UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS DA NATUREZA

CURSO DE GRADUAÇÃO EM CIÊNCIAS BIOLÓGICAS

TRABALHO DE CONCLUSÃO DE CURSO

Um sistema de armazenamento e distribuição de dados georreferenciados para o programa *Bioma Stats*

Gustavo Bilia Marques

BURI - SP
2025

UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS DA NATUREZA

CURSO DE GRADUAÇÃO EM CIÊNCIAS BIOLÓGICAS

TRABALHO DE CONCLUSÃO DE CURSO

Um sistema de armazenamento e distribuição de dados georreferenciados para o programa *Bioma Stats*

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências da Natureza, Universidade Federal de São Carlos, como exigência para obtenção do título de Bacharel em Ciências Biológicas e avaliação obrigatória da atividade curricular Trabalho de Conclusão de Curso

Orientador: Prof. Dr. Iuri Emmanuel de Paula Ferreira

Buri - SP
2025

Marques, Gustavo Bilia

Um sistema de armazenamento e distribuição de dados georreferenciados para o programa Bioma Stats / Gustavo Bilia Marques -- 2025.
56f.

TCC (Graduação) - Universidade Federal de São Carlos, campus Lagoa do Sino, Buri
Orientador (a): Prof. Dr. Iuri Emmanuel de Paula Ferreira
Banca Examinadora: Jorge Luís Rodrigues Pantoja Filho, Denner Augusto de Barros Fabiano
Bibliografia

1. Análise ambiental. 2. Geoprocessamento. 3. API. I. Marques, Gustavo Bilia. II. Título.

Ficha catalográfica desenvolvida pela Secretaria Geral de Informática (SIn)

DADOS FORNECIDOS PELO AUTOR

Bibliotecário responsável: Lissandra Pinhatelli de Britto - CRB/8 7539

FUNDAÇÃO UNIVERSIDADE FEDERAL DE SÃO CARLOS
COORDENAÇÃO DO CURSO DE CIÊNCIAS BIOLÓGICAS (CCCBIO-LS)
Rod. Lauri Simões de Barros km 12 - SP-189, s/n - Bairro Aracaçu, Buri/SP, CEP 18290-000
Telefone: (15) 32569030 - http://www.ufscar.br

DP-TCC-FA nº 43/2025/CCCBio-LS/CCN/R

Graduação: Defesa Pública de Trabalho de Conclusão de Curso

Folha Aprovação (GDP-TCC-FA)

FOLHA DE APROVAÇÃO

GUSTAVO BILIA MARQUES

UM SISTEMA DE ARMAZENAMENTO E DISTRIBUIÇÃO DE DADOS GEORREFERENCIADOS PARA O PROGRAMA BIOMA STATS

Trabalho de Conclusão de Curso

Universidade Federal de São Carlos – Campus Lagoa do Sino

Buri, 05 de dezembro de 2025

ASSINATURAS E CIÊNCIAS

Cargo/Função	Nome Completo
Orientador	Iuri Emmanuel de Paula Ferreira
Membro da Banca 1	Jorge Luís Rodrigues Pantoja Filho
Membro da Banca 2	Denner Augusto de Barros Fabiano

Documento assinado eletronicamente por **Iuri Emmanuel de Paula Ferreira**, **Docente**, em 10/12/2025, às 12:23, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Jorge Luis Rodrigues Pantoja Filho**, **Docente**, em 10/12/2025, às 14:25, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufscar.br/autenticacao, informando o código verificador **2102478** e o código CRC **241CAA81**.

Referência: Caso responda a este documento, indicar expressamente o Processo nº 23112.024353/2025-08 SEI nº 2102478

Modelo de Documento: Grad: Defesa TCC: Folha Aprovação, versão de 02/Agosto/2019

Documento assinado digitalmente
DENNER AUGUSTO DE BARROS FABIANO
Data: 17/12/2025 16:50:50-0300
Verifique em https://validar.iti.gov.br

Documento assinado digitalmente
JORGE LUIS RODRIGUES PANTOJA FILHO
Data: 18/12/2025 11:14:33-0300
Verifique em https://validar.iti.gov.br

Documento assinado digitalmente
IURI EMMANUEL DE PAULA FERREIRA
Data: 19/12/2025 16:46:00-0300
Verifique em https://validar.iti.gov.br

AGRADECIMENTOS

Agradeço imensamente à minha mãe, Cláudia Aparecida Prado Marques, e ao meu pai, Paulo Eduardo Bilia Marques, por todo o apoio e suporte em minhas decisões e por todo o esforço dedicado, ao longo da vida, à educação de seu filho.

Estendo minha gratidão aos meus avós pelo incentivo em seguir meus sonhos e por todo o conhecimento de vida transmitido, que vai além do campo acadêmico.

Agradeço também ao meu irmão e à minha irmã, que foram fonte de encorajamento e compreensão, proporcionando conversas e momentos de descontração valiosos.

Agradeço a toda a minha família pelo apoio incondicional e pelos votos de sucesso que me acompanham em toda a minha trajetória.

Agradeço a todos os professores que passaram por minha vida, deixando em mim um pouco do que há de melhor em cada um deles. Em especial, aos professores do Campus Lagoa do Sino, por sempre serem solícitos nos momentos mais difíceis.

Ao meu orientador, Prof. Dr. Iuri Emmanuel de Paula Ferreira, por ser o profissional exemplar que é, servindo-me de inspiração e admiração, e pela paciência e tempo dedicados a me instruir nos diversos desafios da graduação.

A todos os membros e colaboradores do Centro de Modelagem Estatística em Ciências Ambientais (CeMECA), por todos os momentos compartilhados e pelos desafios superados em conjunto, os quais levarei como aprendizado para a vida toda.

Por fim, agradeço a todos os amigos feitos ao longo da graduação pelo companheirismo e por tornarem esta jornada mais leve.

RESUMO

Com o aumento do volume de dados ambientais e a demanda por ferramentas eficazes e acessíveis para trabalhar com dados espaciais, é imprescindível o desenvolvimento de sistemas que melhorem o acesso e o processamento de dados georreferenciados, especialmente em pesquisas ambientais de larga escala. Nesse cenário, o *Bioma Stats* apresenta-se como uma ferramenta de código aberto, criada em linguagem R, com o objetivo de centralizar e automatizar análises ambientais. A execução do fluxo de trabalho do *Bioma Stats* em sua primeira versão exigia o download manual da sua base de dados diretamente da plataforma MapBiomas, requisitando assim que o usuário realizasse o download da coleção completa com os dados de uso e ocupação do solo do Brasil, sobrecarregando sua máquina com um espaço de armazenamento além do necessário para realização da análise do recorte territorial de interesse. Sendo assim, este trabalho apresenta a implementação de um sistema de armazenamento e distribuição de dados georreferenciados que funciona por meio de uma API RESTful desenvolvida com *Node.js*, *Express* e MySQL. O sistema orquestra a entrega seletiva apenas dos dados dos fragmentos territoriais de interesse do usuário, que estão armazenados em nuvem, substituindo a dependência de servidores centrais. A abordagem de utilizar requisições HTTP para comunicação entre sistemas valida a integração entre o ambiente R e a API web. Os resultados demonstram um ganho significativo no tempo de *download* em comparação ao método tradicional, além de maior resiliência do sistema e otimização do uso de recursos computacionais.

Palavras-chave: Geoprocessamento; Sistemas de Informação Geográfica; API RESTful; Análise Ambiental; Dados Geoespaciais; Linguagem R.

ABSTRACT

With the increasing volume of environmental data and the growing demand for effective and accessible tools to work with spatial information, the development of systems that improve the access to and processing of georeferenced data is essential. In this context, Bioma Stats emerges as an open-source tool built in the R language, aiming to centralize and automate environmental analyses. Executing the Bioma Stats workflow in its first version required manually downloading its database directly from the MapBiomas platform, forcing users to download the entire collection of land use and land cover data for Brazil and overloading their machines with unnecessary storage demands for analyzing their specific area of interest. Therefore, this work presents the implementation of a georeferenced data storage and distribution system that operates through a RESTful API developed using Node.js, Express, and MySQL. The system orchestrates the selective delivery of only the data corresponding to the user's territorial fragments of interest, which are stored in the cloud, thus replacing the dependency on centralized servers. The approach of using HTTP requests for communication between systems validates the integration between the R environment and the web API. The results demonstrate a significant improvement in download time compared to the traditional method, as well as greater system resilience and optimization of computational resource usage.

Keywords: Geoprocessing; Geographic Information Systems; RESTful API; Environmental Analysis; Geospatial Data; R Language

SUMÁRIO

1 **INTRODUÇÃO**..... 11

2 **OBJETIVOS**..... 13

3 **FUNDAMENTAÇÃO TEÓRICA**..... 14

3.1 Geotecnologias..... 14

3.2 Geoprocessamento..... 15

3.3 Sistema de Informação Geográfica 15

3.4 Dados georreferenciados..... 18

3.5 Plataforma MapBiomas..... 21

3.6 Linguagem de programação R..... 21

3.7 *Bioma Stats*……..... 22

3.8 API……..... 23

3.9 *Javascript e Node.js* ……..... 24

4 **METODOLOGIA**……..... 25

4.1 Coleta e processamento dos dados..... 25

4.2 Armazenamento e distribuição descentralizada..... 25

4.3. Arquitetura da API……..... 26

4.4 Stack Tecnológico Aplicado……..... 28

4.5 Detalhamento quanto ao Fluxo de operação……..... 29

5 **RESULTADOS E DISCUSSÕES**..... 35

6 **CONSIDERAÇÕES FINAIS**…..... 36

7 **REFERÊNCIAS BIBLIOGRÁFICAS** …..... 37

8 **APÊNDICES**…..... 41

1. INTRODUÇÃO

Diante das crescentes ameaças ao meio ambiente, compreender os padrões de uso e ocupação do solo tem se tornado um interesse de cada vez mais setores. A partir de uma análise ambiental, por exemplo, é possível identificar zonas prioritárias para conservação, bem como entender as principais ameaças desse ecossistema (Silva, 2015). Com o entendimento de como as atividades antrópicas e os processos naturais que constituem a paisagem interagem, podemos perceber os impactos que causam na biodiversidade, nos recursos hídricos e no clima (Oliveira et al., 2017).

