MINISTÉRIO DA EDUCAÇÃO

SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE PERNAMBUCO CAMPUS GARANHUNS

COORDENAÇÃO DO CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

JOÃO PEDRO MONTEIRO PEREIRA

ECODATA:

PLATAFORMA WEB COMO FERRAMENTA PARA COMPILAÇÃO E COMPARTILHAMENTO DE DADOS AMBIENTAIS NO ESTADO DA PARAÍBA

GARANHUNS-PE, 2024

JOÃO PEDRO MONTEIRO PEREIRA

ECODATA:

PLATAFORMA WEB COMO FERRAMENTA PARA COMPILAÇÃO E COMPARTILHAMENTO DE DADOS AMBIENTAIS NO ESTADO DA PARAÍBA

Trabalho de conclusão de curso apresentado à Coordenação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Ciência e Tecnologia - IFPE-Campus Garanhuns, como parte das exigências para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Dr. Rafael Galvão Mesquita
Coorientador: Prof. Me. Humberto Beltrão da Cunha Júnior
Coorientadora: Profa. Dra. Daniele Jovem S. Azevêdo

Garanhuns - PE

2024

P436e Pereira, João Pedro Monteiro.
Ecodata: plataforma web como ferramenta para compilação e compartilhamento de dados ambientais no Estado da Paraíba. / João Pedro Monteiro Pereira ; orientador Rafael Galvão Mesquita ; Coorientador Humbero Beltrão da Cunha Júnior , Coorientadora Daniele Jovem S. Azevêdo, 2024.
83 f. : il.

Orientador: Rafael Galvão Mesquita.
Coorientador: Humbero Beltrão da Cunha Júnior
Coorientadora: Daniele Jovem S. Azevêdo
Trabalho de Conclusão de Curso (Graduação) – Instituto Federal de Pernambuco. Pró-Reitoria de Ensino. Diretoria de Ensino. Campus Garanhuns. Coordenação do Curso de Tecnólogo em Análise e Desenvolvimento de Sistemas. Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, 2024.

1. Banco de dados. 2. Ecossistemas – Paraíba – Banco de dados. 3. Armazenamento de dados – Ecologia – Paraíba. I. Título.

CDD 005.75

Riane Melo de Freitas Alves –CRB4/1897

JOÃO PEDRO MONTEIRO PEREIRA

ECODATA:

PLATAFORMA WEB COMO FERRAMENTA PARA COMPILAÇÃO E COMPARTILHAMENTO DE DADOS AMBIENTAIS NO ESTADO DA PARAÍBA

Trabalho de Conclusão de Curso, apresentado à Coordenação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, IFPE-Campus Garanhuns, como parte das exigências para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Garanhuns-PE, 15 de Março de 2024.

BANCA EXAMINADORA

Rafael Galvão Mesquita
Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco (IFPE)
Campus Garanhuns-PE

Genesis Jeferson Ferreira Pereira de Lima
Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco (IFPE)
Campus Garanhuns-PE

Jean Elder Santana Araujo
Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco (IFPE)
Campus Garanhuns-PE

Dedico este trabalho aos meus pais que sempre me incentivaram

AGRADECIMENTOS

Agradeço primeiramente a Deus, agradeço aos meus pais Cristiane Maria e José Ricardo e minha companheira Maria Aparecida por todo incentivo, aos meus amigos e professores do curso de TADS, por toda a troca de experiência e conhecimento ao longo da minha trajetória, ao Prof. Me. Humberto Beltrão da Cunha Júnior por toda orientação, apoio e acompanhamento, ao Prof. Dr. Rafael Galvão Mesquita por todo o apoio, ao financiamento do Projeto "PROGRAMA AQUADATA: ferramenta para o monitoramento e a gestão integrada dos recursos hídricos no Estado da Paraíba" (processo nº 3206/2021), aprovado através do EDITAL Nº 010/2021 - FAPESQ/PB - MCTIC/CNPq - PROGRAMA DE INFRAESTRUTURA PARA JOVENS PESQUISADORES / PROGRAMA PRIMEIROSPROJETOS – PPP, concedido pela Fundação de Apoio à Pesquisa do Estado da Paraíba (FAPESQ/PB), o qual é coordenado pela Profa. Dra. Daniele Jovem S. Azevêdo, permitindo a elaboração da plataforma ECODATA e a obtenção do conjunto de dados que foram integrados para teste da plataforma web.

“Se cheguei até aqui foi porque me apoiei no ombro dos gigantes.”

(Isaac Newton)

“A mente que se abre a uma nova ideia jamais voltará ao seu tamanho original.”

(Albert Einstein)

RESUMO

Este trabalho de conclusão de curso teve como objetivo desenvolver uma plataforma web para compilação e compartilhamento de dados de recursos hídricos no estado da Paraíba, visando contribuir para uma gestão mais eficiente.

A plataforma desenvolvida é composta por um banco de dados que armazena informações de natureza múltipla, incluindo: dados de ecossistemas aquáticos (bacias hidrográficas e seus ecossistemas constituintes), da paisagem e etnobiológicos. Os referidos dados são oriundos da compilação de informações obtidas a partir de pesquisas desenvolvidas no estado da Paraíba. Através de uma interface de simples acesso, os usuários podem visualizar, realizar download e upload de dados e gerar relatórios.

A plataforma permite que pesquisadores, técnicos e gestores possam compartilhar informações relevantes de forma estruturada sobre ecossistemas da região. Isso contribui principalmente para a colaboração e o compartilhamento de informações entre diferentes Instituições/Grupos de pesquisa que atuam no monitoramento de ecossistemas no Estado. Além disso, a funcionalidade da compilação estruturada fornecida pela plataforma, também possibilita a melhoria na gestão dos metadados e, consequentemente, na qualidade das informações geradas, podendo ser incluídas para a tomada de decisões mais acertadas no que concerne à gestão e ao manejo dos ecossistemas.

Embora a coleta sistemática de informações relativas à usabilidade da ferramenta não tenha sido realizada no escopo deste trabalho, o desenvolvimento das funcionalidades era seguido do feedback contínuo de profissionais e pesquisadores do setor ambiental.

Do ponto de vista técnico, a plataforma desenvolvida neste trabalho, denominada ECODATA, foi construída utilizando uma arquitetura moderna e eficiente, aproveitando tecnologias de ponta no desenvolvimento web. O frontend foi desenvolvido com ReactJS e Next.js, proporcionando uma interface de usuário responsiva e de alto desempenho. No backend, optou-se por Node.js e NestJS, garantindo uma API robusta e escalável. O MongoDB, um banco de dados NoSQL, foi escolhido para o armazenamento de dados, devido à sua flexibilidade no manejo de diferentes tipos de dados. Além disso, a infraestrutura serverless fornecida pela AWS foi utilizada para maximizar a eficiência operacional e reduzir custos, permitindo que a plataforma escale conforme a demanda sem a necessidade de gerenciamento de servidores físicos.

Espera-se que a plataforma desenvolvida possa ser uma ferramenta valiosa para a gestão de ecossistemas no estado da Paraíba, fornecendo uma fonte centralizada de

informações confiáveis e atualizadas, além de promover a colaboração e o compartilhamento de conhecimento entre os profissionais e pesquisadores da área, contribuindo assim para a gestão mais eficiente dos ecossistemas da região.

Palavras-chave: banco de dados, metadados, *serverless*

ABSTRACT

This thesis aimed to develop a web platform for the compilation and sharing of water resource data in the state of Paraíba, aiming to contribute to more efficient management.

The developed platform is composed of a database that stores information of multiple natures, including data on aquatic ecosystems (watersheds and their constituent ecosystems), landscapes, and ethnobiological data. These data come from the compilation of information obtained from research conducted in the state of Paraíba. Through a simple access interface, users can view, download, and upload data, and generate reports.

The platform allows researchers, technicians, and managers to share relevant information in a structured manner about the ecosystems of the region. This mainly contributes to collaboration and the sharing of information among different Institutions/Research Groups that are active in ecosystem monitoring in the State. In addition, the functionality of structured compilation provided by the platform also allows for improved metadata management and, consequently, the quality of the information generated, which can be included for more accurate decision-making regarding the management and handling of ecosystems.

Although systematic collection of information related to the tool's usability was not conducted within the scope of this work, the development of functionalities was followed by continuous feedback from professionals and researchers in the environmental sector.

From a technical perspective, the platform developed in this work, named ECODATA, was built using a modern and efficient architecture, leveraging cutting-edge web development technologies. The frontend was developed with ReactJS and Next.js, providing a responsive and high-performance user interface. For the backend, Node.js and NestJS were chosen, ensuring a robust and scalable API. MongoDB, a NoSQL database, was selected for data storage, due to its flexibility in handling different types of data. Additionally, the serverless infrastructure provided by AWS was used to maximize operational efficiency and reduce costs, allowing the platform to scale according to demand without the need for physical server management.

It is hoped that the developed platform will be a valuable tool for ecosystem management in the state of Paraíba, providing a centralized source of reliable and updated information, as well as promoting collaboration and knowledge sharing among professionals and researchers in the field, thus contributing to more efficient ecosystem management in the region.

Keywords: *database, metadata, serverless.*

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

AWS	*Amazon Web Services*
CI/CD	Integração Contínua e Entrega Contínua
ECR	*Elastic Container Registry*
ECS	*Elastic Container Service*
ER	Entidade-Relacionamento
ISO	*International Organization for Standardization*
MVP	*Minimum Viable Product*
ORM	*Object-Relational Mapping*
PDF	*Portable Document Format*
S3	*Amazon Simple Storage Service*
SIG	Sistemas de Informação Geográfica
SQL	*Structured Query Language*
SSD	*Diagrama de Sequência do Sistema*
SSR	*Server-Side Rendering*
UML	*Unified Modeling Language*
XLSX	*Microsoft Excel Open XML Spreadsheet*

SUMÁRIO

1 INTRODUÇÃO

Com o crescimento da população, o aumento do consumo dos recursos naturais e os desperdícios gerados pelo ser humano ao longo do tempo, tem sido observado que os efeitos da degradação do meio ambiente vêm sendo crescentes (Silva *et al.*, 2023). Com relação aos recursos hídricos, necessários para manutenção de todas as formas de vida do planeta, essa problemática tem tido um alto impacto, visto que a demanda por água aumenta a cada dia e a contaminação dos ecossistemas aquáticos é algo frequente (Jackson *et al.*, 2001).

As atividades humanas causam impactos ambientais diretamente ligados a processos de desequilíbrio de sistemas aquáticos, incluindo despejo de resíduos industriais e esgoto, contendo substâncias tóxicas que prejudicam organismos e ecossistemas (Ekelund; Häder, 2018). Esses problemas são agravados pelas mudanças climáticas e poluição, colocando os ecossistemas aquáticos entre os mais ameaçados do mundo (Fuoco; Giannarelli, 2019; Liu *et al.*, 2015).

Como forma de acompanhar as mudanças que ocorrem no ecossistema, em busca de mitigar os impactos ambientais, tem sido realizado o monitoramento desses ambientes. A partir do monitoramento, é possível obter dados acerca do ecossistema acompanhado, e, com a análise desses, gerar informações para que seja possível desenvolver práticas de gerenciamento e busca de soluções para a melhoria da qualidade ambiental.

Frequentemente, o monitoramento de ecossistemas inclui a avaliação, diversos dados, como dados abióticos (p.ex., temperatura, pH, turbidez, condutividade, partículas dos sedimentos) (Dudley *et al.*, 2018; Machado *et al.*, 2018) e bióticos (Hauer & Lamberti, 2006), que correspondem aos organismos vivos que compõem o ecossistema. Os dados bióticos podem ser avaliados por múltiplas métricas, como: abundância, densidade, biomassa, riqueza taxonômica, composição, *functional traits*, moleculares (Araujo *et al.*, 2023; Brantschen *et al.*, 2021; Gomes *et al.*, 2017; Martins *et al.*, 2021; Massei *et al.*, 2023). Apesar da importância que o acompanhamento dos ecossistemas representa atualmente, essa não é tarefa fácil, tendo em vista a complexidade de dados que o monitoramento fornece, e também pelo fato de que muitas vezes o processo de monitoramento é realizado de maneira isolada, gerando conflitos de informações. Esses entraves podem resultar em estratégias de gestão ineficientes, com risco de perda dos recursos ambientais.

No cenário atual, com o avanço da tecnologia, é possível vislumbrar a integração entre a tecnologia e a gestão ambiental. Ferramentas tecnológicas têm demonstrado ser um instrumento útil para a gestão de ecossistemas. Por exemplo, sensores ambientais são capazes

de medir diversos parâmetros, como qualidade do ar, temperatura e umidade (Singh *et al*., 2021). Sistemas de Informação Geográfica (SIG) possibilitam o mapeamento e análise de dados espaciais e geográficos, facilitando a visualização de padrões ambientais, podendo ser usados para o monitoramento (Vihervaara *et al*., 2017). A modelagem matemática ambiental é outra ferramenta que utiliza modelos matemáticos e computacionais para simular processos e prever possíveis impactos e avaliar consequências por meio de simulações (Jovem-Azevêdo *et al*., 2020; Jovem-Azevêdo *et al.*, 2022).

Adicionalmente, ferramentas web podem ser instrumentos de cooperação em pesquisas, facilitando o compartilhamento de dados, e possibilitando a elaboração de programas de monitoramento em um nível mais abrangente. Dessa forma, é possível adotar uma visão integrada que une várias fontes de dados e conhecimento, permitindo uma avaliação mais completa da saúde dos ecossistemas hídricos.

Por exemplo, trabalhos buscam integrar a internet das coisas (IoT – internet of things) ao monitoramento ambiental, a qual permite a coleta, análise e gestão de dados ambientais em tempo real (Fang *et al.*, 2014). Outra plataforma, COBWEB (Citizen OBservatory WEB), foi criada com o objetivo principal de facilitar a coleta, transmissão e análise de dados ambientais em tempo real com a finalidade de melhorar monitoramento ambiental, e utiliza redes de sensores sem fio, análise de dados em tempo real, tecnologias de comunicação de longo alcance, integrando dados para gerar uma visão abrangente do ambiente (Higgins *et al*., 2016).

Nesse sentido, o presente estudo teve como objetivo o desenvolvimento de uma plataforma web para compilação de dados ambientais (abióticos, bióticos e etnobiológicos) de ecossistemas no estado da Paraíba. Para a realização do projeto, foi utilizado o framework Next.js para o desenvolvimento do front-end e o Nest.js para o desenvolvimento do back-end. Além disso, a plataforma conta com uma funcionalidade que permite aos usuários compartilhar informações relevantes sobre diferentes ecossistemas da região, utilizando uma planilha padrão de dados. Para viabilizar essa funcionalidade, foi utilizado um AWS Lambda com Python, e Pandas para ler planilhas assim que enviadas ao AWS S3.

O desenvolvimento do presente estudo busca contribuir com uma ferramenta tecnológica voltada à padronização, compilação e compartilhamento de dados ecossistêmicos, estreitando informações entre Instituições/Grupos de Pesquisa de interesse. Ferramentas como a que foi desenvolvida neste trabalho, representam uma fonte centralizada de informações confiáveis e atualizadas, promovendo colaboração e o compartilhamento de conhecimento

que podem ser incluídos na formulação de estratégias e ações voltadas à gestão e manejo de ecossistemas de forma mais eficiente.

1.2 OBJETIVOS

1.2.1 GERAL

O objetivo geral deste trabalho é projetar e implementar uma plataforma web (ECODATA) interativa para a estruturação de dados múltiplos (abióticos, bióticos e etnobiológicos) oriundos de ecossistemas do estado da Paraíba. A plataforma visa oferecer uma interface acessível, destinada à inclusão, compartilhamento e análise de informações sobre a utilização e estado de conservação dos ecossistemas inclusos.

1.2.2 ESPECÍFICOS

Os objetivos específicos são descritos nas alíneas a seguir:

a) desenvolver a interface do usuário da plataforma utilizando o framework Next.js, garantindo uma experiência intuitiva e responsiva para os usuários;
b) implementar o *back-end* da plataforma usando Nest.js, assegurando a eficiência e segurança no processamento e armazenamento de dados;
c) integrar uma funcionalidade de envio de planilhas de dados para o AWS S3, utilizando AWS Lambda com Python e Pandas para processamento e análise dos dados enviados;
d) criar uma base de dados centralizada que consolida informações sobre ecossistemas da Paraíba, permitindo a fácil recuperação e visualização dessas informações;
e) promover a colaboração entre profissionais e pesquisadores da área, proporcionando um espaço para o compartilhamento de conhecimentos e experiências relacionadas à gestão dos recursos hídricos.

1.3 RELEVÂNCIA DO ESTUDO

Este estudo aborda a criação e implementação da plataforma ECODATA, um sistema web projetado especificamente para a compilação e compartilhamento de dados ambientais no Estado da Paraíba. A relevância deste trabalho é destacada por várias dimensões críticas, que refletem tanto a necessidade urgente de inovações tecnológicas na gestão ambiental quanto o

potencial impacto que tais inovações podem ter sobre a sustentabilidade e a conservação de ecossistemas, estas estão descritas nas seguintes alíneas:

a) importância Ambiental:
    - em um momento em que a pressão sobre os recursos naturais é crescente, e a necessidade de ações efetivas de conservação se torna cada vez mais urgente, a ECODATA surge como uma ferramenta vital para pesquisadores, gestores ambientais e o público em geral. A capacidade de acessar, analisar e compartilhar dados ambientais de maneira eficiente pode transformar significativamente a forma como as políticas de conservação são formuladas e implementadas no Estado da Paraíba.
b) avanço Tecnológico:
    - a aplicação de tecnologias web modernas para resolver problemas complexos de gestão de dados ambientais representa um avanço significativo na intersecção entre ciência da computação e ciências ambientais. A ECODATA é um exemplo prático de como soluções digitais podem ser empregadas para facilitar a gestão sustentável dos recursos naturais, promovendo uma maior integração entre dados científicos e ações de conservação.
c) colaboração e Compartilhamento de Conhecimento:
    - além de servir como uma plataforma para a gestão de dados, a ECODATA promove a colaboração interdisciplinar entre diferentes atores envolvidos na gestão ambiental. Ao possibilitar o compartilhamento de dados de forma aberta e acessível, a plataforma estimula a sinergia entre pesquisadores, contribuindo para um corpo de conhecimento ambiental mais rico e diversificado.
d) contribuições para Políticas Públicas:
    - por fim, a capacidade da ECODATA de compilar e disponibilizar dados ambientais detalhados e atualizados tem o potencial de influenciar positivamente a formulação de políticas públicas. Informações precisas e acessíveis são fundamentais para o desenvolvimento de estratégias eficazes de gestão de recursos hídricos, conservação da biodiversidade e mitigação dos efeitos das mudanças climáticas.

