



Fonte: Adaptado de Chirigati et al. (2016).

Para a criação de um pacote reprodutível relativo às execuções de uma pesquisa, é necessário que o pesquisador execute os *scripts* de processamento através da CLI do `ReproZip` em um SO com *Kernel Linux*. Fazendo isso, os elementos necessários são mapeados e utilizados na composição do pacote.

Na criação do pacote, ao executar um *script*, os módulos de *Packing* são utilizados. Após iniciar a execução, o módulo `System Call Tracing` utiliza o `ptrace`, disponível no *Kernel* Linux, para a captura de todas as chamadas de sistema que estão sendo realizadas. Ao final da execução, todas as informações capturadas são registradas no *Trace Database*, um banco de dados `SQLite`.

Com as informações do *Trace Database*, o módulo `Provenance Analysis` determina quais foram os elementos de *software* utilizados pelo *script* durante sua execução. A determinação é feita com base em diferentes regras. Por exemplo, para determinar quais os pacotes de SO foram utilizados, o módulo faz consultas ao gerenciador de pacotes do SO. Os arquivos de entrada utilizados e os de saída produzidos são determinados com base em algumas heurísticas: Arquivos de entrada são aqueles

apenas lidos, que não possuem relação com pacote do SO; Os arquivos de saída são aqueles apenas escritos; Por fim, arquivos intermediários são aqueles lidos e escritos. Os elementos de *software* identificados pelo módulo são salvos no *Configuration File*[35].

O módulo `Package Customization` representa a possibilidade dos pesquisadores realizarem alterações manuais no *Configuration File*. Isto é útil quando há a necessidade de remoção de arquivos de dados de tamanho elevado ou informações sensíveis, como chave de acesso e arquivos de certificados digitais.

O módulo `Package Generation` realiza a criação do pacote reprodutível. Neste processo, que pode ser feito depois da execução de um ou mais *scripts*, são copiados para o pacote tudo o que está listado no *Configuration File*. Por armazenar apenas os elementos necessários para a reexecução dos *scripts* mapeados, os pacotes criados ocupam pouco espaço em disco, permitindo fácil compartilhamento. Em um comparativo feito pelos autores, entre uma máquina virtual e um pacote `ReproZip`, para um mesmo *script*, enquanto a máquina virtual utilizou 4 GB de espaço, o pacote utilizou 67 MB.

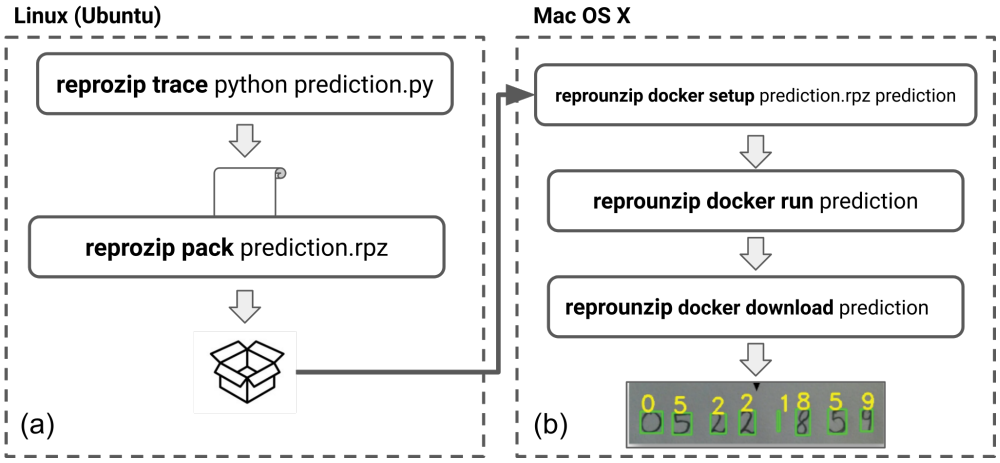
A execução de um pacote reprodutível é feita através da CLI do `ReproZip`. Ao executar o pacote, os módulos de *Unpacking* são utilizados. Na execução, o módulo `Experiment Setup` realiza a configuração do ambiente em que os *scripts* registrados no pacote serão executados. Para tanto, é feito o uso de um `Unpacker`, componente que cria e gerencia o ambiente de execução. Esse componente é modular e pode ser implementado para gerenciar ambientes de execução provisionados por diferentes tecnologias. No `ReproZip`, os `Unpackers` implementados permitem a configuração do ambiente de execução através de `Docker Containers`, máquinas virtuais e ambientes `chroot`. Essa forma de configuração do ambiente, permite que os pesquisadores executem o pacote em qualquer SO que suporte uma das tecnologias utilizadas pelos `Unpackers`. Após a preparação do ambiente, o módulo `Experiment Reproduction`, realiza a execução dos *scripts* registrados.

A Figura 2.9 ilustra um exemplo de uso da CLI do `ReproZip`. No bloco A da Figura é ilustrado a operação de *Packing*. Os comandos `reprozip trace` e `reprozip pack`, realizam, respectivamente, a execução do *script* e o empacotamento dos elementos de *software* utilizados. No bloco B da Figura tem-se o *Unpacking*. Nesse, a

[35]Um exemplo de *Configuration file* completo, para uma atividade de classificação de LULC, foi criado durante o desenvolvimento deste trabalho e está disponível em: https://tinyurl.com/storm-examples-reproziptrc

configuração do ambiente é feita com o comando `reprounzip docker setup`, que utiliza um `Unpacker Docker`. Em seguida, os comandos `reprounzip docker run` e `reprounzip docker download` executam o *script* registrado e exportam os resultados gerados, respectivamente.

Figura 2.9 - Exemplo de *Packing* (A) e *Unpacking* (B) com a CLI do ReproZip para uma tarefa de visão computacional.



Fonte: Adaptado de Chirigati et al. (2016).

`Reana` (ŠIMKO et al., 2019) é uma plataforma *open source* desenvolvida pelo CERN[36] para auxiliar a produção de pesquisa reprodutível e a reutilização de código. Faz isso através da criação de *runnable recipes*, conceito análogo ao pacote reprodutível do `ReproZip`. Nestes, estão presentes os 4 elementos utilizados pela plataforma para a produção reprodutível da pesquisa: Dados de entrada; *Scripts* de análise; Ambiente computacional; e Ordem de execução dos códigos.

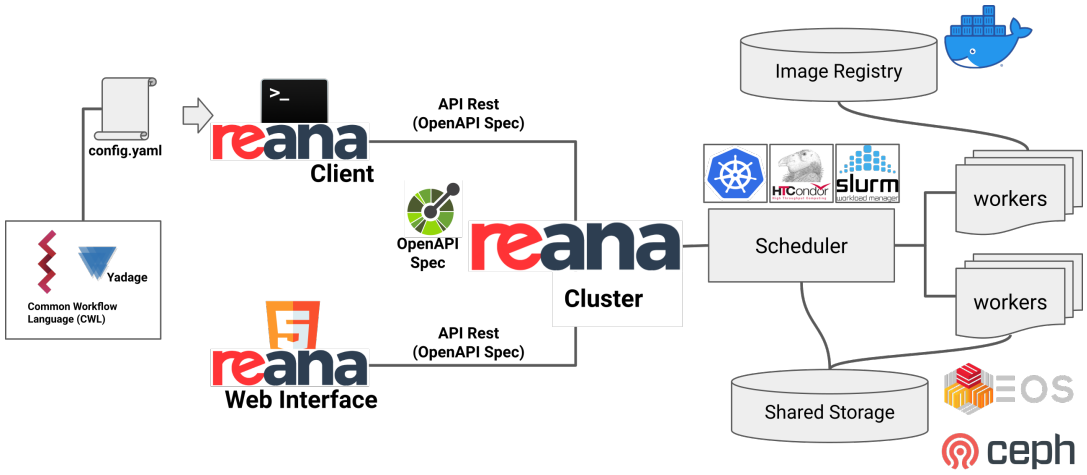
No `Reana`, cada uma dessas informações são postas em um arquivo de configuração no formato `YAML`. Nesse arquivo, tem-se duas seções principais: `inputs` e `workflow`. Na seção `inputs`, faz-se a especificação do caminho absoluto dos dados no sistema de arquivos do pesquisador, sendo diretórios ou arquivos individuais. Os *scripts* também são especificados nessa seção. A seção `workflow` descreve, ao mesmo tempo, o ambiente computacional e a ordem de execução dos *scripts*. Para isso, pode-se

[36]CERN é a sigla em francês para *Conseil Européen pour la Recherche Nucléaire*

utilizar as linguagens de *workflow* CWL e Yadage[37]. Em ambas, quando o fluxo é definido, especifica-se em cada etapa, o *script* de processamento e uma `Docker` *Image* do ambiente de execução. Outra possibilidade é a declaração da análise em modo sequencial. Nesta, não são utilizadas linguagens de *workflow* e a definição dos passos, com o *script* e a `Docker` *Image*, são feitas com subseções da seção `workflow`.

As funcionalidades do `Reana` são implementadas em uma arquitetura cliente-servidor, baseada em microsserviços, a qual é ilustrada na Figura 2.10. No fluxo de uso dessa arquitetura, os pesquisadores preparam um *runnable recipe* e o descrevem no arquivo de configuração. Após criado, faz-se o envio para o servidor, que o valida e executa. Os resultados gerados podem ser obtidos através de *download*.

Figura 2.10 - Visão geral da arquitetura do `Reana`.



Fonte: Adaptado de Šimko et al. (2019).

O `Reana Cluster` é o servidor da arquitetura. Sua responsabilidade é fornecer funcionalidades para o gerenciamento de *runnable recipe*. Como ilustrado na Figura 2.10, as funcionalidades desse componente são fornecidas através de uma API REST, especificada no formato OPENAPI v3.0 (2022). Nessa, são fornecidas operações para executar, reexecutar, parar e excluir os *runnable recipe*. Na operação de execução, quando requisitada, o pesquisador faz o envio de um pacote contendo o arquivo de configuração e todos os materiais referenciados nele. O servidor verifica a con-

[37]Disponível em: <`https://yadage.readthedocs.io/`>. Acesso em: 07/05/2021

sistência do *runnable recipe* recebido e o executa utilizando as descrições da seção `workflow`, do arquivo de configuração. Após a execução, os materiais gerados ficam armazenados no servidor. Na reexecução, esses materiais são utilizados em uma nova execução. A exclusão os remove do servidor.

Para a execução dos processamentos, o `Reana` pode ser configurado com diferentes "*backends* de execução". Estes são componentes da plataforma, que gerenciam os recursos computacionais, os distribuindo entre as requisições de execução e reexecução. Exemplos de *backends* suportados pelo `Reana` são os *softwares* `Kubernetes`, `HTCondor` e `Slurm`. Em um ambiente *cluster*, esses *backends* permitem que o `Reana` distribua, quando possível, as etapas de execução de um *runnable recipe* entre várias máquinas. A persistência dos resultados gerados pode ser configurada para utilizar os sistemas de arquivo `Ceph` ou `EOS`.

No lado cliente da arquitetura, para facilitar a utilização das funcionalidades disponibilizadas pela API REST do `Reana Cluster`, estão disponíveis duas ferramentas. A primeira delas é o `Reana Client`, uma CLI que permite aos pesquisadores utilizar todas as operações fornecidas pelo servidor. Para o início de uma execução, por exemplo, após definir o *runnable recipe* no arquivo de configuração, o pesquisador utiliza a CLI. Esta, automaticamente cria o pacote que contém todos os elementos descritos no arquivo de configuração e em seguida o envia para o servidor. A segunda ferramenta disponível é o `Reana Web Interface`. Esta, fornece apenas funcionalidades que permitem a visualização dos *status* e *logs* de execução dos *runnable recipe*.

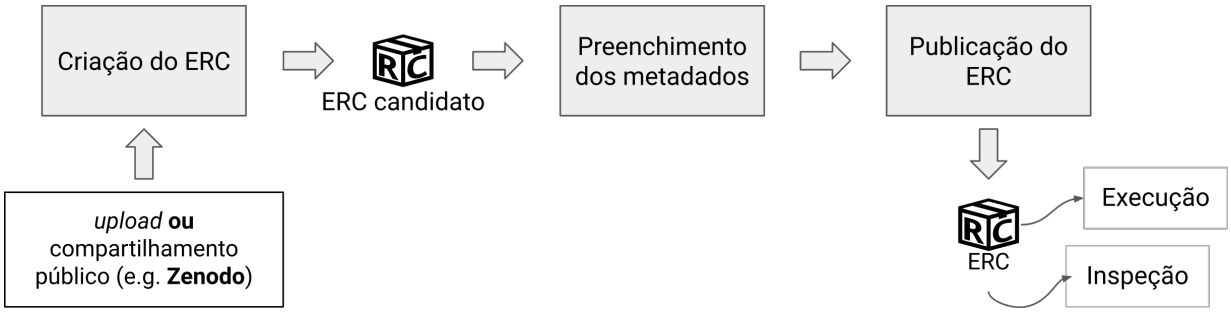
Nüst (2018) apresenta uma solução *open source* denominada de Opening Reproducible Research (`o2r`), que auxilia o processo de produção e publicação de pesquisa reprodutível. Faz isso através da criação de ERCs. Nestes, como apresentado na Seção 2.3.4, estão todos os elementos utilizados para a reprodução dos resultados de uma pesquisa. Isto inclui dados, *scripts* e ambiente computacional. Adicionalmente, no `o2r`, faz-se necessário a existência de um *script* principal, que realiza todas as etapas do fluxo de processamento da pesquisa. Nesse *script*, pode-se fazer a execução de outros *scripts* e mesmo a utilização de funções definidas em outros arquivos de código. É esperado que sua criação seja realizada em um *Computational Notebook*, no formato `R Markdown`[38].

A utilização dos ERCs no `o2r` é feita com base em um ciclo de vida. Neste, fazem

[38] `R Markdown` possui estrutura análoga a um `Jupyter Notebook`

parte as etapas de criação, validação e publicação de um ERC. Na Figura 2.11 é ilustrada cada uma dessas etapas, suas ligações e elementos resultantes. Na primeira etapa, de criação de um ERC, o pesquisador pode fazer o *upload* dos materiais de sua pesquisa ou importar materiais publicados em repositórios digitais, como `Zenodo`[39]. Ao final dessa etapa, tem-se criado um ERC candidato, que pode ser executado e alterado por pesquisadores com permissão de acesso adequado, mas não está disponível ao público. A execução do ERC é feita com o *script* principal. Na segunda etapa, o candidato tem seus metadados preenchidos e a execução certificada pelos pesquisadores responsáveis, gerando um ERC válido para publicação. A terceira etapa realiza a publicação. Uma vez publicado, o ERC pode ser acessado, executado e inspecionado por outros pesquisadores. Os autores, neste estágio, podem ainda enviar o ERC para um repositório de armazenamento e preservação.

Figura 2.11 - Ciclo de vida de um ERC na plataforma `o2r`.

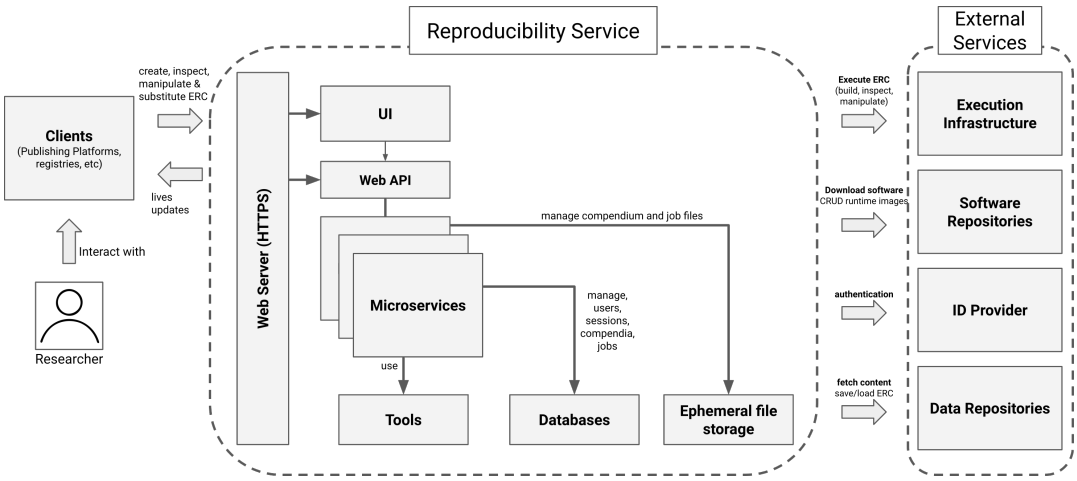


Fonte: Adaptado de Nüst (2018).

As funcionalidades do `o2r`, que materializam o ciclo de vida, são implementadas em uma arquitetura cliente-servidor, baseada em microsserviços. A Figura 2.12 ilustra a visão geral da arquitetura. Todos os componentes implementados na plataforma estão inseridos no contexto do `Reproducibility Service`. O componente principal desse contexto é o `Web API`, uma API REST que fornece as operações do ciclo de vida dos ERCs. As funcionalidades desse componente são implementadas através de microsserviços. Os `External Services` representam serviços externos que a plataforma utiliza para compor suas funcionalidades.

[39]Disponível em: <`https://zenodo.org/`>. Acesso em: 13/04/2021

Figura 2.12 - Visão geral da arquitetura do `o2r`.



Fonte: Adaptado de Nüst (2018).

No lado cliente, a utilização das operações do `Web API` podem ser feitas através do componente `UI`. Este é implementado como uma aplicação *web*, que possibilita o uso do `Web API` em um ambiente de alto nível. Através dessa interface, os pesquisadores podem manipular os ERCs seguindo o ciclo de vida.

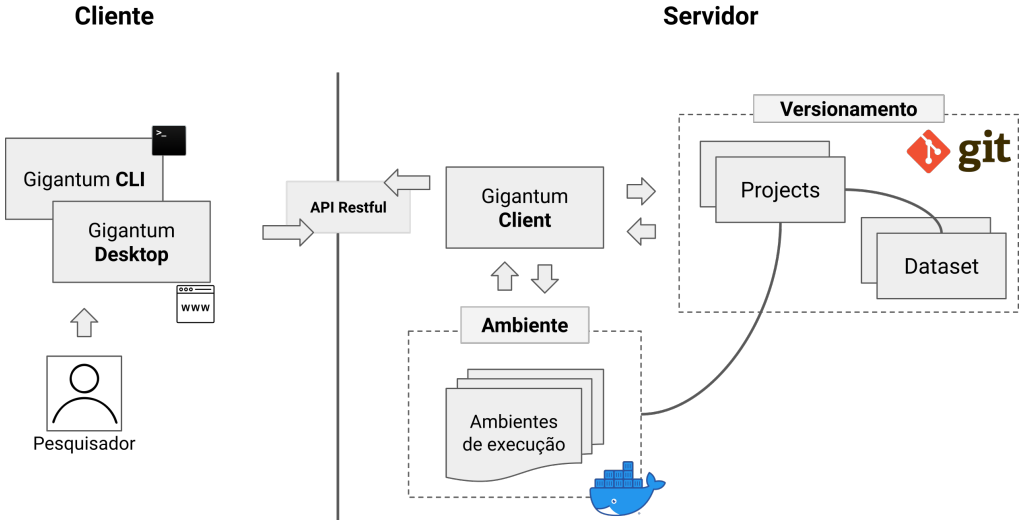
O `Gigantum` (BURNS et al., 2018) é uma plataforma *open source* para a produção de pesquisas colaborativas e reprodutivas. Faz isso através do gerenciamento automatizado dos objetos de pesquisa, os quais incluem dados, *scripts* e ambiente computacional. Nesse gerenciamento são considerados o registro das modificações nos objetos de pesquisa e o provisionamento de ambientes de execução.

No `Gigantum`, os objetos de pesquisa são representados através de *Projects* e *Datasets*. Um *Project* representa um projeto de pesquisa. São associados a essa entidade, os *scripts* de processamento e um `Dockerfile` do ambiente de execução, no qual é especificado o uso de uma interface `RStudio` ou `Jupyter Notebook`. Os dados de um *Project* são armazenados em uma entidade à parte, o *Dataset*. Ambas entidades estão associadas a um repositório `git`, que permite o versionamento dos objetos de pesquisa registrados em cada uma delas.

A implementação das funcionalidades do `Gigantum` é feita com base em uma arquitetura cliente-servidor, a qual é ilustrada na Figura 2.13. De maneira geral, o uso dessa arquitetura é feito pelo pesquisador em uma interface cliente. Nesta interface são

disponibilizadas operações do servidor para a criação de *Projects* e associação com *Datasets* e ambientes de execução. Pode-se executar os ambientes para a produção dos *scripts* de processamento e geração dos resultados da pesquisa. O `Gigantum` é distribuído como um pacote de *software standalone*. Assim, esse fluxo de uso é feito na máquina de cada pesquisador, com os componentes cliente e servidor operando juntos.

Figura 2.13 - Visão geral da arquitetura do `Gigantum`.



Fonte: Produção do autor.

No servidor, as funcionalidades são providas pelo `Gigantum Client`. Este componente é responsável pelas operações de gerenciamento dos *Projects* e *Datasets*, incluindo o versionamento dos objetos de pesquisa e a criação de ambientes de execução. Estas funcionalidades são fornecidas através de uma API REST. Nesta, pode-se fazer a criação, edição e exclusão de *Projects* e *Datasets*. Faz-se também a criação e execução do ambiente associado a um *Project*. Adicionalmente, nos ambientes em execução, o servidor verifica modificações nos arquivos do *Project* e os versiona automaticamente.

Outra funcionalidade disponibilizada pelo `Gigantum Client` é o compartilhamento de *Projects* e *Datasets* entre pesquisadores. Isto facilita o desenvolvimento colaborativo, uma vez que uniformiza os objetos que estão sendo utilizados na pesquisa, indo do ambiente de execução aos *scripts* de processamento. Entretanto, o uso dessa

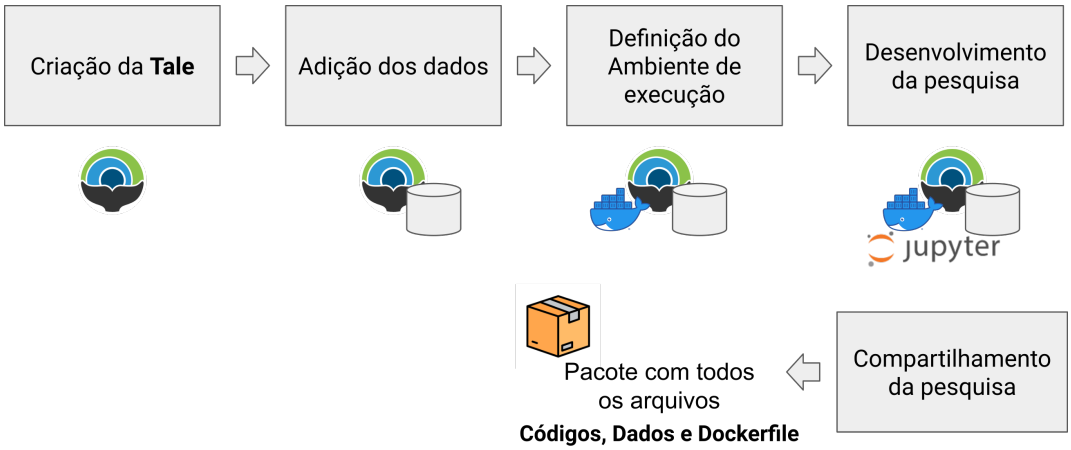
funcionalidade depende do `Gigantum Hub`, um serviço externo à plataforma, para o compartilhamento e execução de ambientes `Gigantum` na nuvem. Seu uso está sujeito a limitações de espaço e cobranças adicionais.

No lado cliente, para a utilização facilitada da API REST do `Gigantum Client`, são disponibilizadas duas ferramentas: `Gigantum CLI` e `Gigantum Desktop`. Estas são, respectivamente, uma CLI e uma interface *web*. Ambas, fornecem as mesmas funcionalidades, utilizando todas as operações de gerenciamento oferecidas pelo servidor.

Apresentada por Brinckman et al. (2019), o `Whole Tale` é uma plataforma *open source* para a produção de pesquisas reprodutíveis. Para isso, a plataforma realiza a criação e uso das *Tales*. Estas são a materialização do conceito de RC na plataforma, sendo utilizadas para centralizar e organizar *scripts*, dados, ambiente computacional e metadados da pesquisa.

No `Whole Tale`, a criação das *Tales* considera as principais etapas do ciclo de vida de um projeto de pesquisa. Isto inclui a ingestão de dados, definição do ambiente computacional, desenvolvimento dos *scripts* de processamento e publicação dos resultados gerados. Na Figura 2.14 é ilustrado esse fluxo e as transformações realizadas na *Tale*. A primeira etapa é a criação da *Tale*. Em seguida, faz-se a adição do conjunto de dados. Para tal, pode-se fazer o *upload* dos dados ou a importação de dados já publicados em repositórios de preservação. Na terceira etapa, faz-se a configuração do ambiente computacional, através da criação de um `Dockerfile`. Neste, especifica-se o uso de uma interface `RStudio` ou `Jupyter Notebook`. A quarta etapa representa o desenvolvimento da pesquisa, em que se faz a criação dos *scripts* de processamento e geração dos resultados. Por fim, na quinta etapa, pode-se compartilhar a pesquisa utilizando o portal da plataforma ou exportando a *Tale* em um arquivo no formato `zip`.

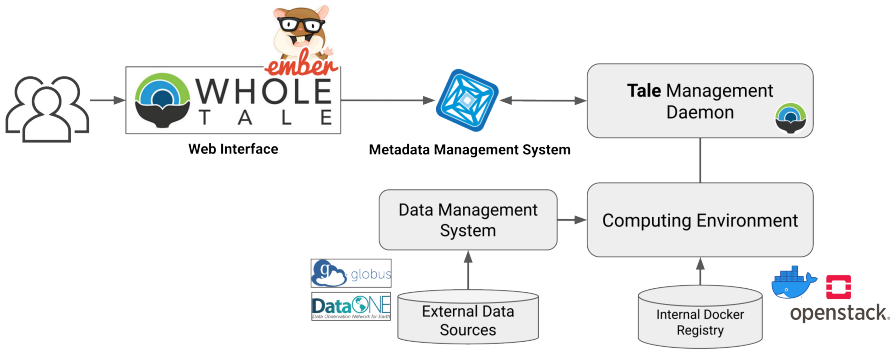
Figura 2.14 - Fluxo de operação da plataforma `Whole Tale`.



Fonte: Produção do autor.

O fluxo de operação do `Whole Tale` é implementado através de microsserviços, em uma arquitetura cliente-servidor, ilustrada na Figura 2.15. As funcionalidades do servidor são implementadas pelo componente `Metadata Management System`. Este é o responsável em fornecer as operações para o gerenciamento das *Tales* e dos ambientes computacionais. Quando necessário a execução dos ambientes, esse módulo faz o uso do `Tale Management Daemon`. As operações do `Metadata Management System` são disponibilizadas através de uma API REST.

Figura 2.15 - Visão geral dos componentes da plataforma Whole Tale.



Fonte: Adaptado de Brinckman et al. (2019).

A utilização das operações fornecidas pelo servidor, no lado cliente, são feitas através do componente `Web Interface`. Este é implementado como uma interface *web*, fornecendo recursos de alto nível para os pesquisadores, permitindo a realização de todas as etapas do fluxo de operação da plataforma.

2.5 Cubos de dados de Observação da Terra

Atualmente, graças a aquisição contínua de dados feita por plataformas orbitais de EO, tem-se disponível grande coleções de dados que permitem aos pesquisadores realizarem a criação de aplicações inovadoras que ajudam no entendimento das dinâmicas da superfície terrestre e os impactos a sua biodiversidade causados por atividades antrópicas (SOILLE et al., 2018). Com o uso desses dados pode-se, por exemplo, realizar avaliações temporais sobre certos fenômenos e então determinar suas causas (GIULIANI et al., 2020). Tais informações, por sua vez, podem ser utilizadas na criação de políticas públicas que auxiliam no desenvolvimento sustentável de atividades resilientes em um país (GROUP ON EARTH OBSERVATIONS, 2022). Por exemplo, a iniciativa Digital Earth Africa[40], por meio de avaliações temporais, tem auxiliado países como a Tanzânia, a combater a seca[41].

No entanto, por conta do grande volume de dados, por vezes, o uso de técnicas convencionais para a manipulação desses dados pode não ser viável, uma vez que isso excede as capacidades de memória, armazenamento e processamento dos computadores tradicionalmente disponíveis para essas atividades (CAMARA et al., 2014). Isso faz com que a extração de informação desses dados e o uso de todo seu potencial, torne-se um desafio (APPEL; PEBESMA, 2019).

Para solucionar esse problema e permitir que os pesquisadores criem aplicações que utilizem desse grande volume de dados, tem-se adotado na comunidade de EO, o uso de Cubos de Dados de Observação da Terra (EODC, do inglês *Earth Observation Data Cubes*). Existem diversas definições para um cubo de dados na literatura (SIMOES et al., 2021). Neste trabalho, será feito o uso da definição adotada por Ferreira et al. (2020), em que se tem que um cubo de dados é uma matriz de quatro dimensões: X (longitude), Y (latitude), *Tempo* e *Bandas*. Além disso, tem-se que um conjunto de imagens para ser considerado um cubo de dados deve possuir as seguintes características:

[40] Disponível em: <https://www.digitalearthafrica.org/>. Acesso em: 02/12/2022

[41] Disponível em: <https://www.digitalearthafrica.org/why-digital-earth-africa/impact-stories/using-satellite-data-combat-drought-monitoring-lake-sulunga>. Acesso em: 16/09/2022

i. A dimensão espacial deve se referir a um único sistema de referência espacial;

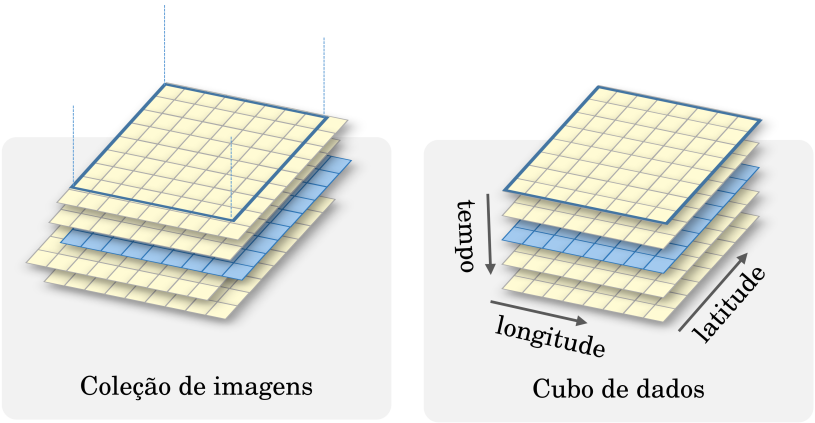
ii. Os *pixels* do cubo têm tamanho espacial constante;

iii. Cada *pixel* do cubo representa um único valor;

iv. Alinhamento espacial e temporal dos *pixels*.

Na Figura 2.16 é ilustrado a diferença entre coleções de imagens e um cubo de dados. Na esquerda, tem-se a coleção de imagens, em que as propriedades referidas não estão presentes. Assim, mesmo que haja várias imagens de uma mesma região, a avaliação temporal pode ser comprometida pela inconsistência espacial. Na direita, por outro lado, tem-se um cubo de dados, no qual os *pixels* tem tamanho constante e estão alinhados no espaço e no tempo.

Figura 2.16 - Diferença entre coleção de imagens e cubos de dados.



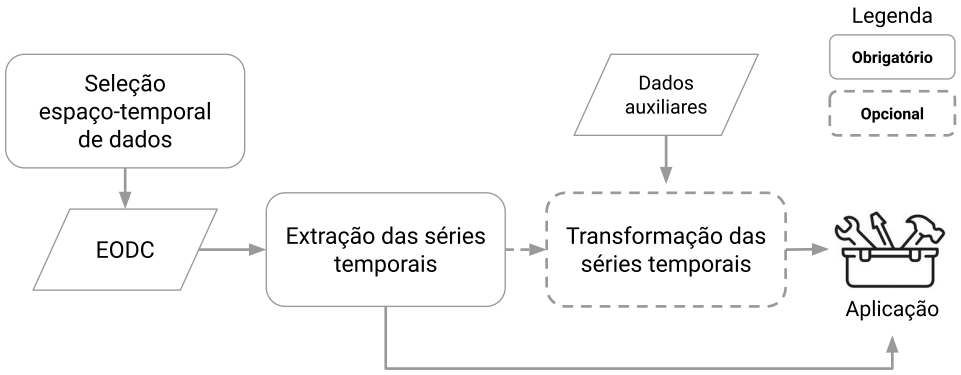
Fonte: Adaptado de Simoes et al. (2021).

O uso dos EODCs beneficia a avaliação temporal das mudanças na superfície terrestre (APPEL; PEBESMA, 2019; KOPP et al., 2019). Tal característica possibilita a realização de avaliações temporais em larga escala para o entendimento de diferentes fenômenos espaciais e temporais na dinâmica de LULC (DHU et al., 2019). De maneira geral, para o uso dos EODCs, aplicam-se as etapas do fluxo de processamento apresentado na Figura 2.17. A definição do fluxo foi realizada com base na generalização das metodologias apresentadas por Sanchez et al. (2019), Picoli et al.

(2020), Ferreira et al. (2020) e Simoes et al. (2021).

No fluxo ilustrado na Figura 2.17, para o uso dos EODCs em uma aplicação específica, inicialmente faz-se a seleção das cenas do EODC que serão utilizadas. Em seguida, realiza-se a extração das séries temporais associadas às cenas selecionadas. Opcionalmente, para necessidades específicas da aplicação, fazem-se transformações nas séries temporais. Por fim, a aplicação utiliza as séries temporais extraídas e também as que foram transformadas, quando necessário.

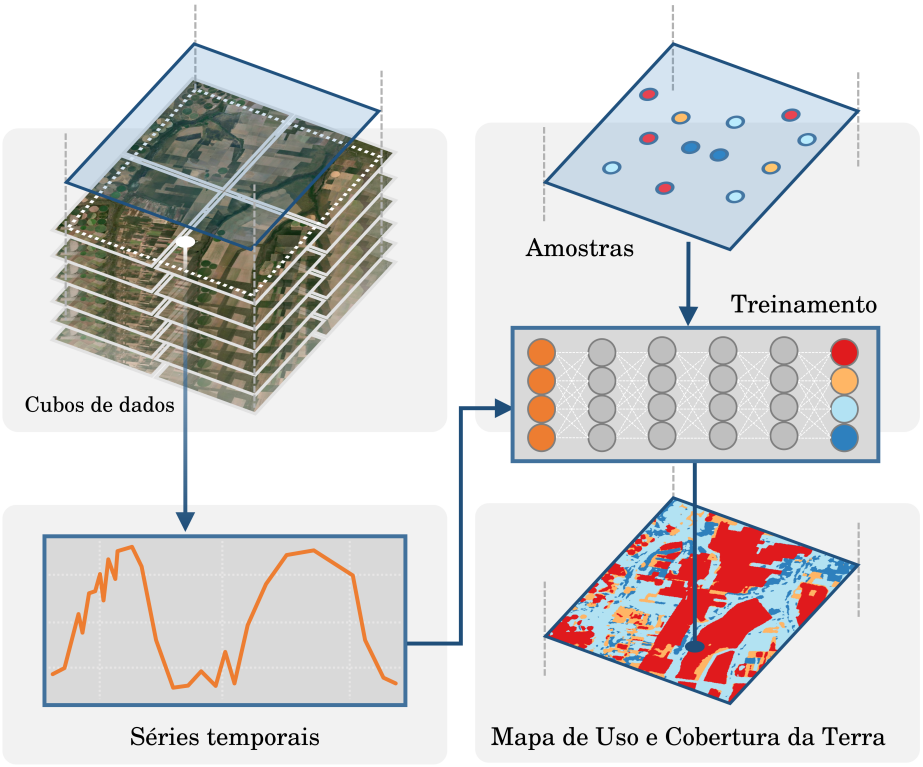
Figura 2.17 - Fluxo geral de uso dos EODC.



Fonte: Produção do autor.

Um exemplo de aplicação em que se faz o uso das séries temporais é o de geração de mapas de LULC. Nesse caso, conforme apresentado por Simoes et al. (2021), cada um dos instantes de tempo disponíveis na série temporal são utilizados como entrada para algoritmos de Aprendizado de Máquina (ML, do inglês *Machine Learning*), que as classificam. Quando essa operação é realizada para toda uma região, tem-se a formação de um mapa temático de LULC. Na Figura 2.18, tem-se um resumo dessa metodologia. Primeiro, faz-se a extração das séries temporais. Em seguida, essas são associadas a amostras de LULC coletadas por especialistas que determinam qual comportamento é representado pela série temporal em questão. Então, essas séries com classes associadas são utilizadas para o treinamento de um algoritmo de ML. Após isso, faz-se o uso do modelo de ML treinado para a classificação das séries temporais de uma determinada região, gerando assim, o mapa de LULC.

Figura 2.18 - Exemplo de aplicação de EODCs para a geração de mapas de LULC.



Fonte: Adaptado de Simoes et al. (2021).

3 SPATIOTEMPORAL OPEN RESEARCH MANAGER

O projeto BDC, criado por pesquisadores do INPE, tem quatro objetivos principais (FERREIRA et al., 2020):

i. Criar e disponibilizar produtos de dados prontos para análise para todo o território nacional;

ii. Criar e disponibilizar cubos de dados multidimensionais;

iii. Aplicar, propor e desenvolver tecnologias para a análise e processamento desses dados;

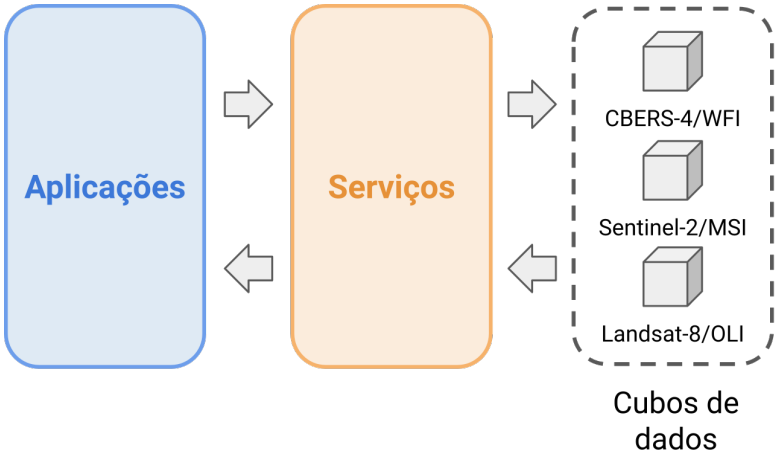
iv. Geração de mapas de LULC.

Para atingir os objetivos (i) e (ii), o BDC realiza a criação de EODC com imagens dos satélites-sensores CBERS-4/WFI, Landsat-8/OLI e Sentinel-2/MSI. Os objetivos (iii) e (iv) estão sendo alcançados através do desenvolvimento da `Plataforma BDC`, a qual disponibiliza um conjunto de serviços que permite a descoberta, busca e acesso aos EODCs. Um exemplo de serviço disponibilizado é o *STAC - SpatioTemporal Asset Catalog* (ZAGLIA et al., 2019; ZAGLIA, 2020), que permite realizar buscas espaço-temporais aos metadados dos produtos de dados disponíveis na plataforma. Outros exemplos de serviços incluem o *WTSS - Web Time Series Service* (QUEIROZ et al., 2015; VINHAS et al., 2017) e o *WLTS - Web Land Trajectory Service* (ZIOTI et al., 2019; ZIOTI, 2020; ZIOTI et al., 2022). A plataforma também torna possível o uso desses serviços no ambiente de trabalho dos usuários, fornecendo bibliotecas para linguagens de programação como R e Python. Assim, pode-se criar aplicações com base nas funcionalidades dos serviços, como o Data Cube Explorer[1] e os *plugins* dos serviços WLTS e WTSS para o QGIS. A Figura 3.1 apresenta uma visão de alto nível da arquitetura da `Plataforma BDC` e a ligação entre os elementos citados.

No projeto BDC, os pesquisadores têm feito o uso dos dados e funcionalidades fornecidos pela `Plataforma BDC` para a criação de pesquisas em cenários que variam da geração de mapas de LULC à avaliação e estudo de mudanças em áreas urbanas. Na realização dessas atividades, no entanto, atualmente não há no BDC uma ferramenta que permita a construção facilitada de análises dos EODCs colaborativamente e reprodutível. Garantir a reprodutibilidade depende completamente dos pesquisadores e sua organização durante o desenvolvimento das etapas de processamento e análise.

[1] Portal de acesso aos dados do BDC: `https://brazildatacube.dpi.inpe.br/portal/`.

Figura 3.1 - Arquitetura resumida da `Plataforma BDC`.



Fonte: Adaptado de Ferreira et al. (2020).