Em um contexto global, existe um aumento contínuo do volume de informações ambientais coletadas ano após ano. Nesse cenário, a demanda por ferramentas que auxiliem a entender aspectos relacionados à estrutura do espaço geográfico e aos aspectos ambientais também tem aumentado (Souza, 2023). De acordo com Rodrigues (2023, p. 4), "o geoprocessamento trata de diversas técnicas empregadas na coleta, armazenamento, processamento, análise e representação de dados com expressão espacial, isto é, possíveis de serem referenciados geograficamente (georreferenciados)". As técnicas empregadas no geoprocessamento vão desde o uso de instrumentos manuais mais simples até a utilização de satélites de posicionamento e imageamento. Dentre as diversas técnicas que permeiam o geoprocessamento, o Sistema de Informação Geográfica ocupa lugar de destaque; por meio dele é possível trabalhar com grandes bases de dados espaciais.

O *Bioma Stats* é um programa SIG, de licença livre, desenvolvido em linguagem R pelo Centro de Modelagem Estatística em Ciências Ambientais (CeMECA) para automatizar e centralizar o fluxo de trabalho estatístico em análises ambientais (Octaviano, 2023). O programa fornece módulos para análise e visualização de informações geográficas e ambientais integradas, como gráficos, tabelas e mapas de uso e ocupação do solo. O *Bioma Stats* se diferencia de outros *softwares* SIGs, como o *Quantum Geographic Information System* (QGIS), por priorizar o uso de comandos, com essa automação o fluxo de trabalho é mais rápido e evita erros manuais.

Usualmente, o trabalho com dados georreferenciados requer uma gama de ferramentas específicas para execução de cada passo na modelagem ambiental. Uma das etapas iniciais para se fazer uma análise ambiental é a obtenção dos dados georreferenciados necessários

para o estudo. Um problema nesse sentido é que trabalhar com arquivos *shapefiles* e *rasters* de grandes dimensões demanda um alto poder de processamento e *download*. Este projeto apresenta um aperfeiçoamento do pacote *Bioma Stats*, através de um sistema de armazenamento e distribuição de dados georreferenciados para uso sob demanda dos usuários do programa.

2. OBJETIVOS

2.1 Objetivo Geral

Analisar, validar e documentar a arquitetura e funcionamento da API de orquestração do *Bioma Stats*, detalhando sua atuação como componente central no fluxo descentralizado de armazenamento e distribuição de dados geoespaciais do programa.

2.2 Objetivos Específicos

- Estudar a representação de dados geoespaciais em ferramentas SIGs, bem como sua relevância em contextos ambientais;
- Estruturar a arquitetura de uma *RESTful* API para comunicação entre duas aplicações;
- Descrever o fluxo de trabalho do *Bioma Stats* após a implementação da API.

3. FUNDAMENTAÇÃO TEÓRICA

Esta seção tem por objetivo estabelecer a base conceitual necessária para a compreensão deste trabalho. Inicialmente, são apresentados os conceitos sobre geotecnologias, geoprocessamento e SIG; em seguida, são expostas as diferentes formas de se retratar os dados georreferenciados; a seguir, a plataforma MapBiomas é abordada; posteriormente, são explanados conceitos sobre a linguagem de programação R; na sequência, o programa *Bioma Stats* é introduzido; e, por fim, apresenta-se a definição de API, *JavaScript* e *Node.js*.

3.1 Geotecnologias

Comumente, na História do Mundo as sociedades e os povos buscavam estratégias para representar as condições e observações referentes a um espaço na superfície terrestre. A localização de recursos naturais, infraestrutura instalada, controle territorial, entre outras informações, sempre fez parte das informações básicas que contribuíram para a gestão e o desenvolvimento dessas associações. A humanidade sempre foi habituada a se utilizar de desenhos e traços como forma de comunicação, antes mesmo da consolidação da escrita. Os esboços produzidos pelos povos primitivos em pedra, os mapas babilônicos e egípcios de aproximadamente 2.500 a.C., a cartografia grega de 600 a.C., entre outros, são alguns exemplos que evidenciam a estreita relação do homem em retratar as paisagens no espaço com a evolução das sociedades (SCOTTON, 2022).

A palavra tecnologia tem sua origem etimológica nos vocábulos gregos "*tekhné*", que significa técnica, arte, ofício, e "*logos*", que se refere a razão, argumento, discussão. Assim, etimologicamente, a palavra tecnologia remete aos conhecimentos, razões e argumentos em torno de uma técnica ou ofício (MAGRANI, 2018). De acordo com Viana (2011), a tecnologia pode ser designada, de forma genérica e direta, como conhecimento aplicado. Nesse sentido Rosa e Brito (2012) definem as geotecnologias da seguinte forma:

> Geotecnologias – conjunto de tecnologias para coleta, processamento, análise e disponibilização de informações com referência geográfica. São compostas por soluções de *hardware, software e peopleware* que juntas constituem-se em poderosos instrumentos como suporte e tomada de decisão. Dentre as geotecnologias podemos destacar: a cartografia digital, o sensoriamento remoto, o sistema de posicionamento global, o sistema de informação geográfica.

3.2 Geoprocessamento

O advento dos computadores na metade do século XX possibilitou o processamento acelerado de cálculos matemáticos e manipulação de grandes quantidades de dados. Com a ascensão de novas tecnologias em diversas áreas - fotogrametria, banco de dados, sensoriamento remoto, computação gráfica, projeto assistido por computador (*Computer Aided Design - CAD)* - associadas a disciplinas que desenvolveram conceitos, teorias e metodologias ligadas a questões espaciais, surgiu uma nova dimensão do conhecimento de caráter multidisciplinar, o geoprocessamento, agregando disciplinas como a Geografia, Cartografia, Geometria, Urbanismo, Geodésia entre outras (PEREIRA e SILVA, 2001). Com isso, o que antes era feito apenas em documentos e mapas em papel, passou a haver a possibilidade de uma análise mais completa que integrasse diversos mapas e diferentes dados (TEIXEIRA, 2016).

Teixeira (2016) define o geoprocessamento como sendo o "processamento informatizado de dados georreferenciados que utiliza programas de computador permitindo o uso de informações cartográficas (mapas, cartas topográficas e plantas) para que se possa associar coordenadas desses mapas, cartas ou plantas. Sua utilização oferece recursos para diversas aplicações".

3.3 Sistema de Informação Geográfica

Dentre as diversas tecnologias de geoprocessamento, temos o Sistema de Informação Geográfica (SIG) como uma delas. Rosa e Brito (2013) definem SIG "como um sistema destinado à aquisição, armazenamento, manipulação, análise, simulação, modelagem e apresentação de dados referidos espacialmente na superfície terrestre, integrando diversas tecnologias". O primeiro sistema deste tipo surgiu em 1964, no Canadá, sendo chamado de *Canadian Geographic Information System*, constituindo parte de um programa governamental para criar um inventário de recursos naturais. Os SIGs por serem um ferramenta poderosa tem tido sua utilização de maneira cada vez mais usual atualmente, no entanto, nos anos seguintes ao seu surgimento (1960-1970), por conta do alto custo e limitações dos equipamentos computacionais da época, essas ferramentas restringiam sua utilização a um grupo restrito de pessoas. A Tabela 1 apresenta a evolução dos Sistemas de Informações Geográficas (SIG), sintetizada a partir das informações descritas por [Emilia Hamada , 2021].

Tabela 1 - Ascensão dos Sistemas de Informações Geográficas (SIGs).

Período	Marcos Tecnológicos	Público Alvo
1960-1970	Origem do primeiro SIG com o Canadian Geographic Information System (CGIS); Refinamentos cartográficos; Avanços nos sistemas computacionais; Revolução na análise espacial;	Uso restrito a pesquisadores e profissionais em aplicações específicas.
1970-1980	Aumento da capacidade computacional; Desenvolvimento de tecnologias relacionadas: sensoriamento remoto, bancos de dados, cartografia digital, CAD, entre outros;	Uso ainda restrito a universidades, órgãos de pesquisa e pequenas empresas
1980-1990	Avanço significativo nos microprocessadores, reduzindo custos e aumentando a capacidade de memória em chips pequenos; Proliferação de *software*s de baixo custo, acessíveis para computadores pessoais	Ampliação do uso devido à proliferação de *softwares* para computadores pessoais; Emergência comercial do SIG como tecnologia amplamente utilizada.

Fonte: Adaptado de Emilia Hamada (2021).

No campo da análise ambiental o progresso tecnológico transformou a maneira como a sociedade entende a relação entre as atividades humanas e o ambiente natural (TEIXEIRA, 2016). O SIG é uma ferramenta capaz de gerenciar e mostrar informações de diferentes aspectos da mesma área geográfica, facilitando assim a visualização de diferentes combinações dos elementos que constituem um mesmo espaço simultaneamente, funcionando assim, como um moderador nas tomadas de decisões. Ao contrário do caráter estático dos mapas de papel, os SIG facilitam a atualização dos dados geográficos, visto que, as fontes de informações são interpretações da realidade obtidas do mundo real que estão em constantes alterações no meio.

Um SIG tem sua aplicabilidade em praticamente todas as atividades humanas, posto que, qualquer atividade é executada em alguma posição geográfica passível de ser definida (Ferreira, 2006). Dito isto, os SIGs encontram aplicação em uma ampla variedade de campos do conhecimento, abrangendo tanto as ciências sociais e humanas quanto as ciências exatas e biológicas. Cada usuário trabalha com os dados a partir das perspectivas e fundamentos específicos de sua área de atuação. Um mesmo programa SIG pode ser utilizado para fins

diferentes a depender do propósito final do seu usuário. O Quadro 2 apresenta algumas das principais aplicações dos Sistemas de Informações Geográficas (SIGs), conforme adaptado de Francisco (2014).

Tabela 2 - Aplicações práticas de Sistemas de Informações Geográficas (SIGs)

Finalidade	Objetivo	Exemplos Práticos
Projetos	Definição das características do projeto.	Projetos de loteamento. Projetos de irrigação.
Planejamento territorial	Delimitação de zoneamentos e estabelecimento de normas e diretrizes de uso.	Planos de manejo de unidades de conservação.
Modelagem	Estudo de processos e comportamentos.	Modelagem de processos hidrológicos.
Gestão	Gestão de serviços de utilidade pública.	Gerenciamento de rede de abastecimento.
Banco de dados	Armazenamento e recuperação de dados.	Cadastro urbano e rural.
Avaliação de riscos e potenciais	Identificação de locais susceptíveis à ocorrência de eventos específicos.	Mapeamento de riscos ou potenciais.
Monitoramento	Acompanhamento da evolução dos fenômenos espaciais.	Monitoramento da cobertura florestal e expansão urbana.
Logístico	Identificação de locais e rotas ideais.	Definição da melhor rota. Implantação de empreendimentos econômicos.