1.3.1 CONCLUSÃO

A relevância deste estudo transcende as fronteiras acadêmicas, apontando para aplicações práticas que podem beneficiar não apenas o Estado da Paraíba, mas também servir como modelo para outras regiões enfrentando desafios semelhantes na gestão de seus recursos naturais. Ao aliar tecnologia avançada com a urgente necessidade de ações ambientais efetivas, a ECODATA estabelece um novo paradigma na conservação ambiental.

1.4 METODOLOGIA

A metodologia adotada para o desenvolvimento desta plataforma web engloba várias etapas:

a) pesquisa e análise de requisitos;
   - realização de uma pesquisa abrangente para identificar as necessidades dos usuários finais e definir os requisitos funcionais e técnicos da plataforma.

b) design e desenvolvimento de interfaces;
   - utilização do Next.js para o design e desenvolvimento de interfaces de usuário que sejam de fácil acesso, intuitivas e adaptáveis a diferentes dispositivos.

c) desenvolvimento de *back-end* e integrações;
   - implementação do *back-end* com Nest.js, integrando funcionalidades como o envio e processamento de planilhas de dados no AWS S3 usando AWS Lambda, Python e Pandas.

d) testes e validação;
   - realização de testes unitários e exploratórios para garantir a funcionalidade, usabilidade e segurança da plataforma.

e) implementação e feedback;
   - lançamento da plataforma com monitoramento contínuo e coleta de feedback dos usuários para melhorias futuras.

f) documentação e treinamento.
   - desenvolvimento de materiais de documentação e treinamento para facilitar a adoção da plataforma pelos usuários finais.

Através desta metodologia, pretende-se assegurar o desenvolvimento eficiente de uma plataforma robusta e de fácil utilização, que atenda às necessidades de gestão dos recursos hídricos na Paraíba. Nesse propósito, foram realizadas reuniões de monitoramento com as principais partes interessadas, que incluiu pesquisadores e membros de grupos de pesquisa na área do Estado.

2 DOCUMENTAÇÃO

2.1 MODELOS DE CASO DE USO

> Quando você modela as interações de um sistema com seu ambiente, deve usar uma abordagem abstrata sem muitos detalhes. Uma maneira de fazer isso é usar um modelo de caso de uso. Como discutido nos capítulos 4 e 5, cada caso de uso representa uma interação com o sistema. Cada interação possível é nomeada em uma elipse, e a entidade externa envolvida na interação é representada por um boneco palito (SOMMERVILLE, 2011, p. 126).

Esse trecho do livro de Ian Sommerville, "Engenharia de Software", esclarece que um modelo de caso de uso é uma representação abstrata das interações entre um sistema e seu ambiente.

A abstração é crucial para evitar a complexidade desnecessária nas fases iniciais do projeto. O modelo de caso de uso captura todas as interações potenciais que o sistema pode ter com os usuários ou outros sistemas, que são representados de forma simplificada, muitas vezes por meio de elipses (para casos de uso) e "bonecos palito" (para atores, sejam eles usuários ou outros sistemas).

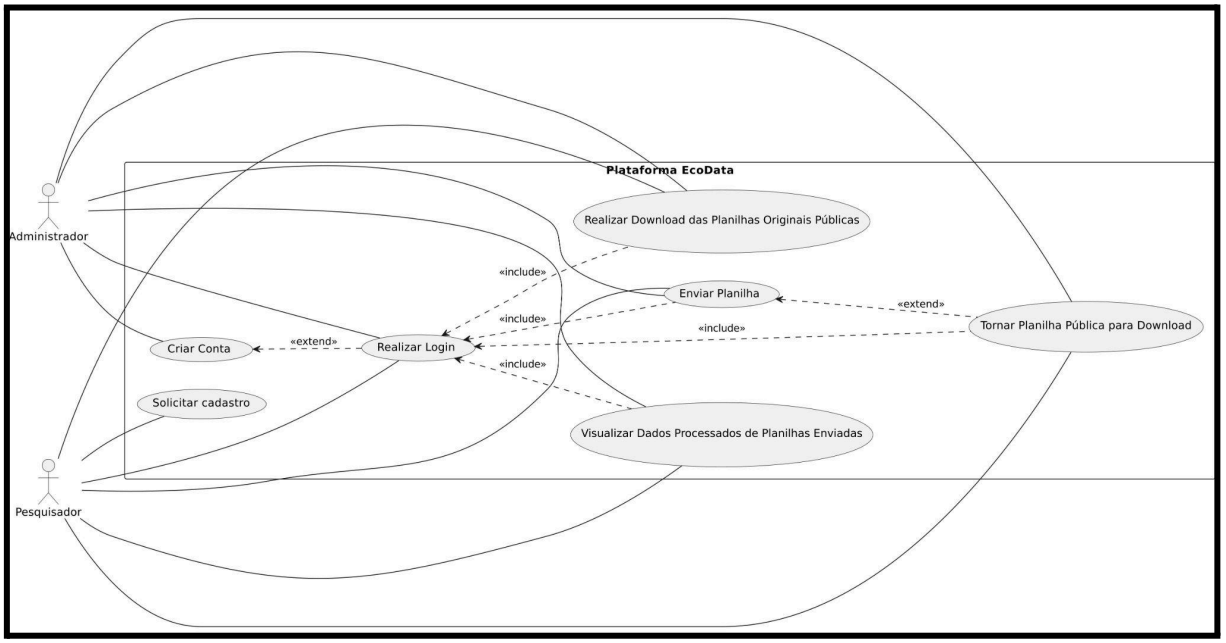
Os atores e os casos de uso da plataforma estão representados nas alíneas a seguir:

a) atores:
   - pesquisador;
   - administrador.

b) identificação dos casos de uso:
   - enviar planilha para processamento;
   - tornar planilha pública para download;
   - visualizar dados processados de planilhas enviadas;
   - realizar o download das planilhas originais públicas;
   - criar conta;
   - realizar login.

Segue o diagrama UML dos casos de uso do sistema (Figura 1).

Figura 1 - Diagrama de caso de uso da plataforma EcoData



Fonte: O autor (2023).

Conforme foi apresentado na Figura 1, temos os casos de uso para que a plataforma atinja seu objetivo. A documentação dos principais destes casos de uso é apresentada nos índices a seguir.

2.1.2 ENVIO DA PLANILHA

O modelo de caso de uso para a funcionalidade de envio da planilha para processamento está descrito nas seguintes alíneas:

a) ator:
   - pesquisador.

b) descrição:
   - o pesquisador que faz login na plataforma seleciona a planilha dos seus arquivos, envia, o sistema lê e processa a planilha e atualiza a lista de dados de planilhas disponíveis.

c) pré-condições:
   - o pesquisador deve estar autenticado na plataforma;
   - o pesquisador deve possuir uma planilha com informações relevantes para a gestão de recursos hídricos da região.

d) fluxo principal:

- o pesquisador seleciona a opção "Enviar planilha" na plataforma.
- o sistema exibe uma interface para seleção de arquivo;
- o pesquisador seleciona a planilha desejada em seus arquivos locais;
- o sistema carrega a planilha e verifica se ela é válida;
- o sistema processa a planilha, adicionando as informações relevantes ao banco de dados;
- o sistema atualiza a lista de planilhas disponíveis na plataforma.

e) pós-condições:
   - a planilha é processada com sucesso e suas informações são adicionadas ao banco de dados georreferenciado;
   - a lista de planilhas disponíveis na plataforma é atualizada.

Esse modelo de caso de uso descreve como o pesquisador pode enviar uma planilha de dados para a plataforma, permitindo que as informações sejam adicionadas ao banco de dados e compartilhadas com outros pesquisadores. O processo envolve seleção da planilha, processamento e atualização da lista de planilhas disponíveis na plataforma (veja APÊNDICE F e H para o fluxo de envio da planilha).

2.1.3 VISUALIZAÇÃO DOS DADOS PROCESSADOS DE PLANILHAS

Segue o modelo de caso de uso para a funcionalidade de visualização dos dados processados de planilhas enviadas:

a) ator:
   - pesquisador.

b) descrição:
   - o pesquisador acessa a plataforma e visualiza a lista de planilhas enviadas com o nome da planilha, data de envio, quem enviou e o ícone de olho (). Ao clicar no ícone de olho, o pesquisador é levado para uma página com um *data grid* com todos os dados processados daquela planilha com paginação.

c) pré-condições:
   - o pesquisador deve estar autenticado na plataforma;
   - o pesquisador deve ter enviado pelo menos uma planilha para processamento.

d) fluxo principal:
    - o pesquisador faz login na plataforma;
    - o sistema exibe uma lista de planilhas enviadas pelos pesquisadores, com informações como nome da planilha, data de envio e quem enviou;
    - o pesquisador seleciona a opção "Visualizar dados processados" disponível no ícone de olho para uma das planilhas enviadas;
    - o sistema exibe uma página com um *data grid* que contém todos os dados processados da planilha selecionada, com paginação;
    - o pesquisador pode navegar pelos dados e realizar buscas, filtragens e outras operações.
e) pós-condições:
    - o pesquisador pode visualizar todos os dados processados de uma planilha enviada por ele.
f) fluxo alternativo:
    - se o pesquisador não tiver enviado nenhuma planilha para processamento, a lista de planilhas não terá a planilha enviada por ele.

Esse modelo de caso de uso descreve como um pesquisador pode visualizar todos os dados processados de uma planilha enviada por ele. A funcionalidade é realizada a partir da tela inicial da plataforma, onde é exibida uma lista de planilhas enviadas pelo pesquisador. Ao selecionar a opção "Visualizar dados processados" para uma planilha específica, o pesquisador é levado para uma página com um *data grid* que contém todos os dados processados daquela planilha com paginação (veja APÊNDICE F e J para o fluxo de visualização dos dados processados da planilha).

2.1.4 TORNAR PLANILHA ORIGINAL PÚBLICA PARA DOWNLOAD

Segue o modelo de caso de uso para a funcionalidade de tornar planilha original pública para download:

a) ator:
    - pesquisador.
b) descrição:

- o pesquisador acessa a plataforma, entra na visualização dos dados da planilha e, caso a planilha tenha sido enviada por ele, há um select para torná-la pública ou não.

c) pré-condições:
   - o pesquisador deve estar autenticado na plataforma;
   - o pesquisador deve ter enviado a planilha que deseja tornar pública.

d) Fluxo principal:
   - o pesquisador seleciona a opção "Visualizar dados processados" clicando no ícone de olho na plataforma;
   - o sistema exibe os dados da planilha;
   - se a planilha foi enviada pelo pesquisador, o sistema exibe uma opção para torná-la pública ou privada;
   - o pesquisador seleciona a opção "Marque para tornar pública" e confirma a operação;
   - o sistema atualiza as informações da planilha, tornando-a pública e disponível para download.

e) pós-condições:
   - a planilha original é tornada pública e disponível para download na plataforma.

f) fluxo alternativo:
   - se a planilha não foi enviada pelo pesquisador, a opção para torná-la pública não é exibida;
   - se o pesquisador selecionar a opção "Desmarque para tornar privada", a planilha não estará mais disponível para download.

Esse modelo de caso de uso descreve como um pesquisador pode tornar uma planilha de dados original pública e disponível para download na plataforma. O processo envolve visualização dos dados da planilha, seleção da opção de torná-la pública, confirmação da operação e atualização das informações da planilha na plataforma (veja APÊNDICE F, J e L para o fluxo de tornar planilha pública).

Esta funcionalidade foi implementada em resposta aos *feedbacks* coletados durante as reuniões com pesquisadores e membros de grupos de pesquisa de diferentes Instituições do Estado. O objetivo é facilitar o compartilhamento de dados, permitindo que outros pesquisadores acessem e utilizem essas informações em suas próprias pesquisas. Essa abordagem promove uma maior colaboração e transparência no meio acadêmico, além de

enriquecer o potencial de descobertas e inovações ao disponibilizar dados originais para análises e estudos diversos.

2.1.5 BAIXAR PLANILHAS ORIGINAIS PÚBLICAS

Segue abaixo um modelo de caso de uso para a funcionalidade de baixar planilhas originais públicas:

a) ator:
   - pesquisador.

b) descrição:
   - o pesquisador faz login na plataforma e, na tela inicial, visualiza a listagem de planilhas. As planilhas que estiverem como públicas terão um ícone de arquivo com uma seta para baixo dentro dele, indicando download. Ao clicar no ícone, o pesquisador pode baixar a planilha original enviada pelo pesquisador que a enviou para processamento.

c) pré-condições:
   - O pesquisador deve estar autenticado na plataforma;
   - Deve haver pelo menos uma planilha pública disponível na plataforma.

d) fluxo principal:
   - o pesquisador faz login na plataforma;
   - o sistema exibe uma lista de planilhas, indicando quais são públicas com um ícone de arquivo com uma seta para baixo dentro dele;
   - o pesquisador clica no ícone de arquivo para realizar o download da planilha original enviada pelo pesquisador que a enviou para processamento;
   - a planilha é baixada para o computador.

e) pós-condições:
   - O pesquisador pode baixar a planilha original enviada por outros pesquisadores para processamento.

f) fluxo alternativo:
   - se não houver planilhas públicas disponíveis na plataforma, a lista de planilhas não terá nenhuma planilha com o ícone de download.

Esse modelo de caso de uso descreve como um pesquisador pode baixar as planilhas originais enviadas por outros pesquisadores que foram tornadas públicas na plataforma. A funcionalidade é realizada a partir da tela inicial da plataforma, onde é exibida uma lista de planilhas. As planilhas públicas têm um ícone de arquivo com uma seta para baixo dentro dele, indicando download. Ao clicar no ícone, o pesquisador pode baixar a planilha original enviada pelo pesquisador que a enviou para processamento (veja APÊNDICE F e K para o fluxo de download da planilha pública original).

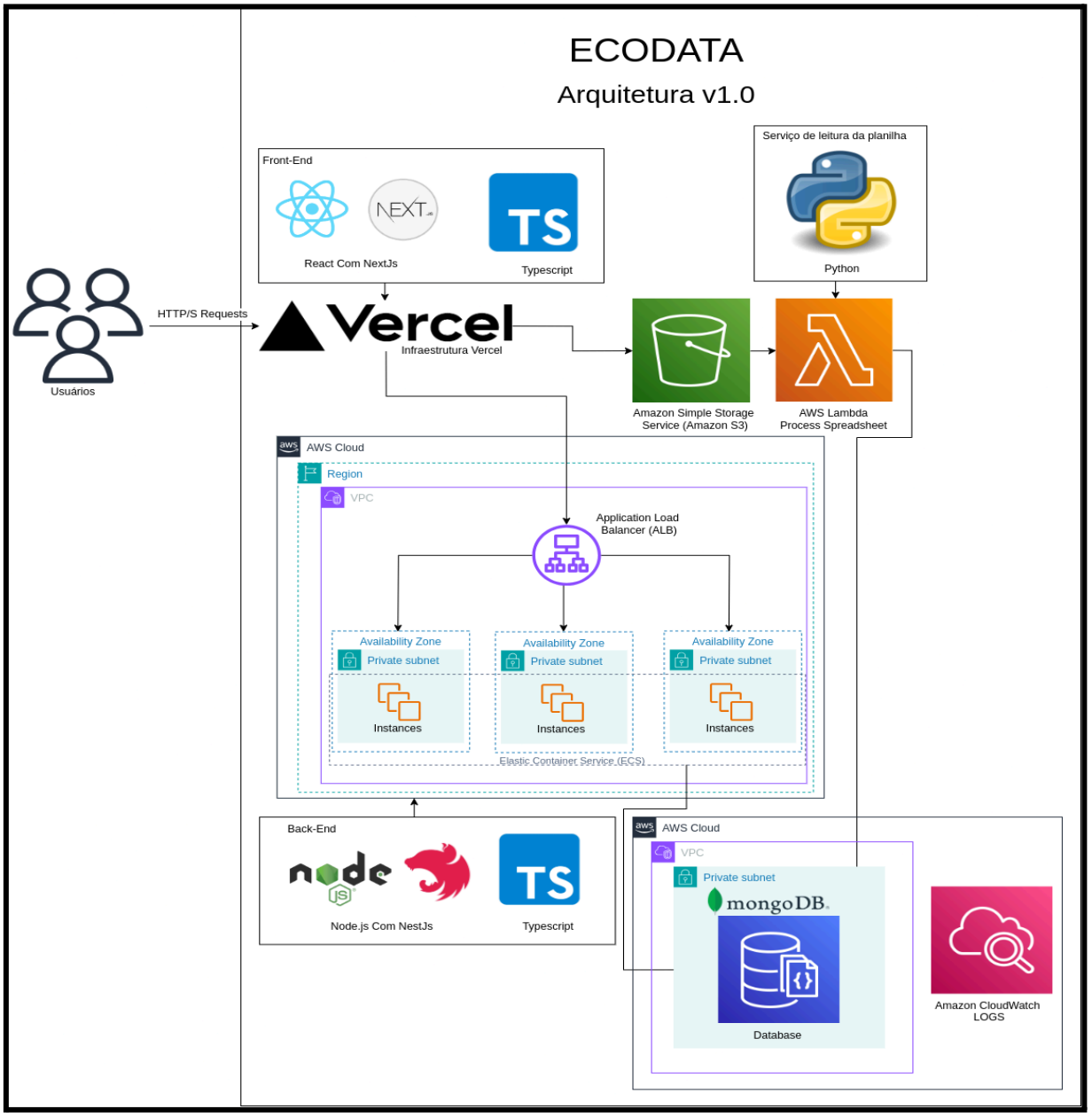
2.2 ARQUITETURA

> A arquitetura de um sistema de software é a forma dada a esse sistema pelos seus criadores. Essa forma está na divisão desse sistema em componentes, na organização desses componentes e nos modos como esses componentes se comunicam entre si. O propósito dessa forma é facilitar o desenvolvimento, implantação, operação e manutenção do sistema de software contido nela (Martin, 2019, cap. 15).

No trecho de "Arquitetura Limpa" de Robert C. Martin, a arquitetura de software é descrita como a estrutura fundamental imposta ao software pelos seus desenvolvedores, enfatizando a importância de uma organização lógica e funcional dos componentes do sistema. Esta estruturação cuidadosa não apenas determina como os componentes individuais interagem e se comunicam, mas também serve como alicerce para simplificar significativamente todos os aspectos do ciclo de vida do software, desde o desenvolvimento inicial até a manutenção e operação contínuas. A ênfase de Martin na divisão estratégica e na organização dos componentes sublinha a visão de que uma arquitetura bem planejada é crucial para a criação de sistemas de software robustos, eficientes e, acima de tudo, sustentáveis, permitindo que adaptações e melhorias sejam realizadas de maneira eficaz ao longo do tempo.