Isto faz com que pequenas falhas na comunicação por qualquer um dos membros envolvidos cause a impossibilidade de rastreio da proveniência dos resultados. Além disso, cada pesquisador utiliza seu próprio ambiente para realizar o processamento dos dados, tornando difícil a definição de um ambiente único que possa ser utilizado para reprodução e reutilização dos resultados.

Uma solução inicial para a reprodutibilidade no BDC foi proposta por Mariano et al. (2020). Na abordagem dos autores, fez-se o desenvolvimento do pacote R `sits.rep`, que disponibiliza funcionalidades que tornam o uso do pacote R `sits` (SIMOES et al., 2021) reprodutível. No contexto do BDC, o pacote `sits` é utilizado nas atividades de geração de mapas de LULC. Isto torna a ferramenta apresentada por Mariano et al. (2020) específica ao domínio da aplicação do `sits`, podendo não ser utilizada para todas as necessidades das etapas de uso dos EODCs apresentado na Seção 2.5. Não sendo o objetivo dos autores, também não são tratadas as questões de trabalho colaborativo, em que cada etapa do processamento pode ser feita por um pesquisador, variando a linguagem de programação, forma de trabalho e ambiente de processamento.

Neste trabalho, com o objetivo de auxiliar os pesquisadores de EO no desenvolvimento colaborativo de pesquisas reprodutíveis, é proposta a Plataforma SpatioTemporal Open Research Manager (`Storm`). Sua concepção e desenvolvimento foram feitas com base nos princípios e práticas que suportam a Pesquisa Aberta e Re-

produtível, dando ênfase, em especial, as etapas da pesquisa que dizem respeito ao processo, análise, compartilhamento e preservação de dados.

Com esse objetivo e a interseção das necessidades do projeto BDC com os objetivos deste trabalho, a especificação e desenvolvimento da `Plataforma Storm` foram realizados considerando o uso de serviços e funcionalidades disponíveis na plataforma computacional desenvolvida e mantida pelo projeto BDC. Além disso, de forma pragmática, a especificação do fluxo de funcionamento da plataforma, bem como os recursos por ela disponibilizados para auxiliar os pesquisadores, foram definidos com base nos requisitos e modo de trabalho do projeto BDC. Isso evita que a plataforma considere características que são irreais e não facilitam verdadeiramente as atividades diárias dos pesquisadores.

No restante deste Capítulo serão apresentados os requisitos e arquitetura definidos para a `Plataforma Storm`, organizados como a seguir. Na Seção 3.1, faz-se a apresentação dos requisitos da solução proposta. As entidades utilizadas no `Storm` são descritas na Seção 3.2. O fluxo de utilização do `Storm` é apresentado na Seção 3.3. Por fim, na Seção 3.4 é detalhado a arquitetura conceitual da plataforma.

3.1 Requisitos

A especificação de uma plataforma computacional, pode considerar diferentes caminhos para atingir seus objetivos. Neste trabalho, buscando realizar a especificação e desenvolvimento da `Plataforma Storm` de forma pragmática, fez-se o levantamento de um conjunto de requisitos.

O levantamento desses requisitos considerou aspectos práticos e teóricos da produção colaborativa de pesquisas reprodutíveis. Para os aspectos teóricos, foram considerados os textos de recomendação da literatura em como pesquisas reprodutíveis podem ser conduzidas (SANDVE et al., 2013; RULE et al., 2019; NüST et al., 2020; ALSTON; RICK, 2021). Para os termos práticos, foram considerados os fluxos de pesquisas desenvolvidos no projeto BDC. Neste segundo caso, o estudo e entendimento da dinâmica do fluxo de desenvolvimento científico foi feita de maneira prática, com o autor colaborando diretamente com as atividades do projeto.

Com base nas informações levantadas, fez-se a especificação de oito requisitos, os quais guiaram toda a especificação da plataforma e as decisões arquiteturais tomadas em sua concepção. Esses requisitos são listados e descritos em detalhes abaixo:

R1. Trabalho colaborativo integrado: Deve ser suportado o trabalho colabo-

rativo, de modo que o fluxo de processamento de uma pesquisa, juntamente com seus materiais, possam ser compartilhados entre vários pesquisadores. O compartilhamento deve considerar que o trabalho iniciado por um pesquisador, possa ser continuado e finalizado por outro;

R2. Registro de proveniência: Em um ambiente colaborativo, os resultados podem ser gerados por vários usuários diferentes, com recursos e metodologias que variam. Para auxiliar a reprodutibilidade do trabalho, a solução proposta deve conseguir registrar a proveniência de todos os dados gerados durante a pesquisa;

R3. Auxílio à reprodutibilidade durante a produção dos resultados: Pesquisas em EO podem ser feitas considerando grande volumes de dados. Neste cenário, a garantia de reprodutibilidade deve ser feita no momento em que os resultados estão sendo gerados, uma vez que pode ser inviável a reexecução de um processamento para validar a possibilidade de reprodução;

R4. Uso de tecnologias *Open Source*: Deve empregar o uso de tecnologias existentes, com licença *Open Source*, que auxiliem o desenvolvimento e permitam sua rápida adoção;

R5. Provisionamento de ambientes de trabalho interativos na produção da análise: Deve facilitar a disponibilização de ambientes interativos para processamento, evitando que o pesquisador tenha que configurá-los;

R6. Suporte para Reprodução/Replicação facilitada: Deve fornecer funcionalidades que auxilie na reprodução das pesquisas produzidas. A replicação, que considera a troca dos dados de entrada e métodos utilizados, também deve ser considerada;

R7. Suporte a múltiplos ambientes de execução: Em um fluxo de processamento com vários usuários, existe a possibilidade de cada etapa ser produzida com uma certa tecnologia ou conjunto de ferramentas. Assim, a solução proposta deve suportar a construção de fluxos de processamento com etapas que possuam ambientes de execução diferentes;

R8. Gerenciamento e preservação dos elementos da pesquisa: Deve conseguir auxiliar o pesquisador no controle de versão, gerenciamento, organização e preservação de todos os elementos que estão sendo produzidos na pesquisa.

Tendo os requisitos definidos, pode-se entender melhor quais as principais necessi-

dades relacionadas ao desenvolvimento de pesquisas em EO no que diz respeito à reprodutibilidade. Neste caso, como pode-se perceber, existe uma grande necessidade de ferramentas que facilitem e automatize o gerenciamento de ambientes para equipes multidisciplinares.

De modo a colocar em perspectiva essas necessidades e entender como as principais soluções tecnológicas disponíveis na literatura lidam com elas, fez-se uma análise comparativa. Nessa análise, foram consideradas as plataformas apresentadas na Seção 2.4, e após uma avaliação teórica de suas arquiteturas e uso prático de cada uma delas, pode-se definir se elas são capazes ou não de lidar com os requisitos apresentados. O resultado dessa análise é apresentado na Tabela 3.1.

Tabela 3.1 - Tabela comparativa entre as plataformas de reprodutibilidade e os requisitos da `Plataforma Storm`.

Requisitos	**Reana**	**Gigantum**	**o2r**	**ReproZip**	**WholeTale**
Trabalho colaborativo integrado		✓			
Registro de proveniência	✓		✓	✓	
Auxílio à reprodutibilidade durante a produção dos resultados	✓		✓	✓	
Uso de tecnologias Open Source	✓	✓	✓	✓	✓
Provisionamento de ambientes de trabalho interativos na produção da análise		✓			✓
Suporte para Reprodução/Replicação	✓		✓	✓	
Múltiplos ambientes de execução	✓				
Gerenciamento e preservação dos elementos da pesquisa			✓		✓

Fonte: Produção do autor.

Como pode-se notar na Tabela 3.1, boa parte das plataformas não possui funcionalidades para as necessidades de pesquisa colaborativa, fácil e reprodutível. Mesmo as que se aproximam, apresentam formas de funcionamento que podem requerer mudanças na maneira como os pesquisadores realizam as análises, como no `Reana` e `o2r`. `Gigantum` oferece opções de compartilhamento, mas deixa a cargo dos pes-

quisadores a organização e registro de proveniência, além de não suportar múltiplos ambientes. `Whole Tale` além de ter os mesmos problemas encontrados no `Gigantum`, não permite trabalho colaborativo. Por fim, o `ReproZip` é uma ferramenta que, através do registro de ambientes, evita a necessidade de provisionamento de ambientes, mas não fornece suporte ao trabalho colaborativo.

3.2 Entidades

Para a especificação da arquitetura e dos fluxos de funcionamento da `Plataforma Storm` foi definido um conjunto de entidades. Estas são utilizadas para representar cada um dos conceitos trabalhados pelo sistema. Nos parágrafos abaixo, divididas pelo tema a qual estão vinculadas, são apresentadas as entidades definidas.

Usuários

Researcher: Representa o registro de um pesquisador-utilizador na lista de usuários da `Plataforma Storm`;

Research Group: Conjunto de *Researcher*, utilizado para representar um laboratório ou grupo de pesquisa;

Conjunto de dados

Dataset: Representa um conjunto de dados utilizados em uma pesquisa. Esta entidade pode representar dados de diferentes tipos, geridos e armazenados na `Plataforma Storm` ou disponíveis em um serviço *web*.

Computação e processamento

Computational Environment: Representa um ambiente computacional utilizado em um processamento. Neste ambiente, tem-se presente pacotes de *software* especializados no processamento e análise de dados e uma interface interativa;

Research Workflow: Representa um DAG com o *workflow* seguido para a obtenção dos resultados de uma pesquisa. Com as informações desta entidade, pode-se identificar *o que*/*quem*/*onde*/*quando*/*como* de todos os elementos da pesquisa, auxiliando a proveniência. Um *Research Workflow* é formado por um conjunto de *Execution Compendium*;

Execution Compendium: Representa a execução de um processo que gera resultados na pesquisa. Por exemplo, processamento com um *script* ou ferramenta de linha de

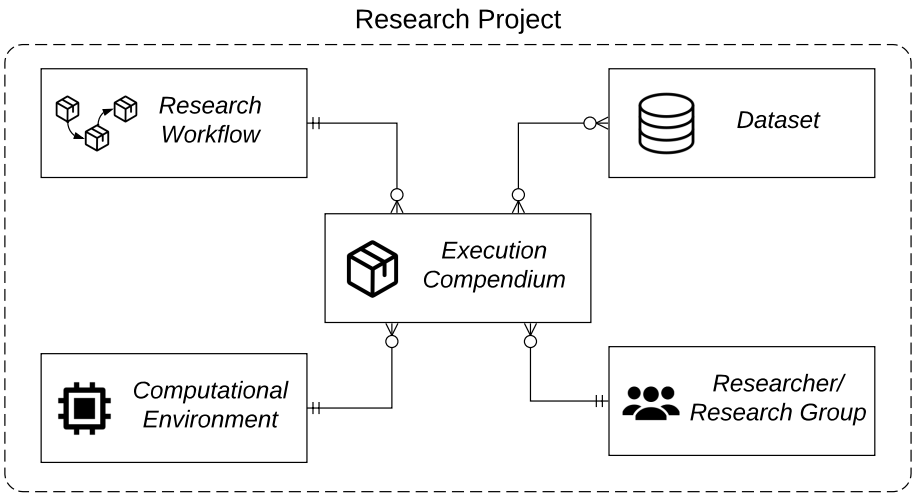
comando. Esta entidade mantém informações do *Researcher* que a criou, do *Research Workflow* a qual faz parte, bem como do *Computational Environment* e `Datasets` utilizados para sua execução. Essa entidade é uma especialização do conceito de RC, sendo utilizada para representar todos os elementos utilizados em um único processamento;

Unidade de gerenciamento

Research Project: Unidade de gerenciamento que representa a agregação de *Researcher/ Research Group*, *Research Workflow* e *Dataset* utilizados durante a criação de uma pesquisa. Esta entidade representa a utilização do conceito de unidade organizacional da pesquisa, vindo do RC, na `Plataforma Storm`.

Na Figura 3.2 é ilustrado, em alto nível, a relação de cada uma das entidades definidas.

Figura 3.2 - Relacionamento das entidades da `Plataforma Storm`.



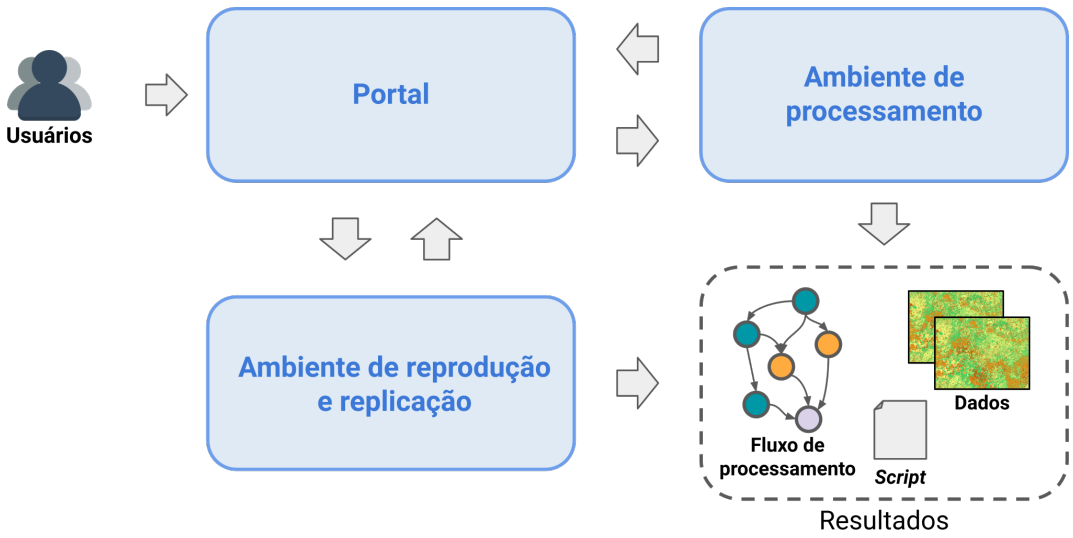
Fonte: Produção do autor.

Nos textos que seguem, as entidades definidas nos tópicos acima serão apresentadas em *itálico*, de modo que fique fácil ao leitor a identificação de quando as entidades estão sendo referenciadas.

3.3 Fluxo de utilização

Para iniciar a especificação da `Plataforma Storm`, primeiro fez-se a definição geral de seus componentes e fluxo de funcionamento. Os elementos definidos nesse processo são ilustrados na Figura 3.3. Como pode ser notado, a `Plataforma Storm` e suas funcionalidades são acessadas através de um portal *web*. Após o acesso, dois cenários de uso são possíveis para os usuários. O primeiro deles é o de realização de pesquisa reprodutível. Neste cenário, o pesquisador pode requisitar ambientes de processamento em que, colaborativamente, pode realizar o desenvolvimento da pesquisa. Ao final, tem-se os resultados, que incluem dados gerados, *scripts* e fluxo de processamento. O segundo cenário considera a reprodução ou replicação de uma pesquisa. Para isso, o pesquisador busca no portal os trabalhos já finalizados. Após encontrar o trabalho desejado, o pesquisador pode reproduzi-lo ou replicá-lo.

Figura 3.3 - Fluxo geral de funcionamento da `Plataforma Storm`.



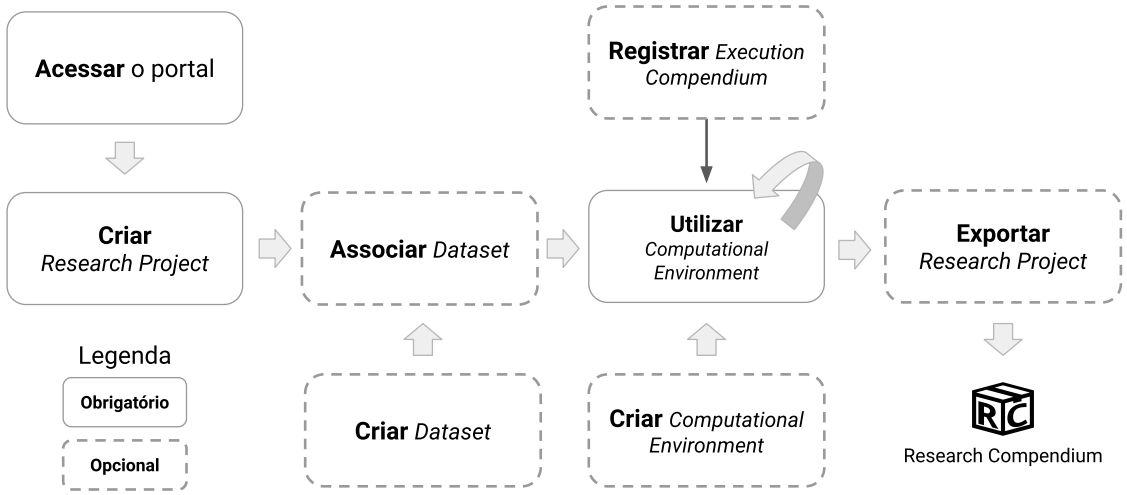
Fonte: Produção do autor.

Utilizando como base o fluxo geral da Figura 3.3, os itens abaixo fazem a especificação do funcionamento da `Plataforma Storm` em cada um de seus cenários de uso. De modo a suplementar as descrições realizadas, estão nos Apêndices A e B, *User Scenarios* que apresentam o uso `Plataforma Storm`, através de usuários e histórias fictícias.

Produção de pesquisa reprodutível

Na Figura 3.4 é apresentado o fluxo de utilização da `Plataforma Storm` para o cenário em que há a necessidade de criação de pesquisa reprodutível e colaborativa. Inicialmente, o pesquisador deve realizar o *login* no portal. Após o acesso, faz-se a criação de um *Research Project*. Isto necessita da declaração de um conjunto mínimo de metadados, como nome, descrição, licenças e visibilidade (público ou privado). Nesta etapa, o pesquisador pode também convidar outros *Researcher* ou *Research Group* para colaborar na produção do *Research Project*.

Figura 3.4 - Fluxo de utilização da `Plataforma Storm` na produção de pesquisa reprodutível.



Fonte: Produção do autor.

Em seguida, pode-se associar ao *Research Project* criado um ou mais *Datasets*. Para isso, duas opções estão disponíveis. Na primeira delas, o pesquisador pode fazer a associação com *Datasets* já publicados. A segunda opção é a de criação de um *Dataset*. Nesta opção, o pesquisador insere um conjunto de metadados e em seguida faz a adição dos dados. Para a adição dos dados, duas formas podem ser utilizadas. A primeira é o *upload* e a segunda é a declaração do uso de serviços *web* externos à plataforma. Nesta segunda forma, o pesquisador declara o endereço do serviço, o protocolo de comunicação e quando necessário, os parâmetros utilizados para a

seleção dos dados. A `Plataforma Storm` então registra essas informações no *Dataset* e as utiliza apenas como metadados.

Feita a configuração do *Research Project* e os *Datasets*, o pesquisador pode iniciar o desenvolvimento e execução dos *scripts* de processamento de sua pesquisa. Isto é realizado através da seleção e uso de um *Computational Environment*. Para tal, duas opções estão disponíveis. A primeira delas, permite ao pesquisador selecionar e iniciar um *Computational Environment* que já esteja disponível na `Plataforma Storm`. Na segunda opção, o pesquisador faz a criação de um novo *Computational Environment*. Neste processo de criação, o pesquisador declara os metadados básicos, como nome e descrição, junto a informações do ambiente, como versão do SO e os pacotes de *software* utilizados. Após configurar o ambiente, o pesquisador realiza sua inicialização. No ambiente inicializado, os *Datasets* que possuem arquivos de dados vinculados (adicionados com *upload*), são adicionados ao ambiente como arquivos e diretórios com permissões restritas à "somente leitura", evitando que eles sejam alterados.

Seguindo uma estrutura não linear, o pesquisador, enquanto está desenvolvendo suas atividades de processamento, realiza diversos testes. Em um determinado momento, o resultado gerado por um teste, pode ser utilizado como o resultado correto e esperado daquela etapa da pesquisa. Assim, é necessário que a `Plataforma Storm` tenha ferramentas que tornem possível registrar as informações dos testes de modo que eles sejam reprodutíveis e permitam ao pesquisador utilizar qualquer teste como resultado, sem a necessidade de reexecução do código para verificar a reprodutibilidade. Considerando isto, em todos os *Computational Environment* criados na `Plataforma Storm` é disponibilizado para o pesquisador a funcionalidade de `Registro`. Esta, realiza a execução completa do *script* de análise e registra todas as informações necessárias para a reexecução do mesmo. As execuções de código podem ser feitas pelo pesquisador sem a realização de `Registros`, mas caso o resultado de uma execução tenha de ser registrado no *Research Project* e compartilhado com outros pesquisadores, a realização desta operação é obrigatória.

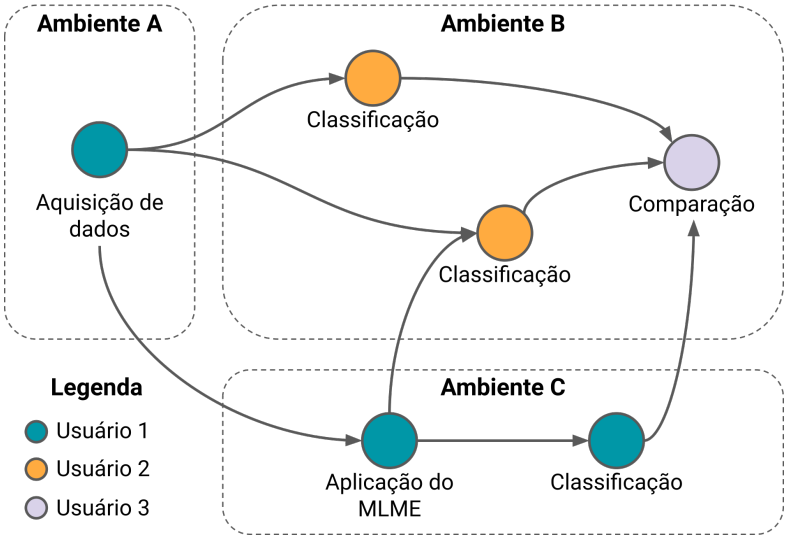
Desta forma, após acessar o *Computational Environment*, o pesquisador começa a desenvolver seus códigos de análise e processamento dos dados. Após a finalização da construção do código, o pesquisador pode fazer as execuções completas para a geração dos resultados finais, utilizando a opção de `Registro`. Ao registrar uma execução, cria-se um *Execution Compendium*. O pesquisador, ao verificar a validade dos resultados, pode salvar o *Execution Compendium* no *Research Workflow* vincu-

lado ao *Research Project* da análise. Opcionalmente, o pesquisador pode adicionar documentação e metadados extras, descrevendo, por exemplo, os parâmetros de entrada e saída do *script* executado. Ao fazer isso, o `Registro` salva essas informações, que podem ser utilizadas pela `Plataforma Storm`, em processos de replicação, para permitir a troca dos valores de entrada de forma facilitada.

Uma vez que um *Execution Compendium* é registrado no *Research Workflow*, define-se também um novo *Computational Environment* no *Research Project*. Isto permite que, um pesquisador comece uma determinada parte da pesquisa, exatamente na mesma parte em que outro pesquisador parou, com os mesmos códigos e ambiente. Além disso, a `Plataforma Storm` permite que durante uma pesquisa, o processo de criação de um *Computational Environment* e o `Registro` para a geração do *Execution Compendium*, possam ser feitos diversas vezes, por diferentes pesquisadores, até a conclusão da pesquisa.

Com essa iteração e adição dos *Execution Compendia*, ao final da pesquisa, tem-se vinculado ao *Research Project* um *Research Workflow* que mostra todos os passos tomados para a obtenção dos resultados gerados. Um exemplo de *Research Workflow* completo é apresentado na Figura 3.5. Como pode ser visto, no fluxo de processamento ilustrado, vários usuários participaram da composição do resultado final com diferentes *Computational Environments*.

Figura 3.5 - Exemplo de *Research Workflow* com o fluxo de classificação de desmatamento apresentado por Sanchez et al. (2019).



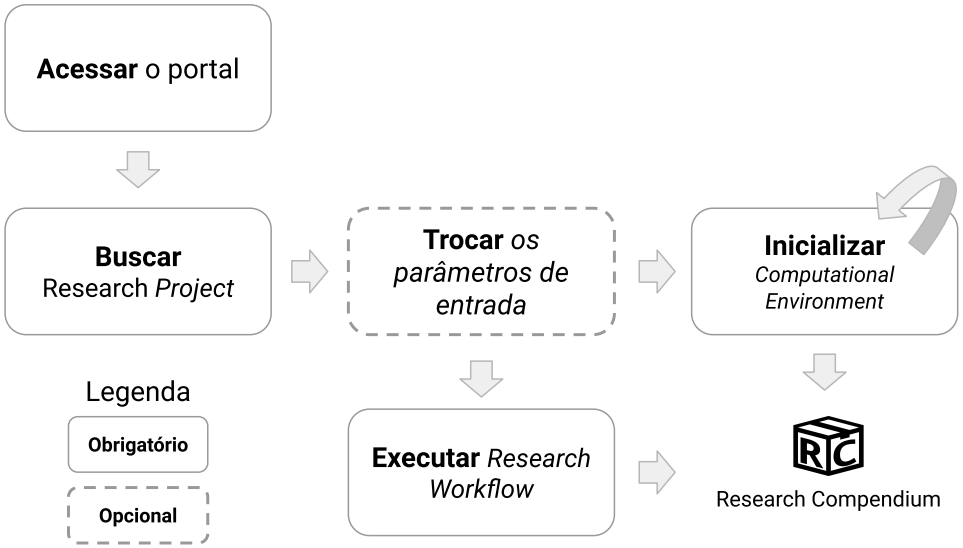
Fonte: Produção do autor.

Ao final da pesquisa, caso o *Research Project* seja privado, o pesquisador pode torná-lo público na `Plataforma Storm`, permitindo que usuários externos ao *Research Project* possam fazer o uso e exploração dos materiais gerados. Também, o pesquisador tem a opção de exportar todos os materiais que foram gerados, como resultados, códigos, fluxo de processamento e ambientes computacionais. Para tal, a `Plataforma Storm` permite que o *Research Project* seja exportado como um RC. Opcionalmente, devido à forma de organização e descrição da pesquisa provida pela plataforma, os *Research Projects* podem ser facilmente depositados em repositórios de disseminação, preservação de longo prazo e execução. Isso inclui repositórios como `Zenodo` e `GEO Knowledge Hub` e plataformas como o `o2r`.

Reprodução e Replicação de pesquisa

No segundo cenário de uso é apresentado a necessidade de reprodução e replicação de uma pesquisa reprodutível disponível na plataforma. Na Figura 3.6 é ilustrado o fluxo de utilização da `Plataforma Storm` para esse cenário. Da mesma forma como feito para a produção da pesquisa, para a reprodução e replicação, inicialmente o pesquisador deve acessar e realizar *login* no portal da `Plataforma Storm`. Após o acesso, utilizando o mecanismo de busca de *Research Projects* publicados, o pesquisador seleciona qual será utilizado.

Figura 3.6 - Fluxo de utilização da `Plataforma Storm` para a reprodução/replicação de pesquisa.



Fonte: Produção do autor.

Acessando a página do *Research Project* selecionado, são apresentados ao pesquisador todos os elementos que o compõem, isto inclui os *Datasets*, *Research Workflow* e os *Researchers/Research Groups* que trabalharam em sua produção. Deste ponto, o pesquisador pode realizar a reexecução da pesquisa. Para isso, são disponibilizadas duas opções. A primeira delas, faz a execução contínua do *Research Workflow*, também chamada de *modo batch*. Nesta opção, não é oferecido nenhuma interação com os ambientes e os códigos que estão sendo executados. O processo ocorre até sua finalização, em que os resultados são disponibilizados para *download*. Diferente disso, a segunda opção permite que o pesquisador navegue entre os *Execution Compendia* da pesquisa e realize a execução dos *scripts* interativamente. Para tal, a `Plataforma Storm` disponibiliza cada um dos *Computational Environment* associados aos *Execution Compendia*. Em cada um dos ambientes, estão, em modo restrito de modificações ou "somente leitura", todos os dados necessários para a execução dos códigos daquela etapa da pesquisa. Nesta segunda opção, o *download* dos resultados pode ser feito à medida que são gerados.

As duas formas de execução descritas podem ser utilizadas para a reprodução e replicação. No caso da reprodução, o pesquisador seleciona a forma de execução e a realiza do início ao fim, sem aplicar modificações em nenhuma das etapas. Na replicação, pode-se trocar os dados de entrada com a substituição de *Datasets* e os códigos de processamento.

Na replicação, cabe considerar que a existência do modo de execução contínua, não depende apenas da `Plataforma Storm`, mas também da maneira a qual os `Registros` dos *scripts* são realizados. Como citado no cenário anterior, ao realizar o `Registro`, o pesquisador tem a opção de definir os parâmetros de entrada e saída do *script* que será executado. Quando esses são especificados em todos os *Execution Compendia*, tem-se a possibilidade de execução contínua. Isto já que, aqueles que farão a reexecução, tem a possibilidade de trocar os parâmetros de entrada utilizados em cada etapa de uma única vez. Ao contrário disso, a replicação fica sujeita ao uso da execução interativa.

3.4 Arquitetura

Embora as plataformas analisadas na Tabela 3.1 não cumpram com todos os requisitos elencados para o contexto deste trabalho, quando considerado o fluxo de funcionamento apresentado na Seção anterior e as operações que o compõem, tem-se que as estratégias adotadas por essas plataformas são extremamente relevantes, uma vez que cobrem diferentes aspectos do desenvolvimento científico e auxiliam

os pesquisadores na produção de pesquisas reprodutíveis. Dessas estratégias, considerando como base as operações da `Plataforma Storm`, pode-se mencionar como relevantes:

Auxílio à virtualização dos ambientes de trabalho: As plataformas `Reana` e `ReproZip` oferecem funcionalidades que auxiliam os pesquisadores na criação de ambientes virtuais que os permitem executar e reexecutar pesquisas de forma reprodutível. De forma mais significativa, o `ReproZip` tem como base para seu funcionamento, a criação automática de ambientes virtuais, facilitando a utilização dessa técnica e evita a necessidade de conhecimentos específicos dos pesquisadores;

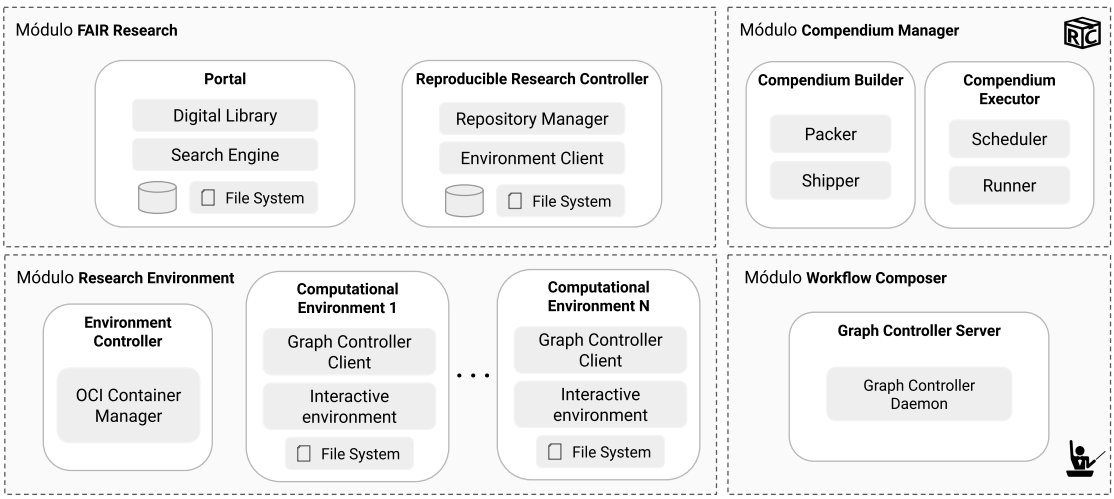
Serviço *web* para suporte à reprodutibilidade: Para seu funcionamento, o `o2r` disponibiliza um serviço *web* que permite a seus utilizadores produzirem pesquisas reprodutíveis. Isso é extremamente relevante já que essa abordagem evita que os pesquisadores tenham de utilizar múltiplas ferramentas e dependências em seu ambiente de trabalho para aplicar os recursos da plataforma. Nesse caso, todo o trabalho de persistência, preservação e versionamento dos objetos de uma pesquisa são feitos pelo serviço *web*. A mesma abordagem é vista na plataforma `Reana`, tendo como base os mesmos princípios mencionados para o `o2r`;

Disponibilização de ambientes virtuais de trabalho: Por vezes, os pesquisadores não terão em seus ambientes básicos de trabalho, os recursos necessários para a realização de uma determinada etapa de sua pesquisa. Por exemplo, ao ter a necessidade de processar grandes volumes de dados. Para a solução desse problema, plataformas como `Gigantum`, `Whole Tale` e `Reana` oferecem ambientes virtuais onde o pesquisador pode realizar o desenvolvimento e execução de suas pesquisas. Esses ambientes virtuais, no caso dessas plataformas, são executados em infraestruturas de computação especializada. No caso do `Reana`, por exemplo, como descrito por Šimko et al. (2019), as execuções dos ambientes dos usuários são feitas em um *cluster* de alto desempenho. Com isso, tem-se que essa disponibilização de ambientes, facilita o trabalho e o acesso a diferentes tipos de recursos computacionais. Além disso, uma vez que o ambiente é padronizado, auxilia na reprodutibilidade, já que suas bibliotecas são padronizadas e as versões são controladas.

Partindo dessas três abordagens e as demais funcionalidades especificadas, fez-se a definição da arquitetura conceitual da `Plataforma Storm`. Para isso, na arquitetura definida, a qual é ilustrada na Figura 3.7, foram incorporados módulos que operam seguindo tais abordagens. Junto a esses, fez-se também a adição de módulos que tornam possível a produção colaborativa, reprodução e replicação de pesquisas.

Nos parágrafos a seguir, cada um desses módulos são apresentados e descritos em detalhes.

Figura 3.7 - Arquitetura conceitual da `Plataforma Storm`.



Fonte: Produção do autor.

No diagrama da arquitetura, observa-se a existência de quatro módulos. O módulo `FAIR Research` contém as funcionalidades de preservação, gerenciamento e disseminação de *Research Projects* da plataforma. O módulo `Research Environment` oferece funcionalidades para o gerenciamento de *Computational Environments*. No módulo `Workflow Composer` estão as funcionalidades para o gerenciamento dos *Research Workflows*. O módulo `Compendium Manager` disponibiliza funcionalidades para a exportação e reexecução de *Research Projects*.

Seguindo os fluxos de funcionamento apresentados nos cenários de uso, o componente principal de interação dos usuários com a `Plataforma Storm` é o `Portal`. Ele possui dois serviços, `Digital Library` e `Search Engine`. O serviço `Digital Library` é o responsável em prover funcionalidades de gerenciamento e preservação dos elementos de um *Research Project*. O serviço `Search Engine`, realiza buscas a *Research Projects* e *Datasets* através dos metadados informados pelos usuários.

Os *Research Projects* e *Datasets* são materializados na `Plataforma Storm` em repositórios de dados. Tais repositórios podem ser versionados. O componente

`Reproducible Research Controller`, através do serviço `Repository Manager` faz o gerenciamento desses repositórios. Outra responsabilidade do `Reproducible Research Controller` é a inicialização de *Computational Environments* com associações a *Datasets*. Para isso, está presente neste componente o serviço `Environment Client`, que faz a comunicação com o componente `Environment Controller` do módulo `Research Environment`, para a criação e inicialização de *Computational Environments*.

No módulo `Research Environment`, para a criação de ambientes de processamento para vários usuários, tem-se no componente `Environment Controller`, o serviço `OCI Container Manager`. Este serviço materializa os *Computational Environments* em *Containers* OCI. Nos *Containers* criados pelo serviço, o acesso aos *Datasets* que têm arquivos de dados é efetuado através da adição de volumes com restrição de modificação "somente leitura".

Em um *Computational Environment*, materializado pelo `OCI Container Manager`, estão dois serviços. O primeiro deles, `Interactive Environment`, disponibiliza interfaces interativas que podem ser utilizadas pelos usuários para interagir com o ambiente. O segundo serviço é o `Graph Controller Client`, ferramenta que realiza a operação de `Registro`. Quando utilizada para executar um *script*, a ferramenta faz o mapeamento de todos os elementos do ambiente, como bibliotecas do SO e da linguagem de programação utilizados na execução. Todos os elementos mapeados são empacotados em um arquivo. Para adição da execução realizada ao *Research Workflow*, os elementos empacotados são enviados para o componente `Graph Controller Server` do módulo `Workflow Composer`.

No módulo `Workflow Composer`, o componente `Graph Controller Server` é o responsável pelas funcionalidades de composição e controle de *Research Workflows*. Para tal, o serviço `Graph Controller Daemon` é utilizado. Uma vez que as informações são recebidas por este serviço, ele realiza a geração de um *Execution Compendium*, considerando as informações do *Research Project*, *Dataset*, *Researcher* e do *Computational Environment* associados à execução. Em seguida, o *Research Workflow* é atualizado e as modificações realizadas nos *scripts* e outros objetos de pesquisa são salvas em uma nova versão dos arquivos no repositório de dados do *Research Project*.

Quando existe a necessidade de exportação de um *Research Project*, a requisição é enviada do `Portal` para o componente `Compendium Builder` do módulo `Compendium Manager`. Com o serviço `Packer`, este componente realiza o empacotamento de todos

os elementos do *Research Project* em um arquivo. A exportação de um *Research Project* para um repositório externo de disseminação e preservação é feita pelo serviço `Shipper`.

O módulo `Compendium Manager` também é utilizado para a reprodução ou replicação de uma pesquisa. Neste caso, usa-se o componente `Compendium Executor`, que possui os serviços `Scheduler` e `Runner`. O `Scheduler` é o responsável em receber do `Portal` a requisição de reexecução de uma pesquisa e determinar, com base em seu tipo, qual componente de execução deve ser utilizado. Quando a reexecução é do tipo interativa, o serviço requisita que o componente `Reproducible Research Controller` faça a execução dos *Computational Environments*, associados aos *Execution Compendia* do *Research Workflow* da pesquisa. Assim, é possível disponibilizar ao usuário, todas as etapas do *Research Workflow* interativamente. Na reexecução de tipo contínuo, a requisição é encaminhada para o serviço `Runner`. Quando este recebe a requisição de execução, utiliza o *Research Workflow* da pesquisa, para iniciar a reexecução. No caso de replicação, quando disponível, os parâmetros de entrada são recebidos através de um arquivo de configuração. Os *Datasets* inseridos com novos dados são montados nos *Computational Environments* necessários.

4 IMPLEMENTAÇÃO DEMONSTRATIVA DA PLATAFORMA PROPOSTA

Para apresentar a capacidade de uso da arquitetura proposta e seus conceitos em cenários reais de pesquisas em EO, fez-se a criação de uma implementação demonstrativa da mesma, promovendo assim, uma demonstração de base empírica de seu funcionamento. Com o uso dessa implementação, conforme apresentado em detalhes no Capítulo 5, foi possível realizar a criação de experimentos com cenários e problemas reais de pesquisa.

Sobre a implementação realizada, deve-se mencionar que, com o objetivo de manter o escopo da implementação alinhado com o restante do trabalho, a implementação realizada considerou apenas as operações relacionadas a produção reprodutível de resultados e a reprodução dos mesmos. Essa redução de escopo na implementação foi feita já que, conforme apresentado no Capítulo 3, as funcionalidades da `Plataforma Storm` são especificadas em função de operações que cobrem o contexto completo da reprodutibilidade, isto é, a produção reprodutível, reprodução e replicação de resultados. Assim, fez-se necessário a realização dessa redução de escopo para a implementação demonstrativa.

No restante deste Capítulo será feita a apresentação dos detalhes da implementação demonstrativa da plataforma. Para isso, na Seção 4.1 faz-se a apresentação dos princípios de *design* utilizados para a implementação. Em seguida, na Seção 4.2 tem-se a descrição da arquitetura adotada na implementação. Então, na Seção 4.3 são apresentados os componentes *Server Side* da arquitetura. De forma complementar, na Seção 4.4, faz-se a apresentação dos componentes *Client side* da arquitetura. Por fim, na Seção 4.5, são apresentados os detalhes do modelo adotado na disponibilização dos códigos da implementação.

4.1 Princípios de design

O dicionário de *Oxford* define "princípio" como sendo "uma verdade ou proposta fundamental que serve de base para um comportamento ou cadeia de raciocínios". Complementarmente, Brignell (2017) apresenta os princípios de *design* como sendo um conjunto de considerações que formam a base de um bom produto, sendo essas considerações utilizadas durante os estágios de desenvolvimento do mesmo, servindo assim como guia para as decisões tomadas.

O projeto da plataforma descrito no Capítulo 3, embora apresente em detalhes

as características da `Plataforma Storm`, suas funcionalidades e como ela pode ser utilizada pelos pesquisadores, sendo uma definição conceitual, não apresenta detalhes de como cada um dos recursos devem ser implementados para atingir as descrições realizadas, nem mesmo do conjunto de tecnologias que devem ser utilizados. Para esse último tópico, tem-se apenas a exceção sobre o uso de tecnologias *Open Source*, apresentado nos requisitos utilizados para a definição do projeto.