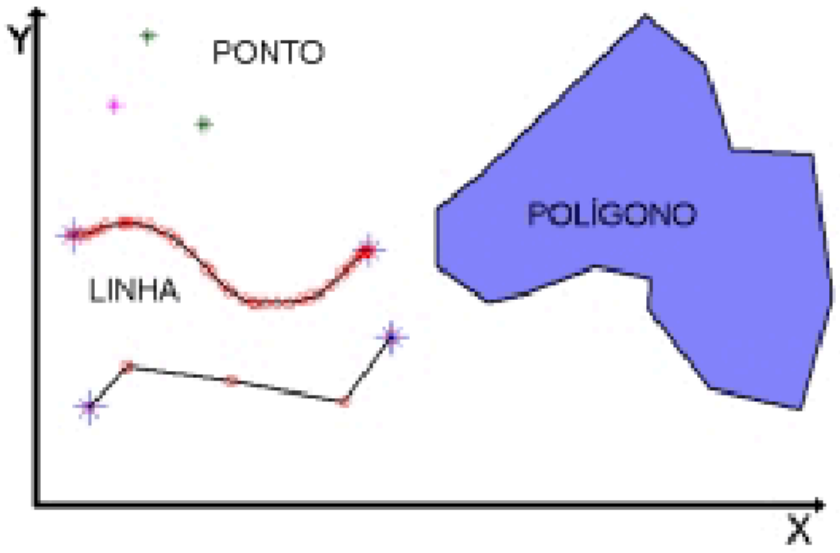
Fonte: Adaptado de Francisco (2014).

3.4 Dados Georreferenciados

Para Tennekes e Nowosad (2021) dados geoespaciais são informações que incluem componentes espaciais e geográficos, abrangendo diversos tipos de dados. Esses dados podem representar objetos com uma localização específica ou áreas delimitadas, geralmente descritas por coordenadas bidimensionais. Tradicionalmente, existem dois modelos amplamente conhecidos para se representar os dados espaciais: dados vetoriais e dados matriciais (*rasters*).

A representação digital do espaço geográfico através de dados vetoriais se dá estruturalmente baseada em elementos geométricos bem definidos. Conforme afirma Silva e Souza (2001), esse modelo é constituído por meio de três elementos primordiais: os pontos, as linhas e os polígonos.

Figura 1 - Componentes fundamentais de dados vetoriais (ponto, linha, polígono).



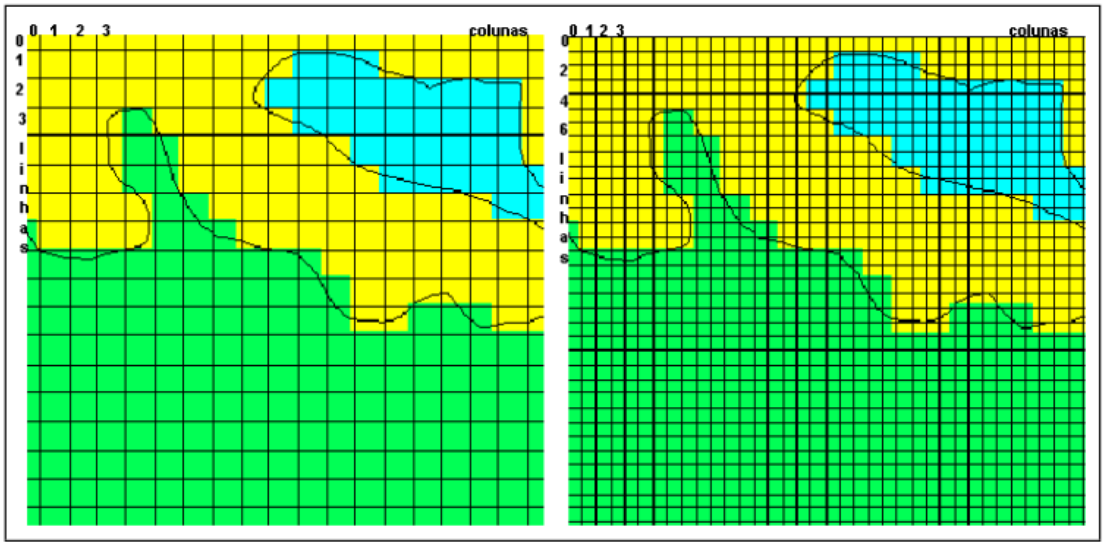
Fonte: Câmara e Monteiro (2001).

Nesse tipo de modelo, os pontos são definidos por um par de coordenadas espaciais (X, Y) que frequentemente representam locais específicos com latitude e longitude definidas, como uma residência. Os pontos podem estar conectados, formando linhas, e em geral representam elementos lineares como rios, redes hídricas e rodovias. No caso dos polígonos, eles são formados por linhas fechadas, ligadas pelo mesmo vértice inicial e final e costumam representar limites territoriais como a delimitação de propriedades, unidades de conservação e limites governamentais (Calles, 2021)

O *Shapefile*, por exemplo, é um formato vetorial amplamente utilizado em Sistemas de Informações Geográficas (SIG). Ele foi criado por volta de 1990, pela empresa ESRI, a mesma criadora dos produtos ArcGIS (ERSI, 1998). Um *Shapefile* é composto, no mínimo, por três arquivos principais: o *.shp*, que armazena as geometrias; o *.shx*, que contém o índice dessas geometrias; e o *.dbf*, que guarda os dados tabulares associados. Graças à sua simplicidade e compatibilidade, o *Shapefile* se consolidou como padrão em diversas aplicações cartográficas e análises espaciais (ESRI, 1998).

No que lhe diz respeito, aos dados matriciais ou *rasters*, eles são representações das informações do mundo real através de uma grade regular de células ou *pixels*, o qual possui 'n' linhas e 'm' colunas. Nessa representação, cada *pixel* da matriz corresponde a uma área no espaço que está atrelado a classes e/ou valores correspondente a área de estudo, podendo serem acessadas por suas coordenadas individualmente.

Figura 2 - Dois exemplos de representação matricial.



Fonte: Câmara e Monteiro (2001).

Em geral, a representação de fenômenos contínuos, como a elevação do terreno, temperatura, umidade, cobertura vegetal entre outros é realizada por meio de dados matriciais, que associam as características de cada ponto no espaço. Os dados matriciais são mais indicados para geração de álgebra de mapas, enquanto, os dados vetoriais possuem maior precisão geográfica, sendo indicados para análises representativas. A escolha do tipo de dado a se utilizar vai depender das necessidades da pesquisa e dos fenômenos analisados

(BARROS, 2022). A tabela comparativa abaixo sintetiza os diferentes aspectos das representações vetorial e matricial:

Tabela 3 - Comparação entre as duas principais formas de representação computacional para dados georreferenciados.

Aspecto	Representação Vetorial	Representação Matricial
Relações espaciais entre objetos	Relacionamentos topológicos entre objetos disponíveis	Relacionamentos espaciais devem ser inferidos
Ligação com banco de dados	Facilita associar atributos a elementos gráficos	Associa atributos apenas a classes do mapa
Análise, Simulação e Modelagem	Representação indireta de fenômenos contínuos, álgebra de mapas é limitada	Representa melhor fenômenos com variação contínua no espaço, simulação e modelagem mais fáceis
Escalas de trabalho	Adequado tanto a grandes quanto a pequenas escala	Mais adequado para pequenas escalas (1:25.000 e menores)
Algoritmos	Problemas com erros geométricos	Processamento mais rápido e eficiente.
Armazenamento	Por coordenadas	Por matrizes

Fonte: Adaptado de Câmara e Monteiro (2001) e Barros (2022).

No ambiente R, é possível criar e manipular esses tipos de dados georreferenciados utilizando pacotes específicos. Para dados vetoriais, os pacotes *sp* e *sf* são amplamente empregados, enquanto o pacote *raster* é ideal para trabalhar com dados matriciais. Essas ferramentas oferecem funcionalidades que facilitam a análise, visualização e integração de informações espaciais no ambiente de programação, ampliando as possibilidades para estudos geográficos e ambientais.

3.5 Plataforma MapBiomas

Nas últimas décadas, o avanço tecnológico proporcionou o acesso a imagens de satélites em altas resoluções, bem como o desenvolvimento de programas computacionais para análises ambientais, com isso, houve um avanço significativo nos estudos relacionados à gestão dos recursos naturais.

O Map Biomas é uma iniciativa do SEEG/OC (Sistema de Estimativas de Emissões de Gases de Efeito Estufa do Observatório do Clima) que busca promover o entendimento sobre a dinâmica do uso do solo nos biomas brasileiros, tendo início em 2015, o projeto opera de forma colaborativa com especialistas de ONGs, universidades, laboratórios e empresas de tecnologia, tendo como objetivo comum o desenvolvimento e implementação de uma metodologia rápida, consistente e de baixo custo para gerar mapas anuais de cobertura e uso do solo do Brasil, entre outros temas transversais, a partir de 1985 até os dias atuais.

Para enfrentar esse desafio, que exigia uma enorme capacidade computacional e um alto nível de automação, o MapBiomas firmou uma parceria técnica com a Google, usando a plataforma Google Earth Engine como base tecnológica. Essa ferramenta ofereceu a infraestrutura de processamento em nuvem e o acesso ao extenso catálogo de imagens de satélite necessários para criar as coleções anuais, que atualmente são a principal fonte de dados ambientais para o Brasil.

No contexto do *Bioma Stats*, o MapBiomas desempenha o papel de fonte primária dos dados ambientais. O MapBiomas disponibiliza dados brutos processados e classificados em escala nacional, enquanto o *Bioma Stats* oferece uma interface para que os pesquisadores possam se concentrar na análise e interpretação desses dados em áreas de interesse específicas. A implementação da API e do sistema de *tiles* no *Bioma Stats*, conforme descrito neste trabalho, melhora o acesso e o *download* seletivo dos dados provenientes do MapBiomas. Isso torna o fluxo de trabalho mais ágil, eficiente e fácil de usar, sem exigir o domínio de todo o complexo processo de manipulação de grandes arquivos raster.