Para determinar as tecnologias utilizadas para que a plataforma funcione conforme o esperado, é importante termos uma arquitetura bem definida (Figura 2).

Figura 2 - Arquitetura do Aplicativo Web.



Fonte: O autor (2023).

Conforme apresentado na Figura 2, a arquitetura do ECODATA incorpora uma gama de tecnologias e serviços avançados. No *front-end*, utilizamos ReactJS e Next.js, enquanto o *back-end* é composto por Node.js, NestJS e Python. A infraestrutura se beneficia do uso da plataforma Vercel, do *Amazon Web Services* (AWS), incluindo o *Amazon Elastic Container Service* (ECS) e o *Application Load Balancer* (ALB) e do MongoDB.

Esta seção detalha a arquitetura desse aplicativo web, destacando sua capacidade de escalar e oferecer alto desempenho. A arquitetura foi cuidadosamente projetada para gerenciar a coleta, o processamento e o armazenamento de dados ecológicos oriundos de planilhas enviadas pelos usuários. A inclusão do AWS ECS e do ALB na infraestrutura contribui

significativamente para a escalabilidade e a eficiência na distribuição do tráfego, garantindo uma operação segura e resiliente do sistema. A combinação dessas tecnologias e serviços permite que o ECODATA atenda às demandas dinâmicas de carga, mantendo a integridade e a segurança dos dados.

2.2.1 FRONT-END

O *front-end* do aplicativo é desenvolvido usando ReactJS com Next.js, uma estrutura que facilita a criação de aplicativos React com *Server-Side Rendering* (SSR)[1]. SSR é uma técnica de renderização em que o conteúdo da página é gerado no servidor, em vez de ser totalmente processado no navegador do cliente. Isso significa que o servidor prepara o conteúdo das páginas em Hypertext Markup Language (HTML)[2] com todos os dados necessários e a envia pronta para o navegador, o que pode melhorar significativamente o tempo de carregamento inicial e a otimização para motores de busca. O Next.js permite uma experiência de carregamento rápido e otimiza o desempenho do aplicativo, fornecendo pré-renderização e renderização do lado do cliente. O aplicativo *front-end* é hospedado na Vercel, uma plataforma de implantação e hospedagem que suporta perfeitamente aplicativos Next.js.

2.2.2 BACK-END

O *back-end* é construído com Node.js, uma plataforma de tempo de execução JavaScript rápida e eficiente. O Node.js é combinado com o framework NestJS, que oferece uma arquitetura modular, escalável e com foco em TypeScript para construir aplicativos server-side[3].

Ao contrário de usar instâncias EC2 tradicionais para hospedagem, este sistema utiliza o ECS que é um dos serviços gerenciados da AWS para orquestração de contêineres, o que implica em uma abordagem baseada em contêineres para a implantação. Isso facilita a

[1] SSR é uma técnica em que o conteúdo de uma página web é gerado no servidor, em vez de no navegador do usuário. Isso permite que páginas inteiras sejam carregadas pré-renderizadas do servidor, melhorando o tempo de carregamento inicial e sendo mais eficaz para SEO.

[2] Hypertext Markup Language (HTML) é a linguagem de marcação padrão usada para estruturar conteúdo na web, definindo elementos como cabeçalhos, parágrafos, links e imagens, interpretados pelos navegadores para exibir as páginas.

[3] Server-side, ou "lado do servidor", refere-se à execução de processos e operações em um servidor, em vez de no dispositivo do cliente (como um navegador). Neste contexto, o servidor processa a lógica da aplicação, gerencia as interações do banco de dados e executa tarefas de *back-end*, enviando os resultados para o cliente.

escalabilidade automática e a gestão de infraestrutura, permitindo que a aplicação se ajuste de forma mais eficiente às variações na demanda.

2.2.3 SERVIÇO DE LEITURA DE PLANILHAS

Para lidar com a leitura de planilhas enviadas pelos usuários, é utilizado um serviço AWS Lambda, escrito em Python. O AWS Lambda é um serviço de computação sem servidor que permite executar código em resposta a eventos. Nesse caso, o Lambda é configurado para ser disparado sempre que uma planilha é enviada para o AWS S3 Bucket. O código Python no Lambda lê a planilha de dados ecológicos e executa as operações necessárias. Os dados processados são então enviados para armazenamento.

2.2.4 ARMAZENAMENTO DE DADOS

Os dados do usuário, bem como os dados ecológicos extraídos das planilhas, são armazenados em um banco de dados MongoDB. O MongoDB é uma base de dados NoSQL[4] altamente escalável e flexível, que oferece uma modelagem de dados ágil e suporte a consultas complexas.

Para garantir a integridade e a durabilidade dos dados, o MongoDB é executado em contêineres Docker dentro de uma instância do AWS EC2. O uso de contêineres Docker facilita a portabilidade e a escalabilidade do serviço de banco de dados. Crucialmente, para assegurar que os dados não sejam perdidos entre as reinicializações do contêiner ou instância, são utilizados volumes do Docker. Estes volumes funcionam como diretórios persistentes que estão fora do ciclo de vida padrão dos contêineres, armazenados no sistema de arquivos do *host*[5] EC2. Isso permite que os dados do MongoDB sejam mantidos seguros e acessíveis, mesmo quando o contêiner precisa ser parado, atualizado ou migrado, assegurando assim a resiliência, segurança e a disponibilidade dos dados.

---

[4] NoSQL, refere-se a uma categoria de sistemas de gerenciamento de banco de dados que diferem do modelo tradicional de bancos de dados relacionais baseados em *Structured Query Language* (SQL). Eles são projetados para lidar com grandes volumes de dados distribuídos e são conhecidos por sua flexibilidade na modelagem de dados, escalabilidade e capacidade de realizar consultas complexas e variadas.

[5] *Host*, no contexto de computação, refere-se ao sistema ou ambiente que aloja e executa aplicações ou serviços. Em uma infraestrutura de nuvem, como a AWS EC2, o host pode ser uma instância virtual que fornece os recursos de hardware e sistema operacional necessários para executar contêineres Docker e outras aplicações.

2.2.5 AUTENTICAÇÃO DO USUÁRIO

Para autenticação do usuário, os dados são armazenados no banco de dados MongoDB mencionado anteriormente. O NestJS é responsável por fornecer as rotas e lógica necessárias para autenticação e gerenciamento dos usuários. O MongoDB é usado para armazenar com segurança as informações de autenticação, como senhas criptografadas[6] e dados pessoais do usuário.

2.2.6 INFRAESTRUTURA E DISTRIBUIÇÃO DE CARGA

A arquitetura do ECODATA foi projetada para maximizar a eficiência operacional e a estabilidade, utilizando serviços de infraestrutura de nuvem para otimizar tanto o *front-end* quanto o *back-end*. No coração da infraestrutura de *back-end*, utilizamos o ALB da AWS, que desempenha um papel crucial na gestão do tráfego de entrada. O ALB é configurado para trabalhar em conjunto com o ECS, distribuindo automaticamente o tráfego de aplicações entre contêineres de serviços que são executados em um *cluster*[7] de instâncias de máquinas virtuais. Isso não apenas garante a alta disponibilidade e a tolerância a falhas, mas também permite a escalabilidade horizontal automática, conforme a demanda do tráfego aumenta ou diminui, otimizando o uso de recursos e reduzindo custos.

A integração do ALB com o ECS oferece uma série de benefícios, incluindo o roteamento inteligente de requisições, a capacidade de realizar deploy[8] de novas versões de aplicações sem interrupção do serviço e a habilidade de manter a sessão do usuário, caso necessário. A capacidade de realizar *health checks*[9] nos contêineres garante que o tráfego seja

---

[6] Refere-se ao processo de transformar informações ou dados em um código secreto, especialmente para prevenir acesso não autorizado. No contexto de armazenamento de dados, como senhas, a criptografia é vital para garantir a segurança e a privacidade, tornando os dados ilegíveis para quem não possui a chave de descriptografia.

[7] Cluster, no contexto de tecnologia da informação, refere-se a uma coleção de servidores interconectados que operam coletivamente, funcionando como um sistema único. Essa configuração permite que as cargas de trabalho sejam distribuídas entre múltiplos nós, aumentando a eficiência, a disponibilidade dos recursos e a capacidade de escalar serviços de forma mais efetiva, comparado ao uso de um único servidor.

[8] Deploy, em contextos de desenvolvimento de software, refere-se ao processo de distribuir e instalar uma aplicação ou uma nova versão dela em um ambiente de produção. Esse procedimento envolve a preparação e a transferência do código de uma aplicação do ambiente de desenvolvimento ou de teste para um ambiente onde ela possa ser acessada por usuários finais, assegurando que novas funcionalidades ou correções sejam disponibilizadas de forma eficiente e segura.

[9] Health check, em sistemas de TI, é um procedimento que verifica o estado e a disponibilidade de um componente ou sistema, como um contêiner ou serviço de rede. Esses testes são usados para detectar problemas ou falhas antecipadamente, garantindo que o tráfego de rede ou as solicitações dos usuários sejam direcionados apenas para recursos que estão funcionando corretamente, contribuindo para a manutenção da qualidade e confiabilidade do serviço.

encaminhado apenas para as instâncias que estão operacionais, mantendo a qualidade e a confiabilidade da aplicação.

Para o *front-end*, a escolha da infraestrutura da Vercel reflete uma estratégia centrada em desempenho e facilidade de uso. A Vercel é uma plataforma de *edge computing*[10] que proporciona a implantação contínua e automática do código do *front-end* assim que ele é atualizado no repositório de versionamento. Além disso, a Vercel otimiza automaticamente as aplicações Next.js para produção, fornecendo funcionalidades como SSR, *Static Site Generation* (SSG)[11] e *Split Testing*[12] sem configurações adicionais.

A Vercel também fornece uma Rede de Distribuição de Conteúdo (CDN)[13] global, garantindo que o conteúdo estático seja servido o mais próximo possível dos usuários finais, reduzindo a latência e melhorando a velocidade de carregamento da página. Isso é especialmente importante para o ECODATA, onde a experiência do usuário é priorizada e o desempenho do site é crucial para o engajamento.

Em suma, a infraestrutura do ECODATA é uma combinação bem pensada de tecnologias de nuvem que visam a escalabilidade, a confiabilidade e a performance. A utilização do AWS ALB e ECS para o *back-end*, juntamente com a infraestrutura de edge computing da Vercel para o *front-end*, estabelece uma base sólida para um sistema robusto e ágil.

2.2.7 INTEGRAÇÃO CONTÍNUA E ENTREGA CONTÍNUA

A Integração Contínua e Entrega Contínua (CI/CD) são fundamentais na arquitetura do ECODATA, promovendo um fluxo de trabalho eficiente e automatizado para o desenvolvimento e implantação de software. A utilização dessas práticas garante que o código seja integrado, testado e liberado de forma contínua e automatizada, melhorando

---

[10] Edge Computing, termo em inglês para "computação de borda", refere-se ao processamento de dados próximo à fonte onde são gerados, reduzindo latência e acelerando a resposta.

[11] SSG refere-se à geração antecipada de páginas web estáticas durante o build do site, proporcionando carregamento rápido e melhor desempenho em SEO, uma vez que não requer renderização no servidor a cada requisição.

[12] Split Testing é um método que compara duas versões de uma página as exibindo simultaneamente a diferentes segmentos do público para avaliar qual gera melhores resultados em objetivos específicos, como conversões ou cliques.

[13] CDN, sigla para "Content Delivery Network" (Rede de Distribuição de Conteúdo), é um sistema de servidores distribuídos geograficamente que trabalha para entregar conteúdo da web de forma rápida. Ele armazena cópias de conteúdo estático, como imagens e arquivos HTML, em vários locais para reduzir a latência, melhorando assim a velocidade de carregamento de páginas para usuários em diferentes regiões.

significativamente a qualidade e a velocidade de lançamento das novas funcionalidades e correções.

Os componentes para configuração do CI/CD estão descritos nas seguintes alíneas:

a) github actions:
   - foi utilizado o Github Actions para automatizar o processo de CI/CD. O Github Actions permite configurar fluxos de trabalho personalizados para compilação, teste e implantação do código assim que as mudanças são feitas no repositório.

b) docker:
   - o uso de contêineres Docker garante consistência entre os ambientes de desenvolvimento, teste e produção. Criamos contêineres Docker para o ambiente de aplicação, facilitando a implantação e a execução em diferentes ambientes.

c) elastic container registry (ECR):
   - foi feita a integração com o AWS ECR para armazenar imagens Docker. Após a construção das imagens pelo processo de CI, elas são enviadas para o ECR, garantindo um repositório seguro e gerenciável para as imagens de contêiner.

d) elastic container service com Fargate:
   - foi utilizado o AWS ECS com a opção serverless Fargate para a execução dos contêineres. Isso elimina a necessidade de gerenciar servidores e permite escalar automaticamente de acordo com a demanda.

e) fluxo de trabalho de implantação:
   - No Github, foram configuradas ações que acionam a construção de imagens Docker após cada *commit* e as enviam para o ECR. Em seguida, atualiza-se as definições de tarefas no ECS com as novas imagens e implementa-se de forma automática as atualizações no serviço ECS, garantindo que a versão mais recente da aplicação esteja sempre em execução.

Os benefícios do CI/CD estão explicitados nas alíneas a seguir:

a) rapidez e eficiência:
   - com a automação do processo de build e deploy, reduziu-se significativamente o tempo necessário para disponibilizar novas funcionalidades e correções aos usuários.

b) consistência e confiabilidade:
   - a padronização de ambientes usando contêineres Docker minimiza os problemas de "funciona na minha máquina", assegurando que o software funcione conforme o esperado em todos os ambientes.

c) escalabilidade:
   - a infraestrutura serverless do ECS com Fargate facilita a escalabilidade automática e eficiente, ajustando-se à demanda sem a necessidade de intervenção manual.

d) segurança e controle:
   - a integração com o AWS ECR e o uso de políticas de segurança garantem que as imagens de contêiner sejam gerenciadas e distribuídas de forma segura.

A implementação do CI/CD no ECODATA é um exemplo de como as modernas práticas de desenvolvimento de software podem ser aplicadas para criar sistemas ágeis, seguros e escaláveis, alinhados com as necessidades de startups e projetos *Minimum Viable Product* (MVP[14]).

2.2.8 RESUMO DA ARQUITETURA

A arquitetura do ECODATA é composta por uma combinação de tecnologias modernas e serviços de infraestrutura de nuvem otimizados. No *front-end*, utilizamos ReactJS com Next.js, que é hospedado e otimizado pela plataforma Vercel, proporcionando uma entrega rápida de conteúdo e uma experiência de usuário aprimorada. O *back-end* é desenvolvido com Node.js e NestJS, operando em contêineres gerenciados pelo Amazon ECS, que, em conjunto com o AWS ALB, garante escalabilidade automática e distribuição eficiente do tráfego.

Para o processamento de planilhas, implementamos um serviço serverless[15] utilizando AWS Lambda, escrito em Python, que é ativado automaticamente ao receber novas planilhas no AWS S3 Bucket. Isso assegura um processamento eficiente e escalável de dados sem a necessidade de gerenciar servidores dedicados.

---

[14] MVP, ou Produto Mínimo Viável em português, é um conceito em desenvolvimento de software que se refere à criação de uma versão inicial de um produto com recursos suficientes para satisfazer os usuários iniciais e fornecer feedback para o desenvolvimento futuro. A ideia é lançar rapidamente um produto com funcionalidades básicas, mas funcionais, para testar hipóteses de mercado e aprender com as reações dos usuários.

[15] Serverless é um modelo de computação onde o provedor de nuvem gerencia a execução do código, eliminando a necessidade de provisionar ou gerenciar servidores. Neste modelo, os desenvolvedores podem se concentrar na escrita do código, enquanto o provedor de nuvem cuida da infraestrutura, escalabilidade e manutenção.

O armazenamento de dados, incluindo informações ecológicas e dados de autenticação do usuário, é feito no MongoDB. Este banco de dados é operado em contêineres Docker dentro de uma instância do AWS EC2, com volumes do Docker para garantir a persistência dos dados. Essa abordagem não só assegura a segurança e a disponibilidade dos dados, mas também facilita a manutenção e a escalabilidade do sistema de armazenamento.

Além disso, a arquitetura incorpora práticas de CI/CD através do uso de Github Actions. Este fluxo automatiza o processo de construção, teste e implantação da aplicação, garantindo atualizações frequentes e estáveis. A combinação do Docker com o AWS ECR e ECS facilita a gerência e implantação de imagens de contêineres, melhorando a eficiência e a confiabilidade do ciclo de vida de desenvolvimento.

Essa combinação de ReactJS e Next.js no *front-end*, Node.js e NestJS no *back-end* com a infraestrutura do ECS e ALB da AWS, uma estratégia de CI/CD robusta juntamente com o processamento serverless de planilhas e o armazenamento de dados eficiente no MongoDB, cria uma arquitetura web coesa, escalável e de alto desempenho, ideal para a coleta, processamento e armazenamento de dados ecológicos.

2.3 MODELO LÓGICO

> Observamos aqui que os fornecedores de ferramentas de bancos de dados costumam usar o termo modelo lógico para se referir ao modelo de dados conceitual e utilizam o termo modelo físico para se referir ao modelo de implementação específico do SGBD (por exemplo, tabelas SQL) (Teorey *et al.*, 2006, p. 6).

O trecho explica a terminologia usada por profissionais de banco de dados ao distinguir entre "modelo lógico" e "modelo físico". O modelo lógico se refere ao modelo de dados conceitual, que é uma abstração que define como os dados são relacionados e como deveriam ser estruturados, independentemente de como serão fisicamente armazenados. Em outras palavras, o modelo lógico enfoca o que os dados significam e como eles se conectam uns aos outros.