Sendo assim, para que a implementação demonstrativa fosse realizada considerando tecnologias e práticas que pudessem ser benéficas aos pesquisadores, fez-se a definição de um conjunto de princípios de *design*. Tais princípios foram utilizados durante todo o processo de desenvolvimento, auxiliando o autor na criação de uma implementação que considerasse as práticas reais de pesquisa, evitando definições arbitrárias que pudessem dificultar o uso da plataforma e suas funcionalidades. Os princípios definidos são apresentados abaixo:

Pesquisa em código

O desenvolvimento de pesquisas com o uso de código em linguagens de programação como R, Python e Julia fornece a possibilidade dos pesquisadores automatizarem todas as etapas de seu trabalho. Essa automatização evita erros involuntários e ações irreprodutíveis.

Sendo assim, a pesquisa criada em código deve ser apoiada e sugerida aos pesquisadores como uma alternativa viável ao desenvolvimento de seus trabalhos, evitando ao máximo a realização de ações manuais ou com ferramentas gráficas que não permitam a reprodução automatizada das pesquisas.

Colaboração

O desenvolvimento colaborativo e multidisciplinar das pesquisas faz com que resultados inovadores sejam gerados. Dito isso, tem-se que essa é uma prática comum no presente meio científico, podendo envolver vários pesquisadores de múltiplas instituições, sendo elas nacionais ou internacionais (WIESER et al., 2021).

De maneira a evitar esforços extras, é preciso apoiar o desenvolvimento colaborativo, em um modelo que viabilize resultados reprodutíveis, mesmo quando há diferentes ambientes e práticas sendo utilizadas.

Reutilização

Na comunidade científica, por conta de seus impactos positivos para os trabalhos gerados, tem-se mantido em constante discussão os tópicos relacionados à Pesquisa Aberta e Reprodutível. Nessas discussões, há muito conhecimento sendo gerado. Por exemplo, assim como pode ser visto no Capítulo 2, têm-se pesquisadores apresentando novas ferramentas para suportar a reprodutibilidade em ambientes e tecnologias específicas, sugestões de boas práticas e discussões sobre como a cultura pode ser alterada para que todos aprendam e entendam a importância da adoção de práticas abertas e reprodutíveis.

Esse conhecimento prévio da comunidade deve ser aproveitado, principalmente para fornecer facilidade de uso e adaptação dos usuários a plataforma desenvolvida. Com isso, tem-se que o funcionamento deve ser orientado a possibilidade de reutilização de ferramentas e práticas já existentes.

Liberdade de escolha

A quantidade de ambientes de trabalho para o desenvolvimento de pesquisas é no mínimo igual à quantidade de pesquisadores ao redor do mundo (NüST et al., 2017). Cada pesquisador possui seu ambiente de trabalho único, com configurações e ferramentas que os tornam mais produtivos.

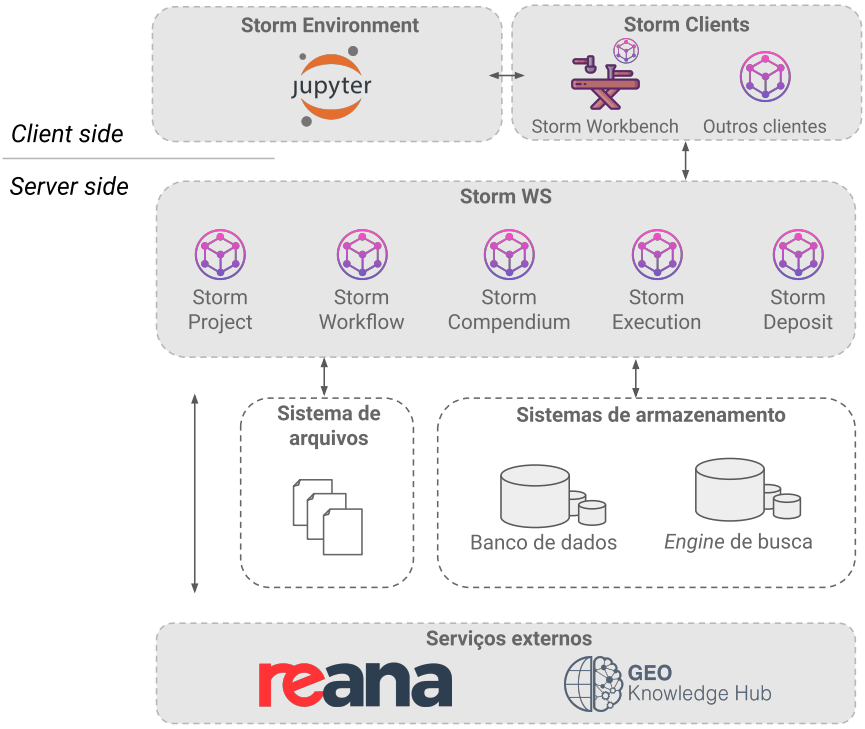
Assim, os pesquisadores devem ter a liberdade de escolher como as ferramentas da plataforma serão integradas a seu fluxo de trabalho. É importante aos recursos da plataforma evitar que o usuário tenha de se adequar a uma ideia de trabalho ou fluxo de pesquisa para que a mesma seja utilizada.

4.2 Arquitetura da plataforma

Conforme apresentado na Figura 4.1, a arquitetura definida para a implementação demonstrativa é composta por duas partes principais: *Client Side* e *Server Side*. No *Client Side*, têm-se os componentes da plataforma utilizados pelos pesquisadores para a produção de resultados reprodutíveis, tendo como base capacidades e recursos fornecidos pelos serviços da plataforma. No *Server Side*, por outro lado, têm-se os componentes da plataforma que suportam o desenvolvimento colaborativo das pesquisas. As funcionalidades dos componentes desse grupo, são acessadas pelos pesquisadores através de APIs em formato de serviços *web*. Isso inclui, por exemplo, serviços de gerenciamento de *Research Projects*, *Research Workflow*, bem como o gerenciamento de dados e tarefas de reprodução.

Nas subseções que seguem serão apresentados em detalhes as partes que compõem ambos os grupos de componentes mencionados.

Figura 4.1 - Arquitetura da implementação demonstrativa da `Plataforma Storm`.



Fonte: Produção do autor.

4.3 Componentes *Server Side*

Como pode-se observar na Figura 4.1, para o *Server Side* definiram-se quatro componentes. Cada um desses componentes possui um conjunto de responsabilidades que quando juntas, conseguem fornecer diversas funcionalidades, incluindo:

- Criação e gerenciamento de *Research Projects*, *Research Workflows*, *Execution Compendia*;
- Reprodução de *Research Workflows*;
- Gerenciamento e preservação de dados;

- Depósito de pesquisas finalizadas e seus materiais em plataformas de gerenciamento e preservação de dados científicos.

Nas Seções abaixo são apresentados, em detalhes, os componentes definidos para a composição dessas funcionalidades no *Server Side*.

4.3.1 Storm WS

O primeiro e mais importante componente do *Server Side* é o `Storm WS`[1]. Esse é o componente que implementa o serviço de suporte a reprodutibilidade na plataforma. Suas funcionalidades são providas aos usuários através de APIs em formato de serviço *web*. Sendo um único serviço, tem-se que todas as funcionalidades do *Server Side* são gerenciadas por esse componente, fazendo com que ele represente diretamente a implementação dos componentes `FAIR Research`, `Compendium Manager` e `Workflow Composer` especificados no projeto da `Plataforma Storm`.

Por possuir uma grande quantidade de responsabilidades e funcionalidades, para a criação de um projeto sustentável e funcional, esse componente é formado por um conjunto de módulos. Cada módulo possui uma responsabilidade única e bem definida. Para a integração desses módulos, a comunicação entre eles em nível de código é feita através da API que cada um deles define. A escolha por esse formato de implementação foi feita com base nas recomendações apresentadas por Bloch (2006), que de forma pragmática indica que "bons programas são modulares, tendo esses módulos limites bem definidos".

Além dessa definição, para o *design* e organização do código, foram seguidas como base as recomendações e conceitos apresentados pelo Grupo de Desenvolvedores Invenio RDM (2022). Dessa forma, cada um dos módulos definidos tem suas funcionalidades organizadas em três partes distintas:

Presentation Layer: Nessa camada, os módulos disponibilizam os recursos que formam a API em forma de serviço *web* do `Storm WS`, isto é, as funcionalidades para o gerenciamento de chamadas de operações feitas através de requisições HTTP (do inglês, *Hypertext Transfer Protocol*);

Service Layer: Camada em que se tem a aplicação das regras de negócio de cada um dos módulos. Essa camada, recebe requisições de operações da *Presentation Layer*, as valida, executa as ações requisitadas e com base na necessidade, consulta

[1]Disponível em: <`https://github.com/storm-platform/storm-ws`>. Acesso em: 03/12/2022

a camada *Data Layer*;

Data Layer: Essa é a camada em que se faz a manipulação dos dados. Com base nas funcionalidades fornecidas nesta camada, os usuários conseguem aplicar todas as operações relacionadas a *Criação*, *Atualização*, *Deleção* e *Leitura* de dados.

Para a implementação do `Storm WS` e seus módulos, fez-se o uso do `Invenio Framework`, que promove uma base de código genérica que pode ser estendida e utilizada em diferentes cenários. Assim, mesmo com suas capacidades especializadas na criação de repositórios digitais, neste trabalho essa tecnologia foi utilizada para a implementação de um serviço de suporte à reprodutibilidade. Ao todo, no serviço provido pelo `Storm WS` foram implementadas 62 operações, divididas em 21 categorias de ação diferentes. Para a documentação dessas operações, fez-se uso do padrão de especificação OPENAPI v3.0 (2022). Essa documentação, incluindo exemplos de utilização, está disponível *online*[2] abertamente e livre de restrições de uso.

Nos tópicos abaixo, são apresentados em detalhes cada um dos módulos que compõem o `Storm WS`. No total, tem-se definido seis módulos, sendo eles:

Storm Commons

O `Storm WS`, conforme mencionado anteriormente, possui diversos módulos. Por vezes, esses módulos possuem necessidades comuns. Para solucionar esse problema, definiu-se o `Storm Commons`[3]. Esse módulo disponibiliza funcionalidades que permitem a construção de novas funcionalidades quando integradas com outros módulos. Dentre suas funcionalidades estão:

Validação de metadados: Operações de validação de metadados. Por exemplo, validação do formato e tipos de dados. Quando utilizado em diferentes módulos, uniformiza as regras aplicadas nas validações;

Sistema de extensão via *plugin*: Conjunto base de operações que permitem a criação de pontos de extensão nos recursos do `Storm WS`;

Sistema base de controle de usuários: Conjunto base de operações que permitem a adição de controle de usuários em recursos da plataforma. Essa funcionalidade é utilizada, por exemplo, para a criação de um controle de acesso a *Research Projects*

[2]Disponível em: <https://spec.storm-platform.org/>. Acesso em: 28/09/2022

[3]Disponível em: <https://github.com/storm-platform/storm-commons>. Acesso em: 03/12/2022

no Storm WS;

Sistema de autenticação: Sistema de autenticação para o Storm WS, implementado com base no serviço de autenticação disponível na Plataforma BDC. Esse sistema é utilizado em todos os módulos.

Storm Project

O Storm Project[4] é o módulo responsável em disponibilizar todos os recursos que permitem a criação e o gerenciamento de *Research Project* no Storm WS. Para isso, esse módulo fornece uma implementação que materializa o conceito da entidade gerenciada através de duas dimensões diferentes: (i) *Metadados*; (ii) *Contexto*.

Na dimensão de *Metadados*, tem-se que o Storm Project materializa o *Research Project* utilizando um conjunto de metadados que permite a especificação de informações como título, descrição, criadores e contribuidores, bem como informações sobre a extensão espacial e temporal dos experimentos realizados no projeto. De forma geral, os campos especificados no modelo de metadados que define um *Research Project* são apresentados na Tabela 4.1. Esses campos foram definidos com o objetivo de facilitar a caracterização do projeto que está sendo construído na plataforma. Para a criação de um modelo que se aproximasse das práticas utilizadas em cenários reais, teve-se como base para a definição dos campos, o modelo de metadados DataCite V4.0 (DATACITE METADATA WORKING GROUP, 2021). Esses metadados são materializados e armazenados no banco de dados associado ao Storm WS no formato de documento JSON[5].

Diferente da dimensão de *Metadados*, utilizada para especificar as características de um *Research Project*, na dimensão de *Contexto*, trabalham-se os objetos associados ao projeto definido. O contexto de um *Research Project* é materializado através de uma área de trabalho. Nessa área, podem ser definidas regras específicas de acesso e visibilidade. Isso possibilita aos pesquisadores definirem quem pode acessar o projeto e quais as informações no projeto podem ser visualizadas e utilizadas. Além disso, nesse espaço de trabalho, podem ser associadas outras entidades definidas no sistema, como um *Researcher*, *Dataset*, *Research Workflow* e também os *Execution Compendia*. Uma vez associado a esse espaço de trabalho, essas entidades seguem as regras de compartilhamento e visibilidade definidas no *Research Project*.

[4]Disponível em: <https://github.com/storm-platform/storm-project>. Acesso em: 03/12/2022

[5]Um exemplo completo de metadados de um *Research Project* está disponível em: https://tinyurl.com/storm-metadata-project

Tabela 4.1 - Tabela de descrição dos campos de metadados utilizados na definição de um *Research Project*.

ID	Propriedade	Descrição
1	*Title*	Título do projeto
2	*Description*	Descrição do projeto
3	*Version*	Nome da versão do projeto
4	*Subjects*	Tópicos com os quais o projeto está relacionado
5	*Creators*	Identificação dos criadores do projeto
6	*Contributors*	Identificação dos contribuidores do projeto
7	*Locations*	Localização geoespacial da área de estudo do projeto
8	*Rights*	Licenças aplicadas ao projeto
9	*Dates*	Datas associadas ao projeto

Fonte: Produção do autor.

Para o gerenciamento de um *Research Project* são disponibilizadas pelo `Storm Project` operações para a *Criação*, *Atualização*, *Busca* e *Ativação/Desativação* de um projeto. Nas operações de *Criação* e *Atualização*, os usuários definem os projetos com base no modelo de metadados da Tabela 4.1, bem como a especificação da visibilidade do projeto. Já nas operações de *Ativação/Desativação*, o usuário pode indicar se um projeto está ativo ou não. A operação de *Busca* permite que projetos sejam descobertos com base nas informações contidas em seus metadados. Também para o compartilhamento e controle de acesso de um projeto faz-se o uso das operações fornecidas pelo `Storm Commons`.

De forma a se manter consistente com o contexto de reprodutibilidade deste trabalho, nota-se que não há nenhuma opção para a exclusão de um *Research Project*. Isso é feito já que, durante a implementação, foi assumido que, uma vez que o projeto é definido, ele contém informações valiosas sobre a pesquisa associada a ele. Sua exclusão pode representar falhas para reprodutibilidade de resultados e dados obtidos. Então, como alternativa à exclusão, fez-se o uso da estratégia de *Ativação/-Desativação* do projeto. Quando ativo, o projeto pode ser modificado, o que inclui mudanças em seus metadados e nos objetos associados a sua área de trabalho. Caso contrário, nenhuma ação além da leitura e acesso aos dados pode ser realizada no projeto e em seus objetos.

Storm Compendium

O `Storm Compendium`[6] é o módulo responsável em prover as funcionalidades e abstrações para a criação e o gerenciamento de *Execution Compendium* no `Storm WS`. Para isso, é fornecido pelo módulo uma implementação que materializa a entidade gerenciada através de um pacote com uma divisão lógica de três partes, sendo elas:

Metadados: Conjunto de metadados utilizados para descrever a execução e o contexto onde a mesma foi realizada. De forma geral, nesse conjunto de metadados, têm-se informações como título, descrição, bem como informações sobre quais foram os dados de entrada, os resultados gerados e também os *scripts* de processamento utilizados;

Arquivos: Arquivos utilizados durante a execução associada ao *Execution Compendium*. Isso inclui os arquivos de dados de entrada, saída, bem como os *scripts* de processamento e arquivos de configuração;

Ambiente computacional: Metadados e arquivos que permitem a descrição completa e reprodutível do ambiente computacional utilizado na execução associada ao *Execution Compendium*.

Para a representação dos *Metadados* da execução, juntamente com os metadados do *Ambiente computacional*, faz-se o uso de um modelo de metadados que contêm os campos descritos na Tabela 4.2. Com seu uso, tem-se a possibilidade de armazenar todas as informações descritas nos tópicos acima. Esses metadados são armazenados no banco de dados associado ao `Storm WS` no formato de documento `JSON`[7].

Tabela 4.2 - Tabela de descrição dos campos de metadados utilizados na definição de um *Execution Compendium*.

ID	Propriedade	Descrição
1	*Title*	Título do *Execution Compendium*
2	*Description*	Descrição do *Execution Compendium*
3	*Execution*	Informações sobre a execução associada ao *Execution Compendium*

Fonte: Produção do autor.

[6]Disponível em: <https://github.com/storm-platform/storm-compendium>. Acesso em: 03/12/2022

[7]Um exemplo completo de metadados de um *Execution Compendium* está disponível em: https://tinyurl.com/storm-metadata-compendium

Para os *Arquivos* utilizados na execução, juntamente com os arquivos que suplementam a descrição e construção do *Ambiente computacional* associado ao *Execution Compendium*, faz-se a criação de um espaço de armazenamento específico no serviço, o qual é nomeado de *bucket*. Tais *buckets* podem ser organizados de diferentes formas, podendo utilizar diferentes tecnologias para o armazenamento dos arquivos.

Além do modelo de metadados e a forma de armazenamento dos arquivos associados a um *Execution Compendium*, tem-se como parte importante de sua definição o formato e estruturas dos arquivos. No caso dos arquivos associados diretamente à execução, como dados de entrada, saída e *scripts*, tem-se que esses podem variar conforme a atividade que está sendo desempenhada pelos utilizadores do serviço. Então, pode-se, por exemplo, fazer parte desses arquivos, *scripts* de diferentes linguagens de programação, além de arquivos de dados em diversos formatos. No entanto, no que tange à representação do ambiente computacional, tem-se a necessidade do uso de arquivos com formato padronizado, para que o serviço consiga interpretar e utilizar tais arquivos para as execuções associadas às etapas de reprodução e replicação.

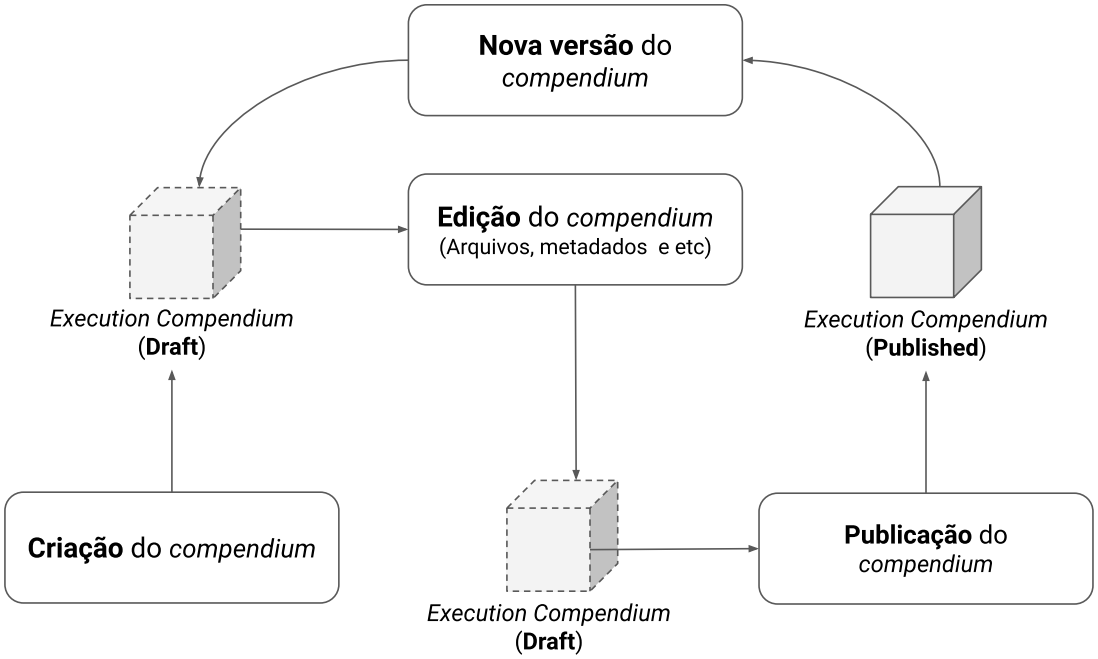
Uma vez que os arquivos que descrevem e representam os ambientes computacionais associados aos *Execution Compendium* são gerados no lado do cliente, tem-se que as decisões tomadas no serviço impactam diretamente a maneira como as ferramentas clientes podem ser utilizadas pelos pesquisadores em seus ambientes de trabalho. Com isso, de modo a evitar que haja limitações nos casos de uso, foi implementada no `Storm Compendium` a possibilidade de uso de diferentes formatos de arquivos que representam um ambiente computacional. Dessa forma, caso o cliente tenha a necessidade de implementar essa representação com o uso de uma estratégia específica, isso pode ser feito. Por exemplo, caso seja necessário representar o ambiente computacional com o uso de uma `Docker` *Image*, `Dockerfile` ou mesmo com o uso de um gerenciador de *Containers* específico, tem-se que tais ações são possíveis.

Para suportar essas diferentes representações do ambiente computacional, o `Storm Compendium` possui um sistema de pontos de extensão via *plugins*, implementado com base nas capacidades do `Storm Commons`, que possibilitam que diferentes interpretadores de ambientes sejam criados. Por padrão, o `Storm Compendium` fornece um interpretador de ambiente, criado com base no `ReproZip`. Esse faz a representação do ambiente computacional do *Execution Compendium* como um único arquivo executável `rpz`. Com a definição desses interpretadores, o cliente, no momento do uso do serviço, define qual dos interpretadores disponíveis deve ser utilizado. Essas informações fazem parte do campo de metadados da Tabela 4.2, referentes a

execução do *Execution Compendium*.

Além da representação do *Execution Compendium*, no `Storm Compendium` tem-se implementado operações que permitem sua criação e o gerenciamento. Essas operações são implementadas seguindo o ciclo de vida apresentado na Figura 4.2.

Figura 4.2 - Ciclo de vida de um *Execution Compendium* definido no `Storm Compendium`.



Fonte: Produção do autor.

Do ciclo de vida apresentado, tem-se que a primeira etapa é a *Criação do compendium*. Nessa etapa, com o uso de metadados, faz-se a criação de um *compendium* em estado *draft*. Nesse estado, o *compendium* pode ser editado e todos os arquivos associados a ele podem ser alterados, formando assim a etapa de *Edição do compendium*. Uma vez que a edição é finalizada, pode-se realizar a *Publicação do compendium*, formando assim um *compendium* em estado *published*. Estando *published*, o *compendium* não pode ser alterado, garantindo assim a preservação e consistência dos elementos nele contidos. Para alterar as informações em um *compendium published* é necessário produzir uma *Nova versão*. Ao fazer isso, um novo *compendium* em estado *draft* é criado e todo o ciclo de vida descrito anteriormente pode ser aplicado a ele. Essas modificações, no entanto, afetam apenas a nova versão, não modificando

versões anteriores. A operação de *Deleção* está disponível somente para *compendia* em estado *draft*. Uma vez em estado *published*, não se tem essa opção.

Além dessas operações, tem-se também suporte a *Busca* de *Execution Compendium*, permitindo assim sua descoberta. Essa operação independe do estado do *compendium*.

Storm Workflow

O `Storm Workflow`[8] é o módulo criado para prover as funcionalidades e abstrações necessárias para a criação e gerenciamento de *Research Workflows* no `Storm WS`. Para o provisionamento de suas funcionalidades, o módulo faz a implementação da entidade gerenciada através de duas dimensões: (i) *Metadados*; e (ii) *Grafo*.

Na dimensão de *Metadados*, tem-se um conjunto de informações que podem ser utilizadas para a descrição do *workflow*, bem como a indicação de sua versão. Para isso, faz-se uso de um modelo de metadados que contêm os campos descritos na Tabela 4.3.

Tabela 4.3 - Tabela de descrição dos campos de metadados utilizados na definição de um *Research Workflow*.

ID	Propriedade	Descrição
1	*Title*	Título do *Research Workflow*
2	*Description*	Descrição do *Research Workflow*
3	*Version*	Nome da versão do *Research Workflow*

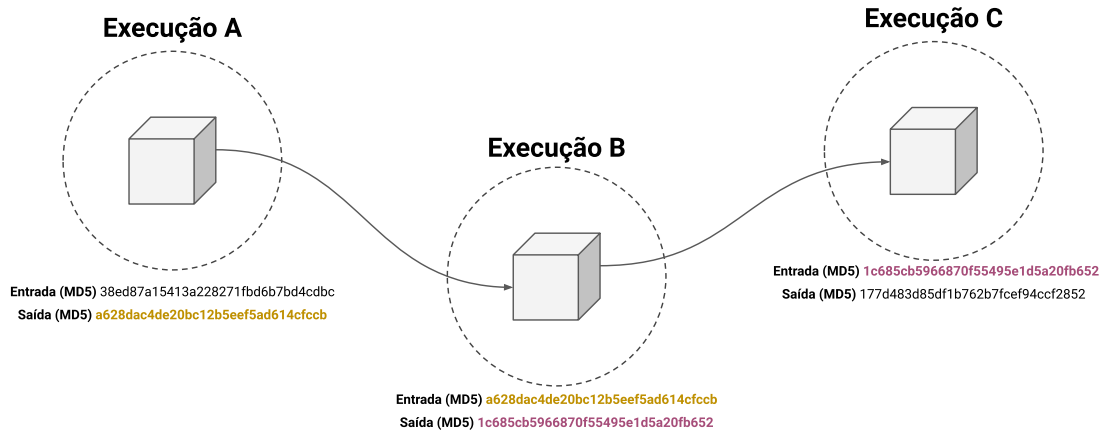
Fonte: Produção do autor

Na dimensão de *Grafo*, diferentemente dos *Metadados* utilizados para a descrição em alto nível do *Research Workflow*, tem-se a representação concreta do fluxo de processamento criado pelos pesquisadores. Para isso, nessa dimensão, define-se um DAG, que representa as etapas do *workflow*. Nesse DAG, cada um dos vértices de sua composição é um *Execution Compendium*. As arestas de ligação entre os vértices, por sua vez, representam o fluxo de dados entre as etapas de processamento da pesquisa. A determinação da ligação entre os vértices é feita com base nos metadados dos *Execution Compendia*, em que conforme apresentado nos tópicos anteriores,

[8]Disponível em: <https://github.com/storm-platform/storm-workflow>. Acesso em: 03/12/2022

têm-se informações sobre quais são os dados de entrada e saída. Para que essa relação seja gerada de forma consistente, seguindo as recomendações apresentadas por Addis (2020), usa-se como base de comparação o *checksum* MD5 desses arquivos. Assim, conforme ilustrado na Figura 4.15, tem-se que um nó de execução fará ligação com outro no DAG, se eles possuírem arquivos de entrada/saída com *checksums* equivalentes.

Figura 4.3 - Exemplo de ligação de um *Research Workflow* implementado no `Storm Workflow`.



Fonte: Produção do autor.

O armazenamento dos *Metadados* e do *Grafo* é feito no banco de dados associado ao `Storm WS` no formato `JSON`[9]. Em específico, para a representação e armazenamento do grafo, faz-se o uso da proposta de representação de grafos *JGF - JSON Graph Format*[10]

Para a manipulação de um *Research Workflow*, tem-se definido operações que permitem a *Criação/Edição* de *workflows*. Essas operações, dizem respeito aos metadados associados a entidade. Há também operações de *Adição* e *Remoção* de *Execution Compendium* no *workflow*. Essas operações especializadas, quando utilizadas, associam o *compendium* ao *workflow*, definindo, com base nos critérios apresentados anteriormente, o fluxo de operação. Essas operações podem ser utilizadas apenas com *compendia* no estado *published*. Isso impede que recursos que ainda estão sendo criados sejam utilizados, evitando assim a geração de fluxos inválidos. Operações

[9]Um exemplo completo de metadados de um *Research Workflow* está disponível em: `https://tinyurl.com/storm-metadata-workflow`

[10]Disponível em: <`http://jsongraphformat.info/`>. Acesso em: 15/11/2022

de *Busca* também estão disponíveis para a descoberta de *Research Workflows*. Caso necessário, tem-se também a operação de *Deleção* do *workflow*.

De modo a criar entidades consistentes, além das operações mencionadas, há também a possibilidade de *Ativação/Desativação* de um *Research Workflow*. Com isso, da mesma forma como feito para um *Research Project*, pode-se controlar quando um *workflow* pode ou não ser alterado. Isso já que, nessa estratégia, quando um *workflow* está ativo, ele pode ser editado. No entanto, quando desativado, ele passa para um modo "somente leitura", que não permite modificações, apenas uso de seus dados e metadados. Quando desativado, um *workflow* não pode ser deletado.

Storm Execution e Storm Deposit

Existem funcionalidades do `Storm WS` que requerem o uso de serviços especializados. Para essas, assim como será apresentado em detalhes na Seção 4.3.2, faz-se o uso de serviços externos à plataforma. Assim, mantém-se a consistência do escopo e responsabilidades implementadas no `Storm WS`. Com o uso de serviços externos, faz-se necessário que o `Storm WS` mantenha comunicação com tais serviços para a troca de informações. Por exemplo, uma das funções oferecidas pelo serviço de reprodutibilidade é a reprodução de um *Research Workflow*. Também, tem-se a necessidade de preservação dos resultados gerados na plataforma. Em ambos casos, serviços externos precisam ser consultados, e a troca de metadados e dados realizada. Para suprir essa necessidade de comunicação e uso de serviços externos nessas funcionalidades, fez-se a definição de dois módulos:

O primeiro módulo definido foi o `Storm Deposit`[11], que provê as funcionalidades que permitem que todos os materiais produzidos em um *Research Project* sejam enviados para um sistema de preservação digital. Na versão do `Storm WS` produzida para esse trabalho, esse componente suporta, por padrão, o envio dos materiais para plataformas criadas com base no `InvenioRDM`[12] e suas derivações. Assim, projetos como `ZenodoRDM`[13] e `Caltech Data`[14] podem ser utilizados para a preservação e disseminação dos resultados das pesquisas. Além desses, também foi implementado suporte ao `GEO Knowledge Hub`. Para esse, os dados são organizados seguindo o formato de empacotamento requerido pela plataforma antes do envio.

[11] Disponível em: <https://github.com/storm-platform/storm-deposit>. Acesso em: 03/12/2022
[12] Disponível em: <https://inveniosoftware.org/products/rdm/>. Acesso em: 16/11/2022
[13] Disponível em: <https://github.com/zenodo/zenodo-rdm>. Acesso em: 16/11/2022
[14] Disponível em: <https://data.caltech.edu/>. Acesso em: 16/11/2022

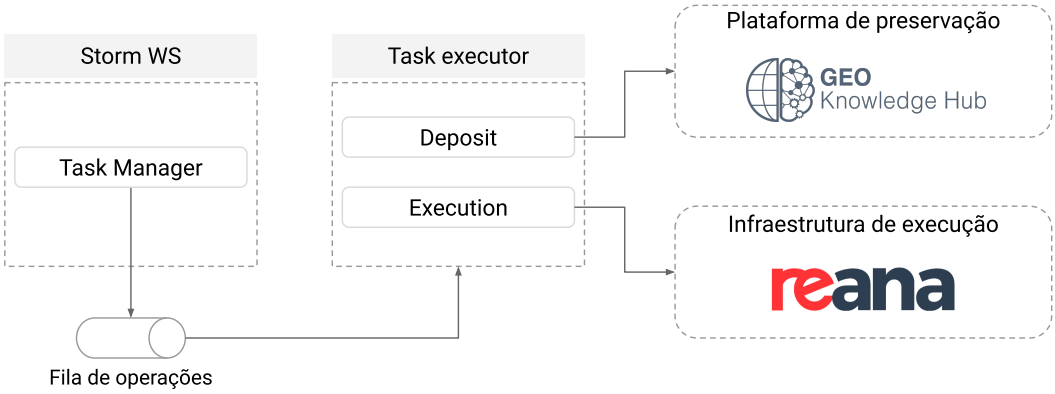
O segundo módulo criado é o `Storm Execution`[15]. Este módulo, disponibiliza funcionalidades que tornam possível a execução de um *Research Workflow*. Para que isso seja possível, cada *Execution Compendium* é preparado e enviado para o sistema externo de execução. Neste trabalho foi implementado o suporte ao uso do `Reana` como plataforma de execução. Assim, usa-se de suas capacidades de processamento distribuído e *cluster* de recursos sem que isso tenha de ser implementado diretamente no `Storm WS`.

Ambos os módulos foram implementados com suporte a extensões de suas funcionalidades. Assim, caso seja necessário a adição de outros serviços de preservação ou execução, isso pode ser facilmente realizado. Essa capacidade é criada com base nos recursos fornecidos pelo `Storm Commons`.

Além dessas capacidades, tem-se também que, o uso de serviços externos requer um tratamento especial durante a execução das operações. Isso é necessário já que, quando é feito o uso de um serviço externo ao sistema, há variáveis que podem influenciar a operação, fazendo com que ela não seja executada corretamente. Por exemplo, falhas na comunicação podem ocorrer, ou mesmo, conexões lentas podem surgir. Ademais, no contexto de EO, quando se faz o uso de grandes volumes de dados em etapas de processamento, pode ser que a operação não seja algo rápido ou trivial de ser feita. Para evitar que esses e outros problemas influenciassem as operações, a implementação de ambos os módulos, `Storm Deposit` e `Storm Execution`, foi feita com base em um sistema de filas de operações. Nesse sistema, conforme apresentado na Figura 4.4, com base em uma fila de operações assíncronas, tem-se que cada operação é executada em diferentes instantes de tempo por um componente especializado em execuções. Quando há falhas na operação, a requisição de execução volta à fila para ser executada novamente. Também, quando a operação é demorada, não se têm bloqueios no sistema para a execução de outras operações.

[15]Disponível em: <https://github.com/storm-platform/storm-execution>. Acesso em: 03/12/2022

Figura 4.4 - Modelo de operação assíncrona adotada no `Storm WS`.



Fonte: Produção do autor.

Com isso, garante-se que o sistema seja resiliente a falhas relacionadas ao meio externo. Também, evita que a indisponibilidade ou erros de comunicação impeçam que os pesquisadores realizem suas atividades.

4.3.1.1 Modelagem de dados

Para o armazenamento e gerenciamento dos dados tratados por cada um dos módulos do `Storm WS`, tem-se que, além de uma API de funcionalidades, cada um deles define um conjunto de entidades de dados, as quais são utilizadas para a criação do modelo de dados utilizado pela aplicação. Para isso, quando todos os módulos são integrados e a aplicação é devidamente configurada, é feita a definição de um modelo de dados completo e integrado, utilizado para armazenar os dados da aplicação. Essa forma de construção do modelo de dados é feita com base nas funcionalidades providas pelo `Invenio Framework`.

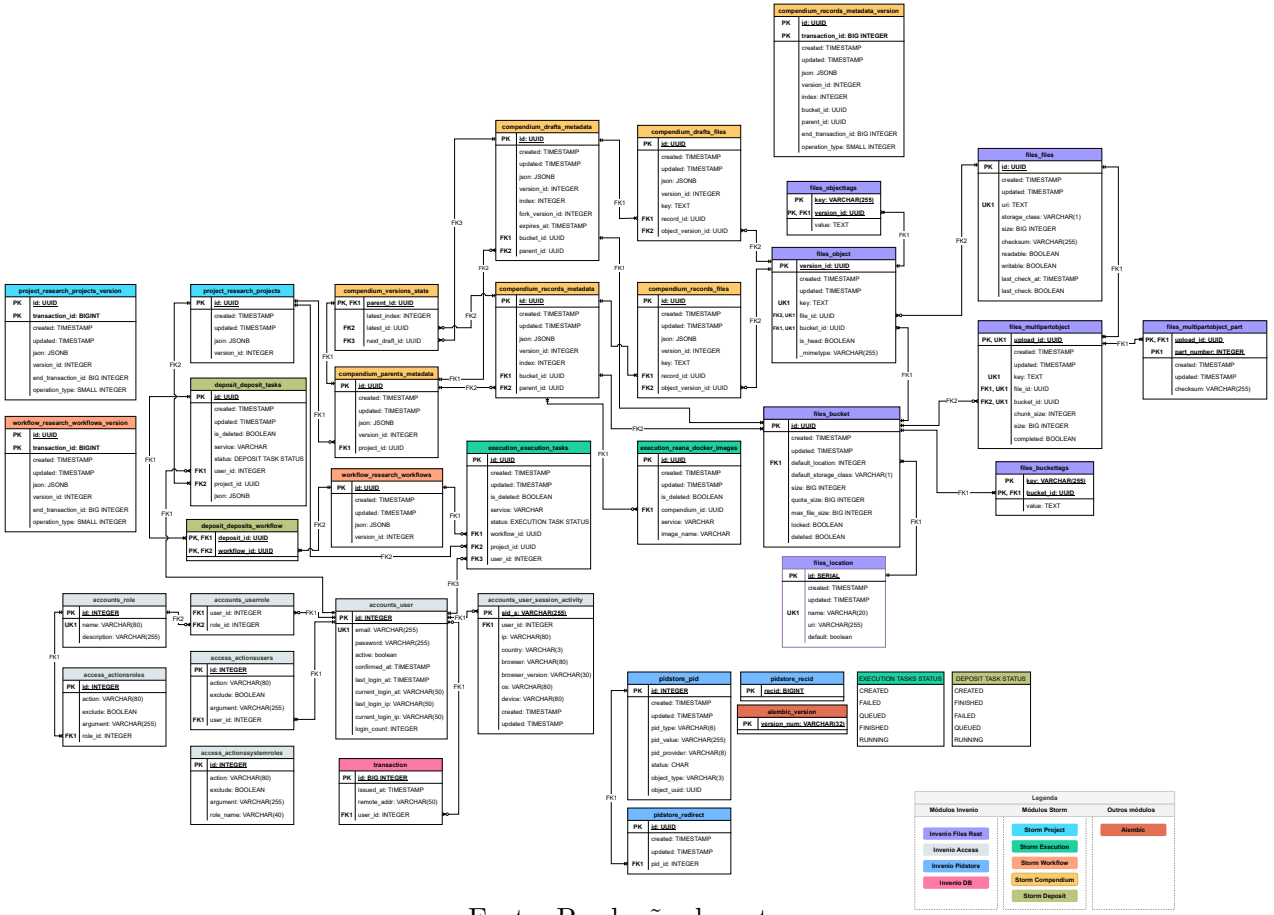
Sendo assim, para o `Storm WS`, quando os módulos estão integrados, tem-se a criação do modelo de dados apresentado na Figura 4.5[16]. Como pode-se notar no modelo, tem-se que cada uma das entidades definidas pelos módulos são representadas nas tabelas. Nesse modelo, para todas as entidades, têm-se informações sobre sua data de criação e atualização. Também, para as entidades que possuem a necessidade de armazenamento de metadados são definidas propriedades especializadas, permitindo que os metadados sejam armazenados como documentos `JSON` no banco de dados.

[16] A versão *online* e interativa do modelo de dados pode ser acessada através do seguinte endereço: <https://docs.storm-platform.org/database/storm-ws-database.html>

Com o uso da estratégia apresentada, quase todas as entidades do `Storm WS` são definidas por um pequeno conjunto de tabelas, simplificando sua manipulação e uso. No entanto, por possuir um ciclo de vida próprio, o qual é baseado na geração e controle de múltiplas versões, a representação do *Execution Compendium*, adota uma estratégia diferente. Nesse caso, como pode-se observar na Figura 4.5, tem-se a definição de tabelas auxiliares, que são operadas em conjunto para o suporte a múltiplas versões. Essa implementação foi realizada seguindo as definições e estratégias apresentadas pelo Grupo de Desenvolvedores Invenio RDM (2022).

Além das entidades definidas pelo `Storm WS`, por ser criado com o uso do `Invenio Framework`, existem tabelas extras definidas no modelo de dados. No `Invenio Access`, tem-se tabelas para o controle dos usuários do serviço. No `Invenio Pidstore`, os identificadores únicos globais utilizados nos *compendia* são armazenados e gerenciados. Por fim, no `Invenio Files Rest`, faz-se a definição de um conjunto de abstrações para o gerenciamento dos arquivos de dados no serviço.

Figura 4.5 - Modelo lógico do banco de dados do `Storm WS`.



Fonte: Produção do autor.

4.3.1.2 Infraestrutura de dados

Como pode-se observar no modelo lógico de dados apresentado na Figura 4.5, nem todos os relacionamentos inicialmente especificados no projeto da `Plataforma Storm` são definidos. Por exemplo, o relacionamento entre o *Research Project* e alguns dos elementos que o constituem não são definidos. Isso é feito já que, tais relacionamentos são definidos em nível de atributo, em especial, na propriedade `JSON` das tabelas das entidades. Quando o relacionamento é definido, faz-se sua cópia para as propriedades `JSON`. Com isso, tem-se que tais informações estão representadas em diversas instâncias de dados. Tal abordagem representa uma forma de desnormalização (ELMASRI; NAVATHE, 2011) feita com base nas funcionalidades providas pelo `Invenio Framework`.