3.6 Linguagem de Programação R

O R é uma linguagem de programação criada em 1992, por Ross Ihaka e Robert Gentleman, no departamento de Estatística da universidade de Auckland, Nova Zelândia, voltada para o trabalho com dados e análises estatísticas. O código fonte do R é um projeto sob licença GNU, esse tipo de licença é conhecido por sua distribuição de código aberto, isso significa que sua utilização para visualização, modificação e distribuição é livre para qualquer

pessoa ou empresa, não sendo permitido sua comercialização. O R é uma linguagem de programação multiparadigma, isso significa, que ela suporta múltiplos paradigmas de programação, o paradigma de linguagem orientada objetos, funcional e dinâmica permitem a resolução do mesmo problema com diferentes abordagens. A linguagem R é compatível com os principais sistemas operacionais, incluindo Linux, Windows e Mac OS.

A comunidade adepta ao R é bem ampla e ativa, além das funcionalidades nativas da linguagem, ela conta com uma biblioteca de pacotes em constante crescimento com a implementação de novas funções criadas por seus usuários. O ecossistema completo da linguagem R compreende uma extensa biblioteca de pacotes colaborativos, totalizando aproximadamente 1000 pacotes disponíveis.

A linguagem de programação R, desenvolvida inicialmente para um nicho estatístico, tem se consolidado como uma ferramenta versátil na utilização em uma ampla gama de aplicações, incluindo geotecnologias e sistemas de informações geográficas (SIGs). Sua característica de distribuição livre tem fomentado o desenvolvimento de pacotes especializados para o trabalho de dados geoespaciais. Pacotes como o *rgdal*, *rgeos*, *mapview*, *sp*, *ggplot* entre outros viabilizam o desenvolvimento de mapas, gráficos e tabelas para análises ambientais e estatísticas integradas ao mesmo ambiente de trabalho, de forma rápida e fácil.

3.7 *Bioma Stats*

O *Bioma Stats* é um pacote desenvolvido em linguagem R, de licença livre, desenvolvido pelo Centro de Modelagem Estatística em Ciências Ambientais (CeMECA) para automatizar e facilitar o processamento e análise de dados geospacias, ele é uma ferramenta SIG capaz integrar a análise de diferentes aspectos de uma área geográfica e fornecer tabelas, gráficos e mapas.

Inicialmente, o programa fornecia análises apenas de uso e ocupação do solo da área de interesse. Atualmente, o *Bioma Stats* possui funções para o cálculo de métricas da paisagem, que são maneiras de quantificar a composição e a configuração dos habitats, como índices de área, borda, fragmentação, diversidade da paisagem etc. Essa evolução permite aos usuários realizar diagnósticos ambientais robustos, identificando padrões espaciais e mudanças temporais de forma precisa por meio de índices como área de fragmentos, densidade de bordas e grau de fragmentação.

A inovação central, abordada neste trabalho, é a implementação de uma arquitetura descentralizada mediada por uma API. Esse sistema substitui a dependência de um servidor central por uma rede de usuários que armazenam e compartilham fragmentos de dados (*tiles*). Como resultado, o *Bioma Stats* torna-se mais resiliente, escalável e eficiente, permitindo que os usuários baixem apenas os recortes territoriais de seu interesse, otimizando tempo e recursos computacionais para análises sob demanda.

3.8 API

Usualmente, os softwares são manipulados pelos usuários via interfaces gráficas (GUI - *Graphical User Interface*). No entanto, os softwares não são usados apenas por pessoas mas também por outras aplicações, no contexto, da integração de sistemas. Nesse cenário, as Interfaces de Programação de Aplicações, popularmente conhecidas pela sigla API (*Application Programming Interface*), atuam como uma ponte na integração e comunicação de programas computacionais.

De acordo com Mulloy (2012), uma API funciona como um contrato entre quem oferece um serviço e o usuário consumidor, definindo claramente as requisições que podem ser feitas, seus formatos, quais dados são necessários e quais respostas são esperadas. No contexto de desenvolvimento web, as APIs RESTful (*Representational State Transfer*) tornaram-se um padrão amplamente adotado, operando através de requisições HTTP para operações de consulta (*GET*), criação (*POST*), atualização (*PUT*) e remoção(*DELETE*) de recursos.

Biehl (2014) complementa a definição de API dizendo que sua concretização vai além da simples implementação técnica, representando um modelo de arquitetura que define como os componentes do sistema interagem, como os dados são estruturados e transferidos, e quais políticas de segurança e governança devem ser seguidas.

No contexto do *Bioma Stats*, a API foi desenvolvida seguindo esses princípios, atuando como o orquestrador central do sistema descentralizado de armazenamento e distribuição de dados geoespaciais. Ela é caracterizada por ser um serviço web independente e externo ao *Bioma Stats* que pode ser acessado via ambiente R através de requisições HTTP, sendo responsável por gerenciar os metadados dos *tiles*, receber requisições dos usuários e identificar o semeador mais adequado para fornecer os fragmentos territoriais solicitados,

implementando assim uma arquitetura resiliente e eficiente para entrega de dados ambientais sob demanda.

3.9 *Javascrip*t e *Node.js*

Desde sua criação por Brendan Eich em 1995, o *JavaScrip*t passou por diversas mudanças importantes. Originalmente criado como uma linguagem de programação para páginas web, permitindo a adição de interatividade a páginas HTML estáticas, o *JavaScript* ampliou seu campo de atuação além do navegador, tornando-se uma das linguagens de programação mais versáteis e amplamente utilizadas no desenvolvimento de sistemas moderno (FLANAGAN, 2020).

A consolidação do *JavaScript* como uma linguagem *full-stack* foi favorecida pelo desenvolvimento do *Node.js* em 2009, por Ryan Dahl. O *Node.js* é um ambiente de execução para *JavaScript* server-side, desenvolvido com base no motor V8 do Google Chrome. Ao contrário de ambientes convencionais que impedem a execução enquanto esperam por operações de entrada e saída, como acesso a arquivos ou bancos de dados, o *Node.js* adota um modelo não bloqueante, possibilitando o atendimento de várias solicitações ao mesmo tempo. Essa particularidade o torna especialmente apropriado para aplicações que exigem atendimento de várias solicitações simultâneas, sendo apropriado para servidores web e APIs (VERDUTTI, 2022).

No contexto do desenvolvimento de APIs RESTful, como a implementada no *Bioma Stats*, o *Node.js* oferece vantagens significativas. Segundo Abramov (2018), o ecossistema *Node.j*s, combinado com frameworks como *Express.js*, proporciona uma base robusta para a criação de serviços web escaláveis. A escolha do *JavaScript/Node.js* para a API do *Bioma Stats* justifica-se entre outros fatores pela facilidade de integração com diferentes bancos de dados e serviços em nuvem e a compatibilidade nativa com o formato JSON (*JavaScript Object Notation*), padrão para APIs web modernas, que simplifica a comunicação entre o cliente R e o servidor *Node.js* (TEIXEIRA, 2021).

4. METODOLOGIA

4.1 Coleta e processamento dos dados

Shapefiles com as delimitações territoriais das unidades federativas do Brasil são carregadas automaticamente no programa *Bioma Stats* através do pacote R *geobr* (PEREIRA & GONÇALVES, 2022). O pacote se utiliza dos *shapefiles* provenientes do Instituto Brasileiro de Geografia e Estatística (IBGE) na escala de 1:250.000 (IBGE, 2024).

Os dados de uso e ocupação do solo foram obtidos diretamente da plataforma MapBiomas. Além da base de dados apresentada, o *Bioma Stats* também se utiliza do OpenSteetMap como fonte de dados complementares para atributos espaciais. O pré-processamento dos dados foi feito com o uso da linguagem R através do RStudio Desktop IDE. Os *shapefiles* delimitando o território brasileiro, obtidos previamente com o pacote geobr, foram ajustados para o sistema de coordenadas UTM (*Universal Transverse Mercator*), considerando o sistema geodésico WGS84. Essa transformação foi necessária para garantir a compatibilidade dos dados com os arquivos *raster* de uso e ocupação do solo fornecidos pela plataforma MapBiomas, originalmente definidos em Long-Lat WGS84.

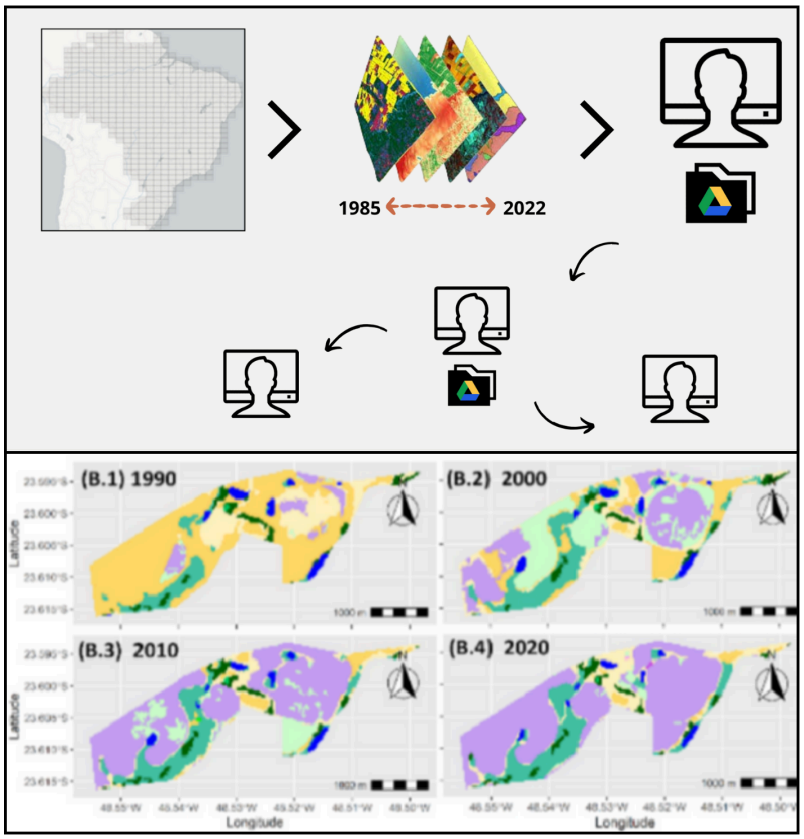
Os mapas anuais de uso e ocupação do solo (1985-2022) foram recortados e armazenados em formato GeoTIFF, resultando em pequenos fragmentos de dados georreferenciados, funcionando como *tiles* de informações espaciais, prontas para serem distribuídas para confecção de análises sob demanda.