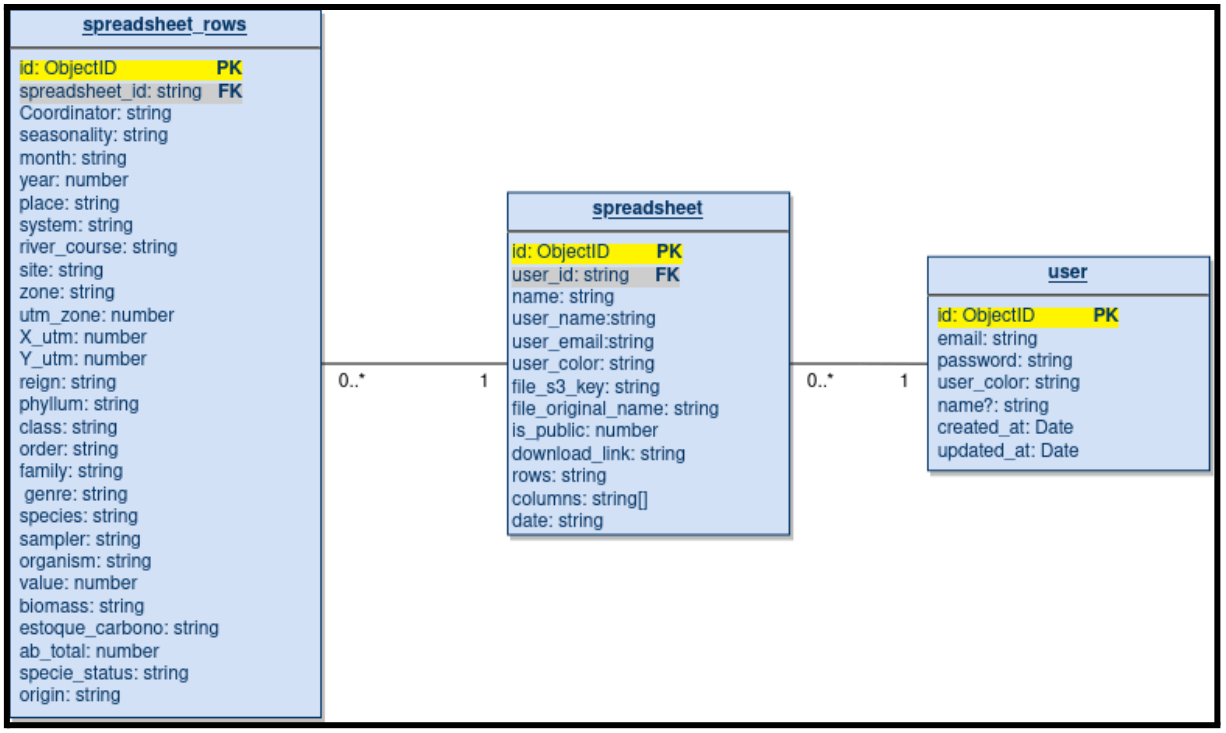
> Projeto lógico. O esquema global, um diagrama de modelo de dados conceitual que mostra todos os dados e seus relacionamentos, é desenvolvido usando técnicas como ER ou UML. As construções do modelo de dados por fim precisam ser transformadas em relações normalizadas (globais), ou tabelas (Teorey *et al*., 2006, p. 3).

O modelo de dados conceitual é uma representação abstrata e simplificada dos dados e de seus relacionamentos dentro do contexto de um sistema de banco de dados. Este modelo é

importante porque define como as informações estão inter-relacionadas e como elas deveriam ser organizadas de uma maneira que faça sentido para os usuários finais e para os objetivos do negócio. O modelo é normalmente criado usando técnicas de modelagem, como diagramas Entidade-Relacionamento (ER) ou *Unified Modeling Language* (UML), que são padrões para visualizar e documentar os elementos de um sistema e seus relacionamentos.

Segue o modelo lógico da plataforma desenvolvida, disponível na Figura 3.

Figura 3 - Modelo lógico da plataforma EcoData



Fonte: O autor (2023).

Conforme apresentado na Figura 3, a cardinalidade no modelo lógico de banco de dados representa a relação entre as entidades. Neste modelo, temos três entidades: user, spreadsheet, e *spreadsheet_rows*. A relação entre user e spreadsheet é de um para muitos (1:N), indicando que um usuário pode ter várias planilhas, mas cada planilha está associada a um único usuário. Da mesma forma, a relação entre *spreadsheet* e *spreadsheet_rows* é também de um para muitos (1:N), o que significa que uma planilha pode conter várias linhas, mas cada linha está vinculada a uma única planilha. Isso é essencial para organizar e relacionar os dados de forma eficiente em um banco de dados.

Na solução, na plataforma ECODATA desenvolvida, que integra a funcionalidade de upload de planilhas para um bucket do AWS S3 seguido pelo processamento desses dados através de uma função AWS Lambda, foi abordada a questão da integridade referencial de

uma maneira inovadora e adaptada às tecnologias NoSQL. Diferentemente dos Sistemas de Gerenciamento de Banco de Dados Relacionais (SGBDR) tradicionais, que utilizam chaves estrangeiras para assegurar a integridade referencial, a abordagem definida confia em uma combinação de validações de front-end, e lógicas de aplicação no back-end para manter a consistência e a precisão dos dados.

No front-end, uma pré-validação assegura que cada tentativa de upload de planilha esteja associada a um usuário autenticado, identificado por um *auth_id* válido. Esta etapa é crucial para prevenir uploads não associados e para reforçar a segurança e a integridade da sessão de usuário. Apesar dessa medida preventiva, validações exclusivamente no cliente podem ser insuficientes para garantir a integridade dos dados em todos os cenários.

Por isso, o processamento subsequente pelo AWS Lambda desempenha um papel complementar nesse esquema de integridade. Ao receber um arquivo, a função Lambda realiza uma consulta no MongoDB para verificar a existência do usuário correspondente ao *user_id* extraído do nome do arquivo. Embora, na prática atual, o processamento não seja interrompido se o usuário não for encontrado - optando-se por registrar o evento no CloudWatch Logs para revisão e auditoria -, este passo é fundamental para monitorar a congruência entre os uploads e os registros de usuários.

A decisão de proceder com o salvamento dos dados da planilha, mesmo na ausência de uma correspondência de usuário válida, foi tomada para evitar a perda de dados críticos, permitindo a intervenção manual e a reconciliação dos dados em circunstâncias excepcionais. Entretanto, reconhecemos a importância da integridade referencial e estamos avaliando mecanismos para reforçá-la ainda mais, incluindo a possibilidade de implementar verificações adicionais de integridade e processos de validação automática que possam intervir de forma mais assertiva, caso discrepâncias sejam detectadas.

Este arranjo reflete o compromisso com a flexibilidade e adaptabilidade do sistema, enquanto outras formas de manter os princípios de integridade dos dados em um ambiente dinâmico e distribuído são exploradas. A semântica de integridade referencial, portanto, enquanto adaptada a este contexto tecnológico específico, permanece uma prioridade fundamental na abordagem de design e desenvolvimento desta solução, assegurando que a ECODATA continue a ser uma solução robusta e confiável para a compilação e compartilhamento de dados ambientais.

2.4 MODELO DE DADOS

O banco de dados escolhido para a aplicação foi o banco NoSQL MongoDB. Bancos não relacionais são feitos para armazenar e recuperar dados de forma não estruturada, ou seja, não seguindo um esquema fixo como no caso de bancos relacionais. Porém, o mesmo possui uma estrutura dados subjacente e pode seguir diferentes modelos de dados como documento, colunas, grafos ou pares chave-valor, fornecendo uma abordagem mais flexível para a organização dos dados.

O modelo de dados utilizado é o de documentos (veja apêndices A, B e C para o modelo de dados do banco da aplicação).

2.4 DIAGRAMA DE SEQUÊNCIA

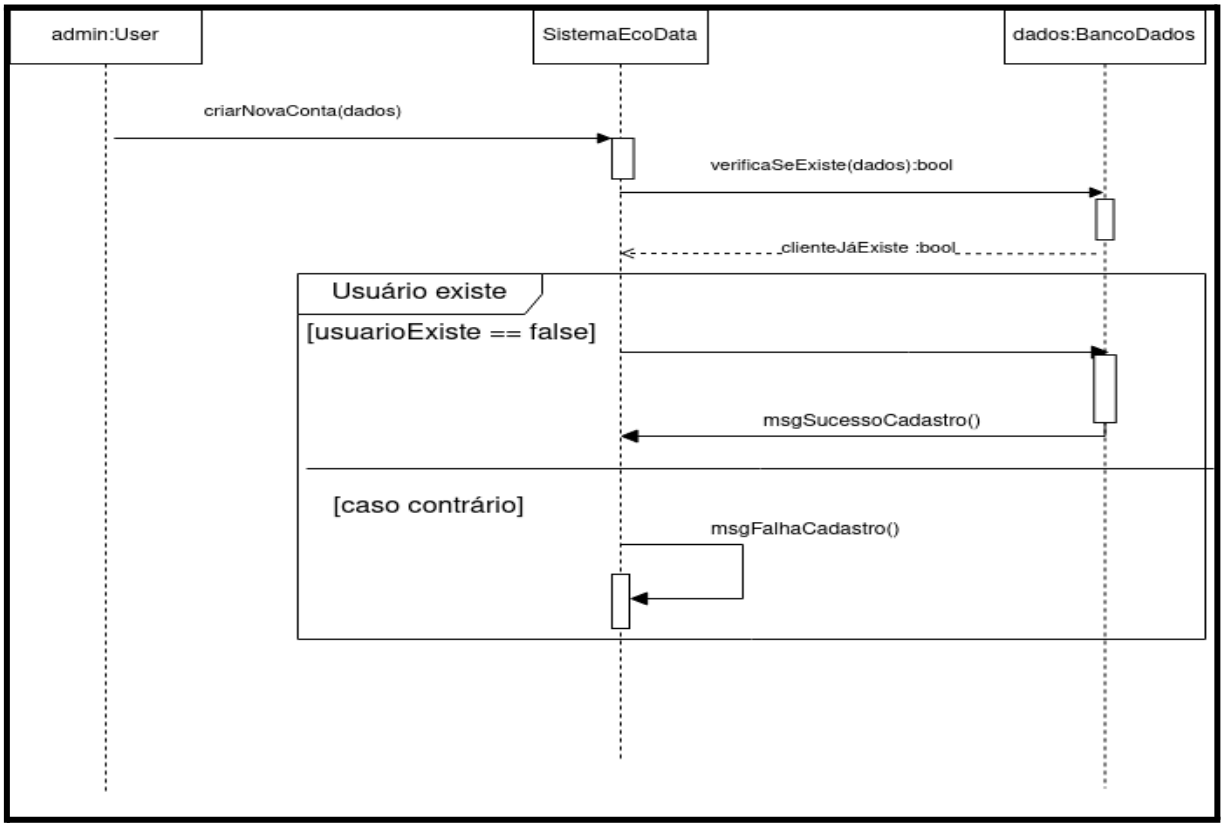
> Um diagrama de sequência do sistema (SSD) é uma imagem que mostra, para um cenário particular de um caso de uso, os eventos que atores externos geram, sua ordem e eventos entre sistemas. Todos os sistemas são tratados como uma caixa preta; a ênfase do diagrama está nos eventos que cruzam o limite do sistema, de atores para sistemas (Larman, 2001, p. 118, tradução própria).

No trecho destacado, Craig Larman desvenda a essência dos diagramas de sequência do sistema (SSD) como ferramentas cruciais no desenvolvimento de software orientado a objetos. Ele esclarece que os SSDs servem como representações visuais que mapeiam a interação entre atores externos e o sistema sob análise, através de um cenário específico de um caso de uso. Essa abordagem permite aos desenvolvedores visualizar não apenas a sequência de eventos gerados por atores externos, mas também como esses eventos interagem e afetam diferentes componentes do sistema. A metáfora da "caixa preta" empregada por Larman enfatiza a abstração do funcionamento interno dos sistemas, focando na interface e na comunicação entre os atores e o sistema. Esta perspectiva é fundamental para compreender a dinâmica de interações em sistemas complexos, facilitando a identificação de requisitos, a análise de fluxos de trabalho e a implementação de soluções eficazes em projetos de software.

2.4.1 CADASTRO DE USUÁRIO

Segue diagrama de sequência (Figura 4) representando a sequência de processos no cadastro do usuário (veja apêndice G para verificar o fluxo de cadastro do usuário).

Figura 4 - Diagrama de sequência para o cadastro do usuário na plataforma EcoData



Fonte: O autor (2023).

Conforme observado na Figura 4, o sistema assim que recebe a solicitação para o cadastro de um novo usuário, verifica se o mesmo já existe no banco de dados, caso exista, retorna uma mensagem de erro, caso não exista, então efetua o cadastro do usuário.

2.5 DIAGRAMA DE ATIVIDADES

> O diagrama de atividade preocupa-se em descrever os passos a serem percorridos para a conclusão de uma atividade específica, podendo esta ser representada por um método com certo grau de complexidade, um algoritmo, ou mesmo um processo completo. O diagrama de atividade concentra-se na representação do fluxo de controle e de objetos de uma atividade. (GUEDES, 2018, cap. 1).

O diagrama de atividades, como descrito na citação, é uma ferramenta da UML que é utilizada para representar sequencialmente os passos necessários para a realização de uma atividade. Essa atividade pode variar em complexidade, desde um método simples até um processo completo e multifacetado.

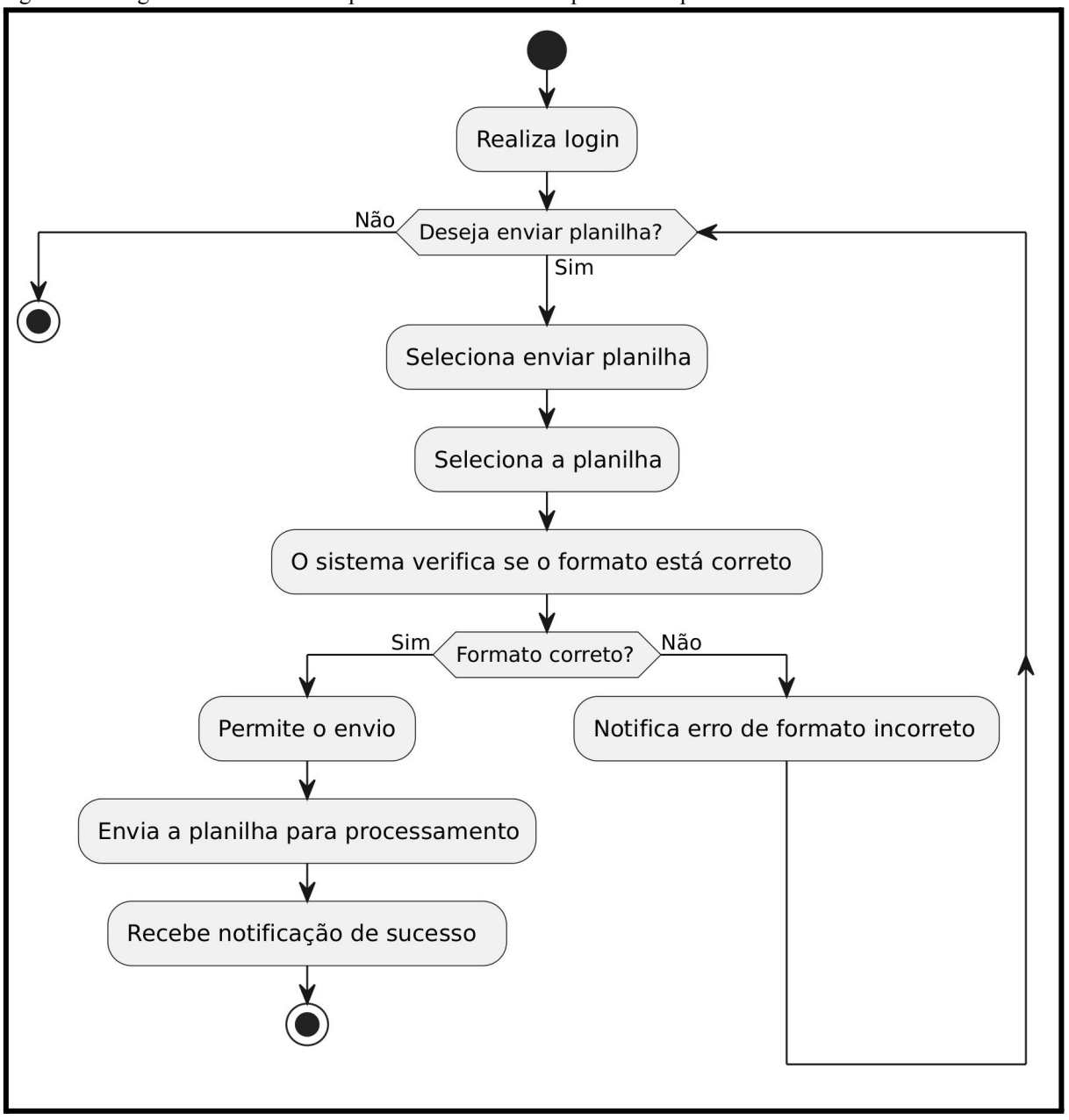
Essa ferramenta é preocupada principalmente com duas coisas: o fluxo de controle e o fluxo de objetos. O fluxo de controle é a ordem em que as ações são realizadas. Ele é representado graficamente de maneira que seja possível entender a sequência lógica dos

eventos ou decisões que guiam o processo do início ao fim. O fluxo de objetos, por outro lado, diz respeito aos dados ou entidades que são manipulados e transformados ao longo do processo.

2.5.1 ENVIAR PLANILHA

Segue o diagrama de atividades para o fluxo de envio da planilha para processamento (veja APÊNDICE F e H para o fluxo de envio da planilha).

Figura 5 - Diagrama de atividades representando o envio da planilha na plataforma EcoData



Fonte: O autor (2023).

A Figura 5 mostra um diagrama de atividades descrevendo o processo de envio de uma planilha para processamento. O fluxo começa quando o usuário realiza o login no sistema. Após a autenticação, o usuário seleciona a opção de enviar uma planilha e, em seguida, escolhe o arquivo específico a ser enviado.

O próximo passo é uma checagem de validação onde o sistema verifica se o formato da planilha está correto. Se o formato for confirmado como correto, o sistema permite que o envio prossiga. Caso contrário, o usuário é notificado do erro, e o processo é interrompido, indicando que o arquivo não segue o formato esperado pelo sistema.

Uma vez que o formato é aprovado, a planilha é enviada para processamento. O sistema então retorna uma notificação de sucesso ao usuário, concluindo o processo de envio.

Este fluxo garante que apenas planilhas no formato correto sejam processadas, o que é importante para manter a integridade dos dados e a eficiência do processamento no sistema.

Este diagrama de atividades é uma representação visual que facilita o entendimento do processo de envio de planilhas, sendo útil para a documentação do sistema e para orientar os usuários e desenvolvedores sobre o procedimento esperado (Veja APÊNDICE H e J para fluxo de envio da planilha).