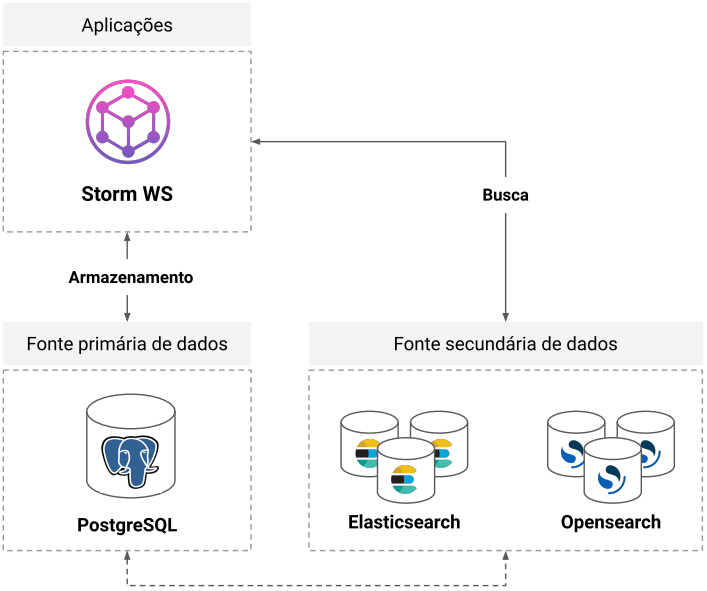
Para usufruir desse formato de organização dos dados e tornar as operações de buscas e recuperação de dados rápidas, assim como apresentado na Figura 4.6, tem-se definido para o `Storm WS` uma infraestrutura de dados composta por dois tipos de sistemas de armazenamento. O primeiro deles é o sistema de armazenamento primário, o qual é constituído por um banco de dados relacional em que os dados são armazenados no formato do modelo de dados apresentado anteriormente. Já o segundo é o sistema de armazenamento secundário, em que se faz o uso de ferramentas e serviços especializados em operações de busca.

Dentre as diferenças entre a fonte primária e secundária, podem-se destacar: (i) a fonte primária armazena o estado atual do dado. Todas as modificações, independente de quais são, são primeiramente salvas na fonte primária para depois serem enviadas as fontes secundárias; (ii) fontes secundárias fornecem métodos especializados para acesso dos dados, como capacidades de busca *Full-Text search* ou buscas espaciais; (iii) fontes secundárias são descartáveis, enquanto a fonte primária é sempre preservada.

Tendo como base essas definições, tem-se que no `Storm WS` a fonte primária é o banco de dados em que todos os dados são armazenados. Esse é implementado com o uso do `PostgreSQL`. Como fonte secundária, têm-se mecanismos de busca especializados como `Elasticsearch`[17] e `OpenSearch`[18], que fornecem funcionalidades como busca espacial e *Full-Text search*.

[17]Disponível em: <https://www.elastic.co/pt/elasticsearch/>. Acesso em: 03/12/2022
[18]Disponível em: <https://opensearch.org/>. Acesso em: 03/12/2022

Figura 4.6 - Infraestrutura de dados do `Storm WS`.



Fonte: Produção do autor.

4.3.2 Serviços externos

O `Storm WS`, conforme mencionado nas Seções anteriores, tem como escopo ser um serviço de reprodutibilidade que auxilia os pesquisadores na produção colaborativa de pesquisas reprodutíveis. No entanto, para que esse serviço alcance seu objetivo, foi definido em sua estrutura, funcionalidades que requerem recursos especializados. Por exemplo, para as etapas de reprodução de um *Research Workflow*, tem-se como necessidade básica, uma infraestrutura de execução, que, no mínimo, forneça capacidades de gerenciamento das operações que estão sendo executadas. Outras capacidades para a resiliência dessas operações, como controle de erros e reexecução, também podem ser necessárias. Outro exemplo está na preservação a longo prazo dos ativos da pesquisa. Essa funcionalidade requer capacidades específicas do sistema.

Para suprir essas necessidades sem que fosse necessário a expansão do escopo de operação do `Storm WS`, fez-se a definição do componente de `Serviços externos` na arquitetura da `Plataforma Storm`. Nesse componente, têm-se serviços que provisionam as capacidades e infraestrutura requeridas pelas operações especializadas do `Storm WS`. Inicialmente, foram consideradas para esse componente duas classes de

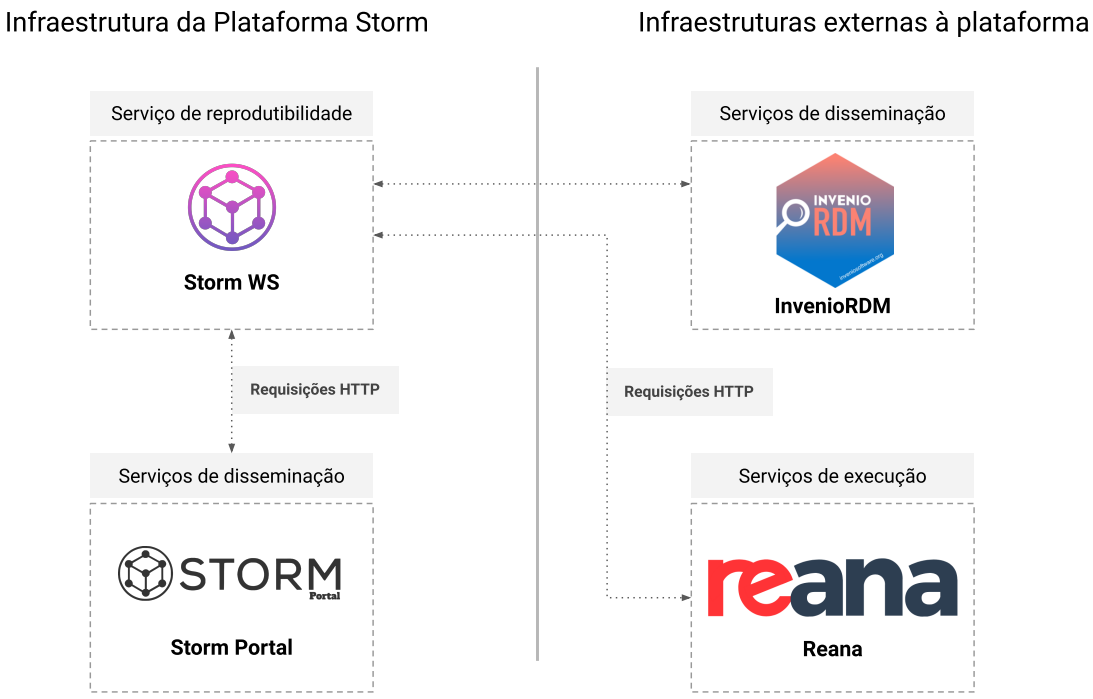
serviços:

Serviços de execução de processos: Serviços que fornecem recursos especializados para a execução e escalonamento de tarefas de processamento de dados;

Serviços de disseminação e preservação de dados: Serviços com capacidade de preservar os ativos da pesquisa a longo prazo, fornecendo também, métodos para a busca e disseminação desses ativos.

A integração desses serviços com o `Storm WS`, assim como ilustrado pela Figura 4.7, é realizada através do uso de suas APIs de serviços *web*. Com isso, tem-se que esses serviços, podem ou não fazer parte da infraestrutura da `Plataforma Storm`, uma vez que sua comunicação é feita via rede.

Figura 4.7 - Esquema de integração entre `Storm WS` e serviços externos.



Fonte: Produção do autor.

Nos tópicos que seguem, serão apresentados os detalhes das tecnologias utilizadas para compor cada uma das classes de serviços definidas anteriormente.

Serviços de execução de processos

Existem muitas tecnologias que podem ser empregadas para as tarefas relacionadas à execução de processos do Storm WS. Exemplos incluem Apache AirFlow[19], Dagster[20] e Argo[21]. No entanto, considerando o escopo desse trabalho e as ferramentas já exploradas durante as etapas iniciais de desenvolvimento, foi escolhida como serviço externo de execução a plataforma Reana. Ela fornece todos os recursos necessários para o gerenciamento de execuções, disponibilizando também aos usuários interfaces para acesso e exploração dos resultados gerados.

A adição do Reana como serviço de execução foi feita com base no uso dos pontos de extensão fornecidos pelo Storm WS e seus módulos. Nesse caso, fez-se a criação do *plugin* Storm Execution Reana[22] para o Storm Execution. Nesse *plugin*, tem-se implementado todas as operações de transformação e organização de dados necessárias para a integração entre os dois serviços. Com isso, o usuário pode requisitar que o serviço seja executado no Reana através do Storm WS. Após a requisição, o usuário pode acompanhar o andamento da execução, bem como fazer a exploração dos resultados gerados diretamente no Reana.

Serviços de disseminação e preservação de dados

Conforme apresentado na Seção 2.3.5, existem diversas opções de plataformas que podem ser utilizadas para a preservação e disseminação dos ativos de pesquisa. Em meio a essa variedade de plataformas disponíveis, no Storm WS, foi implementado suporte a sistemas de preservação que utilizam do InvenioRDM em sua construção. Isso é possível já que, utilizando a mesma tecnologia base, esses serviços normalmente fornecem as mesmas APIs.

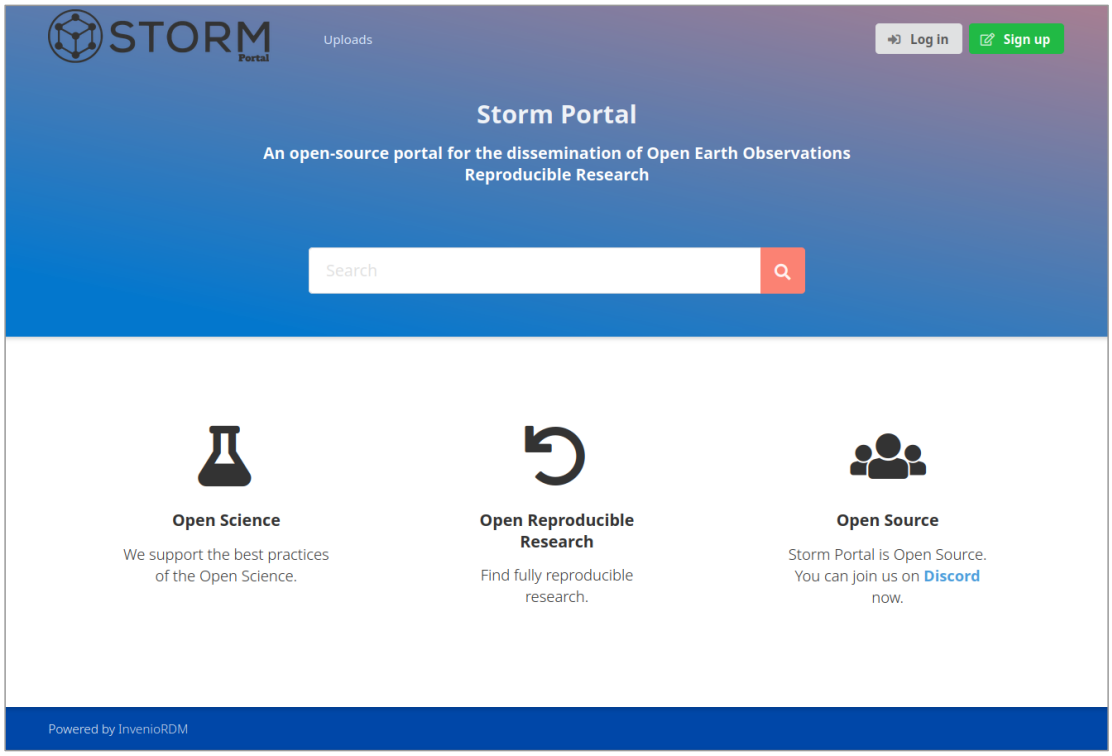
A implementação do suporte ao InvenioRDM foi efetuada através da criação do *plugin* Storm Deposit InvenioRDM[23], que estende as capacidades do Storm Deposit, permitindo que o usuário envie os ativos de pesquisas gerados no Storm WS, para o ambiente de preservação.

Além disso, tem-se que os serviços de preservação e disseminação, quando utili-

[19]Disponível em: <https://airflow.apache.org/>. Acesso em: 20/11/2022
[20]Disponível em: <https://dagster.io/>. Acesso em: 20/11/2022
[21]Disponível em: <https://argoproj.github.io/argo-workflows/>. Acesso em: 20/11/2022
[22]Disponível em: <https://github.com/storm-platform/storm-execution-reana>. Acesso em: 20/11/2022
[23]Disponível em: <https://github.com/storm-platform/storm-deposit-inveniordm/>. Acesso em: 20/11/2022

zados, criam objetos que não podem ser removidos. Isso dificulta etapas de teste. Sendo assim, neste trabalho, para evitar a dependência de serviços externos na apresentação das capacidades do `Storm WS`, foi definido o serviço `Storm Portal`. Esse serviço compõe a arquitetura da plataforma, como sendo um serviço externo de disseminação. Sendo criado com base no `InvenioRDM`, fornece todas as funcionalidades requeridas pelo `Storm WS` para a apresentação e execução de suas funcionalidades. A Figura 4.8 apresenta um exemplo da interface do `Storm Portal`[24] customizada no presente trabalho.

Figura 4.8 - Exemplo de interface do `Storm Portal`.



Fonte: Produção do autor.

4.4 Componentes *Client Side*

No *Service Side*, estão os recursos que auxiliam os pesquisadores na realização de pesquisa reprodutiva e colaborativa. Para que as funcionalidades disponibilizadas

[24]Disponível em: <https://github.com/storm-platform/storm-portal>. Acesso em: 03/12/2022

pudessem ser utilizadas de forma integrada ao ambiente de trabalho dos pesquisadores, fez-se a definição dos componentes *Client Side*. Nas Subseções abaixo, tem-se a apresentação desses componentes.

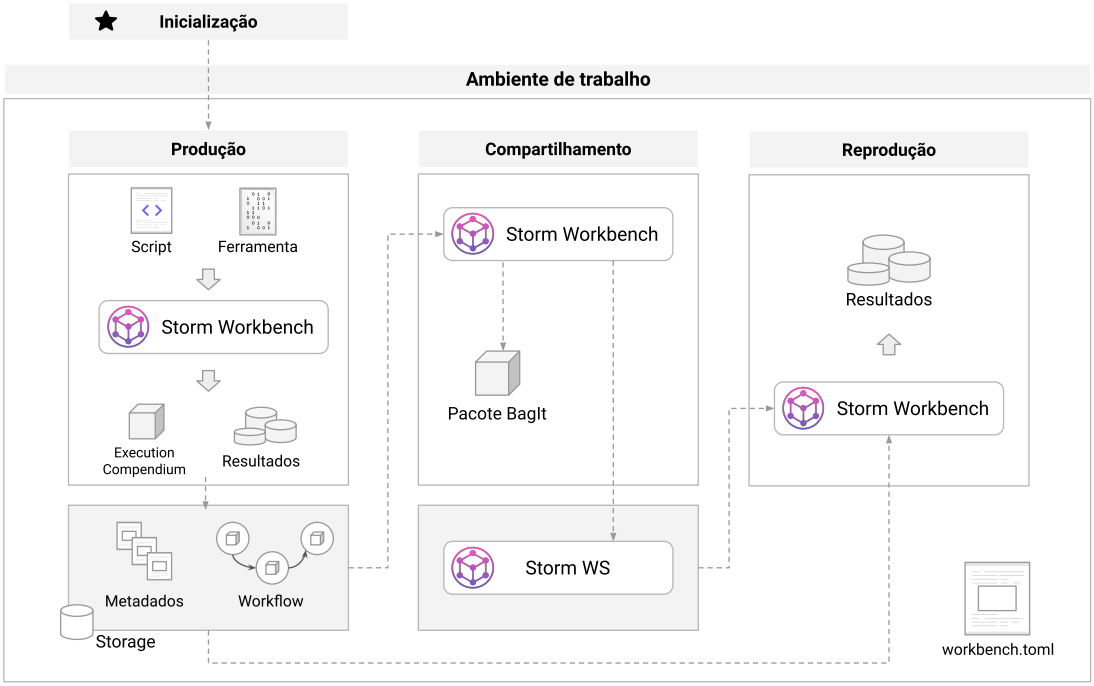
4.4.1 Storm Workbench

O `Storm Workbench` é uma ferramenta de auxílio à reprodutibilidade, que fornece métodos para a produção e reprodução de *Execution Compendia* diretamente no ambiente de trabalho do pesquisador. Além disso, essa ferramenta fornece métodos que facilitam a integração do fluxo de trabalho do pesquisador com serviços externos de reprodutibilidade, incluindo o `Storm WS`. Com isso, tem-se que seu uso pode ser feito independentemente do serviço de reprodutibilidade da `Plataforma Storm`, dando liberdade aos pesquisadores para escolherem como o desenvolvimento e compartilhamento de suas pesquisas reprodutíveis serão realizados. Essa ferramenta representa uma implementação do `Graph Controller Client`, especificado no projeto da `Plataforma Storm`.

Para uso de suas funcionalidades, o `Storm Workbench` disponibiliza aos usuários um CLI de alto nível, com recursos que permitem a fácil visualização e entendimento do *workflow* que está sendo construído com as execuções realizadas pelo usuário. Ademais, a ferramenta também fornece acesso programático através de uma API de alto nível, permitindo que outras aplicações sejam criadas usando suas funcionalidades. A implementação dessa API foi feita tendo como modelo de *design* a estrutura fornecida pela API da ferramenta `git`. Em ambos os casos, a utilização da ferramenta e suas capacidades, assim como ilustrado na Figura 4.9, é feita através de quatro etapas principais, sendo elas:

Inicialização: No `Storm Workbench`, com o objetivo de auxiliar os pesquisadores na organização de seus experimentos para que as etapas de reprodução, entendimento e replicação dos resultados sejam facilitadas, usa-se o conceito de RC. Esse conceito é materializado na ferramenta através do espaço de trabalho onde o pesquisador fará o desenvolvimento de suas pesquisas. Dentro desse ambiente, o pesquisador pode utilizar todas as funcionalidades do `Storm Workbench`. Com a adoção desse modo de operação, tem-se o incentivo a centralização dos objetos de pesquisa em um mesmo diretório, tornando a pesquisa mais organizada. Ao mesmo tempo, tem-se que isso permite que a ferramenta seja utilizada em diferentes projetos, sem que as configurações de operações de um projeto afete outro.

Figura 4.9 - Fluxo de utilização do `Storm Workbench`.



Fonte: Produção do autor.

Para a criação desse ambiente de trabalho, na primeira vez em que o pesquisador está configurando a ferramenta, é necessário realizar a etapa de **Inicialização**. Nessa etapa, o pesquisador especifica em qual diretório de seu SO ficará armazenado o espaço de trabalho onde ele desenvolverá suas pesquisas com o auxílio do `Storm Workbench`. Ao executar essa operação, conforme ilustrado na Figura 4.10, é feita a criação do arquivo `workbench.toml`. Esse arquivo possui um conjunto de metadados base que descrevem o trabalho do pesquisador, bem como um conjunto de configurações que podem ser utilizadas para alterar a forma de funcionamento da ferramenta. Com isso, o usuário pode, por exemplo, configurar quais arquivos não devem ser considerados na construção de um *Execution Compendium*. A configuração do endereço do `Storm WS` também é feita nesse arquivo.

De forma complementar ao arquivo de configuração, na **Inicialização** também é gerado o arquivo auxiliar `.secrets.toml`, em que o usuário pode especificar a chave de acesso que deve ser utilizada pela ferramenta para autenticação no serviço de reprodutibilidade.

Figura 4.10 - Fluxo de inicialização do espaço de trabalho com o `Storm Workbench`.

Fonte: Produção do autor.

Produção: Uma vez que a **Inicialização** é realizada, o ambiente do pesquisador está pronto para uso. Nesse ponto, dá-se início a etapa de **Produção**, em que o pesquisador realiza a produção dos resultados de sua pesquisa. Nessa etapa o pesquisador faz a criação de um fluxo de processamento de dados, que pode incluir etapas de aquisição, exploração, transformação de dados, e desenvolvimento de modelos que explicam ou utilizam os dados para a geração dos resultados (WICKHAM; GROLEMUND, 2017). Durante o desenvolvimento das etapas mencionadas, o pesquisador pode fazer o uso de diferentes ferramentas e técnicas, que vão desde a criação de *scripts ad-hoc* de processamento até ferramentas de linha de comando com funcionalidades prontas para uso. Ao final do processo, o pesquisador tem um fluxo de processamento formado por várias etapas, podendo essas utilizarem diferentes abordagens.

Para tornar cada uma das etapas do fluxo de processamento reprodutível, o pesquisador pode então fazer o uso do `Storm Workbench`. Nesse caso, tem-se que para cada uma das execuções realizadas, seja essa de um *script* ou ferramenta de linha de comando, a criação de um `Execution Compendium`, no qual se tem salvo todas as informações, dados e *softwares* utilizados na execução realizada. Com isso, tem-se a garantia de que a reprodução é tratada enquanto os resultados estão sendo gerados, evitando que novas execuções das mesmas operações tenham de ser feitas para a

verificação da reprodutibilidade dos processos realizados.

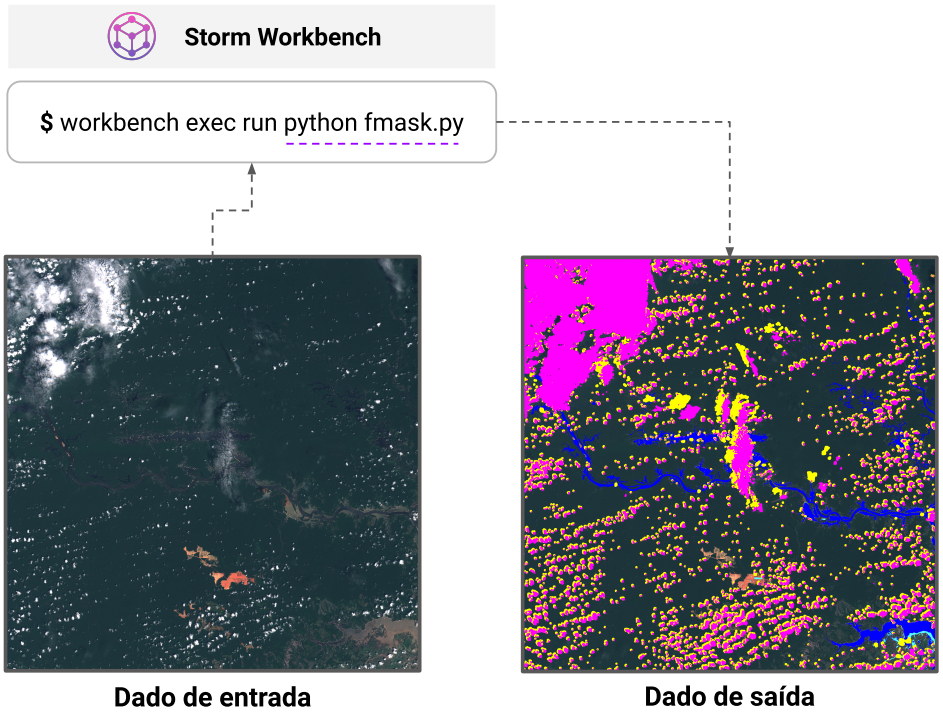
Nesse processo, para a geração dos *Execution Compendia* faz-se o uso das funcionalidades providas pelo `ReproZip`. Nesse caso, tem-se que, para cada uma das execuções, é feita a geração de um *Experiment Package*, que contém todas as informações necessárias para a reexecução da operação realizada pelo pesquisador. Após a geração do *Experiment Package*, as informações geradas são processadas e salvas em um banco de dados local, criado com `SQLite`[25]. Também, fazendo o uso dos metadados que descrevem as entradas e saídas da operação, o pacote gerado é indexado em um banco de dados de grafo criado com o uso do `iGraph`[26]. Isso faz com que o `Storm Workbench`, consiga representar as execuções como um *workflow*, podendo então, fazer seu gerenciamento e reprodução.

Para usufruir dessas funcionalidades, faz-se necessário que o pesquisador, em cada uma das execuções de *scripts* de processamento ou ferramentas de linha de comando que geram resultados para sua pesquisa, realize a operação através do `Storm Workbench`. Na Figura 4.11, tem-se a apresentação de como essa ação é realizada[27]. Nesse caso, o pesquisador deseja fazer a execução de um *script* da linguagem de programação Python. Para isso, primeiro indica-se que o `Storm Workbench` fará uma execução, e então, define-se o comando que deve ser executado (trecho sublinhado na cor roxa). Ao fazer isso, o *script* será executado e os resultados serão gerados. Todos os detalhes relacionados a reprodutibilidade serão transparentes ao usuário. Isso evita ações intrusivas no código e a necessidade de modificações no fluxo de trabalho do pesquisador.

Ao fazer a execução conforme apresentado na Figura 4.11, o `Storm Workbench` fará a geração do *Execution Compendium* contendo todas as informações utilizadas na execução, possibilitando assim que reproduções sejam feitas. Com isso, tem-se que essa operação representa o `Registro` descrito no projeto da `Plataforma Storm`. Ademais, ao realizar essa operação para cada execução, ao final do processo, o *workflow* completo de processamento estará registrado, podendo ser reexecutado ou compartilhado.

[25]Disponível em: <https://www.sqlite.org/index.html>. Acesso em: 21/11/2022
[26]Disponível em: <https://igraph.org/>. Acesso em: 21/11/2022
[27]O código fonte utilizado na construção da Figura está disponível em: https://tinyurl.com/storm-examples-wb01

Figura 4.11 - Fluxo de execução de um *script* via `Storm Workbench`.



Fonte: Produção do autor.

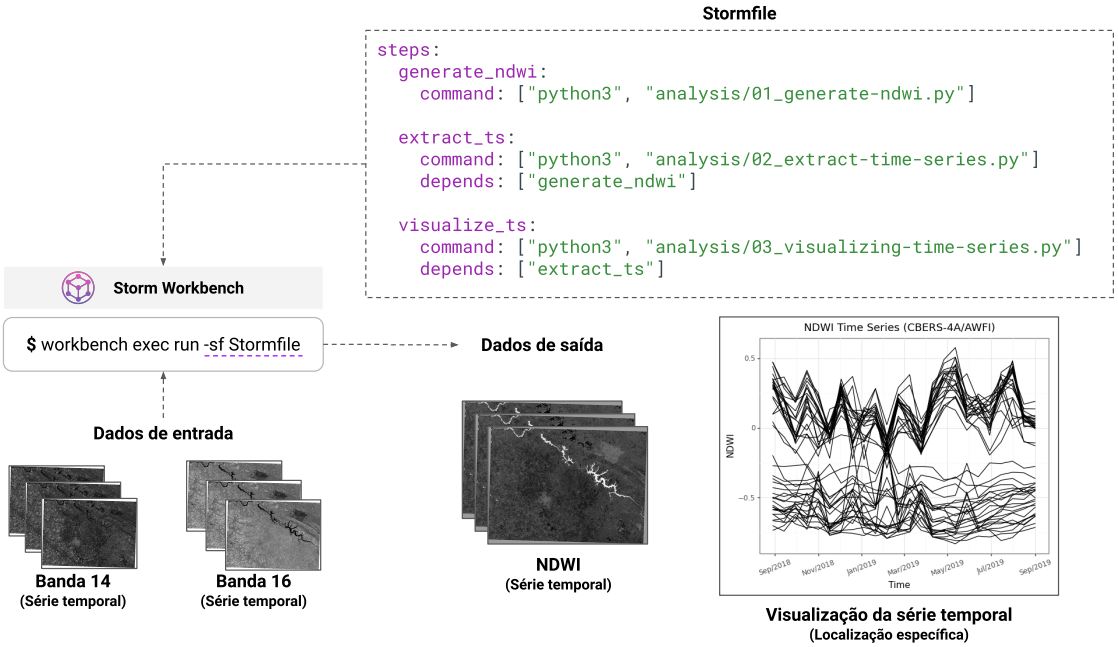
Além desse cenário de execução base, existem casos em que o pesquisador já possui um fluxo de trabalho completo, desejando não ter que executar cada uma das partes individualmente para garantir a reprodutibilidade. Para solucionar esse tipo de necessidade, tem-se no `Storm Workbench`, o suporte a pré-definição do fluxo de processamento. Com o uso desse formato de execução, o pesquisador descreve o fluxo de processamento, declarando o comando que deve ser executado em cada etapa, bem como a relação de dependência entre os processos que serão executados. A declaração de todas essas informações é feita através do `Stormfile`, um arquivo `YAML` que possui um conjunto de campos, que permite aos pesquisadores expressar seu fluxo de processamento.

Na Figura 4.12, tem-se ilustrado um exemplo do `Stormfile` junto a sua forma de uso através do `Storm Workbench`[28]. Como pode-se notar no exemplo, tem-se a declaração de um fluxo de processamento em que é feita a geração de dados e em

[28]O código fonte utilizado na construção da Figura está disponível em: https://tinyurl.com/storm-examples-wb02

seguida a criação de uma figura de visualização. Para isso, os comandos que devem ser executados são definidos com a dependência de operação entre cada um deles.

Figura 4.12 - Exemplo de utilização do `Stormfile` via `Storm Workbench`.



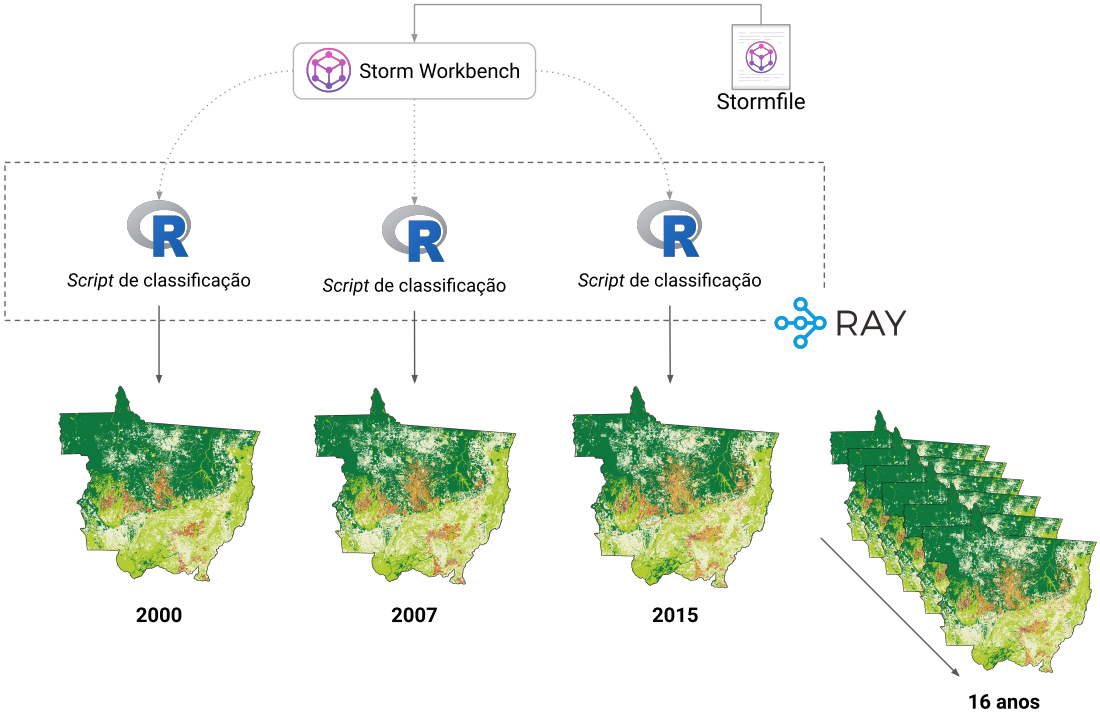
Fonte: Produção do autor.

Com o uso do `Stormfile`, além da facilidade de expressar o fluxo completo de processamento, é disponibilizado também pelo `Storm Workbench`, opções que permitem a execução paralela do fluxo descrito. Nesse caso, tem-se que com base nas dependências entre os comandos declaradas pelo pesquisador, pode ser feita a execução de mais de um processo simultaneamente. O usuário pode escolher a tecnologia que será utilizada para escalonar os processos e realizar a execução. Por padrão, a ferramenta possibilita execuções paralelas utilizando as tecnologias `ParaDag`[29] e `Ray Core`[30]. Para exemplificar a utilização desse modo de operação, fez-se a execução reprodutível da geração de mapas de LULC apresentada por Simoes et al. (2020). Ao todo, são gerados 16 mapas de LULC, cada um deles representando um ano específico na região de estudo. Esses mapas, conforme ilustrado na Figura 4.13, são gerados por várias execuções do mesmo *script*, sendo que em cada execução, faz-se a classificação de um ano específico. Com isso, na execução paralela, mais de um

[29]Disponível em: <https://github.com/xianghuzhao/paradag>. Acesso em: 21/11/2022
[30]Disponível em: <https://www.ray.io/ray-core>. Acesso em: 21/11/2022

mapa pode ser gerado em simultâneo. A execução foi feita utilizando `Ray Core`[31].

Figura 4.13 - Exemplo de execução reprodutível do fluxo de classificação apresentado por Simoes et al. (2020) com execução de *script* em paralelo.



Fonte: Produção do autor.

Compartilhamento: A finalização da etapa de **Produção** permite ao pesquisador compartilhar o trabalho reprodutível desenvolvido. Para isso, duas opções estão disponíveis: (i) compartilhamento via arquivo; (ii) compartilhamento via serviço.

O compartilhamento via arquivo permite ao pesquisador exportar todo o *workflow* mapeado pela ferramenta e seus materiais auxiliares, como os dados, metadados e arquivos de ambiente computacional, em um arquivo único que pode ser compartilhado em ambientes de preservação e disseminação de dados, por exemplo, `Zenodo` e `Pangaea`[32]. Esse arquivo permite a reprodução do *workflow* completo desenvolvido pelos pesquisadores. Para auxiliar na preservação e consistência dos materiais compartilhados, esse arquivo único é criado como um `BagIt` (KUNZE et al., 2018),

[31] O código fonte utilizado na construção da Figura está disponível em: <`https://tinyurl.com/storm-examples-wb03`>

[32] Disponível em: <`https://pangaea.de/`>. Acesso em: 22/11/2022

possuindo metadados úteis com o *checksum* de cada um dos arquivos contidos no pacote. Como segunda opção, tem-se o compartilhamento via serviço. Fazendo o uso dessa opção, o pesquisador pode enviar os resultados gerados juntamente com o *workflow* e materiais auxiliares para o `Storm WS`. Nesse caso, para cada etapa do *workflow* é criado um *Execution Compendium* no serviço de reprodutibilidade. Em seguida, esses *compendia* são vinculados a um *Research Workflow*, salvando assim o fluxo utilizado na geração dos resultados.

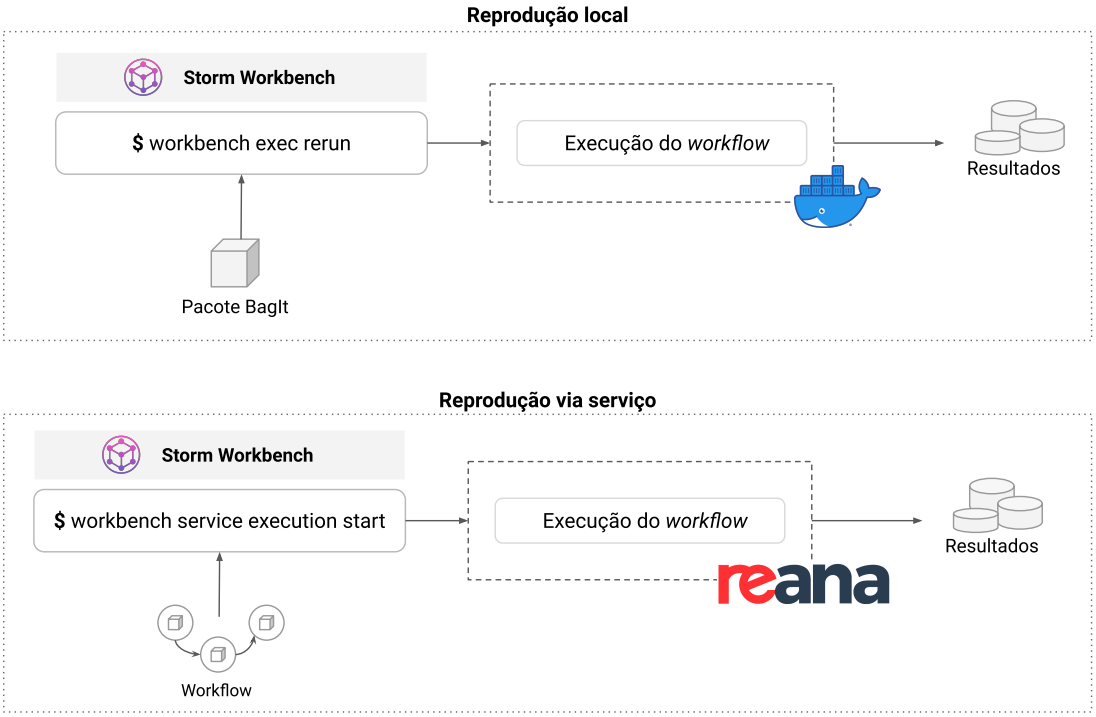
Além disso, tem-se que, caso o pesquisador deseje compartilhar apenas uma etapa específica do *workflow* criado no ambiente local também é possível. Para esse caso, a ferramenta irá identificar todas as etapas dependentes da etapa selecionada e então fará o envio de todas elas para o serviço. Também tem-se que, caso a etapa utilizada pelo pesquisador já tenha uma versão publicada como um *Execution Compendium* no serviço, tem-se a opção de criar uma nova versão para salvar as modificações feitas, caso existam.

Reprodução: Após o **Compartilhamento**, é esperado que o *workflow* gerado possa ser reproduzido. Para isso, conforme ilustrado na Figura 4.14, o `Storm Workbench` disponibiliza as formas de reprodução local e via serviço.

Na reprodução local é esperado que o pesquisador que está realizando a operação, tenha disponível o pacote de reprodução, o qual é gerado na etapa de **Compartilhamento** e salvo no formato `BagIt`. Uma vez tendo o arquivo, o pesquisador o utiliza para realizar a reprodução. Quando isso é feito, a ferramenta, tendo como base os dados contidos no pacote, prepara um ambiente de execução utilizando `Docker` e realiza a reprodução.

No entanto, nota-se que o primeiro formato de reprodução, requer um ambiente pronto para tal atividade. Por vezes, os pesquisadores não terão recursos ou o ambiente configurado para essa atividade. Para isso, tem-se a reprodução via serviço, em que o pesquisador consegue selecionar um `Research Workflow` disponível no `Storm WS` e requisitar sua execução. Nesse caso, seguindo a estrutura do serviço de reprodutibilidade apresentado nas Seções anteriores, tem-se que a reprodução, por padrão, será feita no `Reana`.

Figura 4.14 - Fluxo de reprodução de um *workflow* via `Storm Workbench`.



Fonte: Produção do autor.

Para a implementação dessas operações, tendo como base a estrutura seguida para a implementação do `Storm WS`, fez-se o uso de uma arquitetura modular, em que as diferentes operações foram implementadas por vários módulos. Com isso, o `Storm Workbench` tem suas funcionalidades implementadas com base em três diferentes módulos, sendo eles:

- `Storm Graph`[33]: Módulo que provê funcionalidades que permitem a manipulação de grafos no `Storm Workbench`;
- `Storm Client`[34]: Módulo que provê uma API de alto nível para interação com o `Storm WS` em código Python;
- `Storm Core`[35]: Módulo principal do `Storm Workbench`. Nesse módulo, tem-

[33]Disponível em: <https://github.com/storm-platform/storm-graph>. Acesso em: 02/11/2022
[34]Disponível em: <https://github.com/storm-platform/storm-client>. Acesso em: 02/11/2022
[35]Disponível em: <https://github.com/storm-platform/storm-core>. Acesso em: 02/11/2022

se implementado todas as principais capacidades do `Storm Workbench`, as quais incluem execução reprodutível de *scripts* e ferramentas, feitas com o uso do `ReproZip` e também o gerenciamento de execuções.

A adoção dessa abordagem tornou o processo de desenvolvimento mais simples e rápido, permitindo que diferentes funcionalidades fossem implementadas e modificadas sem grandes dificuldades. Além disso, tem-se também o aumento na capacidade de reúso das ferramentas e funcionalidades disponíveis. Por exemplo, com as funcionalidades implementadas no `Storm Core`, tem-se a possibilidade de expandir o ecossistema de ferramentas da `Plataforma Storm` já que esse pode ser integrado com outras ferramentas. Um exemplo de extensão poderia ser a criação de um *plugin* de execução reprodutível para a ferramenta `Jupyter Notebook`.