4.2 Armazenamento e distribuição descentralizada

Os recortes em *tiles* são armazenados em drives públicos (Google Drive) por usuários pioneiros, esses são desenvolvedores do projeto que consentiram em armazenar e distribuir os arquivos para outros usuários, servindo como fontes de distribuição. O pacote R *googledrive* (D'agostino e Bryan, 2024) foi utilizado para estabelecer a conexão entre o programa R e o Google Drive de cada semeador. A partir dessa conexão, uma funcionalidade permite que o usuário especifique o *tile* de seu interesse e o obtenha no seu repositório de trabalho. Quando os usuários baixam um novo *tile*, o sistema os convida a se tornarem uma fonte de distribuição Se o usuário consente e autentica o acesso ao seu Google Drive, o sistema cria uma pasta *Bioma Stats* com cópias dos recortes *tiles* em formato GeoTIFF e coleta metadados não sensíveis que garantam a orquestração do armazenamento e distribuição desses arquivos.

Quando um novo usuário do *Bioma Stats* faz a requisição de um *tile*, o sistema irá buscar através da API qual semeador é o mais adequado para entrega do recorte desejado. Esse modelo descentralizado aumenta a resiliência do sistema e minimiza a dependência de servidores centrais, sendo especialmente eficiente para a distribuição de dados georreferenciados de grande volume.

Figura 3 - (Parte superior) Esquematização do sistema de distribuição descentralizada (*peer-to-peer*) de dados espaciais sob demanda do Programa *Bioma Stats*. (Parte inferior) Análises de uso e ocupação de solo produzidas pelo *Bioma Stats*.



Fonte: autoria própria (2024).

4.3 Arquitetura da API

Com o objetivo de permitir a comunicação entre pares na rede descentralizada do programa *Bioma Stats*, foi desenvolvida uma API que funciona como interface intermediadora entre os usuários e os arquivos georreferenciados armazenados na rede. Essa API é responsável por organizar as conexões, realizar buscas por fragmentos territoriais (*tiles*) e retornar os metadados associados aos arquivos compartilhados. Para viabilizar esse funcionamento, a aplicação foi construída com um conjunto de tecnologias modernas que garantem desempenho, escalabilidade e facilidade de integração com o ambiente R.

A arquitetura adotada segue o padrão RESTful, caracterizando-se por uma abordagem de interface de programação de aplicações que segue princípios arquiteturais REST(*Representational State Transfer*), utilizando protocolos web padrão para comunicação entre sistemas. Esse paradigma é caracterizado pela operação com recursos identificáveis por URLs, emprego dos verbos HTTP para operações CRUD (*Create, Read, Update, Delete*), sua formatação de dados é em JSON e possui uma natureza *stateless*, isso é, onde cada requisição contém todas as informações necessárias para seu processamento. Esta abordagem proporciona interoperabilidade, escalabilidade e desacoplamento entre cliente e servidor, tornando-se padrão predominante para integração de sistemas distribuídos na web moderna. O código-fonte completo da API está disponível publicamente no repositório GitHub (https://github.com/CreativeGrowDev/api_biomastats), permitindo transparência e colaboração da comunidade (CREATIVE GROW DEV, 2025). Esta escolha estratégica reforça o compromisso do projeto com a ciência aberta e a reprodutibilidade científica.

No contexto de APIs RESTful, os *endpoints* representam os pontos de acesso específicos através dos quais os clientes podem interagir com o serviço web. Cada endpoint corresponde a uma URL única que, quando acessada mediante um verbo HTTP específico (*GET* para consultas, *POST* para criação, *PUT* para atualizações e *DELETE* para remoções), desencadeia uma operação particular no servidor. Na arquitetura do *Bioma Stats*, os *endpoints* funcionam como interfaces bem definidas que estabelecem um contrato claro entre o pacote R e a API de orquestração, garantindo que as requisições para busca de *tiles* geoespaciais (/api/link) e o report de resultados de *download* (/api/res-app) sigam padrões consistentes de comunicação.

O funcionamento do *Bioma Stats* opera em uma arquitetura distribuída composta por dois componentes principais, o primeiro componente, a interface de usuário (*front-end*), se dá por meio do pacote propriamente dito, desenvolvido e usado em linguagem R, o qual emite requisições que são respondidas por uma API (*back-end*) orquestrada em linguagem *Javascript* para entrega de quadrículas atreladas a dados geoespaciais. O fluxo operacional segue a seguinte sequência:

(i) O usuário do *Bioma Stats* faz uma requisição através do endpoint /api/link, enviando os parâmetros fragment e year correspondentes à quadrícula desejada;

(ii) A API identifica a requisição e se conecta com a base de dados distribuída (Google Drive), consultando a tabela all_links para localizar o *tile* solicitado;

(iii) A API busca o link compartilhável específico de acordo com os identificadores da requisição e retorna o metadado completo do arquivo geoespacial;

(iv) Simultaneamente, através do *endpoint* /api/res-app, a API registra metadados sobre a ação na tabela sent-links, capturando informações como IP do requisitante, timestamp e status da operação;

(v) O *Bioma Stats* recebe o objeto geoespacial pronto para análise, permitindo que o usuário execute as funções de interesse.

4.4 Stack Tecnológico Aplicado

A seguir, são apresentadas as principais ferramentas utilizadas na implementação da API, bem como suas funções específicas dentro do sistema.

Tabela 4 - Descrição das tecnologias aplicadas na API do *Bioma Stats*

Tecnologia	Função
Node.js	Ambiente de execução da linguagem JavaScript no servidor
JavaScript	Linguagem de programação utilizada para implementar a lógica de negócio da API
Express.js	Framework para construção das rotas e controle do servidor
Sequelize	ORM (*Object-Relational Mapping*) para interação com o banco de dados
MySQL2	Driver para comunicação com o banco relacional MySQL
Dotenv	Gerenciamento de variáveis de ambiente que segura configurações sensíveis (credenciais DB, portas, chaves API)
CORS	Habilitação de acesso entre diferentes origens, permitindo a comunicação segura entre o *Bioma Stats* e a API.
Swagger UI Express	Geração de documentação interativa da API que estabelece automaticamente a interface de testes dos endpoints
Body-Parser	Interpretação do corpo das requisições HTTP, convertendo JSON em objetos JavaScript utilizáveis
Nodemon	Ferramenta de desenvolvimento que monitora alterações no código e reinicia automaticamente o servidor

Fonte: autoria própria (2025).

4.5 Detalhamento quanto ao Fluxo de operação

A execução do programa *Bioma Stats* é feita seguindo o tutorial que se encontra disponível na página do CeMECA Lagoa do Sino “https://www.cemeca.ufscar.br/dados-c%C3%B3digos-e-programas” (CeMECA Lagoa do Sino, 2024). O pacote *Bioma Stats* é instalado usando o comando devtools::install_github("iep-ferreira/biomastats"), diretamente no ambiente R, através do pacote *devtools*. Com essa etapa inicial o ambiente de desenvolvimento se sincroniza com os códigos do pacote *Bioma Stats* publicados de maneira livre na página do Github;

Em seguida, o usuário precisa definir um recorte do território brasileiro de interesse para análise, para isso o programa oferece duas opções (i) a importação de um arquivo *shapefile* ou (ii) definir um polígono customizado delimitando uma área. Para utilização da segunda opção, o usuário define através da função "make_polygon" um ponto de coordenada central (long-lat) no sistema WGS84, o diâmetro do polígono e o seu formato, podendo ser circular, quadrado ou hexagonal.

Após a definição do recorte de interesse, o programa necessita realizar o carregamento dos dados referentes à área de interesse através da função load_raster. Para isso, o programa fornece duas opções: o *download* automático ou o carregamento de forma manual.

Figura 4 - Código exemplificando as funções load_rasters para *download* automático e carregamento manual de dados no *Bioma Stats*.

```
mapas <- load_rasters(shape_path =  ufscar_shp, start = 1985, end = 2021, method = "download",
                      export_folder_path = "./meus-mapas/")
```

```
mapas <- load_rasters(shape_path = shp_path, start = 1985, end = 2021, method = "library",
                      import_folder_path = "./minha-colecao/")
```

Fonte: autoria própria (2025).

Quando o método de *download* é solicitado, entram em ação as requisições à API através do *endpoint* /api/link, que segue um protocolo baseado no ID do fragmento e no ano. Através desses parâmetros enviados via método POST no corpo da requisição, a API consulta seu banco de dados para identificar o *tile* correspondente armazenado no Google Drive de um administrador semeador do *Bioma Stats*. O *endpoint* processa a requisição, valida os parâmetros (Apêndice A - CreativeController.js), executa a busca na tabela all_links (Apêndice B - CreativeService.js) e retorna um objeto JSON contendo o shareable_link e metadados completos do arquivo geoespacial solicitado, enquanto registra simultaneamente a transação para auditoria no sistema.

O sistema é estruturado em camadas bem definidas, onde o arquivo de rotas (Apêndice C) configura os endpoints disponíveis e mapeia cada URL para seu respectivo controlador, enquanto o servidor principal (Apêndice D) gerencia as configurações do sistema, *middleware* de segurança e inicialização da aplicação, garantindo a comunicação adequada entre o cliente R e a API *Node.js*.

Tabela 5 - Especificações do endpoint da busca de *tiles*

Componente	Especificação	Função
Endpoint	/api/link	Ponto de acesso para busca de *tiles* geoespaciais
Método HTTP	POST	Permite envio de parâmetros complexos no corpo da requisição
Parâmetros	fragment, year	Critérios de busca para identificar o tile específico
Resposta	JSON com shareable_link e metadados	Retorna o link do Google Drive e informações do arquivo
Armazenamento	Tabela all_links	Catálogo central com metadados de todos os *tiles* disponíveis

Fonte: autoria própria (2025).

Espera-se que o código HTTP de resposta à solicitação seja 200, entregando assim o dado geoespacial pronto para análise sob demanda ao usuário do *Bioma Stats*. As respostas para as requisições estão explanadas na tabela a seguir:

Tabela 6 - Respostas HTTP do endpoint de busca de *tiles*

HTTP status	Significado	Quando ocorre
200 (OK)	Requisição bem-sucedida.	Quando os dados solicitados são encontrados e retornados corretamente.
400 (Bad Request)	Erro no pedido do usuário.	Quando faltam parâmetros obrigatórios (por exemplo, *fragment* ou *year*).
404 (Not Found)	Nenhum dado encontrado.	Quando não há registros correspondentes no banco de dados.
500 (Internal Server Error)	Erro interno do servidor.	Quando ocorre uma falha na comunicação com o banco ou outro erro inesperado.