3 RELATÓRIO DE EXECUÇÃO DE TESTES

> Testes são a Pedra Filosofal do programador, transmutando medo em tédio. ‘Não, eu não estraguei nada. Os testes ainda estão verdes.’ Quanto mais estresse eu sentir, mais testes vou rodar. Rodar os testes imediatamente me traz uma sensação agradável e reduz o número de erros que eu cometo, o que reduz ainda mais o estresse que sinto (Beck, 2010, p. 144).

A transformação do medo em tédio, como destacado por Beck, sublinha uma revolução na maneira como encaramos o desenvolvimento de software. Esta citação nos leva a refletir sobre o ciclo vicioso de estresse e erros no desenvolvimento de software e como os testes automáticos oferecem uma saída eficaz. Ao incorporar testes automáticos no fluxo de trabalho, os desenvolvedores podem verificar constantemente a integridade do código, reduzindo a probabilidade de erros e, consequentemente, o estresse associado a possíveis falhas. Este ciclo virtuoso de testes não apenas melhora a qualidade do software mas também o bem-estar dos desenvolvedores, transformando a ansiedade da incerteza em confiança respaldada por resultados de testes consistentemente "verdes".

3.1 API NESTJS COM JEST

3.1.1 OBJETIVO GERAL DOS TESTES

Validar a funcionalidade e a confiabilidade do SpreadsheetsController em um projeto NestJS, garantindo que os métodos find, setIsPublic, e findAll operem conforme esperado sob várias condições.

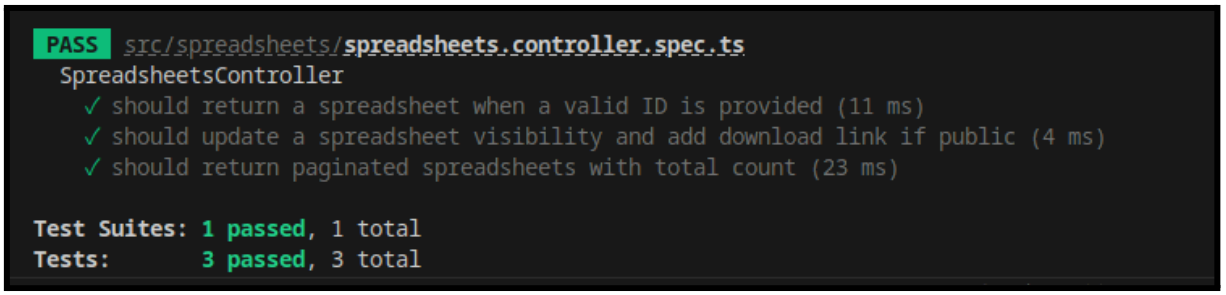
3.1.2 AMBIENTE DE TESTE

Para assegurar a qualidade e a confiabilidade do software ECODATA, um ambiente de teste robusto e detalhadamente configurado é utilizado. Este ambiente inclui componentes chave que são essenciais para realizar testes eficientes e precisos:

a) framework de teste:
   - Jest.
b) linguagem de programação:
   - TypeScript.
c) ambiente de execução:
   - Node.js versão 16.17.1.
d) banco de dados:
   - MongoDB (Mockado).
e) Ferramentas Adicionais:
   - Mocks para simulação de serviços e requests.

3.1.3 RESULTADOS DOS TESTES

Figura 6 - Resultado dos testes



Fonte: O autor (2023).

3.1.3.1 TESTE DO MÉTODO *FIND*

A descrição, resultado e observações do teste do método find estão explicitados nas alíneas a seguir:

a) descrição;
   - testa se o método find retorna uma planilha específica quando fornecido um ID válido;

b) resultado;
   - sucesso. O método retornou corretamente a planilha esperada, indicando que a busca por ID está funcionando como esperado;

c) observações.
   - nenhuma discrepância observada.

3.1.3.2 TESTE DO MÉTODO *SETISPUBLIC*

A descrição, resultado e observações do teste do método setIsPublic estão explicitados nas alíneas a seguir:

a) descrição:
   - verifica se a visibilidade da planilha é atualizada corretamente, e se o link de download é adicionado quando a planilha é configurada como pública.

b) resultado:
   - sucesso. O método atualizou a visibilidade da planilha e gerou o link de download conforme esperado.

c) observações:

- teste fundamental para validar a lógica de negócios relacionada à visibilidade das planilhas.

3.1.3.3 TESTE DO MÉTODO *FINDALL*

A descrição, resultado e observações do teste do método findAll estão explicitados nas alíneas a seguir:

a) descrição:
   - testa se o método findAll retorna planilhas paginadas com a contagem total correta.

b) resultado:
   - sucesso. As planilhas foram retornadas com paginação e a contagem total estava correta.

c) observações:
   - este teste confirma que a funcionalidade de paginação e contagem total está operando corretamente.

3.1.4 CONCLUSÃO SOBRE OS TESTES REALIZADOS

Os testes realizados no SpreadsheetsController demonstraram um alto nível de confiabilidade e funcionalidade dos métodos testados. Todos os testes passaram sem erros, indicando que o controller se comporta conforme esperado nas condições testadas. Este nível de cobertura de teste contribui significativamente para a manutenção da qualidade do código e para a confiabilidade da aplicação.

4 CASOS DE TESTE PARA A PLATAFORMA ECODATA

Esta seção apresenta casos de teste detalhados para a plataforma ECODATA, esses casos de teste são projetados para validar a funcionalidade da plataforma, assegurando que ela atenda aos requisitos especificados e funcione corretamente sob diversas condições. Cada caso de teste incluirá o objetivo, pré-condições, passos, dados de teste, e pós-condições esperadas.

4.1 CASO DE TESTE 1: CADASTRO DE USUÁRIO

Segue o caso de teste para o fluxo de cadastro de usuário:

a) objetivo:
   - verificar se o sistema permite o cadastro de novos usuários com todos os dados necessários.
b) pré-condições:
   - usuário não está cadastrado no sistema.
c) passos:
   - acessar a página de cadastro.
   - preencher todos os campos obrigatórios: nome, e-mail, e senha.
   - submeter o formulário de cadastro.
d) dados de Teste:
   - nome: "Teste Usuário", e-mail: "teste@ecodata.com", senha: "senha123".
e) pós-condições Esperadas:
   - usuário é cadastrado com sucesso e redirecionado para a página de login.

4.2 CASO DE TESTE 2: ENVIO DE PLANILHA PARA PROCESSAMENTO

Segue o caso de teste para o fluxo de envio de planilha para processamento:

a) objetivo:
   - confirmar se o sistema processa e armazena corretamente as planilhas enviadas pelos usuários.
b) pré-condições:
   - usuário está logado e possui uma planilha no formato correto.
c) passos:
   - acessar a seção de envio de planilha.
   - selecionar a planilha a ser enviada.
   - enviar a planilha para processamento.
d) dados de Teste:
   - arquivo de planilha válido.
e) pós-condições Esperadas:

- planilha é processada com sucesso, e os dados são exibidos corretamente na plataforma.

4.3 CASO DE TESTE 3: VISUALIZAÇÃO DE DADOS PROCESSADOS

Segue o caso de teste para o fluxo de visualização de dados processados:

a) objetivo:
   - testar a capacidade do usuário de visualizar os dados processados de planilhas enviadas.
b) pré-condições:
   - usuário está logado e tem pelo menos uma planilha processada disponível.
c) passos:
   - acessar a seção de planilhas processadas.
   - selecionar uma planilha da lista.
   - visualizar os dados processados.
d) dados de Teste:
   - seleção de uma planilha processada existente.
e) pós-condições Esperadas:
   - dados da planilha selecionada são exibidos corretamente.

4.4 CASO DE TESTE 4: TORNAR PLANILHA PÚBLICA PARA DOWNLOAD

Segue o caso de teste para o fluxo de tornar planilha pública para download:

a) objetivo:
   - verificar se o usuário pode tornar uma planilha enviada disponível para download público.
b) pré-condições:
   - usuário está logado e seleciona uma planilha própria que deseja tornar pública.
c) passos:
   - acessar a seção de visualização de planilha.
   - marcar a opção para tornar a planilha pública.
   - confirmar a ação.

d) dados de Teste:
    - seleção de uma planilha própria.
e) pós-condições Esperadas:
    - planilha é tornada pública, e um link para download é disponibilizado.

4.5 CASO DE TESTE 5: DOWNLOAD DE PLANILHA PÚBLICA

Segue o caso de teste para o fluxo de download de planilha pública:

a) objetivo:
    - assegurar que usuários possam baixar planilhas tornadas públicas por outros usuários.
b) pré-condições:
    - existem planilhas marcadas como públicas disponíveis para download.
c) passos:
    - acessar a seção de planilhas públicas.
    - selecionar uma planilha para download.
    - baixar a planilha.
d) dados de Teste:
    - seleção de uma planilha pública disponível.
e) pós-condições Esperadas:
    - planilha é baixada com sucesso.

5 CONCLUSÃO

O desenvolvimento da plataforma web ECODATA para a compilação de dados de ecossistemas do estado da Paraíba, representa um avanço significativo no campo da estruturação de metadados. O presente estudo não só fornece uma solução tecnológica inovadora para o armazenamento, processamento e compartilhamento de dados ecológicos - no levantamento bibliográfico realizado, não foram encontradas ferramentas de mesmo propósito -, mas também contribui para a eficiência do armazenamento de informações que podem ser direcionadas à gestão ao manejo de ecossistemas.

A combinação de tecnologias *front-end* e *back-end* utilizando ReactJS, Next.js, Node.js, NestJS, e a infraestrutura da AWS, oferece uma plataforma robusta, escalável e de fácil utilização. A integração com o MongoDB como sistema de banco de dados NoSQL proporciona flexibilidade e eficiência no manejo de dados não estruturados, essencial para a natureza diversificada dos dados ecológicos coletados.

Os testes de funcionalidade realizados com o Jest para o controle da API, confirmaram a eficácia e a confiabilidade da plataforma. A capacidade de realizar operações complexas de manipulação de dados, como o envio, processamento e disponibilização de planilhas, e a facilidade de uso da interface, asseguram que a plataforma não apenas atende às necessidades atuais de gestão de recursos hídricos, mas também está preparada para se adaptar a futuras demandas e evoluções tecnológicas.

O presente estudo, além de sua aplicabilidade prática, representa um exemplo valioso do potencial que a integração de tecnologias modernas de desenvolvimento web tem na resolução de problemas ambientais críticos. A plataforma "ECODATA" está posicionada para ser uma ferramenta chave na colaboração entre pesquisadores, instituições e administradores de recursos hídricos, promovendo uma gestão uniformizada e compartilhamento de dados com múltiplas naturezas (como os dados ecológicos incluídos na plataforma). Apesar do presente estudo considerar inicialmente a inclusão de dados oriundos de ecossistemas na Paraíba, a plataforma web proposta a partir deste trabalho, representa um modelo de aplicação a nível regional e nacional.

A integração de ferramentas tecnológicas voltadas à resolução de problemáticas ambientais demonstra não apenas uma competência técnica significativa na área de desenvolvimento de sistemas, mas também uma contribuição prática e valiosa para a gestão ambiental. O incremento da plataforma, a partir de sugestões dos usuários, possibilitará a inclusão de aspectos atuais, ampliando seu impacto e eficácia. Estudos futuros podem incluir

a avaliação por um número maior de pesquisadores, grupos de pesquisa e, inclusive, representantes do poder público estadual após a ampliação no escopo da ferramenta.

6 CONTRIBUIÇÕES DO TRABALHO

O presente trabalho contribui significativamente para a área de gestão de dados ambientais no Estado da Paraíba, abordando lacunas tanto no âmbito acadêmico quanto na aplicação prática. As principais contribuições deste TCC são:

a) desenvolvimento de uma Nova Plataforma Web para Dados Ambientais:
    - o trabalho desenvolveu a ECODATA, uma plataforma inovadora que facilita a compilação, análise e compartilhamento de dados ambientais na Paraíba. Isso preenche uma importante lacuna, proporcionando uma ferramenta dedicada ao monitoramento ambiental na região.
b) facilitação do Acesso e Compartilhamento de Dados:
    - ao possibilitar o fácil acesso, download e upload de dados ambientais, a ECODATA promove uma colaboração inédita entre pesquisadores, gestores ambientais e o público em geral. Isso representa um avanço significativo na democratização do acesso à informação ambiental no estado.
c) contribuição para a Gestão de Ecossistemas:
    - através da compilação estruturada e do compartilhamento de dados, a plataforma oferece suporte à tomada de decisões informadas para a gestão e conservação de ecossistemas. Isso tem potencial para impactar positivamente as políticas de sustentabilidade e conservação ambiental na Paraíba.
d) base para Futuras Pesquisas:
    - a ECODATA serve como um repositório centralizado de dados, estabelecendo uma base sólida para futuras pesquisas ambientais na região. Isso facilita estudos longitudinais e comparativos, essenciais para entender as tendências e impactos ambientais ao longo do tempo.
e) adoção de Tecnologias Modernas:
    - a implementação da plataforma emprega tecnologias web modernas e práticas de desenvolvimento ágil, servindo como um caso de estudo valioso para o desenvolvimento de soluções tecnológicas aplicadas à gestão ambiental.

6.1 CONCLUSÃO

As contribuições deste trabalho reforçam a importância de soluções tecnológicas inovadoras na gestão de dados ambientais. A ECODATA não apenas oferece uma nova ferramenta para pesquisadores, gestores e o público em geral no Estado da Paraíba, mas também estabelece um modelo que pode ser adaptado e replicado em outras regiões com desafios semelhantes. Espera-se que este estudo inspire futuras iniciativas e pesquisas na interseção entre tecnologia e sustentabilidade ambiental.

REFERÊNCIAS

ARAÚJO, A. S. F. *et al.* **Environmental DNA sequencing to monitor restoration practices on soil bacterial and archaeal communities in soils under desertification in the Brazilian Semiarid**. Microbial Ecology, v. 85, n. 3, p. 1072-1076, 2023.

BECK, Kent. **TDD Desenvolvimento Guiado por Testes**. Porto Alegre: Bookman, 2001. p. 144. E-book. ISBN 978-85-7780-747-5.

BRANTSCHEN, J.; BLACKMAN, R. C.; WALSER, J. C.; ALTERMATT, F. **Environmental DNA gives comparable results to morphology-based indices of macroinvertebrates in a large-scale ecological assessment**. Plos One, v. 16, n. 9, e0257510, 2021. DOI: https://doi.org/10.1371/journal.pone.0257510.

DUDLEY, N. *et al.* **Measuring progress in status of land under forest landscape restoration using abiotic and biotic indicators**. Restoration Ecology, v. 26, n. 1, p. 5-12, 2018.

EKELUND, N. G. A.; HÄDER, D. P. **Environmental monitoring using bioassays. In: Bioassays**: Advanced Methods and Applications. [S.l.]: [s.n.], 2018. p. 419-437. DOI: https://doi.org/10.1016/B978-0-12-811861-0.00021-8.

FANG, S. *et al.* **An integrated system for regional environmental monitoring and management based on internet of things**. IEEE Transactions on Industrial Informatics, v. 10, n. 2, p. 1596-1605, 2014.

FUOCO, R.; GIANNARELLI, S. **Integrity of aquatic ecosystems**: An overview of a message from the South Pole on the level of persistent organic pollutants (POPs). Microchemical Journal, vol. 148, p. 230–239, 1 jul. 2019. DOI: https://doi.org/10.1016/j.microc.2019.04.076.

GOMES, L. E. O. *et al.* **The impacts of the Samarco mine tailing spill on the Rio Doce estuary, Eastern Brazil**. Marine Pollution Bulletin, v. 120, n. 1-2, p. 28-36, 2017.

GUEDES, G. T. A. **UML 2**: Uma Abordagem Prática. São Paulo: Novatec Editora, 2018. E-book. ISBN 978-85-7522-644-5.

HAUER, F.R.; LAMBERTI, G.A. **Methods in Stream Ecology**. Amsterdam: Elsevier, 2006.

HIGGINS, C. I. *et al.* **Citizen Observatory Web (COBWEB)**: A generic infrastructure platform to facilitate the collection of citizen science data for environmental monitoring. International Journal of Spatial Data Infrastructures Research, v. 11, 2016.

JACKSON, R. B. *et al.* **Water in a changing world. Ecological Applications**, v. 11, n. 4, p. 1027-1045, 2001.

JOVEM-AZEVÊDO, D. *et al.* **Modelling the abundance of a non-native mollusk in tropical semi-arid reservoirs**. Hydrobiologia, p. 1-15, 2022.

JOVEM‐AZEVÊDO, D. *et al.* **Rehabilitation scenarios for reservoirs**: Predicting their effect on invertebrate communities through machine learning. River Research and Applications, v. 36, n. 7, p. 1109-1123, 2020.

KATO, S.; AHERN, J. **"Learning by doing"**: Adaptive planning as a strategy to address uncertainty in planning. Journal of Environmental Planning and Management, v. 51, n. 4, p. 543–559, 2008. DOI 10.1080/09640560802117028.

LARMAN, Craig. **Applying UML and Patterns**. 2. ed. [S.l.]: Prentice Hall, 2001. (629 p.) ISBN 978-0130925695.

LIU, J.; KATTEL, G.; ARP, H. P. H.; YANG, H. **Towards threshold-based management of freshwater ecosystems in the context of climate change**. Ecological Modelling, vol. 318, p. 265–274, 2015. DOI: http://dx.doi.org/10.1016/j.ecolmodel.2014.09.010.

MACHADO, R. C. A. *et al.* **Spatial and seasonal variation of the phytoplankton community structure in a reef ecosystem in North-eastern Brazil**. Journal of the Marine Biological Association of the United Kingdom, v. 98, n. 3, p. 557-566, 2018.

MARTIN, Robert C. **Arquitetura limpa**: o guia do artesão para estrutura e design de software. Rio de Janeiro: Alta Books, 2019. E-book. ISBN 978-85-508-1600-5.

MARTINS, I. *et al.* **Anthropogenic impacts influence the functional traits of Chironomidae (Diptera) assemblages in a neotropical savanna river basin**. Aquatic Ecology, v. 55, p. 1081-1095, 2021.

MASSEI, K. *et al.* **Analysis of marine diversity and anthropogenic pressures on Seixas coral reef ecosystem (northeastern Brazil)**. Science of The Total Environment, v. 905, p. 166984, 2023.

SILVA, O. O. *et al.* **Impactos ambientais ao longo do canal do Estreito, Sousa–PB**. RECIMA21-Revista Científica Multidisciplinar, v. 4, n. 6, e463303, 2023.

SINGH, D. *et al.* **Sensors and systems for air quality assessment monitoring and management**: A review. Journal of Environmental Management, v. 289, p. 112510, 2021.