4.4.2 Storm Environment

Na realização de pesquisas colaborativas, o uso de ambientes padronizados de trabalho pode facilitar as etapas de organização do projeto e também na produção de resultados reprodutíveis. Nesta implementação demonstrativa, com o objetivo de criar uma versão funcional do `Environment Controller`, definido no projeto da `Plataforma Storm` para prover ambientes prontos para uso aos usuários, fez-se a definição do componente `Storm Environment`[36].

Com o uso do `Storm Environment`, os pesquisadores conseguem acessar um ambiente de processamento de dados prontos para uso, que conta com as principais bibliotecas e ferramentas para análise de dados com as linguagens de programação R e Python. Ferramentas para o processamento de dados espaciais também estão inclusas nesses ambientes. Para a fácil utilização das funcionalidades da `Plataforma Storm`, por padrão, os ambientes possuem o `Storm Workbench` instalado e pronto para uso.

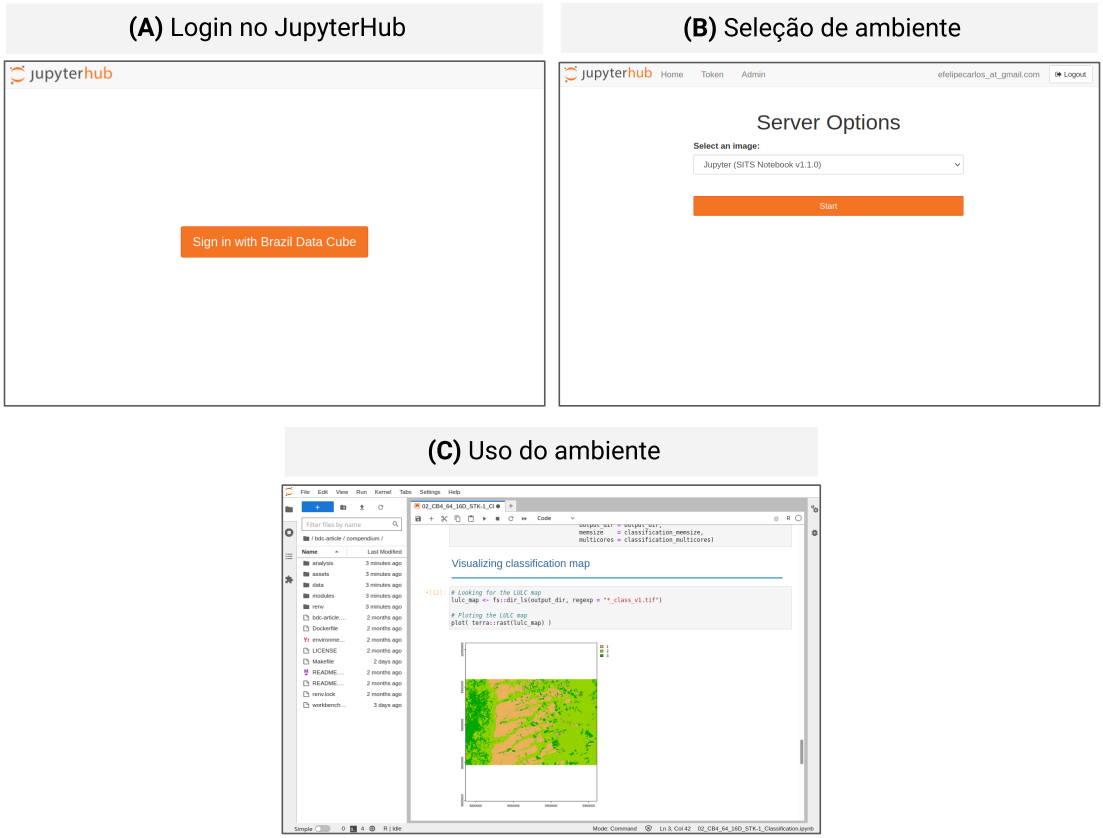
A implementação do `Storm Environment` foi feita com base no `JupyterHub`[37], uma ferramenta para o provisionamento de ambientes virtuais com base em `Jupyter Notebooks` e `Docker` *Containers*. Assim, conforme ilustrado na Figura 4.15, uma vez acessando o ambiente, o usuário deve efetuar o *login*. Após a realização do *login*, que pode ser feito com o uso de uma conta da `Plataforma BDC`, o usuário consegue selecionar o tipo de ambiente no qual ele deseja trabalhar. A escolha do

[36]Disponível em: <https://github.com/storm-platform/storm-environment>. Acesso em: 11/11/2022

[37]Disponível em: <https://jupyter.org/hub>. Acesso em: 11/11/2022

ambiente determina quais serão as ferramentas disponibilizadas para o usuário. Após a seleção, o usuário é redirecionado para uma página em que ele tem acesso ao ambiente selecionado. Então, ele pode começar a processar seus dados e fazer o uso das ferramentas e serviços `Storm` para a produção de resultados reprodutíveis.

Figura 4.15 - Fluxo de utilização do `Storm Environment`.



Fonte: Produção do autor.

Toda a implementação realizada para o correto funcionamento do `Storm Environment`, foi feita em uma biblioteca reutilizável[38], que pode ser aplicada para a geração de um ambiente equivalente ao `Storm Environment` em qualquer projeto ou instituição.

[38]Disponível em: <https://github.com/brazil-data-cube/jupyterhub-docker>. Acesso em: 11/11/2022

4.4.3 Outras aplicações cliente

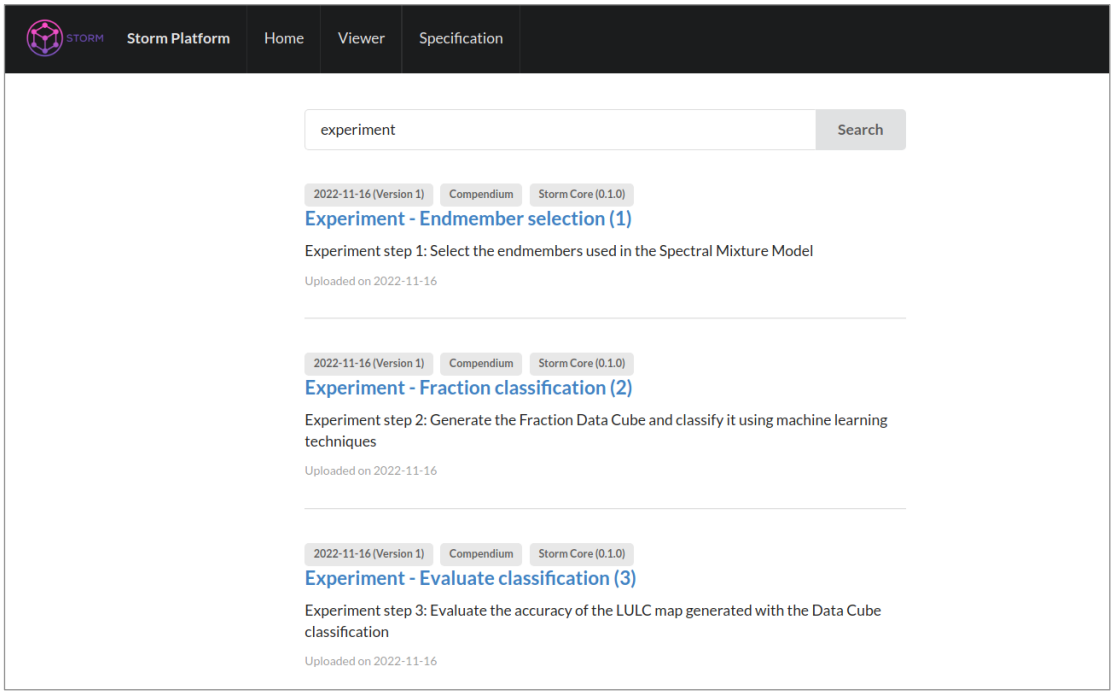
Além do conjunto de ferramentas clientes desenvolvidas para auxiliar os pesquisadores no uso dos serviços da `Plataforma Storm` para a criação de resultados reprodutíveis, fez-se também o desenvolvimento de clientes com funcionalidades que ilustram os diferentes tipos de aplicações que podem ser criados com os serviços da plataforma, indo além das capacidades relacionadas a reprodutibilidade. Nesse contexto, fez-se o desenvolvimento de duas aplicações *web*. Cada uma delas são apresentadas brevemente nos tópicos a seguir:

Storm Searcher

Enquanto a pesquisa está sendo produzida, por conta de sua natureza não linear, muitos *Execution Compendia* podem ter de ser registrados no `Storm WS`. Parte dessas adições, podem, por exemplo, representar novas versões de um *Execution Compendium* ou mesmo execução de uma etapa nova da pesquisa. Sendo assim, uma aplicação que facilita a visualização desses recursos pode ser necessária. Para esse caso, foi implementado o `Storm Searcher`[39], uma aplicação *web* que fornece uma interface para a busca dos *Execution Compendia* que estão disponíveis dentro de um *Research Project*. Um exemplo da interface desta aplicação é apresentado na Figura 4.16.

[39]Disponível em: <https://github.com/storm-platform/storm-searcher>. Acesso em: 03/12/2022

Figura 4.16 - Exemplo da interface de busca do `Storm Searcher`.



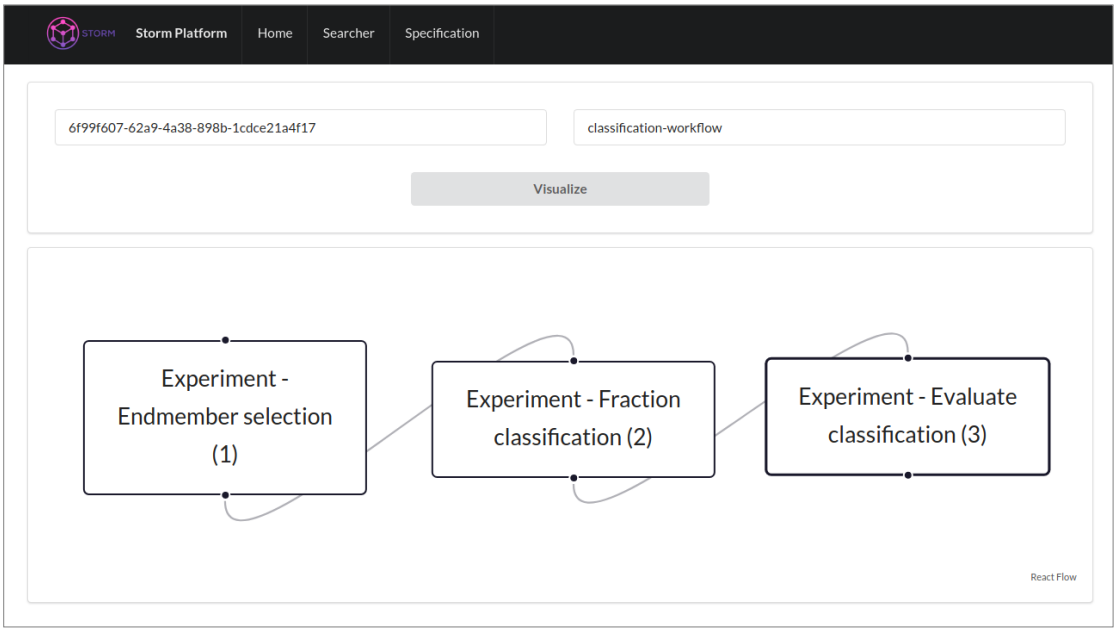
Fonte: Produção do autor.

Storm Viewer

O uso de *workflows* para representar o fluxo utilizado em uma pesquisa para a geração de seus resultados, trás vários benefícios para a reprodutibilidade e o entendimento do trabalho desenvolvido. Por exemplo, com o uso dessa abordagem, pode-se saber com clareza *"com o que"* e *"quem"* fez a geração de um resultado. Também, pode-se entender onde novas etapas e melhorias podem ser realizadas. No `Storm WS`, conforme apresentado nos Capítulos anteriores, toda a representação do fluxo de uma pesquisa é realizada através do uso de *workflows*. Assim, de modo a facilitar a visualização dos *workflows* gerados, fez-se a criação do `Storm Viewer`[40], uma aplicação *web* que permite a visualização dos *workflows* disponíveis em um *Research Project*. A Figura 4.17 apresenta um exemplo da interface desta aplicação. Como pode-se observar, após a definição das informações relativas ao *workflow* escolhido, tem-se sua renderização em modo interativo.

[40]Disponível em: <https://github.com/storm-platform/storm-viewer>. Acesso em: 03/12/2022

Figura 4.17 - Exemplo da interface do `Storm Viewer`.



Fonte: Produção do autor.

4.5 Disponibilidade de código

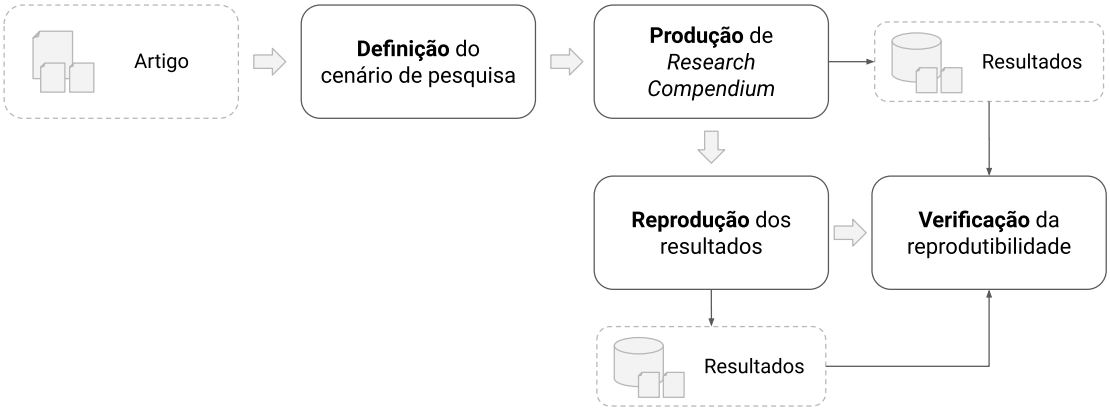
Com base na filosofia da Ciência Aberta e Reprodutível, neste trabalho, todos os materiais desenvolvidos para a composição da implementação demonstrativa, incluindo o código dos módulos, *design* dos logos e *scripts* para *deploy* dos serviços, estão disponíveis *online* sob licença MIT[41]. Ao todo, são 27 repositórios de códigos, que podem ser acessados na organização `SpatioTemporal Open Research Manager` no GitHub[42], criada especialmente para o desenvolvimento desta Dissertação.

[41] Disponível em: <`https://opensource.org/licenses/MIT/`>. Acesso em: 01/12/2022
[42] Disponível em: <`https://github.com/storm-platform`>. Acesso em: 01/12/2022

5 EXPERIMENTOS

Para que fosse possível realizar a avaliação parcial da viabilidade de adoção de uma implementação da `Plataforma Storm` em cenários de pesquisas reais, fez-se a criação de experimentos nos quais as funcionalidades da implementação demonstrativa apresentada no Capítulo 4 foram avaliadas. Esses experimentos foram criados com base em artigos selecionados da literatura. Em cada um deles, fez-se testes da capacidade de produção reprodutível de resultados. Ao todo, foram conduzidos dois experimentos. A realização de cada um deles foi feita seguindo a metodologia apresentada na Figura 5.1.

Figura 5.1 - Metodologia de desenvolvimento dos experimentos deste trabalho.



Fonte: Produção do autor.

Para a realização da metodologia apresentada na Figura 5.1, primeiro é necessário realizar a seleção dos artigos da literatura. Esses artigos, em cada experimento, servem como base teórica e prática, sendo eles utilizados como o guia para a definição da atividade realizada. Neste caso, dado o vasto cenário científico de EO, e os múltiplos temas onde suas técnicas podem ser estudadas e aplicadas, definiu-se que apenas artigos relacionados ao processamento e uso de dados espaço-temporais fossem considerados durante as seleções. Essa definição também torna os experimentos coerentes com o contexto em que o presente trabalho foi desenvolvido.

Com os artigos definidos, pode-se então realizar os experimentos seguindo a metodologia da Figura 5.1. Na primeira etapa, tem-se a criação de um cenário fictício de

pesquisa que representa o ambiente em que o experimento está sendo desenvolvido. Com isso, tem-se a possibilidade de apresentar os diferentes cenários de utilização da plataforma e suas funcionalidades.

Após a definição do cenário de pesquisa, dá-se início a produção do experimento propriamente dito e a geração de seus resultados. Para a realização dessa etapa, faz-se a criação de um RC que armazena todos os elementos da pesquisa. Esse RC é criado com base nas recomendações de organização apresentadas por Marwick et al. (2018). Sendo assim, é modelado como pacote de *software* com as descrições de suas dependências e configurações feitas de forma explícita. Além disso, para a automatização das execuções, com base nas boas práticas apresentadas pela Comunidade Turing Way et al. (2019), todas as operações necessárias para a realização do experimento são declaradas e documentadas utilizando a ferramenta `GNU Make`.

Tendo os resultados gerados, com o uso dos recursos disponíveis na implementação demonstrativa, faz-se a realização da reprodução dos experimentos. Então, com base nos resultados gerados na reprodução, é feita uma análise para a verificação da reprodutibilidade. Nessa análise é feita a comparação de igualdade dos resultados em nível de *bytes*, através do uso de algoritmos de *checksum*. Em especial, são aplicados os algoritmos *MD5*, *SHA256* e *SHA512*.

Para além dos resultados reprodutíveis, neste trabalho, buscou-se tornar a execução dos experimentos reprodutíveis, de modo que as declarações realizadas nesse documento pudessem ser avaliadas pelo leitor. Para isso, foram tomados alguns cuidados específicos no gerenciamento dos ambientes computacionais utilizados. No desenvolvimento do código e na geração dos resultados fez-se a criação e uso de ambientes padronizados com `Docker` *Images*. Esses ambientes foram utilizados através de uma instância do `Storm Environment`, configurada na infraestrutura do projeto BDC durante a realização dos experimentos. Além disso, nas etapas de produção e reprodução, de modo a evitar que o viés ao SO base afetasse o uso das `Docker` *Images* criadas, fez-se sua execução em uma máquina virtual. Dessa forma, conforme apresentado por Leveque et al. (2012), tem-se que todo o ambiente utilizado originalmente fica disponível, evitando que versões ou configurações específicas na forma de uso do `Docker` afete a reprodução dos experimentos. As execuções das máquinas virtuais foram feitas através da abordagem declarativa fornecida pela ferramenta `Vagrant`[1], na qual, através de um arquivo de configuração, faz-se a especificação de todas as características da máquina virtual.

[1]Disponível em: <`https://www.vagrantup.com/`>. Acesso em: 19/11/2022

Além desses detalhes que auxiliam a reprodutibilidade dos experimentos, tem-se também que, em todas as etapas, tendo como base as práticas para a produção de pesquisas reprodutíveis apresentadas por Stanisic et al. (2015), fez-se o uso de uma ferramenta de tomada de notas. No contexto desse trabalho, esse tipo de ferramenta, que embora seja simples, serviu para controlar todas as execuções, resultados e configurações dos experimentos, facilitando principalmente as atividades que dizem respeito à avaliação da reprodução. Para esse propósito, a ferramenta utilizada foi o `Obsidian`[2].

No restante deste Capítulo serão apresentados dois experimentos desenvolvidos. Na Seção 5.1, tem-se a apresentação do primeiro experimento, no qual, com base no artigo e metodologia apresentado por Ferreira et al. (2020), faz-se a apresentação de um cenário básico de uso do ecossistema `Storm`. Em seguida, na Seção 5.2, tem-se o segundo experimento, que com base no trabalho desenvolvido por Sanchez et al. (2019), apresenta um cenário de uso em que as ferramentas da `Plataforma Storm` são utilizadas para a produção colaborativa de pesquisa reprodutível.

5.1 Produção de mapas de Uso e Cobertura de Terra com cubos de dados de Observação da Terra e algoritmos de Aprendizado de Máquina

Nesta Seção é apresentado o primeiro experimento[3] desenvolvido neste trabalho em que é feita a geração reprodutível de mapas de LULC seguindo os passos metodológicos descritos por Ferreira et al. (2020). Essa atividade é desenvolvida com o objetivo de evidenciar as capacidades de produção de pesquisa reprodutível oferecidas pela `Plataforma Storm`. Com isso, embora o artigo selecionado tenha o potencial para representar um cenário de pesquisa complexo, neste experimento, optou-se por sua representação em um cenário de pesquisa simplificado, em que apenas um pesquisador desenvolve todas as etapas da pesquisa.

No restante que segue é feita a contextualização do experimento juntamente com a apresentação dos detalhes relacionados ao cenário de pesquisa, metodologia e dados utilizados.

[2]Disponível em: <`https://obsidian.md/`>. Acesso em: 09/10/2022

[3]O código fonte completo do experimento está disponível em: `https://tinyurl.com/storm-experiment-01`

5.1.1 Contextualização

Dados de EO possuem um enorme potencial na produção de informações que auxiliam setores públicos e privados a construírem uma sociedade melhor, em que se tem melhor gerenciamento de recursos naturais, resiliência a desastres naturais e a possibilidade de avanços contínuos na redução dos impactos causados pelas atividades antrópicas na superfície terrestre (GROUP ON EARTH OBSERVATIONS, 2022). Todo esse potencial, tem sido exaltado nos últimos anos, com o aumento contínuo nos dados de EO que estão sendo disponibilizados (SOILLE et al., 2018). Pesquisadores e especialistas utilizam esses dados para o desenvolvimento de pesquisas e atividades que possuem resultados inovadores, com a criação de informações nunca vistas.

Para extrair todo o potencial informacional presentes nesses dados, nos últimos anos, pesquisadores em EO os tem organizado no formato de EODC (KOPP et al., 2019). Com esse, tem-se a possibilidade de realizar avaliações temporais dos fenômenos da superfície terrestre e tirar conclusões utilizando não apenas o estado atual, mas também, todas as mudanças mapeadas ao longo do tempo (APPEL; PEBESMA, 2019; GIULIANI et al., 2020).

Na busca por respostas sobre os fenômenos do planeta, a realização de avaliações temporais pode ser empregada como parte de diversas aplicações. Por exemplo, Killough (2019) utiliza de EODC e avaliações temporais para o monitoramento da urbanização em Nairóbi, no Quênia. Outra aplicação relevante das avaliações temporais, está na geração de mapas de LULC. Nesse contexto, há diversos trabalhos sendo desenvolvidos aplicando técnicas *time-first, space-later* (CAMARA et al., 2016) para a geração desses mapas com o uso de séries temporais extraídas de EODC. Por exemplo, Picoli et al. (2020) utiliza EODC para a geração de mapas de LULC no estado do Mato Grosso. Tais mapas, são relevantes para o monitoramento de biomas, detecção de desmatamento e para estimar a emissão de gases de efeito estufa.

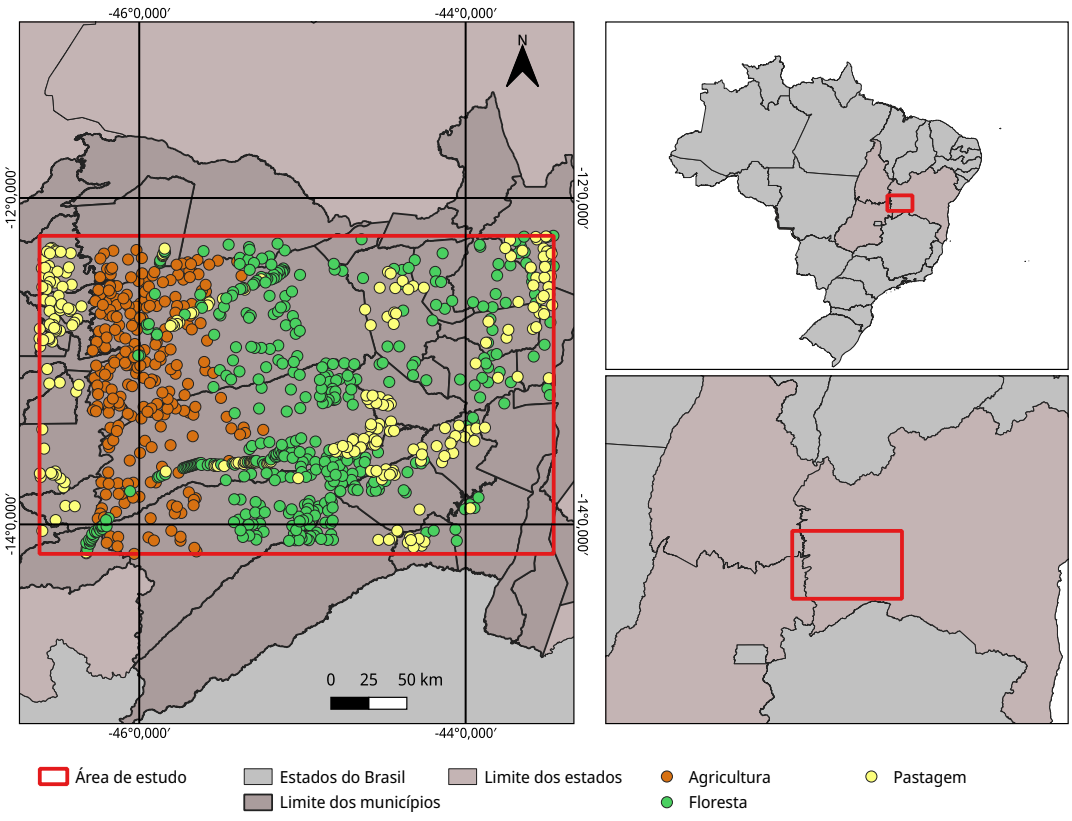
Dado a relevância analítica dos mapas de LULC, junto a possibilidade de geração desses utilizando EODC, nesse trabalho, tem-se como objetivo realizar a geração reprodutível de mapas de LULC. Para isso, será feita a aplicação da metodologia apresentada por Ferreira et al. (2020). Também, como uma primeira abordagem, tem-se que será utilizado a mesma região de estudo e amostras de LULC que os autores supracitados utilizaram no desenvolvimento do trabalho original.

5.1.2 Materiais e métodos

Área de estudo

A área de estudo selecionada para esse experimento, conforme apresentado na Figura 5.2, compreende a mesorregião do extremo oeste da Bahia, estando localizada entre os biomas do Cerrado e da Caatinga. Nessa região, tem-se a presença de acentuada atividades antrópicas, cobrindo as mais diversas atividades como as relacionadas à agricultura e pecuária (BORGES; SANO, 2014).

Figura 5.2 - Área de estudo localizada no Oeste da Bahia utilizada no experimento 1.



Fonte: Produção do autor.

Conjuntos de dados

Para a geração dos mapas de LULC, é feito o uso dos cubos de dados CBERS-

4/WFI[4], Landsat-8/OLI[5] e Sentinel-2/MSI[6] gerados e disponibilizados pelo Projeto BDC. As imagens desses cubos são prontas para análise (COMMITTEE ON EARTH OBSERVATION SATELLITES, 2021), sendo disponibilizadas em nível de reflectância de superfície. Esses cubos são gerados com o uso de composição temporal de 16 dias utilizando a abordagem *Least Cloud Cover First* para a seleção de *pixels* com menos nuvens e sombra de nuvens nesse período (FERREIRA et al., 2020).

Para o treinamento do modelo de *Random Forest* empregado na geração dos mapas de LULC, faz-se uso das séries temporais extraídas dos cubos de dados nas bandas do espectro visível (*red*, *green*, *blue*) e infravermelho próximo (*nir*). Também são utilizados os índices de vegetação *EVI - Enhanced Vegetation Index* e *NDVI - Normalized Diference Vegetation Index*, que são pré-calculados e disponibilizados nos cubos do Projeto BDC. Os dados extraídos são referentes ao período de *2018/08* até *2019/08*. Além disso, essa extração foi realizada considerando um conjunto de 922 amostras das classes: Agricultura (242 amostras), Floresta (422 amostras) e Pastagem (258 amostras). Tais amostras foram disponibilizadas por Ferreira et al. (2020).

5.1.3 Metodologia

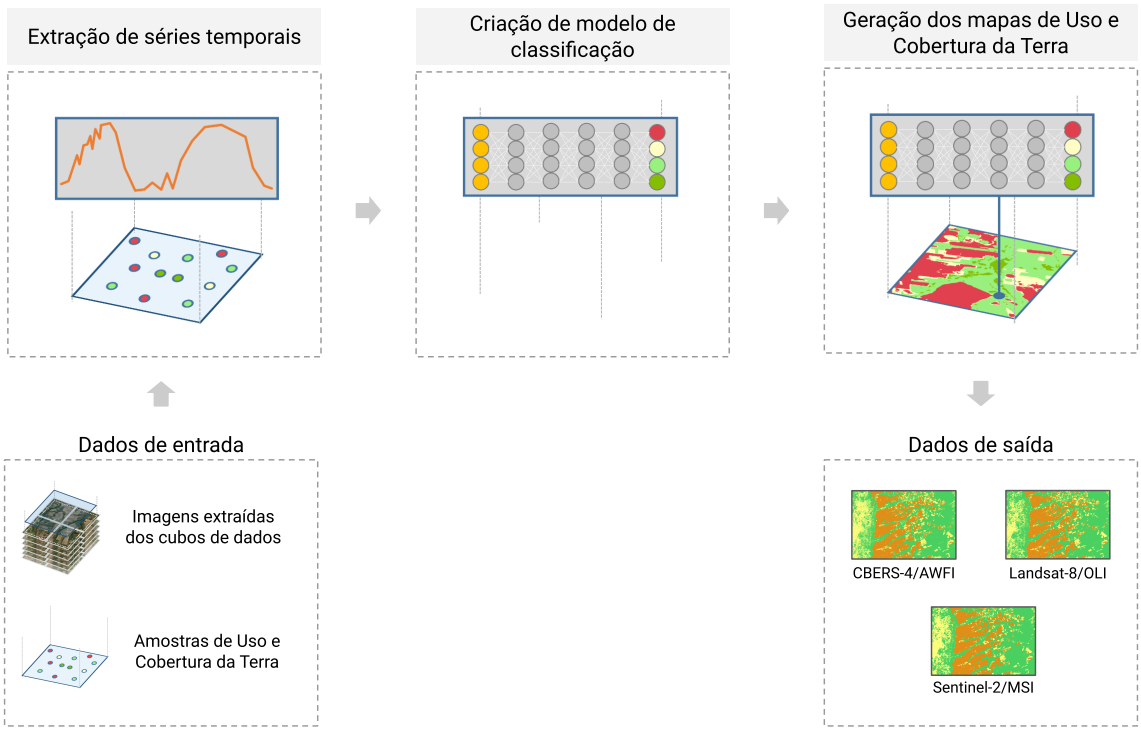
O desenvolvimento desse experimento foi realizado seguindo a metodologia descrita por Ferreira et al. (2020), a qual é ilustrada na Figura 5.3. Na primeira etapa, faz-se a seleção de cenas dos cubos de dados na área de estudo definida para o experimento. Em seguida, faz-se a extração de conjuntos de séries temporais associadas às amostras de LULC. Nesse caso, tem-se um conjunto de séries temporais para cada cubo de dados utilizado. Então, com base nos conjuntos de séries temporais extraídas, é feita a criação dos modelos de classificação. É criado um modelo *Random Forest* para cada conjunto de séries temporais extraídas. Com isso, ao todo se tem três modelos treinados. Por fim, com os modelos prontos, faz-se a aplicação deles para a classificação das séries temporais nas áreas de estudo. No final desse processo, têm-se os mapas de LULC.

[4]Disponível em: <https://brazildatacube.dpi.inpe.br/stac/collections/CB4_64_16D_STK-1>. Acesso em: 14/11/2022

[5]Disponível em: <https://brazildatacube.dpi.inpe.br/stac/collections/LC8_30_16D_STK-1>. Acesso em: 14/11/2022

[6]Disponível em: <https://brazildatacube.dpi.inpe.br/stac/collections/S2-SEN2COR_10_16D_STK-1>. Acesso em: 14/11/2022

Figura 5.3 - Metodologia de desenvolvimento do experimento 1.



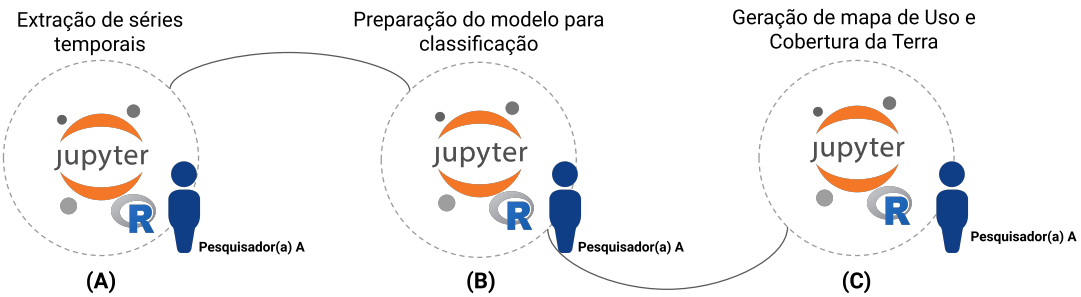
Fonte: Produção do autor.

5.1.4 Cenário de pesquisa

O cenário de pesquisa definido para esse experimento, assim como apresentado na Figura 5.4, tem sua divisão feita com base nas etapas da metodologia. Na primeira etapa, faz-se as atividades relacionadas à extração de séries temporais. Em seguida, é feita a geração do modelo de classificação. Por fim, o modelo é utilizado para a geração dos mapas de LULC.

O principal objetivo desse cenário é apresentar as capacidades básicas relacionadas à reprodutibilidade fornecida pelas ferramentas disponíveis na `Plataforma Storm`. Sendo assim, fez-se a definição de um cenário em que há apenas um pesquisador trabalhando em todas as etapas, de modo que seja simples evidenciar as capacidades supracitadas. Além disso, como ilustrado na Figura 5.4, se tratando de um cenário fictício em que apenas um pesquisador é representado, fez-se o uso do mesmo conjunto tecnológico para todas as etapas. Neste caso, as tecnologias são a linguagem de programação R e o pacote R `sits` (SIMOES et al., 2021). Além disso, tem-se que todos os códigos foram criados em documentos interativos `Jupyter Notebook`.

Figura 5.4 - Cenário de pesquisa do experimento 1.



Fonte: Produção do autor.

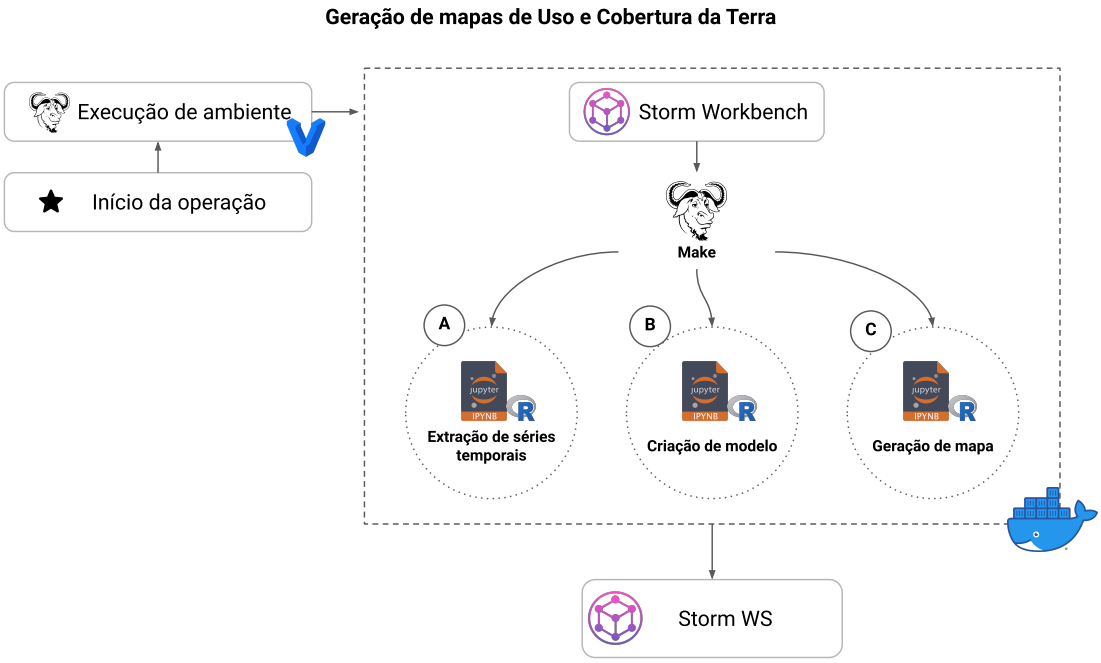
5.1.5 Resultados e discussões

Com o desenvolvimento de cada uma das etapas da metodologia, tem-se que o cenário ilustrado na Figura 5.4 foi materializado e executado através do fluxo de processamento apresentado na Figura 5.5. Sendo assim, após a configuração da máquina virtual e a inicialização do ambiente `Docker`, fez-se a execução de cada uma das etapas da metodologia. Inicialmente, foi feita a execução da extração das séries temporais. Na segunda etapa, com as séries temporais extraídas, fez-se a criação dos modelos de classificação. Por fim, pôde-se fazer a geração dos mapas de LULC. Cada uma das execuções foram feitas com a utilização do `Storm Workbench`, que com seu modelo não intrusivo de reprodutibilidade, não trouxe nenhuma tarefa extra ao desenvolvimento do experimento.

Após a finalização das execuções, o `Storm Workbench`, com base nos metadados coletados durante a realização das operações, tem registrado as informações relacionadas às entradas e saídas de cada uma das operações. Com isso, na ferramenta tem-se gerado o *workflow* da aplicação, com a relação e ordem de todas as etapas. Isso permite que o fluxo completo possa ser reproduzido, de forma local ou via `Storm WS`. Nesse cenário inicial, fez-se a reprodução utilizando o ambiente local. Para isso, primeiro fez-se a exportação dos *Execution Compendia* gerados em cada uma das execuções. A exportação dessas múltiplas etapas foi salva em um único arquivo `zip` com organização interna seguindo o padrão `BagIt`. Com o arquivo exportado, pode-se utilizá-lo para a realização da reprodução.

Após a realização da operação de reprodução, como forma de validar essa operação,

Figura 5.5 - Fluxo de operação do experimento 1.



Fonte: Produção do autor.

fez-se a comparação dos resultados gerados em ambas as execuções, original e reprodução. O resultado dessa comparação, conforme apresentado na Figura 5.6, confirma que a operação de reprodução local consegue produzir resultados reprodutíveis. Essa confirmação é notada na Figura 5.6, quando se faz a comparação dos *checksums* da primeira linha (execução original) e a segunda linha (reprodução).

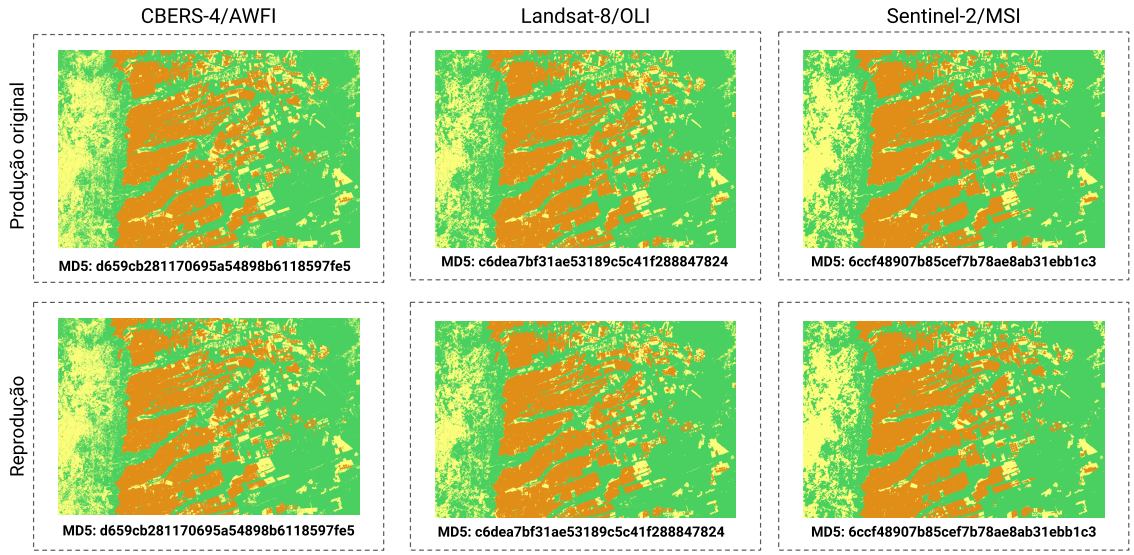
Além da realização dessas operações, uma vez que o cenário modelado emprega o uso das ferramentas do ecossistema `Storm`, tem-se que o pesquisador pode enviar suas execuções e materiais utilizados para o `Storm WS`. Quando isso é realizado, os materiais são salvos, versionados e ficam disponíveis no serviço.