Fonte: autoria própria (2025).

Quando a solicitação ao *endpoint* /api/link é bem-sucedida, um objeto JSON responde a solicitação, servindo como um contrato de comunicação que fornece a estrutura dos metadados que permite a localização automatizada do *tile* solicitado. A resposta contém dados fundamentais, como o identificador único do arquivo, nome do arquivo, fragmento e ano correspondentes, link compartilhável do Google Drive, *checksum* MD5 para verificação de integridade, além de metadados técnicos referentes à fonte e versão dos dados, como mostrado na Figura 5.

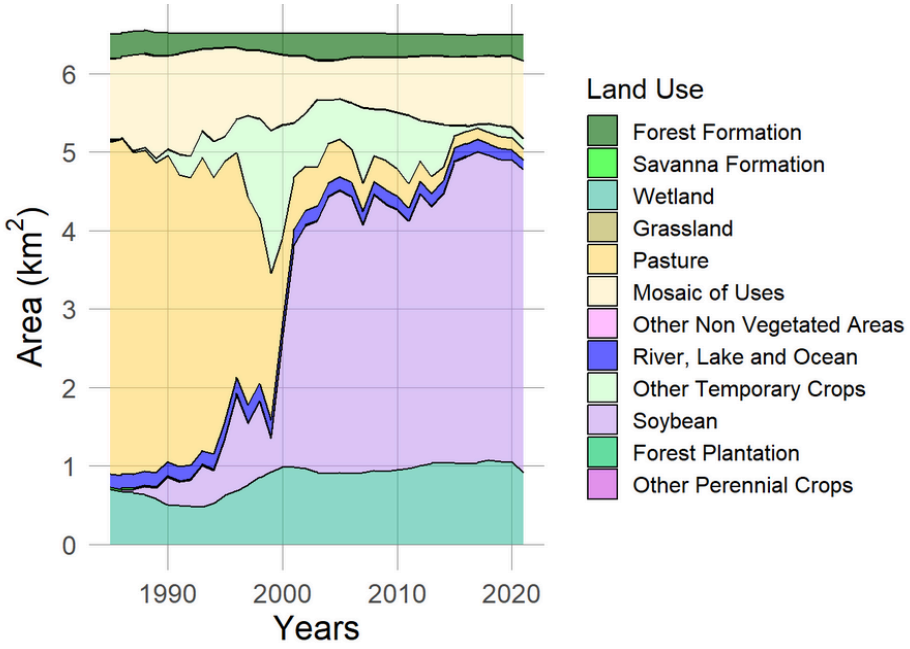
Figura 5 - Exemplo de resposta JSON bem-sucedida da API (/api/link)


```
{
  "error": "",
  "result": {
    "id": 74463,
    "file_name": "coverage-frag-19-year-2021.tif",
    "fragment": "19",
    "year": "2021",
    "info_id": "1FWAePDnEw29XMIj5P60dwnchXWbAdtFu",
    "shareable_link": "https://drive.google.com/file/d/1FWAePDnEw29XMIj5P60dwnchXWbAdtFu/view?usp=drivesdk",
    "md5_checksum": "6c2e062cd97823f5e63912b8b1b457a0",
    "source": "MapBiomas 2023",
    "type": "coverage",
    "version": "collection-7-integration-v1",
    "storage": "gdrive"
  }
}
```


Fonte: Autoria própria (2025).

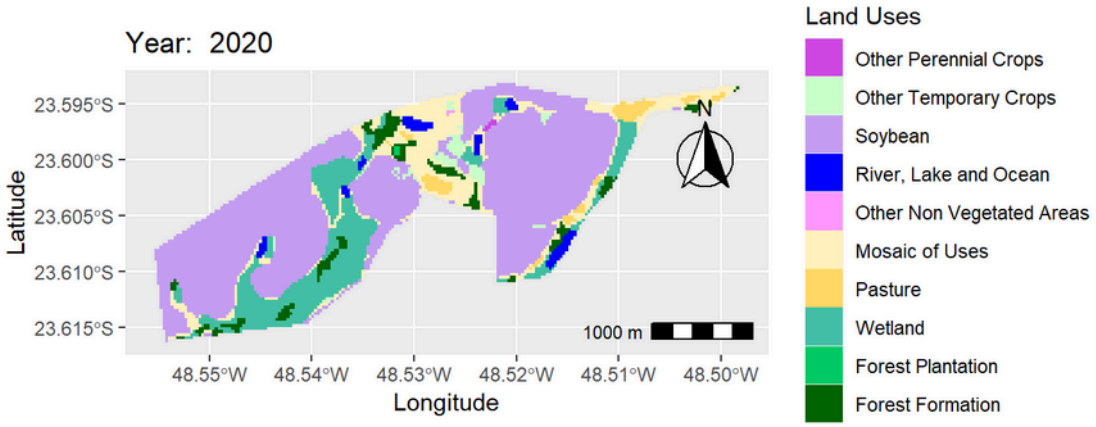
Com o *tile* baixado e carregado no ambiente R, o usuário pode realizar diversas análises ambientais integradas. Os objetos (.Rdata) podem ser utilizados no *Bioma Stats* através de funções nativas, como a land_vis e a get_area, para gerar recursos gráficos de uso e ocupação do solo da área de interesse. Para análises ecológicas, o módulo de métricas da paisagem oferece a função biomastats_metrics, que calcula automaticamente um conjunto abrangente de métricas, incluindo área total, número de fragmentos, comprimento e densidade de bordas, e índice de agregação. Os produtos a seguir foram gerados pelo programa para o recorte do campus Lagoa do Sino - UFSCar, localizado em Buri-SP.

Figura 6 - Gráfico de área mostrando a evolução temporal do uso e ocupação do solo no campus Lagoa do Sino - UFSCar (1990-2020).



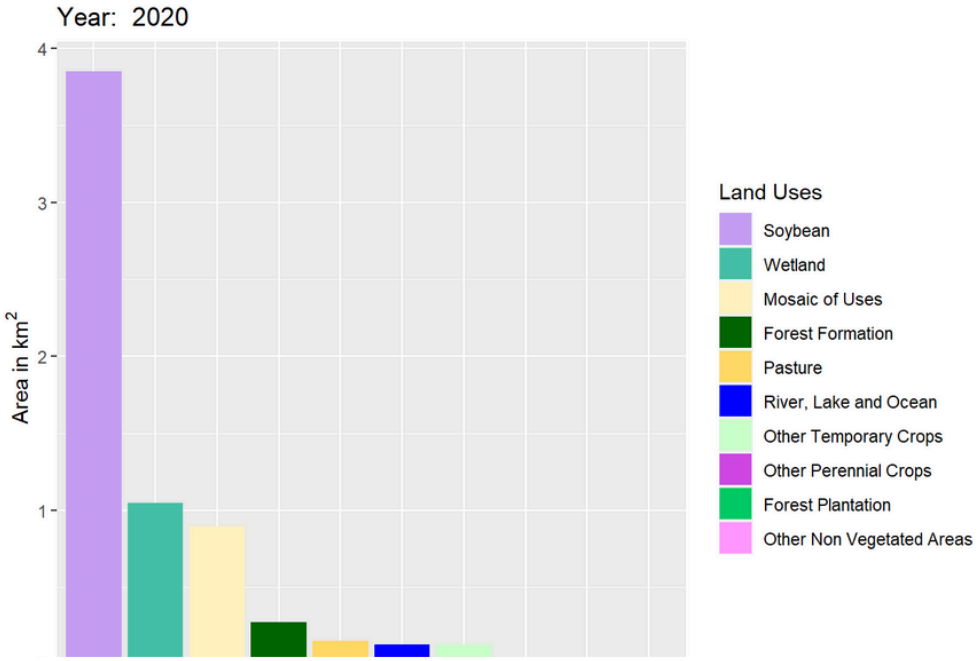
Fonte: CeMECA (2024). Tutorial *Bioma Stats*. Disponível em: https://iep-ferreira.github.io/biomastats/

Figura 7 - Mapa temático de uso e ocupação do solo do campus Lagoa do Sino - UFSCar no ano de 2020.



Fonte: CeMECA (2024). Tutorial *Bioma Stats*. Disponível em: https://iep-ferreira.github.io/biomastats/

Figura 8 - Distribuição das classes de uso do solo no campus Lagoa do Sino - UFSCar no ano de 2020.



Fonte: CeMECA (2024). Tutorial *Bioma Stat*s. Disponível em: https://iep-ferreira.github.io/biomastats/

Ao examinar as figuras 6, 7 e 8, observa-se que houve uma mudança significativa no uso do solo no campus Lagoa do Sino entre 1990 e 2020. Essa mudança foi marcada pela expansão das culturas agrícolas, especialmente da soja, em relação à diminuição acentuada das áreas de pastagem. Simultaneamente, as áreas de mata nativa permaneceram praticamente inalteradas, evidenciando a preservação dos fragmentos florestais. Por outro lado, as áreas de floresta plantada começaram a aparecer na paisagem a partir da década de 2000, indicando uma alteração no padrão de uso antrópico do território. A geração desses recursos gráficos conclui o fluxo operacional do *Bioma Stats*, demonstrando a capacidade do programa em processar requisições por *tiles* de dados geoespaciais e entrega-lós aos usuários em forma de informações ambientais prontas para análise e tomada de decisão.

5. RESULTADOS E DISCUSSÕES

Com a implementação da rede de armazenamento e distribuição de dados georreferenciados, seguindo uma arquitetura descentralizada, o programa *Bioma Stats* torna-se um um *software* mais resiliente e estável por não depender de servidores centrais. Em uma arquitetura linear, quando os servidores centrais estiverem desativados para uma manutenção ou por conta de algum erro, o usuário fica impossibilitado de obter os dados provenientes desse servidor. Na prática, o fornecimento de dados de uso e ocupação do solo do *Bioma Stats* seria ininterrupto, por conta de sempre se ter algum *peer* apto para fornecer os recortes de interesse previamente armazenados em seus drives.