SOMMERVILLE, I. **Engenharia de software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011. p. 126. ISBN 978-85-7936-108-1

TEOREY, T. J.; LIGHTSTONE, S.; NADEAU, T. **Projeto e Modelagem de Banco de Dados**. Rio de Janeiro: Editora Campus/Elsevier, 2006. p. 3-6. ISBN 978-85-352-2114-5.

VIHERVAARA, P. *et al.* **How essential biodiversity variables and remote sensing can help national biodiversity monitoring**. Global Ecology and Conservation, v. 10, p. 43-59, 2017.

APÊNDICE A - MODELO DE DADOS COLLECTION SPREADSHEET

O modelo de dados da *collection*[16] spreadsheet está descrito a seguir no Quadro 1, demonstrando o nome da propriedade da entidade, o tipo e se é uma propriedade obrigatória ou opcional.

Quadro 1 - Modelo de dados da collection spreadsheet

Propriedade	Tipo	Obrigatoriedade
id	ObjectID	Obrigatória
name	string	Obrigatória
user_id	string	Obrigatória
user_name	string	Obrigatória
user_email	string	Obrigatória
user_color	string	Obrigatória
file_s3_key	string	Obrigatória
file_original_name	string	Obrigatória
is_public	number	Obrigatória
download_link	string	Obrigatória
rows	string	Obrigatória
columns	string[]	Obrigatória
date	string	Obrigatória

Fonte: O autor (2023).

A explicação dos campos está descrita nas seguintes alíneas:

a) id:
   - identificador exclusivo gerado automaticamente pelo TypeORM[17] usando o ObjectID.

b) name:
   - nome do spreadsheet. É uma string.

[16] Em contextos de bancos de dados, especialmente nos NoSQL, o termo '*collection*' refere-se a um conjunto de registros ou documentos. Funciona de maneira similar a uma tabela em sistemas de banco de dados relacionais, mas com maior flexibilidade. Cada documento dentro de uma *collection* pode ter sua própria estrutura de dados única, contendo campos e tipos de dados variáveis, que podem ser definidos como obrigatórios ou opcionais conforme as necessidades do sistema.

[17] TypeORM é um *Object-Relational Mapping* (ORM) para TypeScript e JavaScript que facilita a interação com bancos de dados. Ele permite mapear entidades de objetos em tabelas de banco de dados, simplificando a manipulação de dados com uma abordagem orientada a objetos. TypeORM suporta a geração automática de identificadores únicos, como ObjectID, otimizando a gestão e a busca de registros no banco de dados.

c) user_id:
   - id do usuário associado ao spreadsheet. É uma string.

d) user_name:
   - nome do usuário associado ao spreadsheet. É uma string.

e) user_email:
   - email do usuário associado ao spreadsheet. É uma string.

f) user_color:
   - cor associada ao usuário escrita no sistema hexadecimal. É uma string.

g) file_s3_key:
   - chave do arquivo no serviço de armazenamento *Amazon Simple Storage Service* (S3). É uma string.

h) file_original_name:
   - nome original do arquivo do spreadsheet. É uma string.

i) is_public:
   - indicador se o spreadsheet é público ou privado. É um número (0 para privado e 1 para público, por exemplo).

j) download_link:
   - link de download do spreadsheet. É uma string.

k) rows:
   - representa o total de linhas do spreadsheet. É uma string numérica.

l) columns:
   - representa as colunas do spreadsheet. É um array de strings.

m) date:
   - data de criação do spreadsheet. É uma string que segue o *International Organization for Standardization* (ISO) 8601, sendo gerada automaticamente pelo TypeORM através do decorador @CreateDateColumn().

APÊNDICE B - MODELO DE DADOS COLLECTION SPREADSHEET_ROWS

O modelo de dados da *collection* spreadsheet_rows está descrito a seguir no Quadro 2, demonstrando o nome da propriedade da entidade, o tipo e se é uma propriedade obrigatória ou opcional.

Quadro 2 - Modelos de dados collection spreadsheet_rows

Propriedade	Tipo	Obrigatoriedade
id	ObjectID	Obrigatória
spreadsheet_id	string	Obrigatória
Coordinator	string	Obrigatória
seasonality	string	Obrigatória
month	string	Obrigatória
year	number	Obrigatória
place	string	Obrigatória
system	string	Obrigatória
river_course	string	Obrigatória
site	string	Obrigatória
zone	string	Obrigatória
utm_zone	number	Obrigatória
X_utm	number	Obrigatória
Y_utm	number	Obrigatória
reign	string	Obrigatória
phyllum	string	Obrigatória
class	string	Obrigatória
order	string	Obrigatória
family	string	Obrigatória
genre	string	Obrigatória
species	string	Obrigatória
sampler	string	Obrigatória
organism	string	Obrigatória
value	number	Obrigatória
biomass	string	Obrigatória
estoque_carbono	string	Obrigatória
ab_total	number	Obrigatória
specie_status	string	Obrigatória
origin	string	Obrigatória

Fonte: O autor (2023).

A explicação dos campos está descrita nas seguintes alíneas:

a) id:

- identificador exclusivo gerado automaticamente pelo TypeORM usando o ObjectID.

b) spreadsheet_id:
   - id do spreadsheet associado a essa linha. É uma string.

c) campo ‘Coordinator’:
   - nome do coordenador. É uma string.

d) seasonality:
   - temporalidade. É uma string.

e) month:
   - mês. É uma string.

f) year:
   - ano. É um número.

g) place:
   - localização. É uma string.

h) system:
   - sistema. É uma string.

i) river_course:
   - curso do rio. É uma string.

j) site:
   - site. É uma string.

k) zone:
   - zona. É uma string.

l) utm_zone:
   - zona UTM. É um número.

m) campo X_utm:
   - coordenada X UTM. É um número.

n) campo Y_utm:
   - coordenada Y UTM. É um número.

o) reign:
   - reino. É uma string.

p) phyllum:
   - filo. É uma string.

q) class:
   - classe. É uma string.

r) order:
   - ordem. É uma string.

s) family:
   - família. É uma string.

t) genre:
   - gênero. É uma string.

u) species:
   - espécie. É uma string.

v) sampler:
   - amostrador. É uma string.

w) organism:
   - organismo. É uma string.

x) value:
   - valor numérico. É um número.

y) biomass:
   - biomassa. É uma string.

z) estoque_carbono:
   - estoque de carbono. É uma string.

aa) ab_total:
   - total de AB. É um número.

bb) specie_status:
   - status da espécie. É uma string.

cc) origin:
   - origem. É uma string.

APÊNDICE C - MODELO DE DADOS COLLECTION USER

O modelo de dados da *collection* user está descrito a seguir no Quadro 3 com continuação na página 60, demonstrando o nome da propriedade da entidade, o tipo e se é uma propriedade obrigatória ou opcional.

Quadro 3 - Modelos de dados da collection user

Propriedade	Tipo	Obrigatoriedade
id	ObjectID	Obrigatória

email	string	Obrigatória
password	string	Obrigatória
user_color	string	Obrigatória
name	string	Opcional
created_at	Date	Obrigatória
updated_at	Date	Obrigatória

Fonte: O autor (2023).

A explicação dos campos está descrita nas seguintes alíneas:

a) id:
   - identificador exclusivo gerado automaticamente pelo TypeORM usando o ObjectID.

b) email:
   - endereço de e-mail do usuário. É uma string.

c) password:
   - senha do usuário. É uma string.

d) user_color:
   - cor do usuário. É uma string, que será definida automaticamente no momento da criação do usuário pelo método setRandomColor() usando o decorator[18] @BeforeInsert(). Esse método gera uma cor hexadecimal aleatória para cada novo usuário.

e) name:
   - nome do usuário. É uma string e é opcional.

f) created_at:
   - data de criação do usuário. É uma instância de Date, sendo a data de criação do documento.

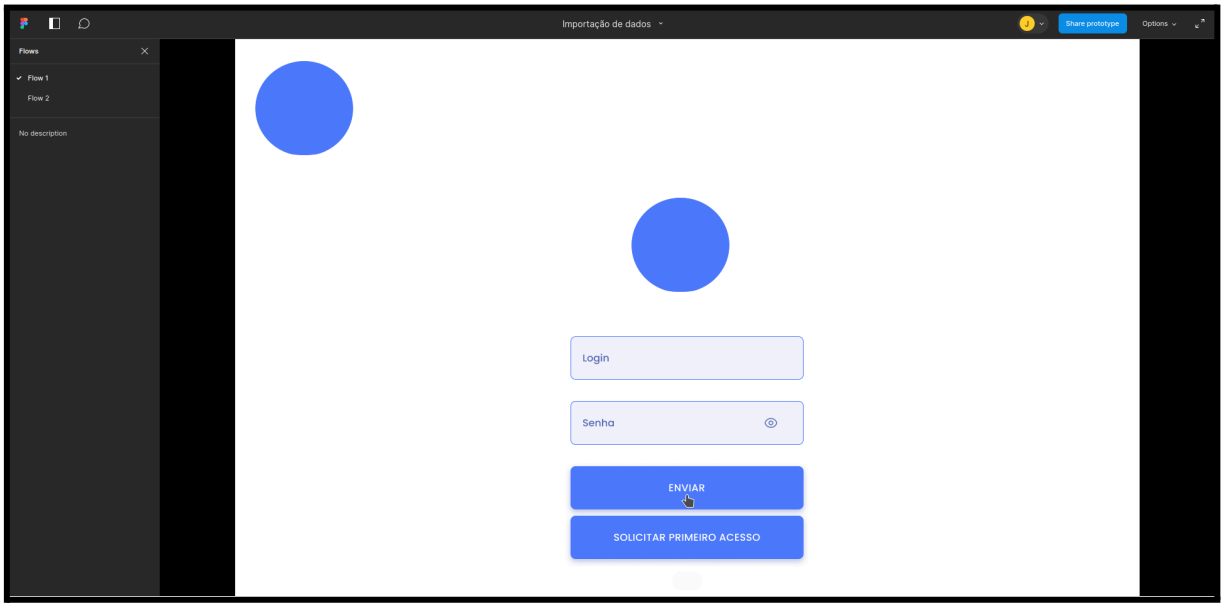
g) updated_at:
   - data da última atualização do usuário. É uma instância de Date, sendo a data de atualização do documento quando o mesmo sofre alterações.

[18] Decorator, no contexto da programação, é um padrão de design que permite adicionar novos comportamentos a objetos ou funções de forma dinâmica. Em linguagens como TypeScript e JavaScript, um decorator é uma declaração especial (precedida por '@') que pode ser anexada a uma declaração de classe, método, acessador, propriedade ou parâmetro. Decorators são utilizados para modificar ou estender o comportamento do elemento ao qual são aplicados, como o @BeforeInsert() que executa uma ação específica antes de uma operação de inserção em um banco de dados.

APÊNDICE D - PROTÓTIPO DO SOFTWARE

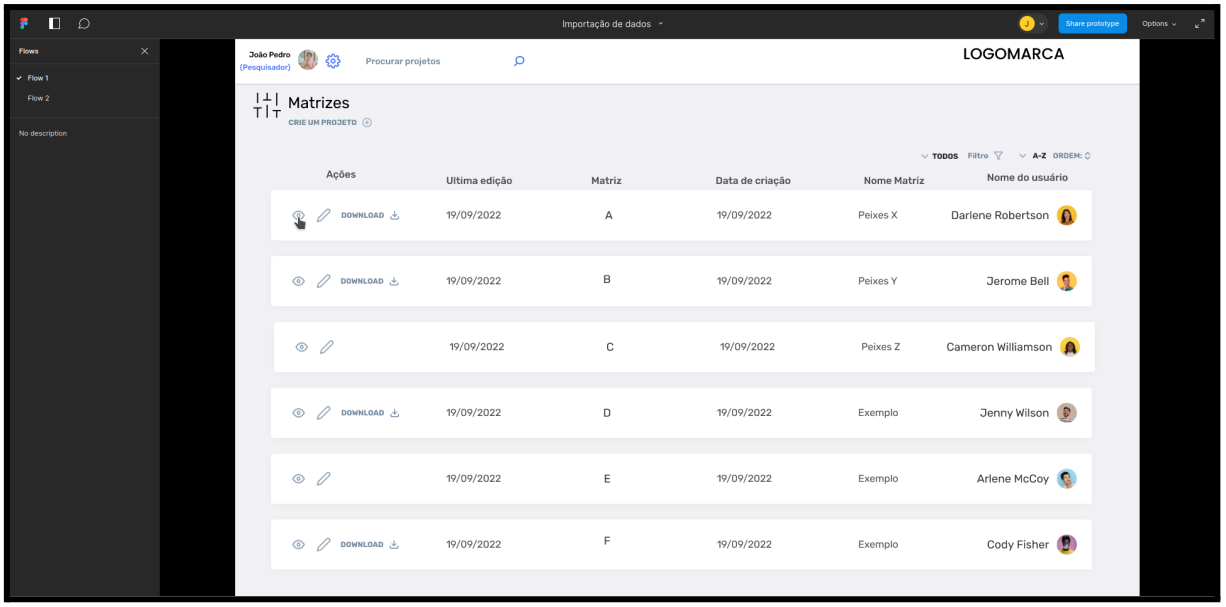
Seguem imagens do protótipo desenvolvido (Figuras 7 a 10) utilizando para tal a ferramenta Figma.

Figura 7 - Protótipo página de login



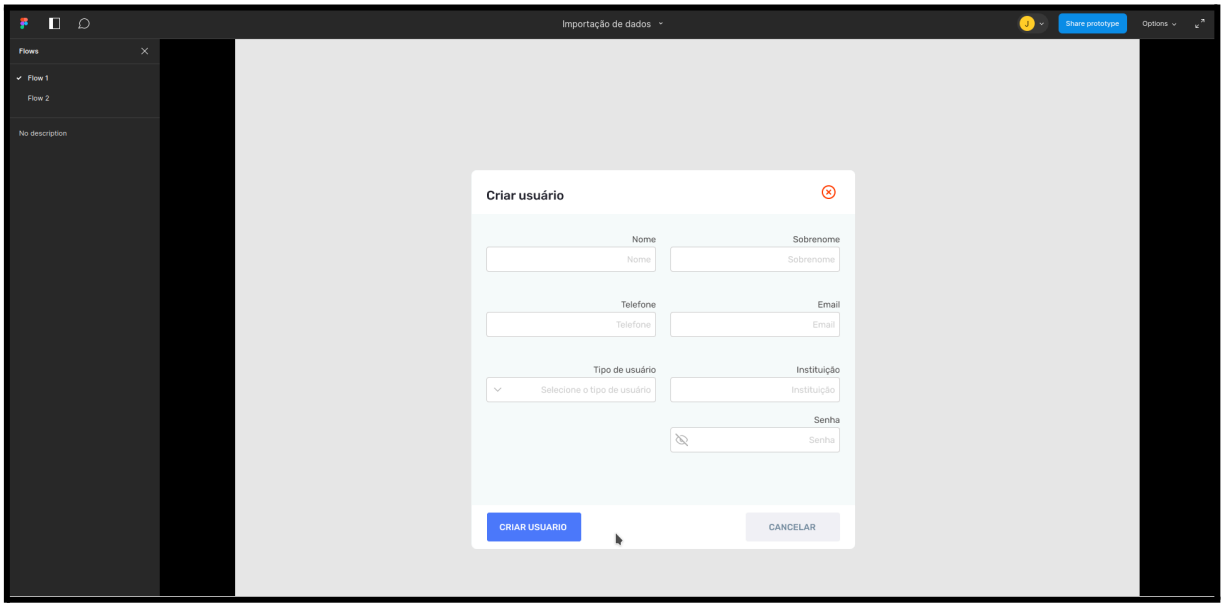
Fonte - O autor (2023).

Figura 8 - Protótipo página de visualização das planilhas



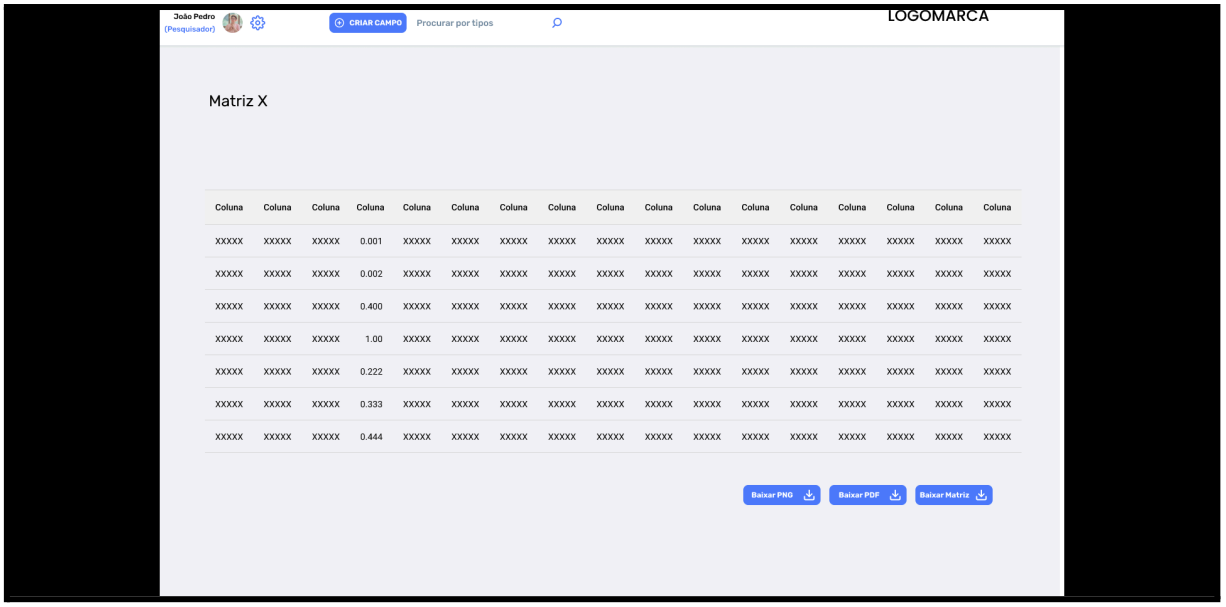
Fonte - O autor (2023).

Figura 9 - Protótipo tela de cadastro do usuário



Fonte - O autor (2023).

Figura 10 - Protótipo tela de visualização dos dados da planilha



Fonte - O autor (2023).

APÊNDICE E - PÁGINA INICIAL DA PLATAFORMA

Segue a Figura 11 da página inicial da plataforma, à qual é visualizada assim que se acessa o link da plataforma.

Figura 11 - Página inicial

Fonte - O autor (2023).

APÊNDICE F - PÁGINA DE LOGIN

Segue a Figura 12 da página de login da plataforma, à qual pode ser acessada ao clicar sobre o botão com a label ‘Login’ na página inicial.