Com a realização dessa ação, além de ter feito a produção de um resultado reprodutível, o pesquisador ainda consegue preservar seus códigos e dados. Isso faz com que, mesmo a `Plataforma Storm`, não tendo sido modelada para ser um repositório de dados de pesquisa, ela serve como um local intermediário, em que durante o curso da pesquisa, os pesquisadores podem preservar seus resultados. Assim, evita-se que perdas de dados e outros problemas técnicos ocorram. Isso é vantajoso por dois motivos principais:

a) Um dos aspectos que fazem com que a reprodutibilidade tenha um alto valor é a preservação. Somente um trabalho preservado poderá ser reproduzido, mesmo anos após a realização da pesquisa. Com um local intermediário, a preservação dos materiais utilizados na geração dos resultados é feita de forma fácil e organizada;

b) A ideia de preservação dos resultados, incentiva os pesquisadores a produzirem pesquisas organizadas, de modo que os elementos preservados sejam compreendidos por outros pesquisadores.

Figura 5.6 - Comparação dos mapas de LULC gerados.



Fonte: Produção do autor.

Além disso, após a finalização da pesquisa, para que os resultados tenham preservação a longo prazo garantidas, com base nas funcionalidades do `Storm WS`, eles podem então ser enviados para repositórios de dados de pesquisa.

Tais características, sendo relevantes, também foram propostas como parte de outras plataformas. Um exemplo é o trabalho feito por Wieser et al. (2021), que mesmo utilizando uma abordagem diferente, apresenta como parte de sua plataforma a mesma capacidade de preservação intermediária dos resultados que estão sendo produzidos pelos pesquisadores.

Como o desenvolvimento das pesquisas é normalmente realizado por um processo não linear, tem-se que, as capacidades de versionamento disponíveis no *Execution Compendium*, são um complemento a essa capacidade de preservação intermediária. Isso já que, caso o pesquisador tenha de executar diversas vezes uma mesma etapa, ele pode salvar o resultado de cada execução no serviço, sem preocupações relacionadas a sobrescrita e perda de resultados gerados por execuções anteriores. Com isso, todo o histórico de resultados pode ser salvo e caso necessário, versões anteriores podem ser recuperadas e utilizadas.

Com essa característica e as capacidades automatizadas de envio dos materiais salvos no `Storm WS` para repositórios de preservação de longo prazo, pode-se dizer que, a plataforma proposta, utilizando de diversos princípios vindos das atividades e práticas de Ciência Aberta, apresenta benefícios que vão além da reprodutibilidade.

5.2 Detecção de áreas desmatadas utilizando modelo de mistura e técnicas de Aprendizado de Máquina

Nesta Seção é apresentado o segundo experimento[7] desenvolvido neste trabalho. A criação desse experimento foi realizada com base no trabalho desenvolvido por Sanchez et al. (2019). Isso foi feito já que, no trabalho, os autores fazem o uso de dados espaço-temporais, enquanto apresentam um cenário que pode ser tipicamente desenvolvido por múltiplos pesquisadores, uma vez que trabalha com etapas em que diferentes áreas do conhecimento em EO podem ser aplicados. Por exemplo, os autores supracitados, fazem o uso de algoritmos de Aprendizado Profundo, junto à seleção e avaliação de amostras de LULC. Essas atividades podem ser realizadas por dois especialistas diferentes. Além disso, os autores fazem o uso dos resultados gerados para a avaliação do uso do solo, neste caso, em atividades de desmatamento. Tal avaliação pode contar com vários tipos de profissionais, indo desde pesquisadores interessados nas dinâmicas das atividades antrópicas, jornalistas que desejam entender os efeitos causados por políticas públicas, além de tomadores de decisão de forma geral.

Além disso, tem-se que, com o objetivo de modelar um cenário mais próximo da realidade, embora o experimento tenha sido criado com base na metodologia apresentada por Sanchez et al. (2019), ao decorrer do experimento, foi adicionado na metodologia o uso de uma abordagem diferente para a criação do EODC de imagens fração geradas com o Modelo Linear de Mistura Espectral (MLME). Com isso, teve-se a

[7] O código fonte completo do experimento está disponível em: `https://tinyurl.com/storm-experiment-02`

possibilidade de criar uma pesquisa em modo iterativo e não linear, utilizando as ferramentas da `Plataforma Storm`.

No restante que segue é feita a contextualização do experimento, juntamente com a apresentação dos detalhes relacionados ao cenário de pesquisa, metodologia e dados utilizados.

5.2.1 Contextualização

A região definida como Amazônia Legal brasileira representa 60% da floresta amazônica, uma das maiores florestas tropicais contínuas do planeta, sendo importante por sua biodiversidade (FEARNSIDE, 2005) e por seus serviços ecossistêmicos, como a regulação do clima e manutenção da biodiversidade (MAURANO et al., 2019). Mesmo com essa importância, a floresta vem sendo desmatada ao longo dos anos (FEARNSIDE, 2005). Buscando diminuir o desmatamento, foram criados diversos sistemas de monitoramento como o PRODES - Projeto de Monitoramento do Desmatamento na Amazônia Legal por Satélite e o DETER - Sistema de Detecção de Desmatamento em Tempo Real. Tais sistemas, atingem altos níveis de acurácia na detecção das áreas desmatadas através do emprego de técnicas de interpretação visual de imagens de SR feitas por especialistas.

Como apresentado por Shimabukuro e Ponzoni (2019), o MLME é uma técnica de transformação de dados utilizada pelos sistemas de monitoramento em seu processo de mapeamento. O MLME decompõe a mistura espectral em cada *pixel* de uma imagem de Sensoriamento Remoto (SR), identificando a influência de cada objeto que compôs a resposta espectral registrada. Para a identificação do desmatamento e degradação florestal, a aplicação do MLME visa identificar a influência dos elementos de Vegetação, Solo e Água (FELSEMBURGH, 2020). Ao aplicar essa técnica em imagens de SR tem-se a redução de dimensionalidade, em que múltiplas bandas espectrais são representadas por um conjunto de imagens fração, que indicam a proporção dos elementos modelados em cada *pixel* da imagem.

Atualmente, impulsionada pelas políticas públicas de dados abertos, a comunidade científica de EO utiliza os EODC para extração das informações na superfície terrestre (DHU et al., 2019; GIULIANI et al., 2020; FERREIRA et al., 2020). O uso dessa organização dos dados, permite aos usuários a aplicação simples e intuitiva da abordagem *time-first, space-later* (CAMARA et al., 2016). Essa abordagem junto a algoritmos de ML como *Support Vector Machines* e redes *Multi-Layer Perceptron* permite a criação de mapas de LULC (PICOLI et al., 2018).

A abordagem de interpretação visual citada, mesmo apresentando bons resultados, não consegue usufruir da grande quantidade de dados atualmente disponíveis e das técnicas de ML que podem ser empregadas para o consumo desses dados. Essa nova abordagem pode ser útil para acelerar os processos de geração dos resultados, evitando a necessidade de escolha entre a velocidade na geração ou acurácia dos produtos (SANCHEZ et al., 2019). Desta forma é definido como objetivo deste trabalho, aplicar a abordagem *time-first, space-later*, unindo EODC de imagens fração a algoritmos de ML para a geração reprodutível e semi-automática de mapas de desmatamento. Em especial, faz-se o uso do algoritmo de ML *Random Forest* para a geração dos mapas.

5.2.2 Materiais e métodos

Área de estudo

A área de estudo deste experimento está localizada na floresta tropical amazônica entre o Brasil e a Bolívia, como apresentada na Figura 5.7. No lado brasileiro, está situado o estado de Rondônia, com duas áreas indígenas (Igarapé Ribeirão e Igarapé Lage) e um conjunto de unidades de conservação. No lado boliviano, há o parque estadual de proteção ambiental na província de Federico Romàn. A área apresenta um forte contraste entre os impactos das atividades antrópicas e as florestas naturais, que possuem diferentes níveis de desmatamento (SANCHEZ et al., 2019).

Conjuntos de dados

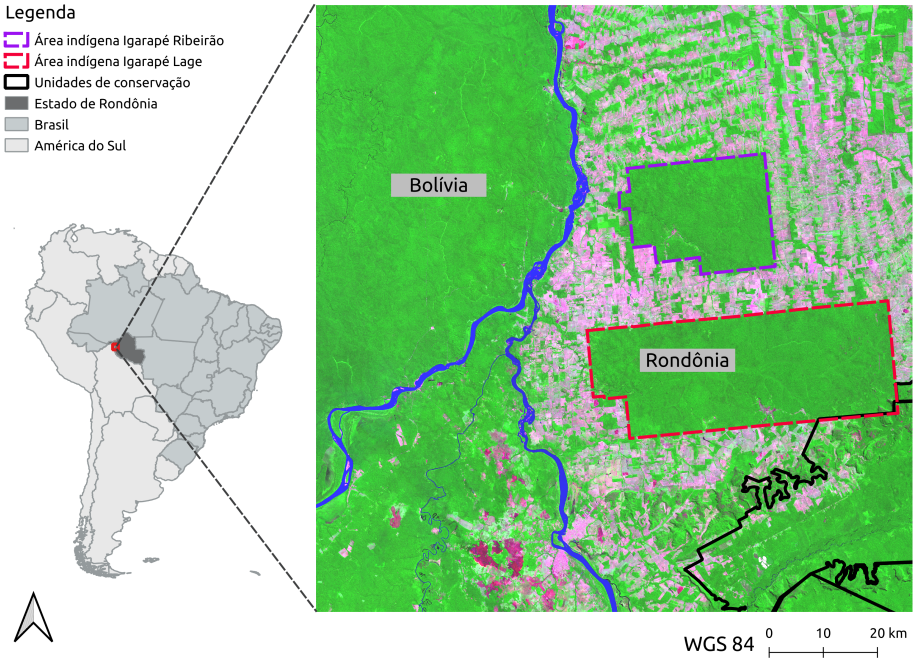
No contexto de treino e teste do algoritmo *Random Forest*, utilizado para a geração do mapa de desmatamento, fez-se o uso dos conjuntos de amostras gerados por Sanchez et al. (2019). O conjunto de treinamento é formado por 481 amostras, divididas nas classes Desmatamento (42), Floresta (144) e Não-Floresta (295). O conjunto de teste possui 240 amostras, divididas igualmente entre as três classes. As amostras de ambos os conjuntos de dados foram coletadas por especialistas em SR com o auxílio de imagens de alta resolução, considerando o ano PRODES 2019 (2018/07 a 2019/09).

Para a criação dos cubos de imagens fração foi realizada a extração de cenas do cubo de dados Landsat-8/OLI[8] gerados pelo Projeto BDC. As imagens deste cubo são prontas para análise (COMMITTEE ON EARTH OBSERVATION SATELLITES, 2021),

[8]Disponível em: <https://brazildatacube.dpi.inpe.br/stac/collections/LC8_30_16D_STK-1>. Acesso em: 16/11/2022

sendo disponibilizadas em nível de reflectância de superfície, com resolução espacial de 30 metros e composição temporal de 16 dias utilizando a abordagem *Least Cloud Cover First* para a seleção de *pixels* com menos nuvens e sombra de nuvens nesse período (FERREIRA et al., 2020). As cenas foram extraídas considerando os critérios de cobertura de nuvem e extensão temporal. Assim, foram consideradas cenas para o mesmo período das coletas de amostras, com menos de 70% de cobertura de nuvem na área de estudo, tendo um total de 13 cenas.

Figura 5.7 - Esquerda: Área de estudo. Direita: Cena da área de estudo extraída do cubo de dados Landsat-8/OLI (07/2019).



Fonte: Produção do autor.

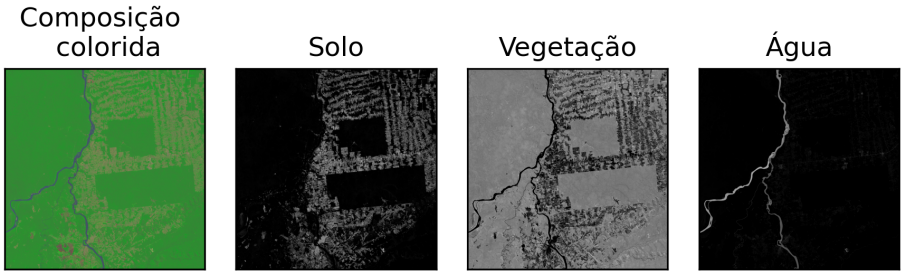
Modelo linear de mistura espectral

As imagens de SR são geradas através do imageamento da superfície terrestre realizada por sensores que estão a bordo de satélites orbitais. Conforme apresentado por Shimabukuro e Ponzoni (2019), durante este processo, as informações do espectro eletromagnético registradas nos *pixels* dessas imagens podem representar a mistura espectral de múltiplos materiais ou objetos que por conta da resolução espacial do sensor imageador são registrados juntos em um único *pixel*. A identificação

da influência que cada elemento imageado tem na informação que está registrada em cada *pixel* pode ser utilizada como base para diversas atividades como a melhoria nos processo de mapeamento de LULC (YANG et al., 2017) e o monitoramento do desmatamento (SCHULTZ et al., 2016; BULLOCK et al., 2020; SOUZA et al., 2013) e queimadas (QUINTANO et al., 2013).

O MLME é uma técnica que pode ser utilizada para modelar a influência dos diferentes elementos nas respostas espectrais registradas por um sensor em cada um dos *pixels* de uma imagem (SHIMABUKURO; PONZONI, 2019). Desta forma, ao aplicar tal técnica, faz-se a geração de uma imagem fração, que contém em cada *pixel* a relação de influência (fração) dos diferentes elementos que compõem sua informação espectral. A Figura 5.8 apresenta um exemplo de três imagens fração, sendo essas modeladas com o MLME para a identificação da influência dos elementos de `Solo`, `Vegetação` e `Água`.

Figura 5.8 - Exemplo de imagens fração para a área de estudo.



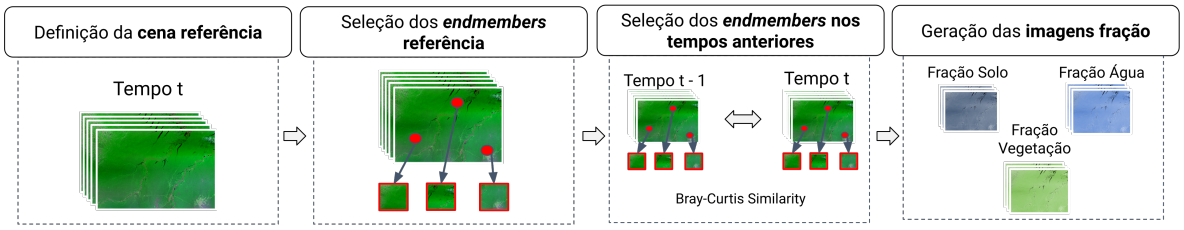
Fonte: Produção do autor.

No exemplo acima, as influências são apresentadas nos *pixels* através de valores que variam de 0 a 1. Note que, as áreas que contém influência de vegetação na composição colorida são mais claras na fração de vegetação, com valores próximos a 1. O comportamento é análogo para as frações de solo e água.

Um ponto importante sobre o uso do MLME é a definição dos comportamentos espectrais de referência (QUINTANO et al., 2011), os chamados *endmembers*, que precisam ser definidos para cada cena em que o MLME será aplicado. Esses são utilizados no método para modelar as influências dos elementos através da solução de sistemas

de equações lineares (SHIMABUKURO; PONZONI, 2019). Existem diversas técnicas que podem ser utilizadas para a definição dos *endmembers*, indo desde a seleção manual ou automática (SHIMABUKURO; PONZONI, 2019; ENDSLEY, 2019) até a aplicação de técnicas de definição global que criam *endmembers* generalizados no tempo e espaço (SOUSA; SMALL, 2017). Neste trabalho, diferente do apresentado por Sanchez et al. (2019), a seleção dos *endmembers* para cada cena Landsat-8/OLI foi feita utilizando uma abordagem semi-automática, a qual tem suas etapas apresentas na Figura 5.9.

Figura 5.9 - Fluxo de definição dos *endmembers*.



Fonte: Produção do autor.

A definição e uso do processo apresentado na Figura 5.9 foi aplicada neste trabalho com o objetivo de reduzir passos manuais e complicações com técnicas avançadas. De maneira geral, como apresentado, o fluxo consiste em realizar a definição manual de um conjunto de *endmembers* de referência em uma cena no instante de tempo T, e então, de maneira automática, utilizando de similaridade espectral com esses *endmembers* de referência, todos os *endmembers* nas cenas dos tempos T anteriores $(T-1, T-2, ..., T-N)$ são definidos. A similaridade é definida através do uso da métrica de dissimilaridade espectral definida por Bray e Curtis (1957).

Classificação dos cubos de dados de imagens fração

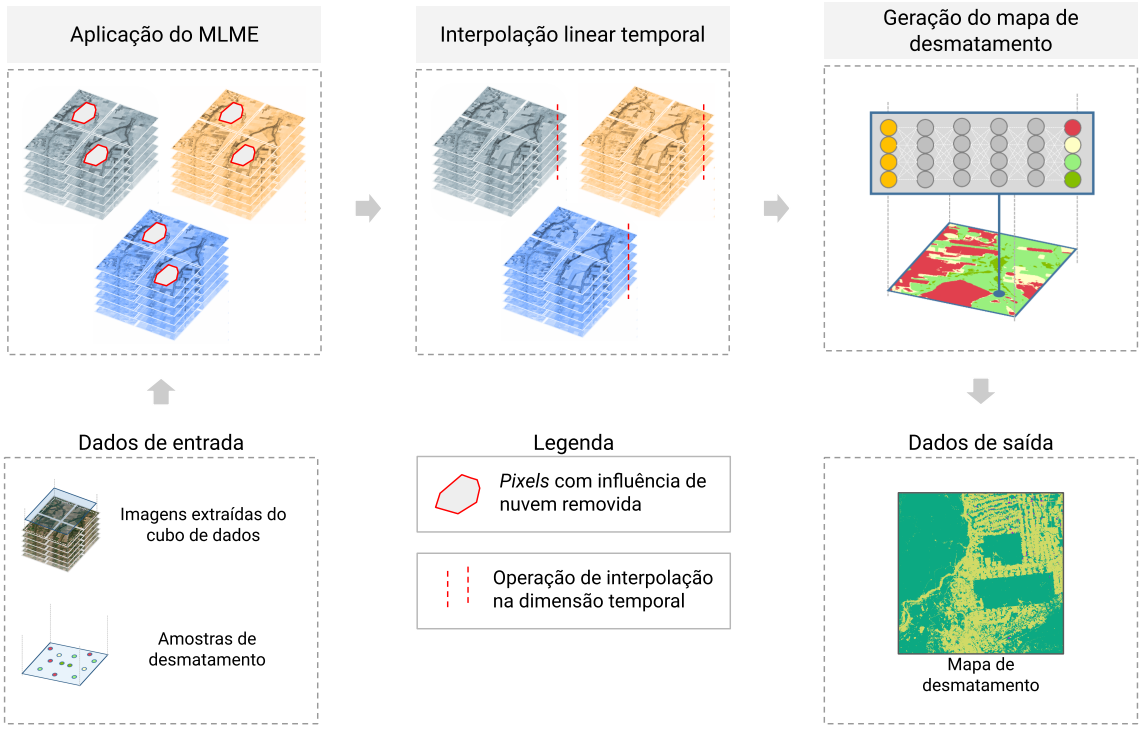
Neste trabalho, a geração dos mapas de desmatamento foi realizada através da classificação de séries temporais extraídas das cenas Landsat-8/OLI selecionadas. Conforme mencionado, o algoritmo utilizado foi o *Random Forest*. Para que esse algoritmo pudesse ser utilizado para a classificação de séries temporais, foi aplicado a forma de organização dos dados especificadas por Simoes et al. (2021) e detalhados

na Seção 2.5.

5.2.3 Metodologia

O desenvolvimento do presente trabalho foi realizado utilizando a metodologia apresentada na Figura 5.10. Inicialmente é feita a aplicação do MLME em cada uma das 13 cenas Landsat-8/OLI selecionadas para a geração das imagens fração de Solo, Vegetação e Água. Nas imagens fração resultantes é feita a remoção de nuvens e em seguida, essas são empilhadas no tempo, formando os cubos de imagens fração. Para preencher as regiões do cubo removidas por presença de nuvens, fez-se uma interpolação linear temporal. Por fim, os cubos de imagens fração foram utilizados para a geração do mapa de desmatamento. Deve-se notar que, para a melhora dos resultados da classificação, fez-se a aplicação de um filtro bayesiano (SIMOES et al., 2021) com janela *5x5* de suavização.

Figura 5.10 - Metodologia de desenvolvimento do experimento 2.

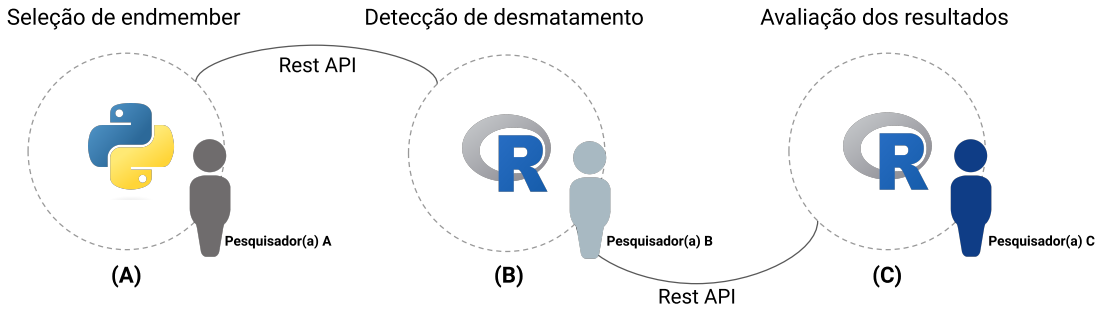


Fonte: Produção do autor.

5.2.4 Cenário de pesquisa

O cenário de pesquisa do presente experimento, conforme ilustrado na Figura 5.11, faz a divisão da metodologia em três partes principais: (i) Seleção de *endmember*; (ii) Detecção de desmatamento; (iii) Avaliação dos resultados.

Figura 5.11 - Cenário de pesquisa do experimento 2.



Fonte: Produção do autor.

Para que as funcionalidades da plataforma pudessem ser utilizadas e evidenciadas, nesse experimento, tem-se que cada etapa representa de forma fictícia um pesquisador contribuindo com a produção dos resultados da pesquisa. Por conta disso, as tecnologias variam, sendo que na etapa inicial faz-se o uso da linguagem de programação Python, enquanto nas demais, faz-se o uso da linguagem R. Para a troca dos resultados entre as etapas, faz-se o uso do `Storm Workbench` e as funcionalidades providas pelo `Storm WS`. Em todas as etapas, conforme mencionado anteriormente, faz-se a criação de um RC que mantém o experimento organizado.

O objetivo do presente experimento foi a construção de um cenário em que a complexidade das pesquisas reais pudesse ser representada. Para isso, fez-se, em cada etapa, a utilização de um conjunto amplo e diversificado de pacotes de *software* auxiliares. Por exemplo, na seleção de *endmembers*, tem-se a aplicação da metodologia apresentada na Figura 5.9. Para que essa fosse utilizada de forma otimizada, o processamento realizado possui operação paralela. Isso é feito com o auxílio da biblioteca `joblib`[9]. Já na etapa de detecção de desmatamento fez-se o uso conjunto dos pacotes R `RStoolbox` (LEUTNER et al., 2019) e `Snow` (TIERNEY et al., 2018), de

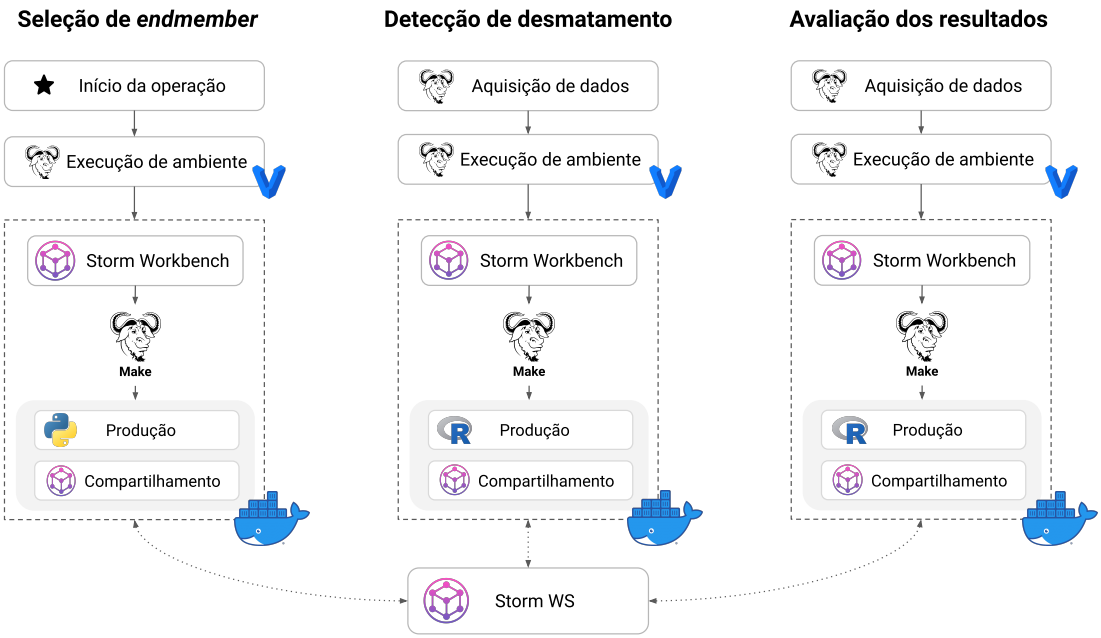
[9]Disponível em: <`https://joblib.readthedocs.io/en/latest/`>. Acesso em: 14/11/2022

modo que o modelo linear de mistura espectral fosse aplicado de forma paralela. Por fim, para a manipulação dos cubos de dados, faz-se o uso do pacote R `sits` (SIMOES et al., 2021).

5.2.5 Resultados e discussões

O desenvolvimento das etapas apresentadas na Figura 5.11, teve como resultado o fluxo de processamento ilustrado na Figura 5.12. Em cada etapa, primeiro fez-se a configuração da máquina virtual utilizada como ambiente base. Em seguida, no ambiente `Docker` criado especialmente para cada uma das etapas, com suas dependências e configurações próprias, fez-se a execução da operação subjacente à etapa trabalhada. Para essa execução, a ferramenta `Storm Workbench` foi utilizada. Ao final de cada operação, fez-se o uso das funcionalidades de compartilhamento da ferramenta para que os dados e metadados coletados sobre a execução pudessem ser enviados ao `Storm WS`. Nas etapas em que havia a dependência de informações geradas em etapas anteriores, antes da execução, fez-se a aquisição dos dados salvos no `Storm WS`.

Figura 5.12 - Fluxo de execução utilizado no experimento 2.

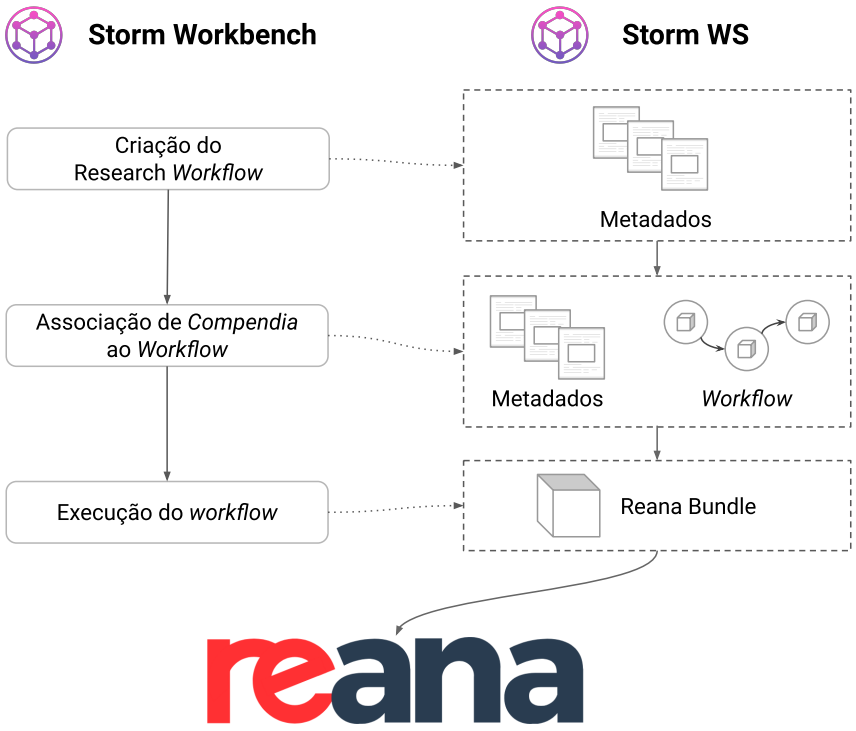


Fonte: Produção do autor.

Após a execução do fluxo da Figura 5.12, no `Storm WS`, tinha-se um *Execution*

Compendium para cada uma das etapas. Isso possibilitou a criação de um *Research Workflow* no serviço. Então, após a criação do *workflow* e a associação dos *Execution Compendium* a ele, fez-se a reprodução dos resultados. O fluxo da operação descrita é ilustrado na Figura 5.13.

Figura 5.13 - Fluxo de reprodução do experimento 2.



Fonte: Produção do autor.

Após a realização da operação de reprodução, para verificar a consistência dos resultados e a capacidade de reprodução fornecida pelo serviço de reprodutibilidade `Storm WS`, fizeram-se algumas análises comparativas entre os resultados gerados na execução original e na reprodução.

A primeira comparação realizada foi de equivalência entre os *checksums* dos arquivos gerados na execução original e na reprodução. Para essa comparação, uma vez que em cada etapa tem-se dois diretórios com os resultados, original e reprodução, fez-se testes de equivalência em nível de diretório. Nessa comparação, o objetivo foi verificar se todo o conteúdo disponível no diretório de resultados originais era o mesmo que o

diretório da reprodução. Então, com o uso da ferramenta `is.equal`[10], desenvolvida neste trabalho, essa comparação foi realizada. Essa ferramenta, faz o cálculo do *checksum* de diretórios. Para isso, faz-se o somatório dos *bytes* de cada arquivo no diretório, e a partir desse valor é calculado o *checksum*. Com essa abordagem, detalhes como a estrutura de diretório ou nome dos arquivos não são considerados para a determinação de equivalência entre dois diretórios.

Ao aplicar a abordagem de comparação em nível de diretório, conforme apresentado na Tabela 5.1, pode-se confirmar a obtenção dos mesmos resultados em ambas as execuções, original e reprodução. Desta forma, tem-se que o uso da `Plataforma Storm`, auxilia na produção de resultados reprodutíveis, mesmo quando a pesquisa está sendo desenvolvida em cenários colaborativos.

Tabela 5.1 - Tabela de comparação entre dados originais e dados da reprodução.

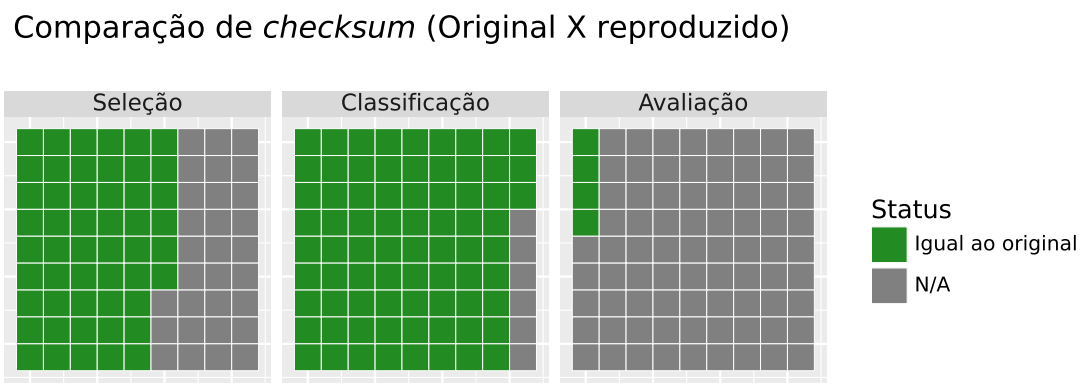
Algoritmos	**Etapa**	**Status**
MD5/SHA256/SHA512	Seleção de *endmembers*	*Conteúdos iguais*
MD5/SHA256/SHA512	Detecção de desmatamento	*Conteúdos iguais*
MD5/SHA256/SHA512	Avaliação dos resultados	*Conteúdos iguais*

Fonte: Produção do autor.

Esses resultados indicam que todos os arquivos nos diretórios de suas respectivas etapas de processamento são iguais. De forma complementar a esse resultado, tem-se na Figura 5.14, a comparação individual entre o *checksum* dos arquivos originais e reproduzidos. Cada um dos blocos representa um arquivo gerado durante a execução do experimento. Para os arquivos em que o *checksum* MD5 era igual na execução original e na reprodução, definiu-se verde. Aqueles que diferem são colocados com vermelho. Assim, como pode-se notar, o resultado corrobora com a análise realizada anteriormente.

[10]Disponível em: <https://github.com/storm-platform/is.equal>. Acesso em: 24/11/2022

Figura 5.14 - Comparação de igualdade entre os arquivos originais e reproduzidos.

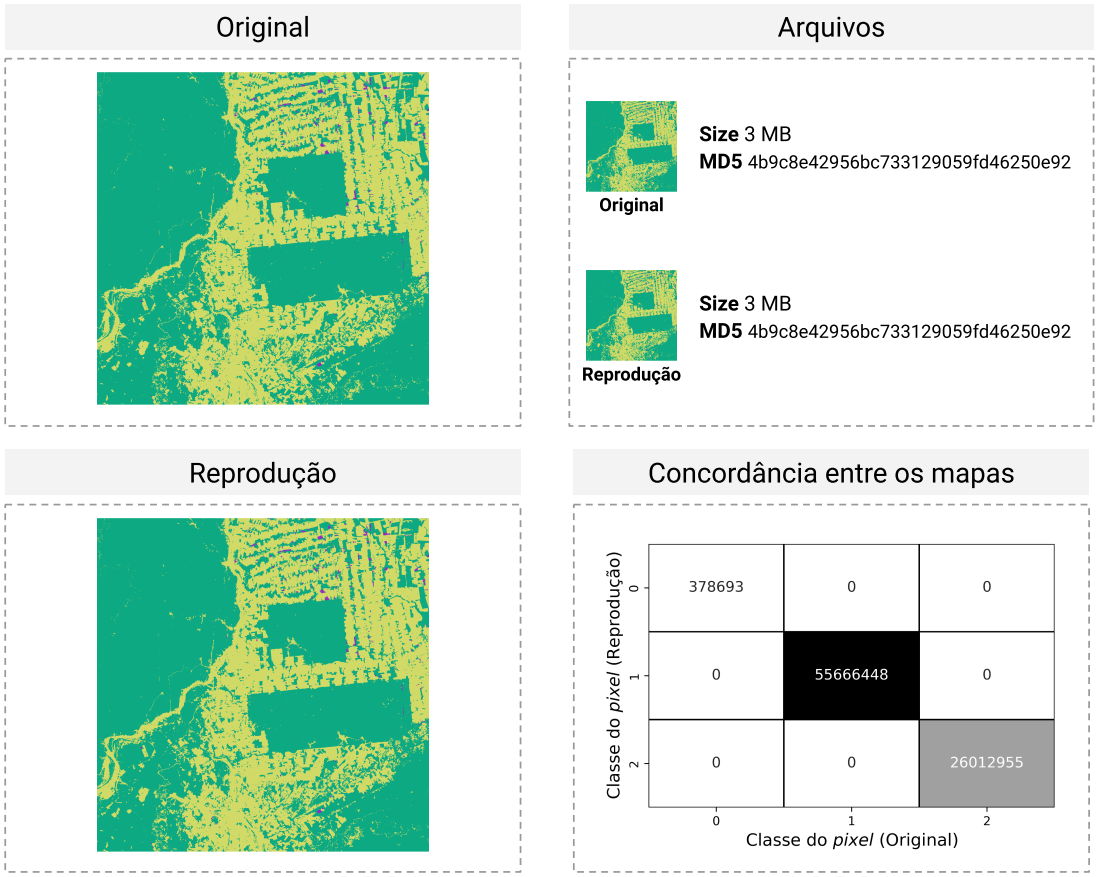


Fonte: Produção do autor.

Para tornar essa equivalência entre os resultados mais visual, tem-se na Figura 5.15, a comparação entre cada um dos mapas de desmatamento gerados, bem como seus respectivos *checksums*. Dessa forma, pode-se observar que os resultados são idênticos, não apresentando nenhuma variação numérica. Tal resultado é condizente com a análise de *checksum* feita anteriormente.

As análises apresentadas anteriormente são formas diferentes de verificar e entender que, ao ser utilizado em um cenário colaborativo, a abordagem utilizada pelas ferramentas implementadas na `Plataforma Storm`, tornou possível a reprodução dos resultados. No entanto, essa abordagem possui limitações que devem ser explicitadas. Para que isso seja feito, primeiro, deve-se considerar o cenário de pesquisa utilizado, o qual é ilustrado na Figura 5.12. Para cada uma das etapas nesse fluxo, nota-se que há pontos em que o ecossistema de ferramentas proposto, pode não auxiliar na reprodutibilidade:

Figura 5.15 - Comparação dos resultados (Original X Reprodução).



Fonte: Produção do autor.

Seleção de *endmember*: Na seleção de *endmember*, tem-se como operação base, a aquisição de dados. Como definição, nesse experimento essa operação foi feita com base nos serviços de disseminação e acesso a dados do projeto BDC. Com isso, embora os dados sejam mais fáceis de serem acessados, caso o serviço não esteja disponível, a reprodutibilidade será comprometida.

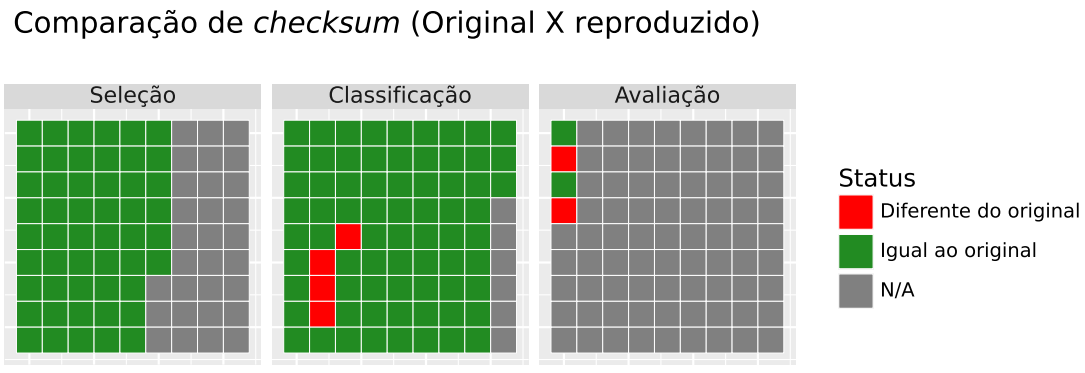
Para esses casos, o uso da `Plataforma Storm` se dá pela aplicação das funcionalidades de versionamento. Com o uso dessa capacidade, os pesquisadores têm a possibilidade de salvar, no serviço de reprodutibilidade, diferentes versões de uma mesma etapa de sua pesquisa. Desta forma, quando há o uso de um serviço para acesso aos dados, o pesquisador pode trabalhar com duas versões de suas etapas de pesquisa. Na primeira versão, a execução é feita normalmente, com a utilização dos devidos serviços para acesso aos dados. Já na segunda versão, o que o pesquisador pode fazer é disponibilizar um exemplo de execução com um pequeno conjunto de dados local.

Desta forma, caso o serviço utilizado para aquisição dos dados não esteja disponível, ainda estará disponível um exemplo executável, que permite o entendimento do que foi realizado na pesquisa. Tal ação, não cria uma pesquisa reprodutível, mas, auxilia aqueles que desejam entender como a pesquisa funciona, mesmo quando não há disponível todos os recursos utilizados na execução original.

Detecção de desmatamento: Nessa etapa, além da necessidade de uso de dados disponibilizados pelos serviços do projeto BDC, há também influências internas na operação que podem causar a irreprodutibilidade dos resultados.

Para a geração dos mapas de desmatamento, conforme mencionado, faz-se o uso do algoritmo de ML *Random Forest*. Durante as etapas de treinamento desse algoritmo, têm-se estocásticas. Para essas, caso o pesquisador não define uma semente de valores pseudo-aleatórios, possivelmente, os resultados não serão reprodutíveis. Veja que essa ação não tem relação direta com a capacidade da reexecução da operação fornecida pelas ferramentas `Storm`. Isso será possível, no entanto, o resultado gerado será diferente. Para exemplificar o efeito da não definição dessas sementes para cenários como esse, tem-se na Figura 5.16, a ilustração da equivalência entre os arquivos de duas execuções da etapa de detecção de desmatamento. Como pode-se notar, os arquivos gerados como resultados diferem em nível de *checksum*, o que, consequentemente, representa que os resultados não são iguais.

Figura 5.16 - Comparação de igualdade entre os arquivos originais e reproduzidos quando controle de semente de valores pseudo-aleatórios não é definido.



Fonte: Produção do autor.