Outro ponto relevante é que, com a implementação da rede, o usuário é capaz de fazer o *download* apenas do recorte de interesse, diminuindo, assim, o espaço de armazenamento utilizado pela máquina para realizar uma análise ambiental. Para contextualizar a eficiência do sistema descentralizado, foi realizado um teste comparativo de tempo de *download*. O download automático de um *tile* específico do recorte do Campus Lagoa do Sino via API do *Bioma Stats* foi concluído em aproximadamente 15-30 segundos, dependendo da conectividade da rede. Em contraste, o *download* manual da coleção completa do MapBiomas para a mesma área, seguindo o fluxo tradicional pela plataforma online, demandou entre 8-12 minutos para aquisição e processamento dos dados. Esta diferença de 16 a 24 vezes no tempo de *download* bruto dos dados evidencia a eficiência operacional alcançada pela arquitetura de *tiles* sob demanda. O ganho é ainda mais significativo considerando que o usuário do *Bioma Stats* obtém apenas os dados estritamente necessários para sua análise, enquanto o método tradicional exige o *download* de conjuntos de dados mais amplos que subsequentemente precisam ser recortados e processados localmente, consumindo recursos computacionais adicionais.

6. CONSIDERAÇÕES FINAIS

Este estudo mostrou a viabilidade e os benefícios da criação de um sistema descentralizado para a entrega sob demanda de análises ambientais no programa *Bioma Stats*. A arquitetura proposta, que utiliza uma API RESTful como orquestradora central de uma rede de armazenamento e distribuição de dados georreferenciados, provou ser eficaz para superar as restrições dos sistemas centralizados tradicionais, principalmente no que diz respeito à escalabilidade, resiliência e democratização do acesso a dados geoespaciais.

A API apresentada, utilizando tecnologias atuais como *Node.js, Express* e MySQL, demonstrou ser um elemento sólido para administrar metadados e coordenar a distribuição de *tiles* entre os usuários-semeadores. O sistema em funcionamento possibilita que os usuários acessem apenas as partes territoriais necessárias para suas análises específicas, o que otimiza o uso do armazenamento local da máquina e diminui consideravelmente o tempo de *download* e processamento.

Metodologicamente, a integração bem-sucedida entre o ambiente R do *Bioma Stats* e a API web confirma a estratégia de usar requisições HTTP para a comunicação entre sistemas e o ambiente R. A documentação interativa do sistema é assegurada por meio do Swagger UI, e a implementação de *endpoints* específicos garante a usabilidade e a clareza no funcionamento do sistema. Ademais, os códigos da API foram disponibilizados publicamente no GitHub, o que facilita o acesso para os interessados e contribui para a ciência aberta e reprodutível.

Como perspectivas futuras planeja-se expandir a API de forma que os usuários possam escolher a coleção do MapBiomas que desejam utilizar como fonte para sua análises (como a Coleção 8 e futuras atualizações subsequentes). A arquitetura atual permite a adição de novos metadados e *tiles* sem necessidade de modificações profundas no código. Em conclusão, este trabalho evidencia que soluções tecnológicas simples e eficientes podem ser desenvolvidas com recursos acessíveis, promovendo significativos avanços na maneira como a comunidade científica acessa e utiliza dados ambientais em grande escala.

REFERÊNCIAS BIBLIOGRÁFICAS

BIEHL, M. API architecture: the big picture for building APIs. San Francisco: API Academy, 2014. Disponível em: https://www.accorsi.net/docs/api-design-ebook-2012-03.pdf. Acesso em: 24 out. 2024.

CÂMARA, G.; MONTEIRO, A. M. V. Conceitos básicos em ciência da geoinformação. In: Introdução à ciência da geoinformação. São José dos Campos: INPE, 2001. p. 12–41.

CREATIVE GROW DEV. api_biomastats: API de orquestração para o programa Bioma Stats. Versão 1.0.0. GitHub, 2025. Disponível em: https://github.com/CreativeGrowDev/api_biomastats. Acesso em: 24 out. 2024.

D'AGOSTINO McGOWAN, L.; BRYAN, J. googledrive: an interface to Google Drive. R package version 2.1.1, 2023. Disponível em: https://CRAN.R-project.org/package=googledrive. Acesso em: 24 out. 2024.

FERREIRA, I. E. P.; et al. biomastats: pacote R para análise ambiental integrada. Versão 1.0.0. GitHub, 2025. Disponível em: https://github.com/iep-ferreira/biomastats. Acesso em: 24 out. 2024.

FERREIRA, N. C. Apostila de sistema de informações geográficas. Goiânia: Centro Federal de Educação Tecnológica de Goiás, 2006. Disponível em: https://www.faed.udesc.br/arquivos/id_submenu/1414/apostila_sig.pdf. Acesso em: 24 out. 2024.

FLANAGAN, D. JavaScript: O Guia Definitivo. 7. ed. Porto Alegre: Bookman, 2020. Disponível em: https://books.google.com.br/books?hl=pt-BR&lr=&id=zWNyDgAAQBAJ&oi=fnd&pg=PR1&ots=IBAez1O8mK&sig=XMxE7ma3EPtIXM9bBRo6cNutokA&redir_esc=y#v=onepage&q&f=false. Acesso em: 11 out. 2024.

MAPBIOMAS. Coleção 7 da série de mapas de cobertura e uso da terra do Brasil. Disponível em: https://mapbiomas.org. Acesso em: 20 out. 2024.

MAGRANI, E. A internet das coisas. Rio de Janeiro: Editora FGV, 2018. Disponível em: https://books.google.com.br/books?hl=pt-BR&lr=&id=qYtlDwAAQBAJ. Acesso em: 24 out. 2024.

MULLOY, B. Web API design. 2013. [Falta local e editora]. Disponível em: https://www.apigee.com/about/resources/ebooks/web-api-design. Acesso em: 24 out. 2024.

OCTAVIANO, K. F. Bioma Stats: um pacote R para análise de uso e ocupação do solo no território brasileiro. 2023. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Ambiental) – Universidade Federal de São Carlos, Centro de Ciências da Natureza, Buri (SP), 2023.

OLIVEIRA, R. C. M.; LIMA, P. V. P. S.; SOUSA, R. P. Gestão ambiental e gestão dos recursos hídricos no contexto do uso e ocupação do solo nos municípios. Gestão & Regionalidade, v. 33, n. 97, p. 48–64, 2017. Disponível em: http://www.redalyc.org/articulo.oa?id=133450056004. Acesso em: 24 out. 2024.

PEREIRA, G. C.; SILVA, B. C. N. Geoprocessamento e urbanismo. 2001. Disponível em: https://repositorio.ufba.br/bitstream/ri/7961/1/geopro.pdf. Acesso em: 24 out. 2024.

PEREIRA, R.; GONÇALVES, C. geobr: download official spatial data sets of Brazil. R package version 1.9.1, 2024. Disponível em: https://CRAN.R-project.org/package=geobr. Acesso em: 24 out. 2024.

PORTAL EMBRAPA. Sistema de informações geográficas. Disponível em: https://www.embrapa.br/agencia-de-informacao-tecnologica/tematicas/agricultura-e-meio-ambiente/politicas/analise-espacial/sistema-de-informacoes-geograficas. Acesso em: 24 out. 2024.

R CORE TEAM. R: a language and environment for statistical computing. Vienna: R Foundation for Statistical Computing, 2024. Disponível em: https://www.R-project.org/. Acesso em: 24 out. 2024.

RODRIGUES, M. C. Geoprocessamento. 2023. Trabalho de Conclusão de Curso (Técnico em Agropecuária) – Centro Estadual de Educação Tecnológica Paula Souza, Etec Professor Carmelino Corrêa Júnior, Franca.

ROSA, R.; BRITO, J. L. S. Introdução ao geoprocessamento. Uberlândia: Universidade Federal de Uberlândia (UFU), 2013. Disponível em: https://cetesb.sp.gov.br/wp-content/uploads/sites/33/2016/12/Introducao-ao-Geoprocessamento-Roberto-Rosa.pdf. Acesso em: 24 out. 2024.

SANTOS, Z. M. S. A. Tecnologia em saúde: aspectos teóricos-conceituais. In: Tecnologias em saúde: da abordagem teórica à construção e aplicação no cenário do cuidado. Fortaleza: EdUECE, 2016. p. 12–22. Disponível em: https://www.researchgate.net/profile/Carminda-Lamboglia/publication/311176731_O_VILAO_SE_TORNA_MOCINHO_UMA_PERSPECTIVA_INOVADORA_DA_UTILIZACAO_DAS_TECNOLOGIAS_DE_ENTRETENIMENTO_E_COMUNICACAO_PARA_A_PROMOCAO_E_PRATICAS_EM_SAUDE/links/58469f4f08ae2d21756e0f2b/O-VILAO-SE-TORNA-MOCINHO-UMA-PERSPECTIVA-INOVADORA-DA-UTILIZACAO-DAS-T...

SCOTTON, G. C. Os primeiros mapas: o Brasil na cartografia do século XVI. São Paulo: Editora Dialética, 2022. Disponível em: https://books.google.com.br/books?hl=pt-BR&lr=&id=YzZwEAAAQBAJ. Acesso em: 24 out. 2024.

SILVA, P. J. Uso e ocupação do solo urbano: uma análise dos impactos ambientais nas áreas de dunas no bairro de Felipe Camarão/Natal-RN. Holos, v. 5, p. 91–103, 2015. Disponível em: http://www.redalyc.org/articulo.oa?id=481547288010. Acesso em: 24 out. 2024.

SINGJAI, Apitchaka et al. Patterns on Designing API Endpoint Operations. 2021. Disponível em: https://eprints.cs.univie.ac.at/7194/. Acesso em: 11 out. 2024.

SLABADACK, A. R.; DOS SANTOS, M. Implementação do método multicritério MABAC na linguagem R: uma ferramenta para tomada de decisão. Disponível em: https://www.researchgate.net/profile/Marcos-Santos-85/publication/376093013_IMPLEMENTACAO_DO_METODO_MULTICRITERIO_MABAC_NA_LINGUAGEM_R_UMA_FERRAMENTA_PARA_TOMADA_DE_DECISAO/links/656c96a93fa26f66f44d757f/IMPLEMENTACAO-DO-METODO-MULTICRITERIO-MABAC-NA-LINGUAGEM-R-pUMA-FERRAMENTA-PARA-TOMADA-DE-DECISAO.pdf. Acesso em: 24 out. 2024.