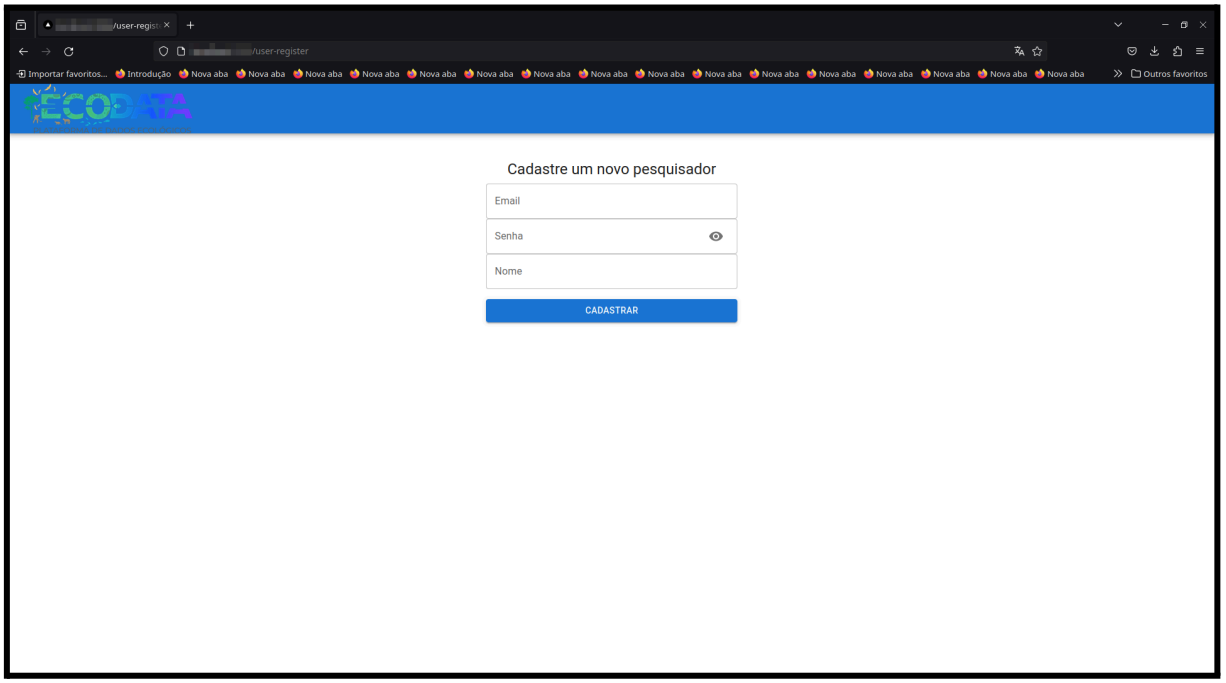
Figura 12 - Página de login

Fonte - O autor (2023).

APÊNDICE G - CADASTRO DO USUÁRIO

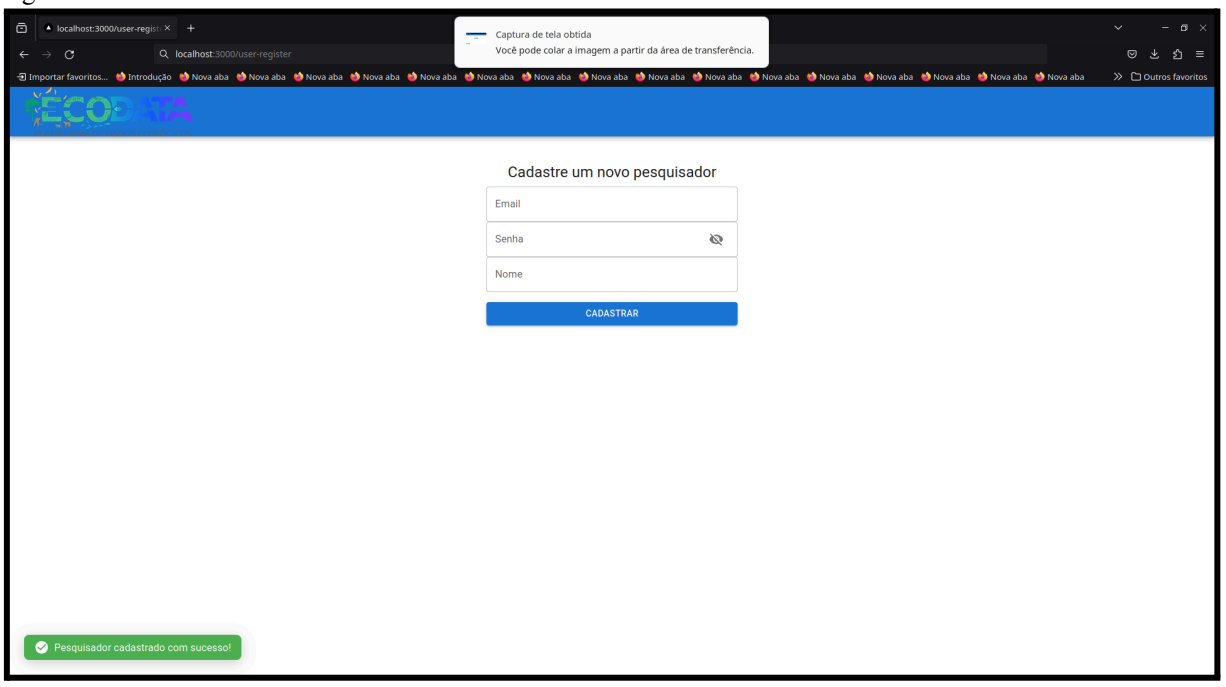
Segue a página de cadastro de usuário demonstrando o cadastro de um usuário (Figuras 13 e 14) a qual deve ser acessada via link, no path ‘<link-da-plataforma>/user-register’.

Figura 13 - Página de cadastro de usuário



Fonte - O autor (2023).

Figura 14 - Cadastro do usuário com sucesso

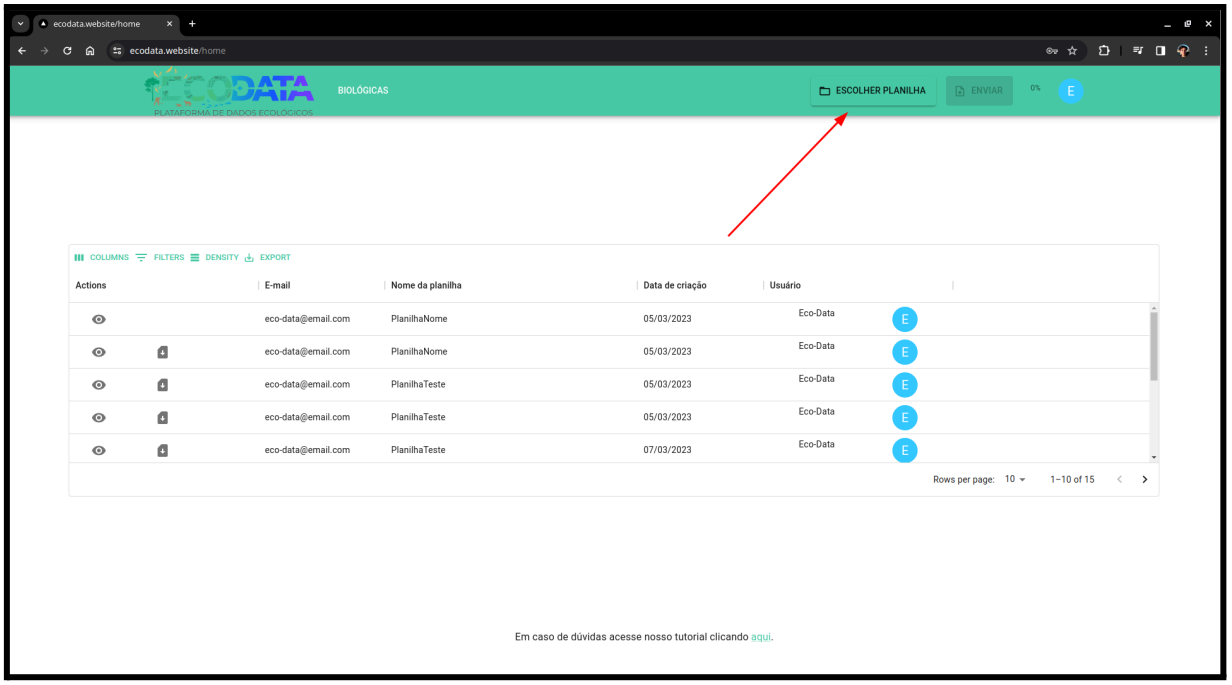


Fonte - O autor (2023).

APÊNDICE H - ENVIO DA PLANILHA PARA PROCESSAMENTO

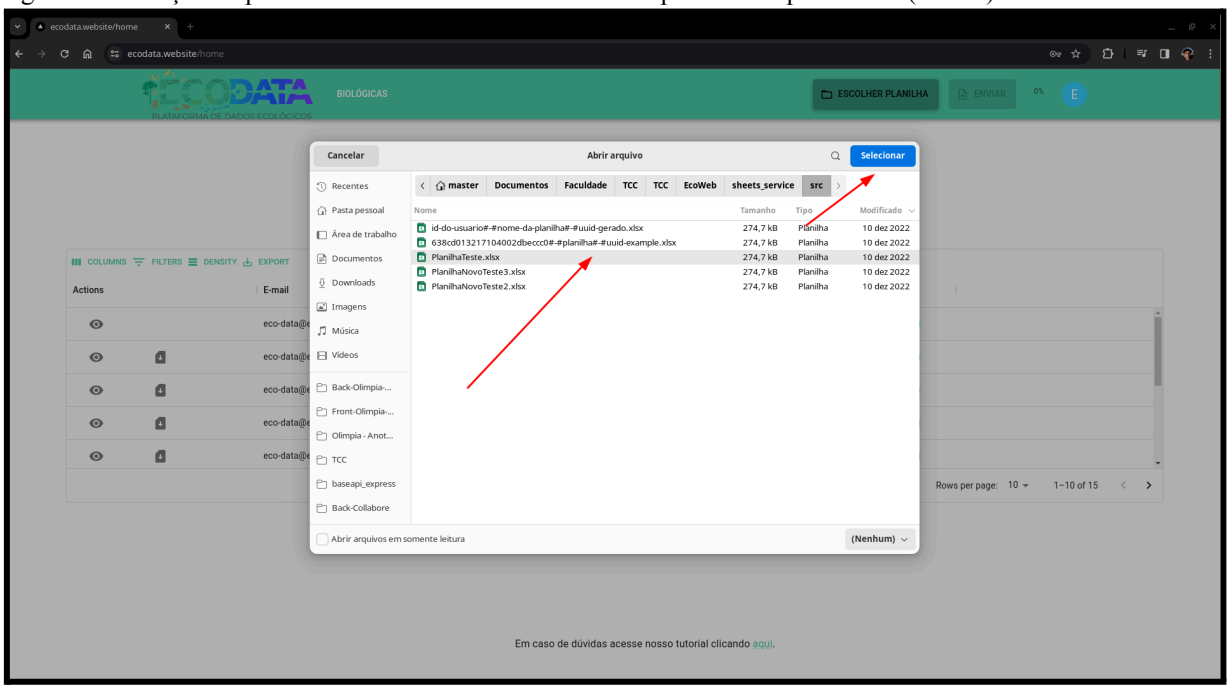
Segue o fluxo de envio da planilha para processamento (Figuras 15 a 19) demonstrando as etapas para realizar o envio da planilha para processamento.

Figura 15 - Botão de seleção da planilha



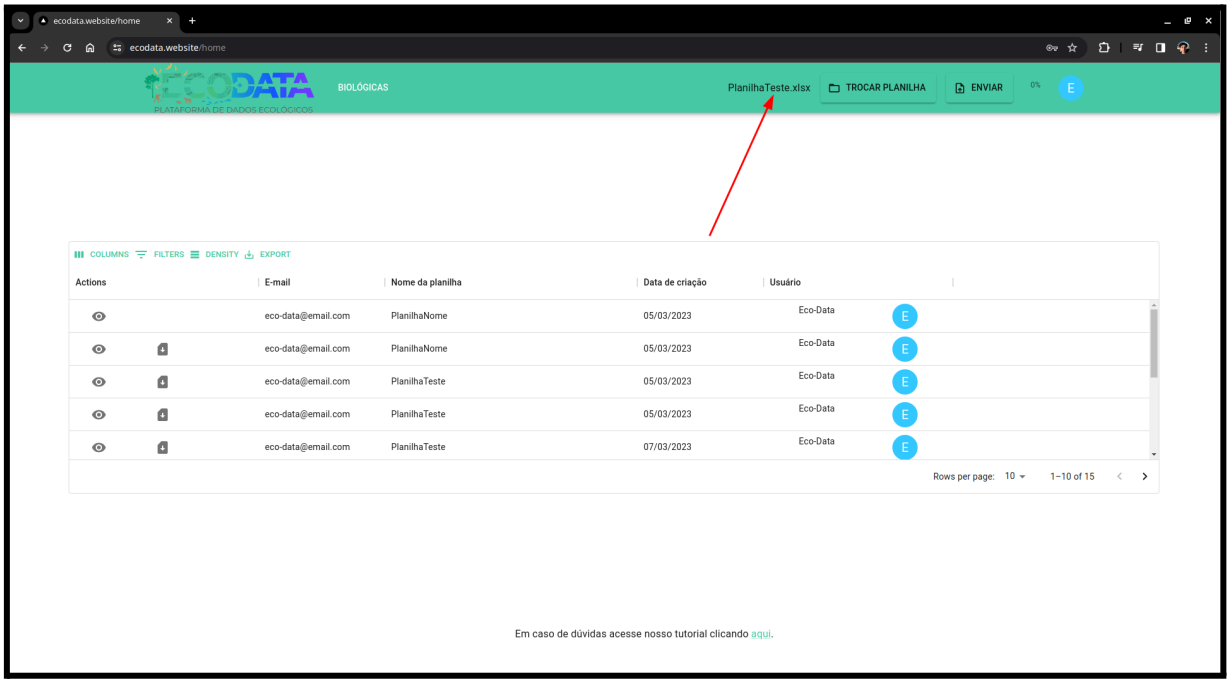
Fonte - O autor (2023).

Figura 16 - Seleção da planilha no formato Microsoft Excel Open XML Spreadsheet (XLSX)



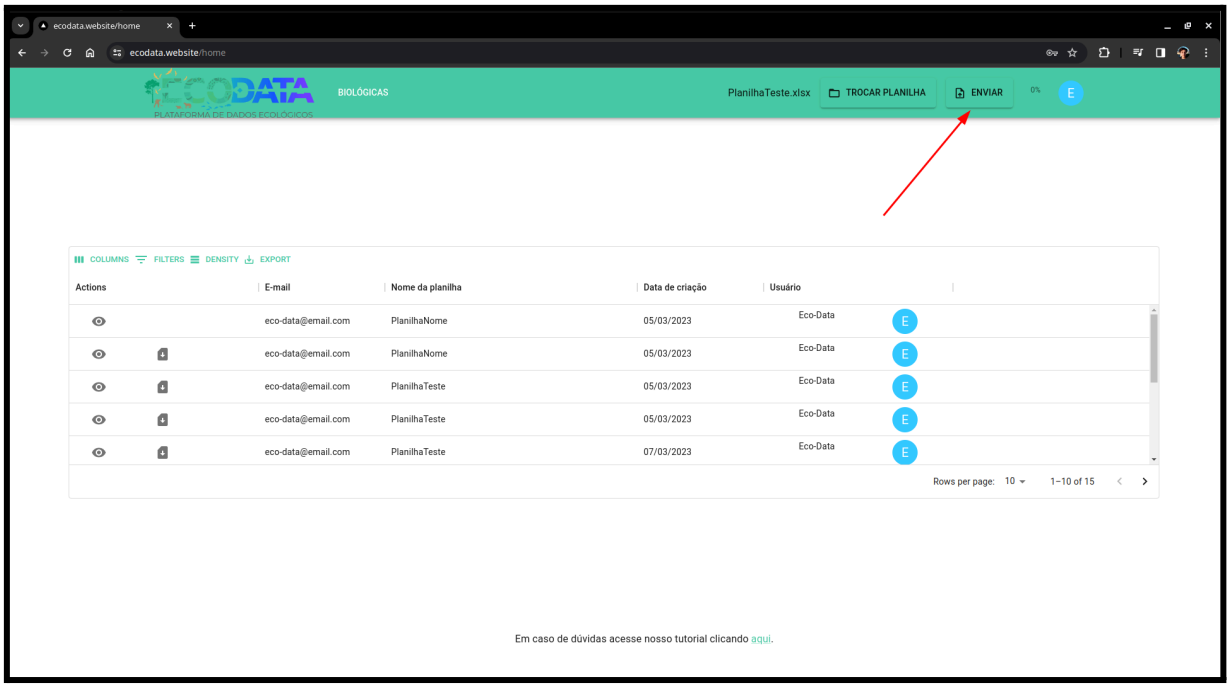
Fonte - O autor (2023).

Figura 17 - Planilha selecionada



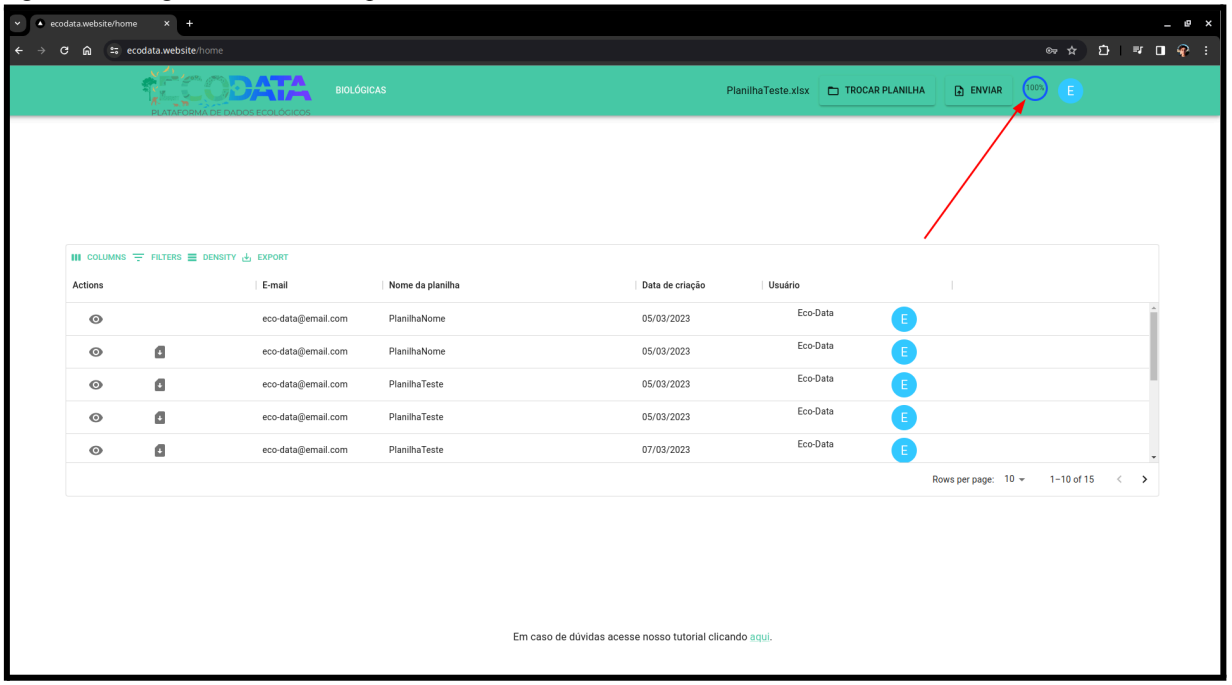
Fonte - O autor (2023).