Isso faz com que, mesmo que a `Plataforma Storm` auxilie na reexecução, como foi

feito durante a geração da Figura 5.16, é tido que as escolhas tecnológicas feitas na implementação demonstrativa não conseguem lidar com características específicas de ferramentas ou linguagens de programação, fazendo com que resultados irreprodutíveis sejam gerados quando tais características não são tradas pelo usuário. Possivelmente, com a implementação de outras ferramentas para a geração de *Execution Compendium*, tal problema pode ser resolvido. Por exemplo, o pacote R `targets` (LANDAU, 2021), é criado especialmente para a produção de *pipelines* reprodutíveis na linguagem R. Nesse caso, a ferramenta consegue lidar com características específicas do ambiente da linguagem. Tal pacote, poderia ser a base para a construção de uma ferramenta que faz a geração de *Execution Compendium* específicos para ambientes R. Com isso, o controle e a garantia de reprodutibilidade poderiam ser realizados, mesmo quando o usuário não toma certos cuidados em seu código.

Ademais, no que diz respeito a implementação demonstrativa da `Plataforma Storm`, tal problema não limita sua utilização. No entanto, sua ampla adoção possivelmente teria de ser feita em conjunto com boas práticas para evitar que o não controle de características específicas de ferramentas e linguagens de programação causem a geração de resultados irreprodutíveis.

Avaliação dos resultados: Nessa etapa, tem-se apenas a realização de operações determinísticas, com o uso de dados locais já gerados nas etapas anteriores. Assim, não há muitas barreiras para a reprodutibilidade quando se faz o uso das ferramentas da implementação demonstrativa `Plataforma Storm`.

6 CONSIDERAÇÕES FINAIS

Neste trabalho foi proposto o projeto da `Storm`, uma plataforma para suporte ao desenvolvimento colaborativo de pesquisas reprodutíveis. Nesse projeto, adotando uma abordagem modular, fez-se a especificação de funcionalidades que suportam as práticas de produção reprodutível e de replicação de resultados científicos. Para a avaliação parcial da viabilidade de uso e adoção desse projeto de plataforma em cenários reais, fez-se a criação de uma implementação demonstrativa, que implementa as principais funcionalidades relativas à produção reprodutível de pesquisas propostas no projeto da plataforma. Partindo dessa implementação, fez-se a realização de experimentos que colocassem à prova a efetividade das abordagens especificadas no projeto proposto na produção de resultados reprodutíveis. Ao todo se fez a realização de dois experimentos, os quais foram desenvolvidos com base em pesquisas reais, neste caso, a aplicação de cubos de dados para a criação de mapas de LULC e desmatamento.

No primeiro experimento, observou-se que a abordagem base da plataforma, disponível na implementação realizada, conseguiu auxiliar o desenvolvimento de uma pesquisa reprodutível. Além disso, notou-se que a abordagem não intrusiva adotada pela plataforma facilitou a utilização das ferramentas para a produção de resultados reprodutíveis. Também tem-se que, com base nas capacidades do serviço de reprodutibilidade implementado pela plataforma, notou-se a possibilidade de preservação dos resultados gerados, no mínimo, durante o curso da pesquisa. Não sendo uma plataforma de preservação, seus serviços ainda oferecem a possibilidade de enviar os resultados gerados para plataformas de preservação de longo prazo. Essas ações automatizadas são um convite aos pesquisadores para a adoção de práticas abertas e reprodutíveis com base no uso das funcionalidades da `Plataforma Storm`.

No segundo experimento, com o uso de um cenário fictício em que vários pesquisadores foram representados nas etapas de desenvolvimento da pesquisa, pode-se observar que as funcionalidades da plataforma conseguiram auxiliar o desenvolvimento colaborativo de pesquisas reprodutíveis. Nesse caso, notou-se que o uso do serviço de reprodutibilidade facilitou a integração do trabalho de diversos pesquisadores, tornando transparente a complexidade envolvida na troca de materiais entre eles. Além disso, notou-se que para certas etapas realizadas no desenvolvimento do experimento, como a produção de resultados com métodos estocásticos, por conta das escolhas tecnológicas, teve-se a obtenção de resultados irreprodutíveis. No entanto, para esses, entende-se que, como são problemas relacionados à implementação

realizada, não afetam a consistência das abordagens apresentadas no projeto da plataforma.

Desta forma, com base nos resultados gerados nesta dissertação, pode-se concluir que, o projeto de plataforma proposto, juntamente às abordagens nele aplicadas para a produção de pesquisas reprodutíveis em cenários colaborativos, tem potencial para ser utilizado na produção de pesquisas reais. Além disso, nota-se que, as escolhas tecnológicas feitas para a implementação demonstrativa, produziram resultados promissores, que podem ser considerados em projetos completos de implementação da `Plataforma Storm`.

6.1 Trabalhos futuros

Tendo como base a implementação demonstrativa realizada, juntamente à conclusão desta Dissertação, há trabalhos futuros que são passíveis de serem realizados:

- Criação de uma implementação completa do projeto da `Plataforma Storm`, integrando-o com mecanismos institucionais que possibilitem sua operacionalização, como infraestruturas de processamento de dados distribuídos, sistemas de arquivos distribuídos e sistemas federados, que possibilitem a colaboração multi-institucional de forma facilitada;
- Avaliação da abordagem com o uso de grandes volumes de dados manipulados diretamente pela ferramenta, uma vez que, há cenários em que tais ações são necessárias;
- Implementação de outras ferramentas, complementares ao `Storm Workbench`, que possibilitem a apresentação das capacidades de múltiplas representações de um *Execution Compendium*;
- Integração da `Plataforma Storm` com outros serviços externos de preservação e execução dos resultados da pesquisa;
- Integração da `Plataforma Storm` com outras plataformas de reprodutibilidade disponíveis, como no `o2r`;
- Adição de funcionalidades à implementação demonstrativa, que torne possível o uso de *scripts* CWL na representação do *Research Workflow*.

REFERÊNCIAS BIBLIOGRÁFICAS

ADDIS, M. **Which checksum algorithm should I use?** Digital Preservation Coalition, dez. 2020. Disponível em: <https://doi.org/10.7207/twgn20-12>. 73

ALLISON, D. B.; BROWN, A. W.; GEORGE, B. J.; KAISER, K. A. Reproducibility: a tragedy of errors. **Nature**, v. 530, n. 7588, p. 27–29, fev. 2016. Disponível em: <https://doi.org/10.1038/530027a>. 1

ALSTON, J. M.; RICK, J. A. A beginner's guide to conducting reproducible research. **The Bulletin of the Ecological Society of America**, v. 102, n. 2, jan. 2021. Disponível em: <https://doi.org/10.1002/bes2.1801>. 45

AMORIM, R. C.; CASTRO, J. A.; ROCHA DA SILVA, J.; RIBEIRO, C. A comparison of research data management platforms: architecture, flexible metadata and interoperability. **Universal Access in the Information Society**, v. 16, n. 4, p. 851–862, 2017. ISSN 16155297. 26, 27

AMSTUTZ, P.; CRUSOE, M. R.; TIJANIć, N.; CHAPMAN, B.; CHILTON, J.; HEUER, M.; KARTASHOV, A.; KERN, J.; LEEHR, D.; MéNAGER, H.; NEDELJKOVICH, M.; SCALES, M.; SOILAND-REYES, S.; STOJANOVIC, L. Common workflow language, v1.0. Figshare, 2016. Disponível em: <https://doi.org/10.6084/m9.figshare.3115156.v2>. 20

APPEL, M.; PEBESMA, E. On-demand processing of data cubes from satellite image collections with the gdalcubes library. **Data**, v. 4, n. 3, p. 92, jun. 2019. Disponível em: <https://doi.org/10.3390/data4030092>. 39, 40, 102

ATKINSON, M.; GESING, S.; MONTAGNAT, J.; TAYLOR, I. Scientific workflows: past, present and future. **Future Generation Computer Systems**, v. 75, p. 216–227, 2017. ISSN 0167739X. Disponível em: <http://dx.doi.org/10.1016/j.future.2017.05.041>. 20

BAKER, M. 1, 500 scientists lift the lid on reproducibility. **Nature**, v. 533, n. 7604, p. 452–454, maio 2016. Disponível em: <https://doi.org/10.1038/533452a>. 2

BARBA, L. A. The hard road to reproducibility. **Science**, v. 354, n. 6308, p. 142–142, out. 2016. Disponível em: <https://doi.org/10.1126/science.354.6308.142>. 9

_____. Terminologies for reproducible research. **CoRR**, abs/1802.03311, 2018. Disponível em: <http://arxiv.org/abs/1802.03311>. 7

_____. Praxis of reproducible computational science. **Computing in Science & Engineering**, v. 21, n. 1, p. 73–78, jan. 2019. Disponível em: <https://doi.org/10.1109/mcse.2018.2881905>. 22, 24

BARGA, R.; GANNON, D. **Workflows for e-Science**. Springer, 2007. Disponível em: <https://doi.org/10.1007/978-1-84628-757-2>. 19

BAUMER, B.; UDWIN, D. R markdown. **Wiley Interdisciplinary Reviews: Computational Statistics**, v. 7, n. 3, p. 167–177, fev. 2015. Disponível em: <https://doi.org/10.1002/wics.1348>. 13

BEAULIEU-JONES, B. K.; GREENE, C. S. Reproducibility of computational workflows is automated using continuous analysis. **Nature Biotechnology**, v. 35, n. 4, p. 342–346, mar. 2017. Disponível em: <https://doi.org/10.1038/nbt.3780>. 15, 16

BENEDICIC, L.; CRUZ, F. A.; MADONNA, A.; MARIOTTI, K. Sarus: highly scalable docker containers for HPC systems. In: WEILAND, M.; JUCKELAND, G.; ALAM, S.; JAGODE, H. (Ed.). **High performance computing**. Springer, 2019. p. 46–60. Disponível em: <https://doi.org/10.1007/978-3-030-34356-9_5>. 17

BEZJAK, S.; CLYBURNE-SHERIN, A.; CONZETT, P.; FERNANDES, P.; GöRöGH, E.; HELBIG, K.; KRAMER, B.; LABASTIDA, I.; NIEMEYER, K.; PSOMOPOULOS, F.; ROSS-HELLAUER, T.; SCHNEIDER, R.; TENNANT, J.; VERBAKEL, E.; BRINKEN, H.; HELLER, L. **Open Science training handbook**. Zenodo, 2018. Disponível em: <https://zenodo.org/record/1212496>. 10

BHATT, A.; VALENTIC, T.; REIMER, A.; LAMARCHE, L.; REYES, P.; COSGROVE, R. Reproducible software environment: a tool enabling computational reproducibility in geospace sciences and facilitating collaboration. **Journal of Space Weather and Space Climate**, v. 10, p. 12, 2020. Disponível em: <https://doi.org/10.1051/swsc/2020011>. 1

BLISCHAK, J. D.; CARBONETTO, P.; STEPHENS, M. Creating and sharing reproducible research code the workflowr way. **F1000Research**, v. 8, p. 1749, out. 2019. Disponível em: <https://doi.org/10.12688/f1000research.20843.1>. 11, 19, 22

BLOCH, J. How to design a good API and why it matters. In: ACM SIGPLAN CONFERENCE ON OBJECT-ORIENTED PROGRAMMING SYSTEMS, LANGUAGES, AND APPLICATIONS, 21., 2006. **Proceedings...** ACM Press, 2006. Disponível em: <https://doi.org/10.1145/1176617.1176622>. 65

BOETTIGER, C. An introduction to docker for reproducible research. **ACM SIGOPS Operating Systems Review**, v. 49, n. 1, p. 71–79, jan. 2015. Disponível em: <https://doi.org/10.1145/2723872.2723882>. 17, 18

BOETTIGER, C.; EDDELBUETTEL, D. An introduction to rocker: docker containers for R. **The R Journal**, v. 9, n. 2, p. 527–536, 2017. Disponível em: <https://doi.org/10.32614/RJ-2017-065>. 18

BOETTIGER, C.; MANGEL, M.; MUNCH, S. **Nonparametric-Bayes: V0.2.0**. Zenodo, 2014. Disponível em: <https://zenodo.org/record/13794>. 25

BOETTIGER, C.; MEMARZADEH, M. **boettiger-lab/pomdp-intro: Resolving the measurement uncertainty paradox in ecological management**. Zenodo, 2018. Disponível em: <https://zenodo.org/record/2528507>. 25

BONTEMPS, C.; OROZCO, V. **Toward a FAIR Reproducible Research**. nov 2020. Disponível em: <https://www.tse-fr.eu/publications/toward-fair-reproducible-research>. 25

BORGES, E. F.; SANO, E. E. Caracterização fenológica da cobertura vegetal do oeste da Bahia a partir de séries temporais de EVI do sensor MODIS. **Revista Brasileira de Cartografia**, n. 66, p. 1265–1280, 2014. 103

BRAY, J. R.; CURTIS, J. T. An ordination of the upland forest communities of southern wisconsin. **Ecological Monographs**, v. 27, n. 4, p. 325–349, out. 1957. Disponível em: <https://doi.org/10.2307/1942268>. 114

BRIGNELL, B. **Design principles**. 2017. Disponível em: <https://principles.design/>. 61

BRINCKMAN, A.; CHARD, K.; GAFFNEY, N.; HATEGAN, M.; JONES, M. B.; KOWALIK, K.; KULASEKARAN, S.; LUDäSCHER, B.; MECUM, B. D.; NABRZYSKI, J.; STODDEN, V.; TAYLOR, I. J.; TURK, M. J.; TURNER, K. Computing environments for reproducibility: capturing the “whole tale”. **Future Generation Computer Systems**, v. 94, p. 854–867, maio 2019. Disponível em: <https://doi.org/10.1016/j.future.2017.12.029>. 3, 37, 38

BRUNSDON, C.; COMBER, A. Opening practice: supporting reproducibility and critical spatial data science. **Journal of Geographical Systems**, n. 0123456789, 2020. ISSN 14355949. Disponível em: <https://doi.org/10.1007/s10109-020-00334-2>. 1

BRYAN, J.; HESTER, J. **What they forgot to teach you about R**. sep 2018. Disponível em: <https://rstats.wtf/>. 19

BRYAN, J.; TAS, S. .; HESTER, J. **Happy git and gitHub for the user**. dec 2016. Disponível em: <https://happygitwithr.com/>. 24

BULLOCK, E. L.; WOODCOCK, C. E.; OLOFSSON, P. Monitoring tropical forest degradation using spectral unmixing and landsat time series analysis. **Remote Sensing of Environment**, v. 238, p. 110968, mar. 2020. Disponível em: <https://doi.org/10.1016/j.rse.2018.11.011>. 113

BURNS, R.; KLEISSAS, D.; VOGELSTEIN, J.; VOGELSTEIN, J.; WHITEHOUSE, T. **Gigantum – a simple way to create and share reproducible data science and research**. jul 2018. Disponível em: <https://elifesciences.org/labs/bdbeac92/gigantum-a-simple-way-to-create-and-share-reproducible-data-science-and-research>. 3, 35

CAMARA, G.; ASSIS, L. F.; RIBEIRO, G.; FERREIRA, K. R.; LLAPA, E.; VINHAS, L. Big earth observation data analytics: matching requirements to system architectures. In: ACM SIGSPATIAL INTERNATIONAL WORKSHOP ON ANALYTICS FOR BIG GEOSPATIAL DATA, 5., 2016. **Proceedings...** ACM Press, 2016. Disponível em: <https://doi.org/10.1145/3006386.3006393>. 102, 110

CAMARA, G.; EGENHOFER, M. J.; FERREIRA, K.; ANDRADE, P.; QUEIROZ, G.; SANCHEZ, A.; JONES, J.; VINHAS, L. Fields as a generic data type for big spatial data. **Geographic Information Science**, p. in press, 2014. ISSN 16113349. Disponível em: <https://doi.org/10.1007/978-3-319-11593-1_11>. 39

CARLOS, F. M.; GOMES, V. C. F.; QUEIROZ, G. R. de; SOUZA, F. C. de; FERREIRA, K. R.; SANTOS, R. Integrating open data cube and brazil data cube platforms for land use and cover classifications. **Revista Brasileira de Cartografia**, v. 73, n. 4, p. 1036–1047, out. 2021. Disponível em: <https://doi.org/10.14393/rbcv73n4-60387>. 6

CARLOS, F. M.; GOMES, V. C. F. G.; QUEIROZ, G. R.; FERREIRA, K. R.; RAFAEL, S. Integração dos ambientes brazil data cube e open data cube. In: GEOINFO, 21., 2020. **Proceedings...** INPE, 2020. p. 168–173. Disponível em: <http://mtc-m16c.sid.inpe.br/col/sid.inpe.br/mtc-m16c/2020/12.15.12.11/doc/s2.pdf>. 5

CARPENTER, C. M.; FRANK, D. N.; WILLIAMSON, K.; ARBET, J.; WAGNER, B. D.; KECHRIS, K.; KROEHL, M. E. tidyMicro: a pipeline for microbiome data analysis and visualization using the tidyverse in r. **BMC Bioinformatics**, v. 22, n. 1, fev. 2021. Disponível em: <https://doi.org/10.1186/s12859-021-03967-2>. 14

CHEN, X.; DALLMEIER-TIESSEN, S.; DANI, A.; DASLER, R.; FERNÁNDEZ, J. D.; FOKIANOS, P.; HERTERICH, P.; ŠIMKO, T. CERN analysis preservation: a novel digital library service to enable reusable and reproducible research. In: FUHR, N.; KOVÁCS, L.; RISSE, T.; NEJDL, W. (Ed.). **Research and advanced technology for digital libraries**. Springer, 2016. p. 347–356. Disponível em: <https://doi.org/10.1007/978-3-319-43997-6_27>. 26

CHIRIGATI, F.; RAMPIN, R.; SHASHA, D.; FREIRE, J. ReproZip: computational reproducibility with ease. In: INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 16., 2016. **Proceedings..** ACM, 2016. p. 2085–2088. Disponível em: <https://doi.org/10.1145/2882903.2899401>. 3, 28, 29, 31

CLAERBOUT, J. F.; KARRENBACH, M. Electronic documents give reproducible research a new meaning. In: SEG TECHNICAL PROGRAM, 1992. **Proceedings..** Society of Exploration Geophysicists, 1992. Disponível em: <https://doi.org/10.1190/1.1822162>. 2, 7, 10

CLYBURNE-SHERIN, A.; FEI, X.; GREEN, S. A. Computational reproducibility via containers in psychology. **Meta-Psychology**, v. 3, nov. 2019. Disponível em: <https://doi.org/10.15626/mp.2018.892>. 19

COHEN-BOULAKIA, S.; BELHAJJAME, K.; COLLIN, O.; CHOPARD, J.; FROIDEVAUX, C.; GAIGNARD, A.; HINSEN, K.; LARMANDE, P.; BRAS, Y. L.; LEMOINE, F.; MAREUIL, F.; MÉNAGER, H.; PRADAL, C.; BLANCHET, C. Scientific workflows for computational reproducibility in the life sciences: status, challenges and opportunities. **Future Generation Computer Systems**, v. 75, p. 284–298, out. 2017. Disponível em: <https://doi.org/10.1016/j.future.2017.01.012>. 20

COMMITTEE ON EARTH OBSERVATION SATELLITES. **CEOS analysis ready data governance framework**. [S.l.: s.n.], 2021. 104, 111

COMUNIDADE TURING WAY; ARNOLD, B.; BOWLER, L.; GIBSON, S.; HERTERICH, P.; HIGMAN, R.; KRYSTALLI, A.; MORLEY, A.; OŔEILLY, M.; WHITAKER, K. **The turing way: a handbook for reproducible data science**. Zenodo, 2019. Disponível em: <https://zenodo.org/record/3233986>. 9, 10, 11, 23, 24, 25, 100

CRAGLIA, M.; NATIVI, S. Mind the gap: big data vs. interoperability and reproducibility of science. In: MATHIEU, P. P.; AUBRECHT, C. (Ed.). **Earth observation open science and innovation**. Springer, 2018. p. 121–141. Disponível em: <https://doi.org/10.1007/978-3-319-65633-5_6>. 3

CURTARELLI, V. P.; BARBOSA, C. C. F.; MACIEL, D. A.; JÚNIOR, R. F.; CARLOS, F. M.; NOVO, E. M. L. de M.; CURTARELLI, M. P.; SILVA, E. F. F. da. Diffuse attenuation of clear water tropical reservoir: a remote sensing semi-analytical approach. **Remote Sensing**, v. 12, n. 17, p. 2828, set. 2020. Disponível em: <https://doi.org/10.3390/rs12172828>. 5

D'ANDREA, F. **Mi próximo artículo científico en R - compendios de investigación, reproducibilidad e interactividad en las publicaciones académicas**. dec 2020. Disponível em: <https://flor14.github.io/rladies-jujuy/presentacion.html>. 23

DATACITE METADATA WORKING GROUP. Datacite metadata schema documentation for the publication and citation of research data and other research outputs v4.4. DataCite, 2021. Disponível em: <https://schema.datacite.org/meta/kernel-4.4/>. 67

DECAN, A.; MENS, T.; CLAES, M. On the topology of package dependency networks. In: EUROPEAN CONFERENCE ON SOFTWARE ARCHITECTURE WORKSHOPS, 10., 2016. **Proceedings..** ACM, 2016. Disponível em: <https://doi.org/10.1145/2993412.3003382>. 15

DHU, T.; GIULIANI, G.; JUÁREZ, J.; KAVVADA, A.; KILLOUGH, B.; MERODIO, P.; MINCHIN, S.; RAMAGE, S. National open data cubes and their contribution to country-level development policies and practices. **Data**, MDPI AG, v. 4, n. 4, p. 144, nov. 2019. Disponível em: <https://doi.org/10.3390/data4040144>. 40, 110

DONOHO, D. L. An invitation to reproducible computational research. **Biostatistics**, v. 11, n. 3, p. 385–388, jun. 2010. Disponível em: <https://doi.org/10.1093/biostatistics/kxq028>. 11

DUFFY, C.; GIL, Y.; DEELMAN, E.; MARRU, S.; PIERCE, M.; DEMIR, I.; WIENER, G. Designing a road map for geoscience workflows. **Eos, Transactions American Geophysical Union**, v. 93, n. 24, p. 225–226, jun. 2012. Disponível em: <https://doi.org/10.1029/2012eo240002>. 19

EASTERBROOK, S. M. Open code for open science? **Nature Geoscience**, v. 7, n. 11, p. 779–781, out. 2014. Disponível em: <https://doi.org/10.1038/ngeo2283>. 1

EISNER, D. Reproducibility of science: fraud, impact factors and carelessness. **Journal of Molecular and Cellular Cardiology**, v. 114, p. 364–368, jan. 2018. Disponível em: <https://doi.org/10.1016/j.yjmcc.2017.10.009>. 2

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 6. ed. Sao Paulo (SP): Pearson Addison Wesley, 2011. OCLC: 817222804. ISBN 9788579360855. 79

ENDSLEY, K. A. **The unmixing library: interactive tools for spectral mixture analysis of multispectral raster data in Python**. Zenodo, ago. 2019. Disponível em: <https://doi.org/10.5281/zenodo.3585979>. 114

FEARNSIDE, P. M. Deforestation in brazilian Amazonia: history, rates, and consequences. **Conservation Biology**, v. 19, n. 3, p. 680–688, jun. 2005. Disponível em: <https://doi.org/10.1111/j.1523-1739.2005.00697.x>. 110

FELSEMBURGH, C. A. **A produção do conhecimento na engenharia florestal**. Ponta Grossa: Atena, 2020. Disponível em: <https://doi.org/10.22533/at.ed.006202610>. 110

FERREIRA, K. R.; QUEIROZ, G. R.; VINHAS, L.; MARUJO, R. F. B.; SIMOES, R. E. O.; PICOLI, M. C. A.; CAMARA, G.; CARTAXO, R.; GOMES, V. C. F.; SANTOS, L. A.; SANCHEZ, A. H.; ARCANJO, J. S.; FRONZA, J. G.; NORONHA, C. A.; COSTA, R. W.; ZAGLIA, M. C.; ZIOTI, F.; KORTING, T. S.; SOARES, A. R.; CHAVES, M. E. D.; FONSECA, L. M. G. Earth observation data cubes for Brazil: requirements, methodology and products. **Remote Sensing**, v. 12, n. 24, p. 4033, dez. 2020. Disponível em: <https://doi.org/10.3390/rs12244033>. 39, 41, 43, 44, 101, 102, 104, 110, 112

FILGUIERA, R.; KRAUSE, A.; ATKINSON, M.; KLAMPANOS, I.; MORENO, A. dispel4py: a python framework for data-intensive scientific computing. **The International Journal of High Performance Computing Applications**, v. 31, n. 4, p. 316–334, jun. 2016. Disponível em: <https://doi.org/10.1177/1094342016649766>. 19

FIRESMITH, D. **Virtualization via containers**. sep 2017. Disponível em: <https://insights.sei.cmu.edu/blog/virtualization-via-containers/>. 16

FLORES JÚNIOR, R.; BARBOSA, C. C. F.; MACIEL, D. A.; NOVO, E. M. L. de M.; MARTINS, V. S.; LOBO, F. de L.; CARVALHO, L. A. S. de; CARLOS, F. M. Hybrid semi-analytical algorithm for estimating chlorophyll-a concentration in lower Amazon floodplain waters. **Frontiers in Remote Sensing**, v. 3, abr. 2022. Disponível em: <https://doi.org/10.3389/frsen.2022.834576>. 6

FOMEL, S.; CLAERBOUT, J. F. Guest editors' introduction: reproducible research. **Computing in Science Engineering**, v. 11, n. 1, p. 5–7, jan 2009. ISSN 1521-9615. Disponível em: <http://ieeexplore.ieee.org/document/4720217/>. 14

FORTUNATO, S.; BERGSTROM, C. T.; BöRNER, K.; EVANS, J. A.; HELBING, D.; MILOJEVIĆ, S.; PETERSEN, A. M.; RADICCHI, F.; SINATRA, R.; UZZI, B.; VESPIGNANI, A.; WALTMAN, L.; WANG, D.; BARABÁSI, A.-L. Science of science. **Science**, v. 359, n. 6379, p. eaao0185, mar. 2018. Disponível em: <https://doi.org/10.1126/science.aao0185>. 3

FREEDMAN, L. P.; COCKBURN, I. M.; SIMCOE, T. S. The economics of reproducibility in preclinical research. **PLOS Biology**, v. 13, n. 6, p. e1002165, jun. 2015. Disponível em: <https://doi.org/10.1371/journal.pbio.1002165>. 2

FREIRE, J.; KOOP, D.; CHIRIGATI, F.; SILVA, C. T. In: _____. **Implementing Reproducible Research**. [S.l.]: CRC Press, 2014. cap. Reproducibility Using VisTrails. 1

FUNDO AMAZÔNIA. **Fundo Amazônia**. 2021. Disponível em: <http://www.fundoamazonia.gov.br/pt/projeto/Monitoramento-Ambiental-dos-Biomas-Brasileiros/>. 4

GENTLEMAN, R.; LANG, D. T. Statistical analyses and reproducible research. 2004. 23, 24

_____. _____. 2007. 9

GIL, Y. From data to knowledge to discoveries: artificial intelligence and scientific workflows. **Scientific Programming**, v. 17, n. 3, p. 231–246, 2009. ISSN 10589244. 20

GIULIANI, G.; CAMARA, G.; KILLOUGH, B.; MINCHIN, S. Earth observation open science: enhancing reproducible science using data cubes. **Data**, v. 4, n. 4, p. 147, nov. 2019. Disponível em: <https://doi.org/10.3390/data4040147>. 3

GIULIANI, G.; CHATENOUX, B.; PILLER, T.; MOSER, F.; LACROIX, P. Data Cube on Demand (DCoD): generating an earth observation data cube anywhere in the world. **International Journal of Applied Earth Observation and Geoinformation**, v. 87, p. 102035, 2020. ISSN 03032434. Disponível em: <https://doi.org/10.1016/j.jag.2019.102035>. 39, 102, 110

GOBLE, C.; COHEN-BOULAKIA, S.; SOILAND-REYES, S.; GARIJO, D.; GIL, Y.; CRUSOE, M. R.; PETERS, K.; SCHOBER, D. FAIR computational workflows. **Data Intelligence**, v. 2, n. 1-2, p. 108–121, jan. 2020. Disponível em: <https://doi.org/10.1162/dint_a_00033>. 19, 20

GOMES, V. C. F.; CARLOS, F. M.; QUEIROZ, G. R.; FERREIRA, K. R.; SANTOS, R. Accessing and processing brazilian earth observation data cubes with the open data cube platform. **ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences**, V-4-2021, p. 153–159, jun. 2021. Disponível em: <https://doi.org/10.5194/isprs-annals-v-4-2021-153-2021>. 6

GROUP ON EARTH OBSERVATIONS. **Integrating earth observations into the formulation and implementation of national adaptation plans: agriculture and food security**. 2022. Disponível em: <https://earthobservations.org/documents/cc_wg/GEO_NAP_Supplement_final.pdf>. 39, 102

GRüNING, B.; ; DALE, R.; SJöDIN, A.; CHAPMAN, B. A.; ROWE, J.; TOMKINS-TINCH, C. H.; VALIERIS, R.; KöSTER, J. Bioconda: sustainable and comprehensive software distribution for the life sciences. **Nature Methods**, v. 15, n. 7, p. 475–476, jul. 2018. Disponível em: <https://doi.org/10.1038/s41592-018-0046-7>. 16

GRUPO DE DESENVOLVEDORES INVENIO RDM. **Arquitetura e boas práticas do InvenioRDM**. november 2022. Disponível em:

<https://inveniordm.docs.cern.ch/develop/architecture/software/>. 65, 77

HEROUX, M. A.; BARBA, L.; PARASHAR, M.; STODDEN, V.; TAUFER, M. **Toward a compatible reproducibility taxonomy for computational and computing sciences.** Office of Scientific and Technical Information (OSTI), out. 2018. Disponível em: <https://doi.org/10.2172/1481626>. 7

HERRERA, W. Best practices for reproducible research. **XRDS: Crossroads, The ACM Magazine for Students**, v. 25, n. 3, p. 10–11, abr. 2019. Disponível em: <https://doi.org/10.1145/3312549>. 24

HOWE, B. Virtual appliances, cloud computing, and reproducible research. **Computing in Science Engineering**, v. 14, n. 4, p. 36–41, jul 2012. ISSN 1521-9615. Disponível em: <http://ieeexplore.ieee.org/document/6193081/>. 16

IVIE, P.; THAIN, D. Reproducibility in scientific computing. **ACM Computing Surveys**, v. 51, n. 3, p. 1–36, jul. 2018. Disponível em: <https://doi.org/10.1145/3186266>. 1

JACKSON, M.; KAVOUSSANAKIS, K.; WALLACE, E. W. Using prototyping to choose a bioinformatics workflow management system. **PLoS Computational Biology**, v. 17, n. 2, p. 1–13, 2021. ISSN 15537358. Disponível em: <http://dx.doi.org/10.1371/journal.pcbi.1008622>. 21

JOSEPH, M. **mbjoseph/intro-research-compendia: Initial release**. Zenodo, 2020. Disponível em: <https://zenodo.org/record/3717288>. 23

JUPYTER, P.; BUSSONNIER, M.; FORDE, J.; FREEMAN, J.; GRANGER, B.; HEAD, T.; KELLEY, K.; NALVARTE, G.; OSHEROFF, A.; PACER, M.; PANDA, Y.; PEREZ, F.; RAGAN-KELLEY, B.; WILLING, C. Binder 2.0: reproducible, interactive, sharable environments for science at scale. In: PYTHON IN SCIENCE CONFERENCE, 17., 2018. **Proceedings...** 2018. p. 113–120. Disponível em: <https://conference.scipy.org/proceedings/scipy2018/pdfs/project_jupyter.pdf>. 18

KEDRON, P.; LI, W.; FOTHERINGHAM, S.; GOODCHILD, M. Reproducibility and replicability: opportunities and challenges for geospatial research. **International Journal of Geographical Information Science**, Informa UK Limited, v. 35, n. 3, p. 427–445, ago. 2020. Disponível em: <https://doi.org/10.1080/13658816.2020.1802032>. 3, 10, 15

KEITH, P. Anything you can do I can do better. **Computer Vision, Graphics, and Image Processing**, v. 36, p. 387–391, 1986. ISSN 10672516. 8, 9

KILLOUGH, B. The impact of analysis ready data in the africa regional data cube. In: IEEE INTERNATIONAL GEOSCIENCE AND REMOTE SENSING SYMPOSIUM, 2019. **Proceedings...** IEEE, 2019. Disponível em: <https://doi.org/10.1109/igarss.2019.8898321>. 102

KLUYVER, T.; RAGAN-KELLEY, B.; PÉREZ, F.; GRANGER, B.; BUSSONNIER, M.; FREDERIC, J.; KELLEY, K.; HAMRICK, J.; GROUT, J.; CORLAY, S.; IVANOV, P.; AVILA, D.; ABDALLA, S.; WILLING, C. Jupyter notebooks — a publishing format for reproducible computational workflows. In: INTERNATIONAL CONFERENCE ON ELECTRONIC PUBLISHING, 20., 2016. **Proceedings...** ELPUB, 2016. p. 87–90. ISBN 9781614996484. Disponível em: <https://doi.org/10.3233/978-1-61499-649-1-87>. 12, 13

KNOTH, C.; NüST, D. Reproducibility and practical adoption of GEOBIA with open-source software in docker containers. **Remote Sensing**, v. 9, n. 3, p. 290, mar. 2017. Disponível em: <https://doi.org/10.3390/rs9030290>. 18

KNUTH, D. E. Literate programming. **Computer Journal**, v. 27, n. 2, p. 97–111, 1984. ISSN 00104620. Disponível em: <http://doi.org/10.1093/comjnl/27.2.97>. 12

KONKOL, M.; KRAY, C.; PFEIFFER, M. Computational reproducibility in geoscientific papers: insights from a series of studies with geoscientists and a reproduction study. **International Journal of Geographical Information Science**, v. 33, n. 2, p. 408–429, ago. 2018. Disponível em: <https://doi.org/10.1080/13658816.2018.1508687>. 1, 10, 14

KONKOL, M.; NÜST, D.; GOULIER, L. Publishing computational research – a review of infrastructures for reproducible and transparent scholarly communication. Research Integrity and Peer Review, p. 1–8, 2020. ISSN 2058-8615. Disponível em: <http://arxiv.org/abs/2001.00484>. 2

KOPP, S.; BECKER, P.; DOSHI, A.; WRIGHT, D. J.; ZHANG, K.; XU, H. Achieving the full vision of earth observation data cubes. **Data**, v. 4, n. 3, p. 94, jul. 2019. Disponível em: <https://doi.org/10.3390/data4030094>. 40, 102

KORNFELD, D. S.; TITUS, S. L. Stop ignoring misconduct. **Nature**, v. 537, n. 7618, p. 29–30, ago. 2016. Disponível em: <https://doi.org/10.1038/537029a>. 1

KOTLIAR, M.; KARTASHOV, A. V.; BARSKI, A. CWL-Airflow: a lightweight pipeline manager supporting Common Workflow Language. **GigaScience**, v. 8, n. 7, p. 1–8, 2019. ISSN 2047217X. 22

KUNZE, J.; LITTMAN, J.; MADDEN, E.; ADAMS, C. **The BagIt file packaging format (V1.0)**. RFC Editor, October 2018. 1-25 p. Disponível em: <https://www.rfc-editor.org/rfc/rfc8493>. 90

KURTZER, G. M.; SOCHAT, V.; BAUER, M. W. Singularity: scientific containers for mobility of compute. **PLOS ONE**, v. 12, n. 5, p. e0177459, maio 2017. Disponível em: <https://doi.org/10.1371/journal.pone.0177459>. 17

LAMPA, S.; DAHLö, M.; ALVARSSON, J.; SPJUTH, O. SciPipe: a workflow library for agile development of complex and dynamic bioinformatics pipelines. **GigaScience**, v. 8, n. 5, abr. 2019. Disponível em: <https://doi.org/10.1093/gigascience/giz044>. 20

LANDAU, W. The targets r package: a dynamic make-like function-oriented pipeline toolkit for reproducibility and high-performance computing. **Journal of Open Source Software**, v. 6, n. 57, p. 2959, jan. 2021. Disponível em: <https://doi.org/10.21105/joss.02959>. 123

LASSER, J. Creating an executable paper is a journey through open science. **Communications Physics**, v. 3, n. 1, ago. 2020. Disponível em: <https://doi.org/10.1038/s42005-020-00403-4>. 13

LEIPZIG, J. A review of bioinformatic pipeline frameworks. **Briefings in Bioinformatics**, p. bbw020, mar. 2016. Disponível em: <https://doi.org/10.1093/bib/bbw020>. 20

LEUTNER, B.; HORNING, N.; SCHWALB-WILLMANN, J. **RStoolbox: tools for remote sensing data analysis**. [s.n.], 2019. Disponível em: <http://bleutner.github.io/RStoolbox>. 116

LEVEQUE, R.; MITCHELL, I.; STODDEN, V. Reproducible research for scientific computing: tools and strategies for changing the culture. **Computing in Science and Engineering**, v. 14, n. 4, p. 13–17, 2012. ISSN 15219615. 7, 9, 100

LEWIS, K. P.; WAL, E. V.; FIFIELD, D. A. Wildlife biology, big data, and reproducible research. **Wildlife Society Bulletin**, v. 42, n. 1, p. 172–179, jan. 2018. Disponível em: <https://doi.org/10.1002/wsb.847>. 10

LOPES, L.; SADLER, P.; BERNARDES-GÉNISSOND, V.; MOURA, J.; CHAUVIN, R.; BERNHARDT, P. V.; SOUSA, E. The fundamental importance of basic science: examples of high-impact discoveries from an international chemistry network. **Química Nova**, 2020. Disponível em: <https://doi.org/10.21577/0100-4042.20170584>. 1

LOWNDES, J. S. S.; BEST, B. D.; SCARBOROUGH, C.; AFFLERBACH, J. C.; FRAZIER, M. R.; O'HARA, C. C.; JIANG, N.; HALPERN, B. S. Our path to better science in less time using open data science tools. **Nature Ecology & Evolution**, Springer Science and Business Media LLC, v. 1, n. 6, maio 2017. Disponível em: <https://doi.org/10.1038/s41559-017-0160>. 14

LUDÄSCHER, B.; ALTINTAS, I.; BERKLEY, C.; HIGGINS, D.; JAEGER, E.; JONES, M.; LEE, E. A.; TAO, J.; ZHAO, Y. Scientific workflow management and the kepler system. **Concurrency and Computation: Practice and Experience**, Wiley, v. 18, n. 10, p. 1039–1065, 2006. Disponível em: <https://doi.org/10.1002/cpe.994>. 19

MARIANO, R.; QUEIROZ, G.; ANDRADE, P.; SANTOS, R. sits.rep: pesquisa reprodutível em classificações de uso e cobertura da terra. In: WORKSHOP DE COMPUTAÇÃO APLICADA À GESTÃO DO MEIO AMBIENTE E RECURSOS NATURAIS, 2020. **Anais...** Sociedade Brasileira de Computação, 2020. Disponível em: <https://doi.org/10.5753/wcama.2020.11019>. 44

MARWICK, B. **Research compendium for the 1989 excavations at Madjedbebe rockshelter, NT, Australia**. figshare, 2017. Disponível em: <https://figshare.com/articles/software/1989_excavation_report_Madjebebe/1297059>. 25

MARWICK, B.; ATTALI, D.; HOLLISTER, J. W.; BRYAN, J. **rrrpkg - use of an R package to facilitate reproducible research**. jan 2017. Disponível em: <https://github.com/ropensci/rrrpkg>. 23

MARWICK, B.; BOETTIGER, C.; MULLEN, L. Packaging data analytical work reproducibly using R (and Friends). **American Statistician**, v. 72, n. 1, p. 80–88, 2018. ISSN 15372731. Disponível em: <https://doi.org/10.1080/00031305.2017.1375986>. 11, 22, 23, 24, 100

MAURANO, L. E. P.; ESCADA, M. I. S.; RENNO, C. D. Padrões espaciais de desmatamento e a estimativa da exatidão dos mapas do PRODES para Amazônia

Legal Brasileira. **Ciência Florestal**, v. 29, n. 4, p. 1763, 2019. ISSN 0103-9954. 110

MCKINNEY, W. Data structures for statistical computing in python. In: PYTHON IN SCIENCE CONFERENCE, 9., 2010. **Proceedings...** 2010. p. 56 – 61. Disponível em: <https://doi.org/10.25080/Majora-92bf1922-00a>. 17

MENEGIDIO, F. B.; JABES, D. L.; OLIVEIRA, R. C. de; NUNES, L. R. Dugong: a docker image, based on ubuntu linux, focused on reproducibility and replicability for bioinformatics analyses. **Bioinformatics**, v. 34, n. 3, p. 514–515, set. 2017. Disponível em: <https://doi.org/10.1093/bioinformatics/btx554>. 19

MIELL, I. **Docker in practice**. Shelter Island, NY: Manning Publications, 2016. ISBN 9781617292729. 17