SOUZA, R. S. Análise de dados georreferenciados via R com a assistência de inteligência artificial: aplicações no estudo do uso e ocupação do solo. Trabalho de Conclusão de Curso

(Bacharelado em Engenharia Ambiental) – Universidade Federal de São Carlos, Centro de Ciências da Natureza, Buri (SP), 2023.

TEIXEIRA, C. API Design Patterns: Best Practices for Building RESTful APIs. 2021. Disponível em: https://github.com/goldbergyoni/nodebestpractices. Acesso em: 24 out. 2024.

TEIXEIRA, M. R. A utilização do geoprocessamento na análise ambiental e sua importância na tomada de decisões: um estudo sobre o uso do geoprocessamento em pesquisas sobre ocupação desordenada do solo. 2016. Disponível em: https://repositorio.uniceub.br/jspui/handle/235/7788. Acesso em: 24 out. 2024.

VERDUTTI, R. Node.js: Aplicações web real-time com Node.js. São Paulo: Casa do Código, 2022. Disponível em: https://books.google.com.br/books?hl=pt-BR&lr=&id=Wm-CCwAAQBAJ&oi=fnd&pg=PT7&dq=Node.js:+Aplica%C3%A7%C3%B5es+web+real-time+com+Node.js&ots=_fr_-mDoeM&sig=wxG7a6wvQhZYQMRjSXIbmp5IvGo&redir_esc=y#v=onepage&q=Node.js%3A%20Aplica%C3%A7%C3%B5es%20web%20real-time%20com%20Node.js&f=false. Acesso em: 11 out. 2024.

APÊNDICE A - Controlador principal dos endpoints da API (arquivo CreativeController.js do repositório público)

```
const CreativeService = require('../services/CreativeService');


module.exports = {
   logsAll: async (req, res) => {
      let json = { error: '', result: [] };
      let logs = await CreativeService.logsAll();


      for (let i in logs) {
         json.result.push({
            id: logs[i].id,
            url: logs[i].url,
            service: logs[i].service,
            created_at: logs[i].created_at,
            updated_at: logs[i].updated_at,
            ip: logs[i].ip,
            user_agent: logs[i].user_agent,
            referer: logs[i].referer,
         });
      }
      res.json(json);
   },
```

```
pushlinksAll: async (req, res) => {
    let json = { error: '', result: [] };
    let links = await CreativeService.pushlinksAll();

    for (let i in links) {
        json.result.push({
            id: links[i].id,
            file_name: links[i].file_name,
            fragment: links[i].fragment,
            year: links[i].year,
            info_id: links[i].info_id,
            shareable_link: links[i].shareable_link,
            md5_checksum: links[i].md5_checksum,
            source: links[i].source,
            type: links[i].type,
            version: links[i].version,
            storage: links[i].storage,
            collection: links[i].collection,
            data_type: links[i].data_type,
            resolution: links[i].resolution
        });
    }
    res.json(json);
},
```

```
buscarReferer: async (req, res) => {
    let json = { error: '', result: [] };
    let logs = await CreativeService.buscarReferer(req.params.referer);

    if (logs) {
        json.result = logs;
    }
    res.json(json);
},

responseApp: async (req, res) => {
    let json = { error: '', result: [] };

    const {
        shareable_link,
        file_name,
        storage,
        was_successful,
        request_application,
        collection,
        data_type,
        resolution,
        source
```

```
    } = req.body;

            const requesterIp = req.headers['x-forwarded-for'] || req.connection.remoteAddress ||
req.socket.remoteAddress || (req.connection.socket ? req.connection.socket.remoteAddress : null);

    if (!shareable_link || !file_name || !storage || !was_successful || !request_application) {
        json.error = 'The parameters were not properly sent';
        return res.status(400).json(json);
    }

    try {
        console.log("IP:", requesterIp);
        console.log("Link:", shareable_link);

        const link = {
            shareable_link,
            file_name,
            storage,
            was_successful,
            collection,
            data_type,
            resolution,
            source
        };
```

```
        await CreativeService.gravarRespostaApp(link, requesterIp, request_application);
        json.result = 'System response recorded';
        return res.status(200).json(json);

    } catch (error) {
        json.error = 'Error fetching link: ' + error.message;
        return res.status(500).json(json);
    }
},

buscarLink: async (req, res) => {
    let json = { error: '', result: [] };
    const { fragment, year } = req.body;

    if (!fragment || !year) {
        json.error = 'Missing fragment or year parameter';
        return res.status(400).json(json);
    }

    try {
        let links = await CreativeService.buscarLink(fragment, year);

        if (links && links.length > 0) {
            const randomIndex = Math.floor(Math.random() * links.length);
```

```
            const selectedLink = links[randomIndex];


            json.result = selectedLink;


                    const requesterIp = req.headers['x-forwarded-for'] || req.connection.remoteAddress ||
req.socket.remoteAddress || (req.connection.socket ? req.connection.socket.remoteAddress : null);
            const requestUrl = `${req.protocol}://${req.get('host')}${req.originalUrl}`;


            console.log("IP:", requesterIp);
            console.log("URL:", requestUrl);
            console.log("Link:", selectedLink);


            await CreativeService.gravarLinkEnviado(selectedLink, requesterIp, requestUrl);


            return res.status(200).json(json);
          } else {
            json.error = 'No links found';
            return res.status(404).json(json);
          }
        } catch (error) {
          json.error = 'Error fetching link: ' + error.message;
          return res.status(500).json(json);
        }
      },
```

```
  create: async (req, res) => {
    let json = { error: '', result: {} };

    const { url, user_agent, referer, ip, service, created_at } = req.body;

    if (url && user_agent && referer && ip && service && created_at) {
      let logId = await CreativeService.create(url, user_agent, referer, ip, service, created_at);
      json.result = {
        id: logId,
        url,
        user_agent,
        referer,
        ip,
        service,
        created_at,
        updated_at: created_at,
      };
    } else {
      json.error = 'Dados inválidos';
    }

    res.json(json);
  }
};
```

APÊNDICE B - Serviços de negócio e acesso ao banco de dados (arquivo CreativeService.js do repositório público)

```
const db = require('../db');


module.exports = {
  logsAll: () => {
    return new Promise((aceito, rejeitado) => {
      db.query('SELECT * FROM logs', (error, results) => {
        if (error) {
          rejeitado(error);
        } else {
          aceito(results);
        }
      });
    });
  },


  pushlinksAll: () => {
    return new Promise((aceito, rejeitado) => {
      db.query('SELECT * FROM `all_links`', (error, results) => {
        if (error) {
          rejeitado(error);
        } else {
          aceito(results);
```

```
            }
        });
    });
},

buscarReferer: (referer) => {
    return new Promise((aceito, rejeitado) => {
        db.query('SELECT * FROM logs WHERE referer=?', [referer], (error, results) => {
            if (error) {
                rejeitado(error);
            } else {
                aceito(results);
            }
        });
    });
},

buscarLink: (fragment, year) => {
    return new Promise((resolve, reject) => {
        db.query(
            'SELECT * FROM `all_links` WHERE fragment = ? AND year = ?',
            [fragment, year],
            (error, results) => {
                if (error) {
```

```
                reject(error);
            } else {
                resolve(results);
            }
        }
    );
  });
},

gravarLinkEnviado: (link, requesterIp, requestUrl) => {
  return new Promise((resolve, reject) => {
    const query = `
      INSERT INTO \`sent-links\` (
        sent_datetime,
        shareable_link,
        file_name,
        storage,
        status,
        fragment,
        year,
        requester_ip,
        request_url,
        collection,
        data_type,
```

```
        resolution,
        source
    ) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
`;

const values = [
    new Date(),
    link.shareable_link,
    link.file_name,
    link.storage,
    true,
    link.fragment,
    link.year,
    requesterIp,
    requestUrl,
    link.collection || null,
    link.data_type || null,
    link.resolution || null,
    link.source || null
];

db.query(query, values, (error, results) => {
    if (error) {
        reject(error);
```

```
            } else {
                resolve(results);
            }
        });
    });
},

gravarRespostaApp: (link, requesterIp, request_application) => {
    return new Promise((resolve, reject) => {
        const query = `
            INSERT INTO \`res_app\` (
                sent_datetime,
                shareable_link,
                file_name,
                storage,
                was_successful,
                requester_ip,
                request_application,
                collection,
                data_type,
                resolution,
                source
            ) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
        `;
```

```
        const values = [
            new Date(),
            link.shareable_link,
            link.file_name,
            link.storage,
            link.was_successful,
            requesterIp,
            request_application,
            link.collection || null,
            link.data_type || null,
            link.resolution || null,
            link.source || null
        ];

        db.query(query, values, (error, results) => {
            if (error) {
                reject(error);
            } else {
                resolve(results);
            }
        });
    });
},
```

```
  create: (url, user_agent, referer, ip, service, created_at) => {
    return new Promise((aceito, rejeitado) => {
      db.query(
        'INSERT INTO logs (url, user_agent, referer, ip, service, created_at) VALUES (?, ?, ?, ?, ?, ?)',
        [url, user_agent, referer, ip, service, created_at],
        (error, results) => {
          if (error) {
            rejeitado(error);
          } else {
            aceito(results.createId);
          }
        }
      );
    });
  }
};
```

APÊNDICE C - Configuração das rotas e endpoints da API (arquivo routes.js do repositório público)

```
const express = require('express');

const router = express.Router();

const swaggerUi = require('swagger-ui-express');

const swaggerDocument = require('./swagger.json');


const CreativeController = require('./controllers/CreativeController');


router.post('/link', CreativeController.buscarLink);


router.post('/res-app', CreativeController.responseApp);


router.post('/logs', CreativeController.create);


router.use('/api-docs', swaggerUi.serve);

router.get('/api-docs', swaggerUi.setup(swaggerDocument));


module.exports = router;
```

APÊNDICE D - Servidor principal e configurações do sistema (arquivo server.js do repositório público)

```
require('dotenv').config({path:'conf.env'});

const express = require('express');

const cors = require('cors');

const bodyParser = require('body-parser');

const swaggerUi = require('swagger-ui-express');

const swaggerDocument = require('./swagger.json');


const routes = require('./routes');


const server = express();


server.use(cors());


server.use(bodyParser.urlencoded({extended:false}));


server.use('/api', routes);


server.listen(process.env.PORT,()=>{

    console.log(`Server is running on port ${process.env.PORT}`);

})
```