MORABITO, R.; KJALLMAN, J.; KOMU, M. Hypervisors vs. lightweight virtualization: a performance comparison. In: IEEE INTERNATIONAL CONFERENCE ON CLOUD ENGINEERING, 2015. **Proceedings...** IEEE, 2015. Disponível em: <https://doi.org/10.1109/ic2e.2015.74>. 16

MUNAFÒ, M. R.; NOSEK, B. A.; BISHOP, D. V. M.; BUTTON, K. S.; CHAMBERS, C. D.; SERT, N. P. du; SIMONSOHN, U.; WAGENMAKERS, E.-J.; WARE, J. J.; IOANNIDIS, J. P. A. A manifesto for reproducible science. **Nature Human Behaviour**, v. 1, n. 1, jan. 2017. Disponível em: <https://doi.org/10.1038/s41562-016-0021>. 1, 10

MUNAFÒ, M. R.; SMITH, G. D. Robust research needs many lines of evidence. **Nature**, v. 553, n. 7689, p. 399–401, jan. 2018. Disponível em: <https://doi.org/10.1038/d41586-018-01023-3>. 2

MURALIKRISHNA, I. V.; MANICKAM, V. Natural resource management and biodiversity conservation. In: _____. **Environmental management**. [S.l.]: Elsevier, 2017. cap. 3, p. 23–35. Disponível em: <https://doi.org/10.1016/b978-0-12-811989-1.00003-8>. 1

NATIONAL ACADEMIES OF SCIENCES. **Reproducibility and replicability in science**. [S.l.: s.n.], 2019. ISBN 9780309486194. 1, 7

NATURE. Must try harder. **Nature**, v. 483, n. 7391, p. 509–509, mar. 2012. Disponível em: <https://doi.org/10.1038/483509a>. 1

_____. The scientific events that shaped the decade. **Nature**, v. 576, n. 7787, p. 337–338, dez. 2019. Disponível em: <https://doi.org/10.1038/d41586-019-03857-x>. 1

NüST, D. **Reproducibility service for executable research compendia: technical specifications and reference implementation**. 2018. Disponível em: <https://zenodo.org/record/2203844>. 33, 34, 35

NÜST, D.; BOETTIGER, C.; MARWICK, B. How to read a research compendium. **arXiv**, n. Keshav, 2018. ISSN 23318422. 24

NüST, D.; EGLEN, S. J. CODECHECK: an open science initiative for the independent execution of computations underlying research articles during peer review to improve reproducibility. **F1000Research**, v. 10, p. 253, mar. 2021. Disponível em: <https://doi.org/10.12688/f1000research.51738.1>. 1, 9

NüST, D.; HINZ, M. Containerit: generating dockerfiles for reproducible research with r. **Journal of Open Source Software**, v. 4, n. 40, p. 1603, ago. 2019. Disponível em: <https://doi.org/10.21105/joss.01603>. 17

NüST, D.; KONKOL, M.; PEBESMA, E.; KRAY, C.; SCHUTZEICHEL, M.; PRZIBYTZIN, H.; LORENZ, J. Opening the publication process with executable research compendia. **D-Lib Magazine**, v. 23, n. 1/2, jan. 2017. Disponível em: <https://doi.org/10.1045/january2017-nuest>. 3, 10, 24, 25, 63

NÜST, D.; PEBESMA, E. Practical reproducibility in geography and geosciences. **Annals of the American Association of Geographers**, p. 1–11, 2020. ISSN 24694460. Disponível em: <https://doi.org/10.1080/24694452.2020.1806028>. 9, 10, 24

NüST, D.; SOCHAT, V.; MARWICK, B.; EGLEN, S. J.; HEAD, T.; HIRST, T.; EVANS, B. D. Ten simple rules for writing dockerfiles for reproducible data science. **PLOS Computational Biology**, v. 16, n. 11, p. e1008316, nov. 2020. Disponível em: <https://doi.org/10.1371/journal.pcbi.1008316>. 45

OCI. **Open container initiative runtime specification**. mar 2020. Disponível em: <https://github.com/opencontainers/runtime-spec>. 17

OPENAPI v3.0. **OpenAPI specification**. november 2022. Disponível em: <https://github.com/OAI/OpenAPI-Specification>. 32, 66

OSTERMANN, F.; NüST, D.; GRANELL, C.; HOFER, B.; KONKOL, M. Reproducible research and GIScience: an evaluation using GIScience conference papers. California Digital Library (CDL), out. 2020. Disponível em: <https://doi.org/10.31223/x5zk5v>. 10

PENG, R. D. Reproducible research in computing science. **Science**, v. 334, n. 6060, p. 1226–1227, 2011. ISSN 1095-9203. 1, 7, 8, 9, 10

PÉREZ, F.; GRANGER, B. E. IPython: a system for interactive scientific computing. **Computing in Science and Engineering**, v. 9, n. 3, p. 21–29, 2007. ISSN 15219615. 12

PERKEL, J. M. Workflow systems turn raw data into scientific knowledge. **Nature**, v. 573, n. 7772, p. 149–150, set. 2019. Disponível em: <https://doi.org/10.1038/d41586-019-02619-z>. 19

PICCOLO, S. R.; FRAMPTON, M. B. Tools and techniques for computational reproducibility. **GigaScience**, v. 5, n. 1, p. 1–13, 2016. ISSN 2047217X. Disponível em: <http://dx.doi.org/10.1186/s13742-016-0135-4>. 11, 12, 14, 16

PICOLI, M. C.; SIMOES, R.; CHAVES, M.; SANTOS, L. A.; SANCHEZ, A.; SOARES, A.; SANCHES, I. D.; FERREIRA, K. R.; QUEIROZ, G. R. Cbers data cube: a powerful technology for mapping and monitoring brazilian biomes. In: PHOTOGRAMMETRY, REMOTE SENSING AND SPATIAL INFORMATION SCIENCES, 3., 2020. **Proceedings...** ISPRS, 2020. p. 533–539. Disponível em: <http://doi.org/10.5194/isprs-Annals-V-3-2020-533-2020>. 41, 102

PICOLI, M. C. A.; CAMARA, G.; SANCHES, I.; SIMÕES, R.; CARVALHO, A.; MACIEL, A.; COUTINHO, A.; ESQUERDO, J.; ANTUNES, J.; BEGOTTI, R. A.; ARVOR, D.; ALMEIDA, C. Big earth observation time series analysis for monitoring brazilian agriculture. **ISPRS Journal of Photogrammetry and Remote Sensing**, v. 145, p. 328–339, nov. 2018. Disponível em: <https://doi.org/10.1016/j.isprsjprs.2018.08.007>. 110

PONTIKA, N.; KNOTH, P.; CANCELLIERI, M.; PEARCE, S. Fostering open science to research using a taxonomy and an eLearning portal. In: INTERNATIONAL CONFERENCE ON KNOWLEDGE TECHNOLOGIES AND DATA-DRIVEN BUSINESS, 15., 2015. **Proceedings...** ACM, 2015. Disponível em: <https://doi.org/10.1145/2809563.2809571>. 10

POPPER, K. R. **A lógica da pesquisa científica**. [S.l.: s.n.], 2013. ISBN 9788531612503. 9

PRIEDHORSKY, R.; RANDLES, T. Charliecloud. In: INTERNATIONAL CONFERENCE FOR HIGH PERFORMANCE COMPUTING, NETWORKING, STORAGE AND ANALYSIS, 2017. **Proceedings...** ACM, 2017. Disponível em: <https://doi.org/10.1145/3126908.3126925>. 17

QUEIROZ, G. R. d.; FERREIRA, K. R.; VINHAS, L.; CAMARA, G.; COSTA, R. W. da; SOUZA, R. C. M. de; MAUS, V. W.; SANCHEZ, A. Wtss: um serviço web para extração de séries temporais de imagens de sensoriamento remoto. In: SIMPÓSIO BRASILEIRO DE SENSORIAMENTO REMOTO, 17., 2015, João Pessoa. **Anais...** São José dos Campos: INPE, 2015. 43

QUINTANO, C.; FERNÁNDEZ-MANSO, A.; ROBERTS, D. A. Multiple Endmember Spectral Mixture Analysis (MESMA) to map burn severity levels from Landsat images in Mediterranean countries. **Remote Sensing of Environment**, v. 136, p. 76–88, 2013. ISSN 00344257. Disponível em: <http://dx.doi.org/10.1016/j.rse.2013.04.017>. 113

QUINTANO, C.; FÉRNANDEZ-MANSO, A.; SHIMABUKURO, Y. E.; PEREIRA, G. Aplicação do modelo linear de mistura espectral para o mapeamento de queimadas no Parque Nacional das Emas. In: SIMPÓSIO BRASILEIRO DE SENSORIAMENTO REMOTO, 15., 2015. **Anais...** São José dos Campos: INPE, 2011. Disponível em: <http://marte.sid.inpe.br/col/dpi.inpe.br/marte/2011/07.11.14.50/doc/p0685.pdf>. 113

RAPHAEL, M. P.; SHEEHAN, P. E.; VORA, G. J. A controlled trial for reproducibility. **Nature**, v. 579, n. 7798, p. 190–192, mar. 2020. Disponível em: <https://doi.org/10.1038/d41586-020-00672-7>. 2

RULE, A.; BIRMINGHAM, A.; ZUNIGA, C.; ALTINTAS, I.; HUANG, S.-C.; KNIGHT, R.; MOSHIRI, N.; NGUYEN, M. H.; ROSENTHAL, S. B.; PÉREZ, F.; ROSE, P. W. Ten simple rules for writing and sharing computational analyses in jupyter notebooks. **PLOS Computational Biology**, v. 15, n. 7, p. e1007007, jul. 2019. Disponível em: <https://doi.org/10.1371/journal.pcbi.1007007>. 2, 45

RULE, A.; TABARD, A.; HOLLAN, J. D. Exploration and explanation in computational notebooks. In: CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, 2018. **Proceedings...** ACM, 2018. p. 1–12. Disponível em: <https://doi.org/10.1145/3173574.3173606>. 12, 14

RULL, V. The most important application of science. **EMBO reports**, v. 15, n. 9, p. 919–922, ago. 2014. Disponível em: <https://doi.org/10.15252/embr.201438848>. 1

RUSSELL, J. F. If a job is worth doing, it is worth doing twice. **Nature**, v. 496, n. 7443, p. 7–7, abr. 2013. Disponível em: <https://doi.org/10.1038/496007a>. 2

SANCHEZ, A.; PICOLI, M.; ANDRADE, P. R.; SIMõES, R.; SANTOS, L.; CHAVES, M.; BEGOTTI, R.; CAMARA, G. Land cover classifications of clear-cut deforestation using deep learning. In: GEOINFO, 20., 2019. **Proceedings...** São José dos Campos: INPE, 2019. p. 48–56. Disponível em: <http://mtc-m16d.sid.inpe.br/col/sid.inpe.br/mtc-m16d/2019/11.27.13.29/doc/48-56.pdf>. xiii, 40, 53, 101, 109, 111, 114

SANDVE, G. K.; NEKRUTENKO, A.; TAYLOR, J.; HOVIG, E. Ten simple rules for reproducible computational research. **PLoS Computational Biology**, v. 9, n. 10, p. e1003285, out. 2013. Disponível em: <https://doi.org/10.1371/journal.pcbi.1003285>. 2, 45

SANSIGOLO, G. **Terrabrasilis research data: uma plataforma para compatilhamento de dados científicos geoespaciais**. Dissertação (Mestrado em Computação Aplicada) — Instituto Nacional de Pesquisas Espaciais (INPE), 2020. 26

SAYÃO, L. F.; SALES, L. F. Algumas considerações sobre os repositórios digitais de dados de pesquisa. **Informação & Informação**, v. 21, n. 2, p. 90, dez. 2016. Disponível em: <https://doi.org/10.5433/1981-8920.2016v21n2p90>. 25, 26

SCHULTZ, M.; CLEVERS, J. G.; CARTER, S.; VERBESSELT, J.; AVITABILE, V.; QUANG, H. V.; HEROLD, M. Performance of vegetation indices from Landsat time series in deforestation monitoring. **International Journal of Applied Earth Observation and Geoinformation**, v. 52, n. May 2012, p. 318–327, 2016. ISSN 1872826X. Disponível em: <http://dx.doi.org/10.1016/j.jag.2016.06.020>. 113

SHIMABUKURO, Y. E.; PONZONI, F. J. **Spectral mixture for remote sensing**. Springer, 2019. Disponível em: <https://doi.org/10.1007/978-3-030-02017-0>. 110, 112, 113, 114

ŠIMKO, T.; HEINRICH, L.; HIRVONSALO, H.; KOUSIDIS, D.; RODRÍGUEZ, D. REANA: A system for reusable research data analyses. **EPJ Web of**

Conferences, v. 214, 2019. Disponível em: <https://doi.org/10.1051/epjconf/201921406034>. 3, 22, 31, 32, 56

SIMOES, R.; CAMARA, G.; QUEIROZ, G.; SOUZA, F.; ANDRADE, P. R.; SANTOS, L.; CARVALHO, A.; FERREIRA, K. Satellite image time series analysis for big earth observation data. **Remote Sensing**, v. 13, n. 13, p. 2428, jun. 2021. Disponível em: <https://doi.org/10.3390/rs13132428>. 39, 40, 41, 42, 44, 105, 114, 115, 117

SIMOES, R.; PICOLI, M. C. A.; CAMARA, G.; MACIEL, A.; SANTOS, L.; ANDRADE, P. R.; SÁNCHEZ, A.; FERREIRA, K.; CARVALHO, A. Land use and cover maps for Mato Grosso State in Brazil from 2001 to 2017. **Scientific Data**, v. 7, n. 1, p. 34, 2020. Disponível em: <http://www.nature.com/articles/s41597-020-0371-4>. xiv, 89, 90

SOCIETY, A. P. **What is science?** nov 1999. Disponível em: <https://www.aps.org/policy/statements/99_6.cfm>. 9

SOILLE, P.; BURGER, A.; DE MARCHI, D.; KEMPENEERS, P.; RODRIGUEZ, D.; SYRRIS, V.; VASILEV, V. A versatile data-intensive computing platform for information retrieval from big geospatial data. **Future Generation Computer Systems**, v. 81, p. 30–40, 2018. Disponível em: <https://doi.org/10.1016/j.future.2017.11.007>. 3, 39, 102

SOUSA, D.; SMALL, C. Global cross-calibration of Landsat spectral mixture models. **Remote Sensing of Environment**, v. 192, p. 139–149, 2017. ISSN 00344257. Disponível em: <http://dx.doi.org/10.1016/j.rse.2017.01.033>. 114

SOUZA, C. M.; SIQUEIRA, J. V.; SALES, M. H.; FONSECA, A. V.; RIBEIRO, J. G.; NUMATA, I.; COCHRANE, M. A.; BARBER, C. P.; ROBERTS, D. A.; BARLOW, J. Ten-year landsat classification of deforestation and forest degradation in the brazilian amazon. **Remote Sensing**, v. 5, n. 11, p. 5493–5513, 2013. ISSN 20724292. 113

STANISIC, L.; LEGRAND, A.; DANJEAN, V. An effective git and org-mode based workflow for reproducible research. **ACM SIGOPS Operating Systems Review**, v. 49, n. 1, p. 61–70, jan. 2015. Disponível em: <https://doi.org/10.1145/2723872.2723881>. 101

STARK, P. B. Before reproducibility must come preproducibility. **Nature**, v. 557, n. 7707, p. 613–613, maio 2018. Disponível em: <https://doi.org/10.1038/d41586-018-05256-0>. 2

STODDEN, V. The legal framework for reproducible scientific research: Licensing and copyright. v. 11, n. 1, p. 35–40, 2009. Disponível em: <https://doi.org/10.1109/MCSE.2009.19>. 24

_____. Reproducible research: tools and strategies for scientific computing. **Computing in Science & Engineering**, v. 14, n. 4, p. 11–12, jul. 2012. Disponível em: <https://doi.org/10.1109/mcse.2012.82>. 10

STODDEN, V.; MCNUTT, M.; BAILEY, D. H.; DEELMAN, E.; GIL, Y.; HANSON, B.; HEROUX, M. A.; IOANNIDIS, J. P.; TAUFER, M. Enhancing reproducibility for computational methods. **Science**, v. 354, n. 6317, p. 1240–1241, 2016. ISSN 10959203. Disponível em: <https://doi.org/10.1126/science.aah6168>. 10, 22

STODDEN, V.; MIGUEZ, S. Best practices for computational science: software infrastructure and environments for reproducible and extensible research. **Journal of Open Research Software**, v. 2, n. 1, p. 1–6, 2014. ISSN 2049-9647. Disponível em: <https://doi.org/10.5334/jors.ay>. 2, 11, 12

STROZZI, F.; JANSSEN, R.; WURMUS, R.; CRUSOE, M. R.; GITHINJI, G.; TOMMASO, P. D.; BELHACHEMI, D.; MöLLER, S.; SMANT, G.; LIGT, J. de; PRINS, P. Scalable workflows and reproducible data analysis for genomics. In: ANISIMOVA, M. (Ed.). **Evolutionary genomics: statistical and computational methods**. Springer, 2019. p. 723–745. Disponível em: <https://doi.org/10.1007/978-1-4939-9074-0_24>. 19, 20

TALIA, D. Workflow Systems for Science: Concepts and Tools. **ISRN Software Engineering**, v. 2013, p. 1–15, jan 2013. ISSN 2090-7680. Disponível em: <https://www.hindawi.com/journals/isrn/2013/404525/>. 20

TIERNEY, L.; ROSSINI, A. J.; LI, N.; SEVCIKOVA, H. **Snow: simple network of workstations**. [S.l.: s.n.], 2018. R package version 0.4-3. 116

TURNBULL, J. **The Docker book**. [S.l.]: James Turnbull, 2014. ISBN 0988820234. 17

VINHAS, L.; QUEIROZ, G. R. de; FERREIRA, K. R.; CAMARA, G. Web services for big earth observation data. **Brazilian Journal of Cartography**, 2017. 43

VIVIAN, J.; RAO, A. A.; NOTHAFT, F. A.; KETCHUM, C.; ARMSTRONG, J.; NOVAK, A.; PFEIL, J.; NARKIZIAN, J.; DERAN, A. D.;

MUSSELMAN-BROWN, A.; SCHMIDT, H.; AMSTUTZ, P.; CRAFT, B.; GOLDMAN, M.; ROSENBLOOM, K.; CLINE, M.; O'CONNOR, B.; HANNA, M.; BIRGER, C.; KENT, W. J.; PATTERSON, D. A.; JOSEPH, A. D.; ZHU, J.; ZARANEK, S.; GETZ, G.; HAUSSLER, D.; PATEN, B. Toil enables reproducible, open source, big biomedical data analyses. **Nature Biotechnology**, v. 35, n. 4, p. 314–316, abr. 2017. Disponível em: <https://doi.org/10.1038/nbt.3772>. 22

VUORRE, M.; CRUMP, M. J. C. Sharing and organizing research products as r packages. **Behavior Research Methods**, set. 2020. Disponível em: <https://doi.org/10.3758/s13428-020-01436-x>. 11

WANG, A. Y.; MITTAL, A.; BROOKS, C.; ONEY, S. How data scientists use computational notebooks for real-time collaboration. **Proceedings of the ACM on Human-Computer Interaction**, v. 3, n. CSCW, 2019. ISSN 25730142. 12, 13

WICKHAM, H. Tidy data. **Journal of Statistical Software**, v. 59, n. 10, 2014. Disponível em: <https://doi.org/10.18637/jss.v059.i10>. 14

WICKHAM, H.; GROLEMUND, G. **R for data science**. [S.l.]: O'Reilly Media, 2017. 86

WIESER, F.; STRYECK, S.; LANG, K.; HAHN, C.; THALLINGER, G. G.; FEICHTINGER, J.; HACK, P.; STEPPONAT, M.; MERCHANT, N.; LINDSTAEDT, S.; OBERDORFER, G. A local platform for user-friendly FAIR data management and reproducible analytics. **Journal of Biotechnology**, v. 341, p. 43–50, nov. 2021. ISSN 0168-1656. 62, 108

WILSON, G.; BRYAN, J.; CRANSTON, K.; KITZES, J.; NEDERBRAGT, L.; TEAL, T. K. Good enough practices in scientific computing. **PLOS Computational Biology**, v. 13, n. 6, p. e1005510, jun. 2017. Disponível em: <https://doi.org/10.1371/journal.pcbi.1005510>. 24

WITTER, I. H.; BAINBRIDGE, D. **How to build a digital library**. [S.l.: s.n.], 2003. 57 p. ISBN 1-55860-790-0. 25

XIE, Y. **R Markdown : the definitive guide**. Boca Raton: [s.n.], 2019. ISBN 978-1-138-35933-8. 13

YANG, C.; WU, G.; DING, K.; SHI, T.; LI, Q.; WANG, J. Improving land use/land cover classification by integrating pixel unmixing and decision tree methods. **Remote Sensing**, v. 9, n. 12, 2017. ISSN 20724292. 113

YANIV, Z.; LOWEKAMP, B. C.; JOHNSON, H. J.; BEARE, R. SimpleITK image-analysis notebooks: a collaborative environment for education and reproducible research. **Journal of Digital Imaging**, v. 31, n. 3, p. 290–303, nov. 2017. Disponível em: <https://doi.org/10.1007/s10278-017-0037-8>. 3

ZAGLIA, M. C. **Catalogação de imagens de satélites de sensoriamento remoto voltada para acesso a cubos de dados de observação da terra**. Dissertação (Mestrado em Computação Aplicada), 2020. 43

ZAGLIA, M. C.; VINHAS, L.; QUEIROZ, G. R. D.; SIMOES, R. Catalogação de metadados do cubo de dados do brasil com o spatiotemporal asset catalog. In: GEOINFO, 20., 2019. **Proceedings...** São José dos Campos: INPE, 2019. p. 280–285. Disponível em: <http://mtc-m16d.sid.inpe.br/col/sid.inpe.br/mtc-m16d/2019/11.27.19.13/doc/280-285.pdf>. 43

ZARAGOZÍ, B. M.; TRILLES, S.; NAVARRO-CARRIÓN, J. T. Leveraging container technologies in a GIScience project: a perspective from open reproducible research. **ISPRS International Journal of Geo-Information**, v. 9, n. 3, p. 138, fev. 2020. Disponível em: <https://doi.org/10.3390/ijgi9030138>. 18

ZIOTI, F. **Plataforma para acesso e análise de trajetórias de uso e cobertura da terra**. Dissertação (Mestrado em Computação Aplicada), 2020. Disponível em: <http://mtc-m21c.sid.inpe.br/col/sid.inpe.br/mtc-m21c/2020/10.09.14.57/doc/publicacao.pdf>. 43

ZIOTI, F.; CONRADO, V.; LLAPA, E.; QUEIROZ, G.; FERREIRA, K. Um ambiente para acesso e análise de trajetórias de uso e cobertura da terra. In: SIMPÓSIO BRASILEIRO DE SENSORIAMENTO REMOTO, 19., 2019. **Anais...** São José dos Campos: INPE, 2019. 43

ZIOTI, F.; FERREIRA, K. R.; QUEIROZ, G. R.; NEVES, A. K.; CARLOS, F. M.; SOUZA, F. C.; SANTOS, L. A.; SIMOES, R. E. A platform for land use and land cover data integration and trajectory analysis. **International Journal of Applied Earth Observation and Geoinformation**, v. 106, fev. 2022. Disponível em: <https://doi.org/10.1016/j.jag.2021.102655>. 5, 43

APÊNDICE A - USER SCENARIO: PRODUÇÃO DE PESQUISA REPRODUTÍVEL

Daniel é um estudante de doutorado do curso de pós-graduação em Sensoriamento Remoto. Ele é conhecido por ser curioso e sempre estar realizando testes com novos dados e métodos. Todas as suas análises são feitas utilizando a linguagem de programação R. Sua tese está sendo construída com base nos artigos que ele está publicando em eventos nacionais e revistas internacionais. Recentemente, enquanto Daniel realizava alguns testes de estimativa de propriedades de qualidade d'água com cubos de dados Sentinel-2/MSI, ele obteve resultados que o ajudam a apresentar a eficácia de um dos modelos de estimativa de qualidade d'água que ele está propondo em sua tese de doutorado. Partindo deste resultado, ele trabalhou durante dias para fazer a geração das estimativas com seu modelo em um cubo com toda a extensão espaço-temporal de sua região de estudo. Durante este processo, Daniel contou com a colaboração de seus colegas de laboratório.

Nessa colaboração, o processamento dos dados foi realizado em vários passos, considerando múltiplas máquinas do laboratório. Na primeira etapa, relativa à correção atmosférica, um colega o ajudou criando um *script* em Python que fazia a recuperação das cenas do cubo de dados de um serviço *SpatioTemporal Asset Catalog* (STAC) e aplicava a correção atmosférica utilizando o modelo 6SV. O código Python foi executado em um sistema operacional Ubuntu 18.04. Todos os resultados foram salvos em um serviço de armazenamento em nuvem.

Após o primeiro colega ter aplicado a correção atmosférica nas cenas, um segundo colega, fez o *download* dos dados corrigidos, para gerar máscaras de nuvem. Para tal, foi aplicado o algoritmo `Fmask 4.1` em todas as cenas. O processamento foi feito utilizando o sistema operacional Windows 10 com uma ferramenta de linha de comando criada com código Python. As bibliotecas e dependências deste segundo colega eram bem diferentes das aplicadas na máquina utilizada para a realização da etapa anterior.

Por fim, todos os dados gerados foram entregues para Daniel, que os centralizou, organizando-os em diretórios separados. Esta foi a primeira vez que todos os membros do laboratório trabalharam em uma estrutura de processamento colaborativa, onde cada integrante operou uma etapa do processo.

Semanas após o esforço de processamento, Daniel concluiu as análise e decidiu relatar os resultados obtidos em um artigo. O artigo foi escrito e submetido para uma

revista conceituada de sua área. De maneira surpreendentemente rápida, a submissão realizada por ele teve uma resposta positiva. O editor gostou do trabalho e achou relevante. Os três revisores que participaram do processo compartilhavam da opinião do editor. Por conta da alta qualidade dos resultados, o editor e os revisores, requisitaram a Daniel os códigos utilizados para a realização do processamento. Eles exigiram resultados reprodutíveis.

Daniel então conversa com seus colegas de laboratório e pede para eles enviarem os códigos utilizados em suas respectivas etapas de processamento. No momento da centralização, Daniel percebeu problemas. A etapa de correção atmosférica foi feita utilizando código Python, enquanto a geração da máscara de nuvem foi feita com uma ferramenta de linha de comando.

Percebendo que isso poderia ser um problema para o editor e os revisores entenderem o que foi feito, Daniel começou a organizar os *scripts*. Para isso, ele criou um documento com o passo a passo da execução. Após dias para configurar o ambiente com todas as dependências, ele decide fazer a execução do fluxo de processamento para algumas imagens, apenas para verificar se estava tudo certo. Após algumas horas processando, ele descobre que os resultados diferem dos gerados originalmente. A qualidade geral se manteve, porém, em alguns casos houve mudanças significativas. Daniel conversa com seus colegas e após horas tentando, eles percebem que não seria possível chegar nos mesmos resultados. Haviam diversos problemas para a reconstrução do ambiente original de processamento.

Um dos colegas ficou sabendo que uma nova plataforma *web* para suporte à reprodutibilidade colaborativa estava sendo desenvolvida em sua instituição. Os três colegas então acessam a plataforma e decidem refazer toda a execução, em um fluxo reprodutível.

Para começar a desenvolver o código de processamento, Daniel acessa a plataforma que possui uma interface simples e intuitiva e identifica que é necessário criar um *Project*. Para isso, ele teve que preencher alguns campos de metadados como título, descrição e as licenças que devem ser atribuídas aos códigos e dados gerados. Em seguida, é apresentada uma mensagem sobre a adição de um *Dataset*. A mensagem informa que a adição pode ser feita naquele momento, ou depois. Daniel decide fazer naquele momento. Selecionando essa opção, foi aberta uma página para a criação do *Dataset*, na qual estavam disponíveis duas opções. Na primeira opção, nomeada de *Service*, é possível declarar um serviço externo que será utilizado durante o desenvolvimento da análise. Como segunda opção, está disponível o *Storage*, na

qual é possível fazer o *upload* dos dados diretamente para a plataforma.

O primeiro *Dataset* adicionado por Daniel foi um *Service*, o qual ele utilizou para definir um serviço STAC para acesso às cenas do cubo de dados da região de interesse. Na escolha desta opção, foi informado a Daniel que ele estava assumindo um compromisso com a comunidade que iria utilizar a pesquisa quando pronta. Isto porque, as informações declaradas para o tipo *Service*, não são utilizadas para o *download* de dados, mas sim, para a composição dos metadados do *Project*. Assim, fica a cargo do usuário, no momento da análise, trabalhar com a mesma região declarada. Em seguida, ele fez a adição de um *Dataset* em modo *Storage*, no qual ele fez o *upload* de algumas tabelas de dados auxiliares para que seu modelo pudesse ser utilizado.

Ao finalizar essas etapas iniciais, Daniel decide novamente trabalhar em conjunto com seus colegas. Ele, então, compartilhou o *Project* que criou, com seus colegas. A realização desta ação foi feita através de um menu de compartilhamento na tela inicial do *Project*. Neste menu, ele pode inserir o *e-mail* de cada um de seus colegas.

O primeiro colega ficou responsável novamente em fazer a correção atmosférica. Para começar, ele acessou a plataforma e conferiu o convite de colaboração enviado por Daniel. Ele aceitou o convite sendo redirecionado para a página do *Project* compartilhado. Na página, ele vê uma opção que o permite criar um ambiente interativo para executar seus códigos. Como é a sua primeira utilização, lhe são apresentadas instruções de uso. Após a leitura, ele entende que para a garantia de reprodutibilidade no ambiente colaborativo, a plataforma utiliza um conceito de fluxo de operação. Basicamente, cada etapa desse fluxo representa um *script* executado. Após criada, ficam associadas a etapa o ambiente utilizado, o usuário que criou e o *script* que é executado. A ordem de execução, com relação às outras etapas, é definida no momento em que o *script* é executado.

Após a apresentação dessas informações, uma janela é aberta com a opção para iniciar um novo ambiente computacional, o qual será utilizado para criar e executar os *scripts*. Na plataforma, os ambientes são chamados *Computational Environment*. Na janela, são apresentadas *Computational Environment* prontos que permitem uso das linguagens de programação Python e R com Jupyter Notebook. Ele, então, seleciona o ambiente Python. Antes de finalizar, ele vê uma opção para a adição de alguns pacotes extras. Sabendo de suas necessidades específicas, selecionou a opção de customização e na janela aberta, especificou cada pacote e as versões que deveriam ser instaladas. Por fim, ele iniciou o ambiente.

Como é a primeira vez que ele acessa o ambiente, lhe foi apresentado instruções de utilização. É informado que o uso do ambiente não possui restrições, então é possível criar os `Jupyter Notebooks`, executar os códigos, instalar novos pacotes, alterar arquivos de configuração no ambiente, tudo como se fosse a própria máquina do usuário. É informado também que para a adição de uma etapa no fluxo de operação é necessário utilizar uma opção especial nomeada de `Registrar`, que está disponível na interface do `Jupyter Notebook`. Esta, deve ser utilizada toda vez que se tem o objetivo de gerar os resultados finais da etapa de processamento. A opção pode ser utilizada quantas vezes forem necessárias, já que isso não adiciona etapas diretamente no fluxo de operação. A etapa só é adicionada quando explicitamente o usuário acessa o painel no `Jupyter` e aceita o *script* executado como uma nova etapa do fluxo. Nos diretórios do ambiente, ele nota que há uma estrutura pré-definida, simples e intuitiva, para auxiliar o pesquisador na organização de seu projeto. Dentre os diretórios criados, um especial é aquele nomeado de "Dados", que armazena os *Datasets* e os resultados gerados no *Project*.

Após entender o fluxo de funcionamento, o colega de Daniel realiza a criação e execução de seu código. Ele salvou os resultados em um sub-diretório de "Dados" que permite leitura/escrita. Ao final do processo, ele gerou o conjunto de cenas processadas e prontas para serem passadas para a próxima etapa. Ele, então, vai até o painel do `Jupyter Notebook`, seleciona a opção de gerenciamento de execuções e adiciona a última execução que fez de seu *script* ao fluxo de operação.

Sendo notificado que novos resultados foram adicionados no *Project*, o segundo colega começa sua etapa de processamento. Mesmo utilizando uma ferramenta de linha de comando, o fluxo de uso da plataforma é o mesmo feito pelo colega no passo anterior, que criou um código em Python. Por ter dependências de versões bem específicas e antigas do Python, o segundo colega decide criar um *Computational Environment*. Para isso, ele acessa a página de criação de ambientes e coloca as versões antigas necessárias. Ao entrar no ambiente interativo, ele abre um terminal de linha de comando. Ao conferir o diretório de dados, nota haver, em modo "somente leitura", os dados gerados na etapa anterior de processamento. Ele então utiliza esses dados como entrada e faz a criação das máscaras de nuvem. Para o `Registro`, ele utilizou um comando disponibilizado pela plataforma no terminal, com nome `Registro-CLI`, que executou sua ferramenta de linha de comando e fez o registro das informações. Ao validar os resultados, ele salva a execução realizada como uma etapa nova do fluxo de operação.

Com a finalização das duas primeiras etapas, Daniel pode trabalhar na geração dos resultados finais. Por utilizar R, fez a criação de um *Computational Environment*. Após criar e acessar o ambiente, ele valida o código e então realiza o processamento e aplicação de seu modelo, gerando novamente os resultados do artigo.

Agora que o processamento está finalizado, Daniel decide que responderá ao editor com os novos materiais. Para isso, ele planeja exportar todos os códigos e resultados gerados e então fazer seu compartilhamento. Ao acessar a página de compartilhamento, ele nota que a plataforma tem várias opções possíveis. A primeira delas é exportar todo o material gerado em um arquivo `zip`. Outra opção é o compartilhamento somente leitura, em que se permite a outros usuários acessarem e executarem o projeto em modo "somente leitura", sem modificar os resultados originais. Há também a opção de enviar um `zip` para o `Zenodo`.

Com tantas opções, Daniel acha por bem o compartilhamento no `Zenodo`. Mas também associa o editor ao ambiente, para que ele possa fazer a execução por si próprio. Por fim, Daniel também faz a exportação dos resultados em um arquivo único no formato `zip`.

Essa foi a experiência de uso do Daniel na plataforma. Ele espera poder utilizá-la em mais trabalhos, por conta da comodidade oferecida no uso colaborativo.

APÊNDICE B - USER SCENARIO: NECESSIDADE DE REPLICAÇÃO DE PESQUISA REPRODUTÍVEL

Rogério é um pesquisador na área de Aprendizado de Máquina. Durante seu doutorado trabalhou com algoritmos de Aprendizado Profundo para a segmentação semântica de imagens de Sensoriamento Remoto. Recentemente, Rogério leu um artigo que utilizando de uma *MultiLayer Perceptron Neural Network*, obteve resultados satisfatórios na segmentação de imagens. O autor do artigo, lido por Rogério, fez algumas transformações simples nos dados e modificações na organização das camadas e acabou gerando bons resultados. Rogério, ficou muito interessado em verificar os resultados e possivelmente aplicará o modelo criado pelo autor em seus estudos. Atualmente, ele enfrenta problemas com a complexidade computacional envolvida na aplicação de seus algoritmos para segmentação de grandes quantidades de imagens, tendo problemas especialmente no Sentinel-2/MSI. O modelo desenvolvido pelo autor do artigo é mais simples e consequentemente mais rápido para o uso, o que poderia acelerar a pesquisa de Rogério.

Curioso com os resultados e querendo fazer testes, após ler o artigo, Rogério viu que o trabalho está publicado em uma plataforma de pesquisas reprodutíveis que sua instituição está desenvolvendo. Interessado, ele faz um cadastro para testar a plataforma e recebe a permissão para uso. Ao acessar a plataforma, uma interface simples e intuitiva é apresentada. Há um "*feed*" com as últimas análises reprodutíveis publicadas. Além disso, a barra de busca fica no centro da primeira página, da mesma forma como feito por outros sites, como no `Zenodo`. Usufruindo da simplicidade, ele digita na barra de pesquisa o nome do artigo que está procurando. Então, ele é redirecionado para a página de resultados. Na lista de artigos encontrados, o artigo procurado aparece em primeiro. Rogério nota também que, a página na lateral esquerda, apresenta um sistema de filtro por palavras-chave e ano, o que é útil para uma busca mais acurada, quando necessário. Ele então verifica o título do artigo, confirma os autores e depois acessa a página da análise do artigo que ele quer testar.

Ao acessar a página, Rogério vê vários conteúdos vinculados e separados por *frames*. Há um *frame* com o fluxo de operação utilizado para a obtenção dos resultados. Em cada etapa do fluxo estão vinculados os diferentes autores do artigo. Cada autor utilizou um ambiente diferente, variando entre as linguagens Python, Julia e R. No *frame* de dados, são listados 2 *Datasets*, ambas imagens do satélite-sensor Sentinel-2/MSI. Em um *frame* de visualização interativa são apresentadas as locali-

zações geográficas dos dados. O *frame* de código apresenta os `Jupyter Notebooks` da análise. Olhando para esses elementos, ele confirma serem as mesmas informações apresentadas no artigo, ao menos na quantidade e aparência de cada elemento, como dados e figuras.

Além dessas opções, na parte superior da página, Rogério nota a opção `Run`. Como ele está com pressa e quer testar o código, ele clica no botão de imediato. Ao clicar, lhe são apresentadas duas opções, `Execução imediata` ou `Execução por partes`. Na opção `Execução imediata`, todo o fluxo de operação definido para gerar os resultados do artigo são executados em um processamento *batch*, não oferecendo elementos interativos. Como o autor do artigo preparou tudo corretamente, além da execução imediata, esta opção permite a troca de parâmetros, o que pode ser utilizado para a replicação. Na segunda opção, `Execução por partes`, é oferecido o suporte a execução interativa. Como o processamento foi feito com vários ambientes, Rogério fica curioso em saber como a `Execução por partes` funciona. Então, ele clica no botão de ajuda. Na janela que foi aberta, é informado que para os casos onde há múltiplos ambientes, um painel especializado é disponibilizado para o usuário, permitindo navegação entre os ambientes.

Como está curioso e deseja explorar em detalhes o que foi feito, Rogério escolhe inicialmente a segunda opção. Ao fazer isso, ele é redirecionado para uma página onde na parte superior há o fluxo de operação completo, representado através de vértices (`Etapas`) e arestas (`Ligação entre etapas`). No total são apresentados três passos, conforme Rogério havia visto na página inicial. Ao mesmo tempo, é aberta uma janela de ajuda que informa a Rogério sobre a forma de uso. A informação passada para Rogério é que, ao clicar em um vértice dentre os que estão representados, o *frame* que está logo abaixo na página, abre uma tela para o `Jupyter Notebook` do ambiente.

Rogério então fecha a janela de ajuda e começa o trabalho, clicando na primeira etapa do fluxo e iniciando a execução. Ao final da execução da primeira etapa, ele nota que os resultados são os mesmos apresentados no artigo. Feliz com isso, ele decide ir navegando entre as etapas e executando parte a parte. Com as anotações disponibilizadas pelo autor nos notebooks, tudo fica mais simples de entender. Após horas escrutinando o código e o ambiente, ele confirma o modelo e assume concordar com as explicações do artigo. Sabendo que isso poderia melhorar seus resultados, ele começa então a pensar em fazer a replicação do modelo para seu contexto.

Para a replicação, ele decidiu ir para o modo `Execução imediata`. Ao clicar para

começar essa forma de execução, uma janela informa Rogério de que a `Execução imediata` permite a troca de parâmetros, mas isso pode não ocorrer em outros projetos. Isto porque, depende do pesquisador declarar quais os parâmetros de cada etapa do fluxo de operação, o que nem sempre é feito. Ele, então, se sente com sorte e vai em frente com a execução. Na janela que foi aberta, é informado que a troca de parâmetros, por uma questão de comodidade e reprodutibilidade da replicação, é realizada através de um arquivo no formato `YAML`. Além disso, novos dados devem ser inseridos através da adição de novos *Datasets*. Tendo essas informações, Rogério primeiro faz o *download* do *template* do arquivo `YAML` e começa a fazer sua edição. Após o preenchimento, ele faz o *upload* do arquivo, que é validado pela plataforma. Os pesquisadores que criaram a análise utilizaram um serviço *SpatioTemporal Asset Catalog* (STAC) para a aquisição dos dados. Assim, no arquivo `YAML` Rogério declarou apenas a consulta e a URL para um serviço STAC com os dados da região de estudo de Rogério. Isso evitou a criação de novos *Datasets*. Após estas configurações, ele executa a análise completa.

Após esperar um tempo, Rogério notou que os resultados estavam prontos para serem utilizados. Realizando a análise, ele percebeu que o modelo apresentado pelo autor do artigo que leu, realmente melhorou os resultados de seu trabalho, finalizando o processo em menos tempo. Esta foi a experiência de Rogério com a plataforma. Ele planeja utilizá-la mais vezes, inclusive, para iniciar uma colaboração com o autor do artigo que ele reproduziu e replicou.