Figura 18 - Botão de envio da planilha



Fonte - O autor (2023).

Figura 19 - Progresso do envio da planilha ao servidor

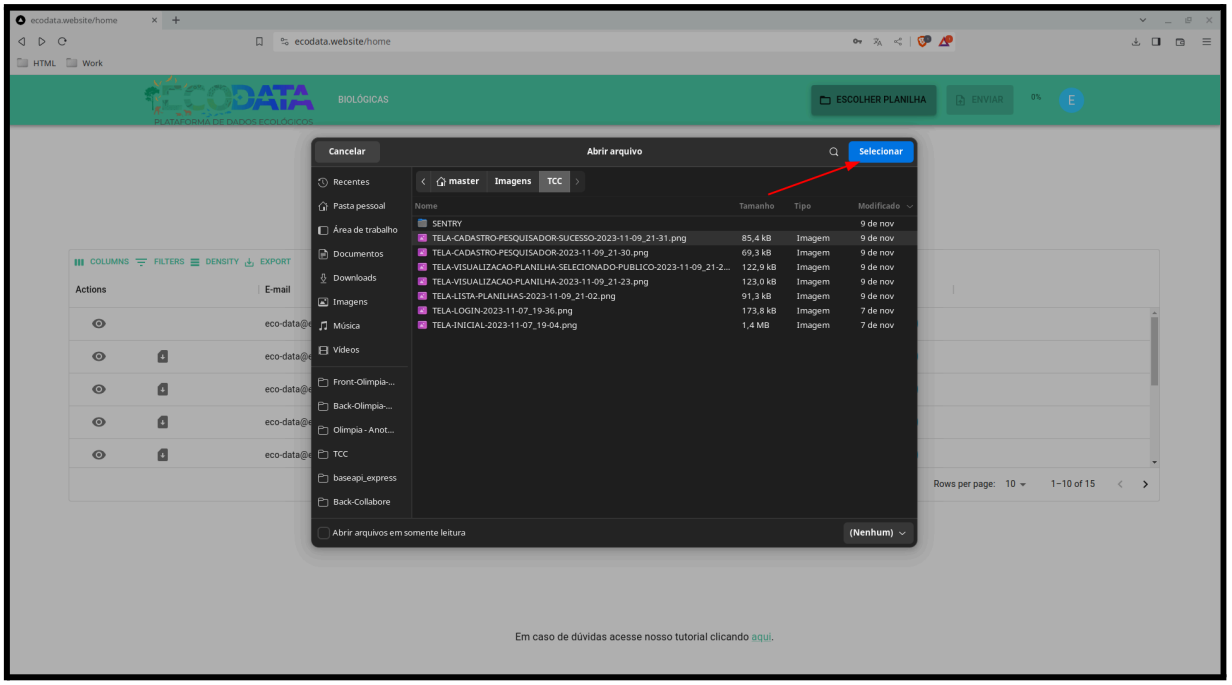


Fonte - O autor (2023).

APÊNDICE I - SELEÇÃO DO FORMATO DE ARQUIVO ERRADO PARA ENVIO

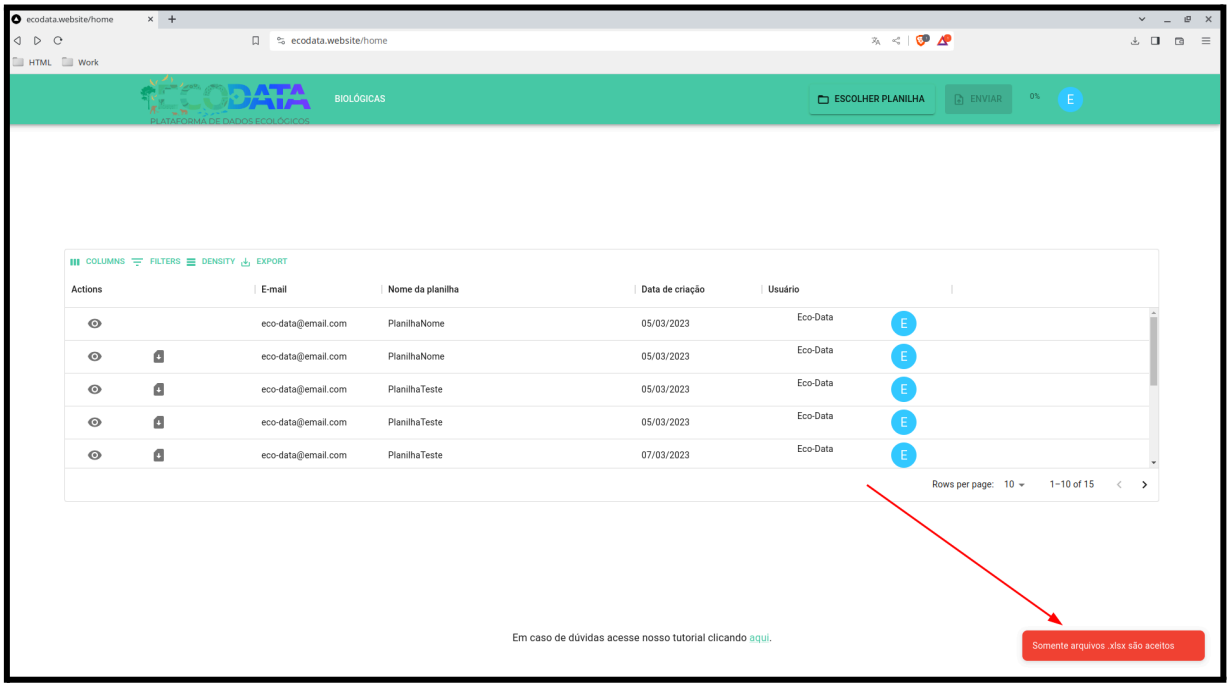
Segue exemplo de fluxo da seleção de arquivo do formato incorreto para o envio (Figuras 20 e 21) demonstrando a notificação de erro.

Figura 20 - Seleção de arquivo de formato errado



Fonte - O autor (2023).

Figura 21 - Notificação de formato da planilha inválido

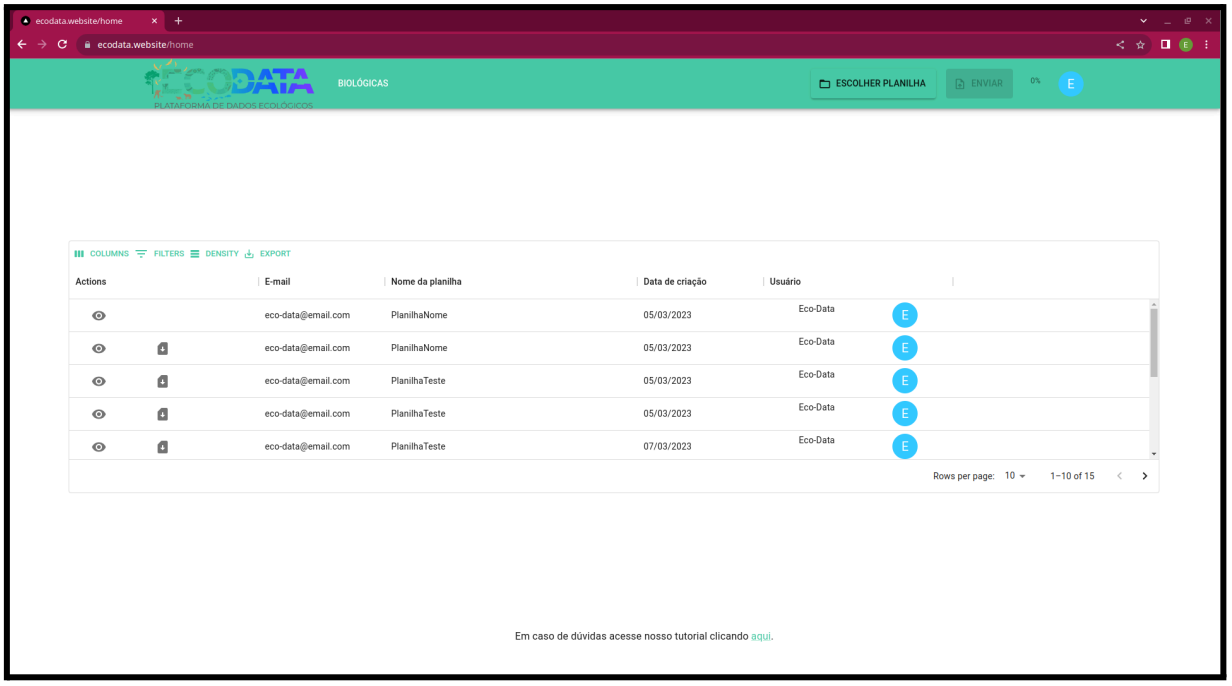


Fonte - O autor (2023).

APÊNDICE J - FLUXO DE VISUALIZAÇÃO DOS DADOS PROCESSADOS DA PLANILHA

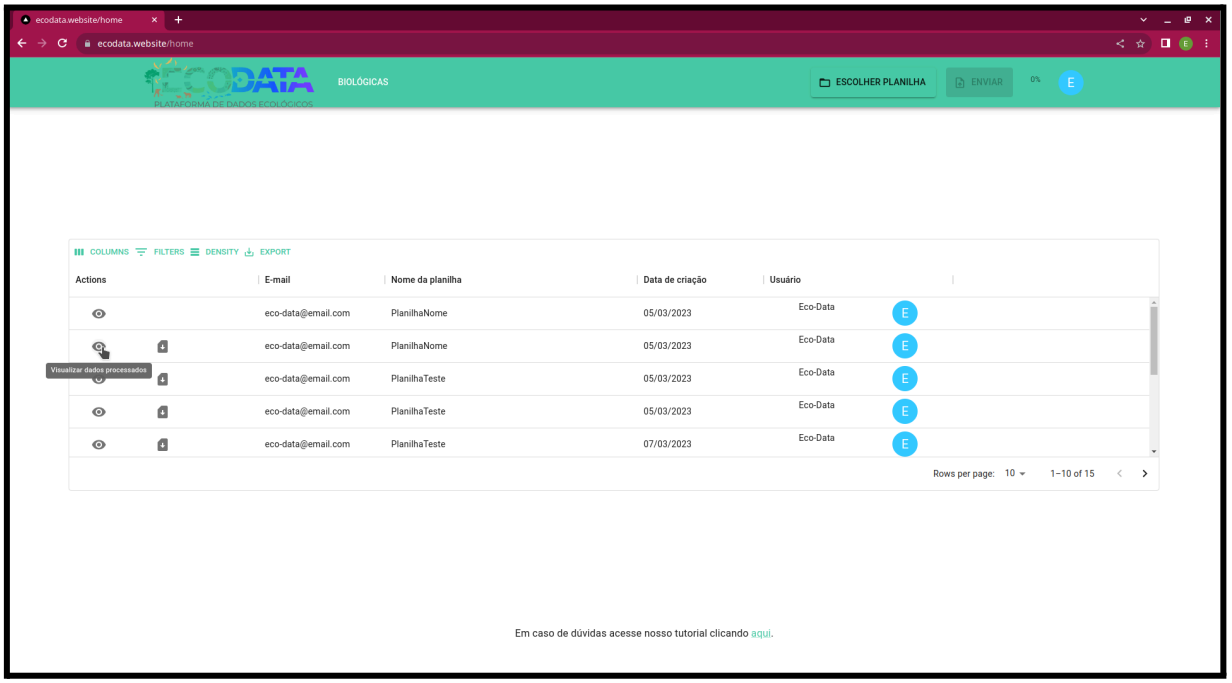
Segue o fluxo de visualização dos dados processados (Figuras 22 a 24) demonstrando as etapas em sequências para visualizar os dados processados da planilha.

Figura 22 - Página de planilhas processadas



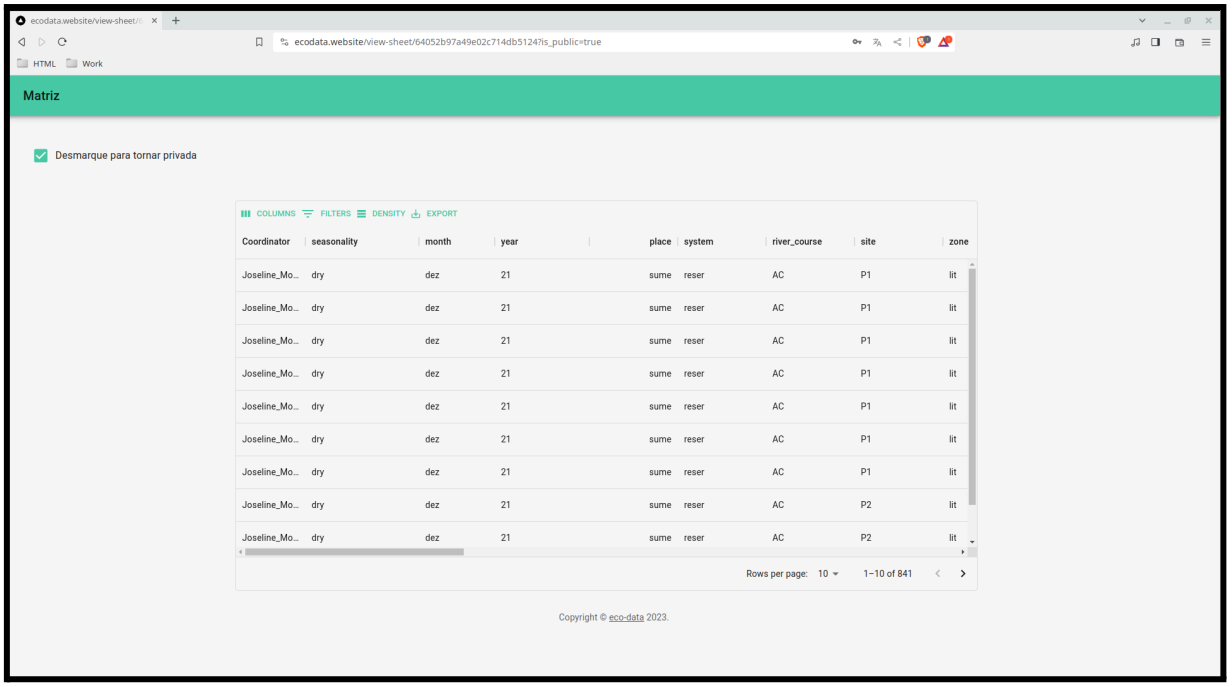
Fonte - O autor (2023).

Figura 23 - Seleção do ícone de visualizar dados processados



Fonte - O autor (2023).

Figura 24 - Página de visualização da planilha

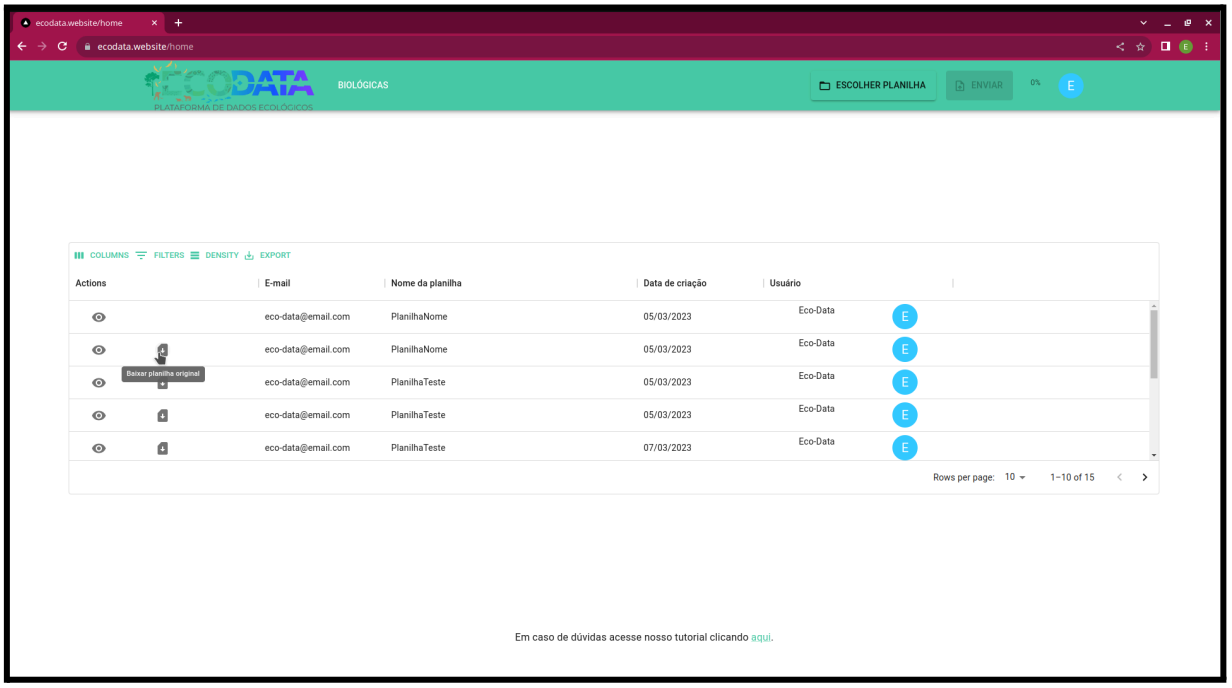


Fonte - O autor (2023).

APÊNDICE K - BAIXAR PLANILHA PÚBLICA ORIGINAL

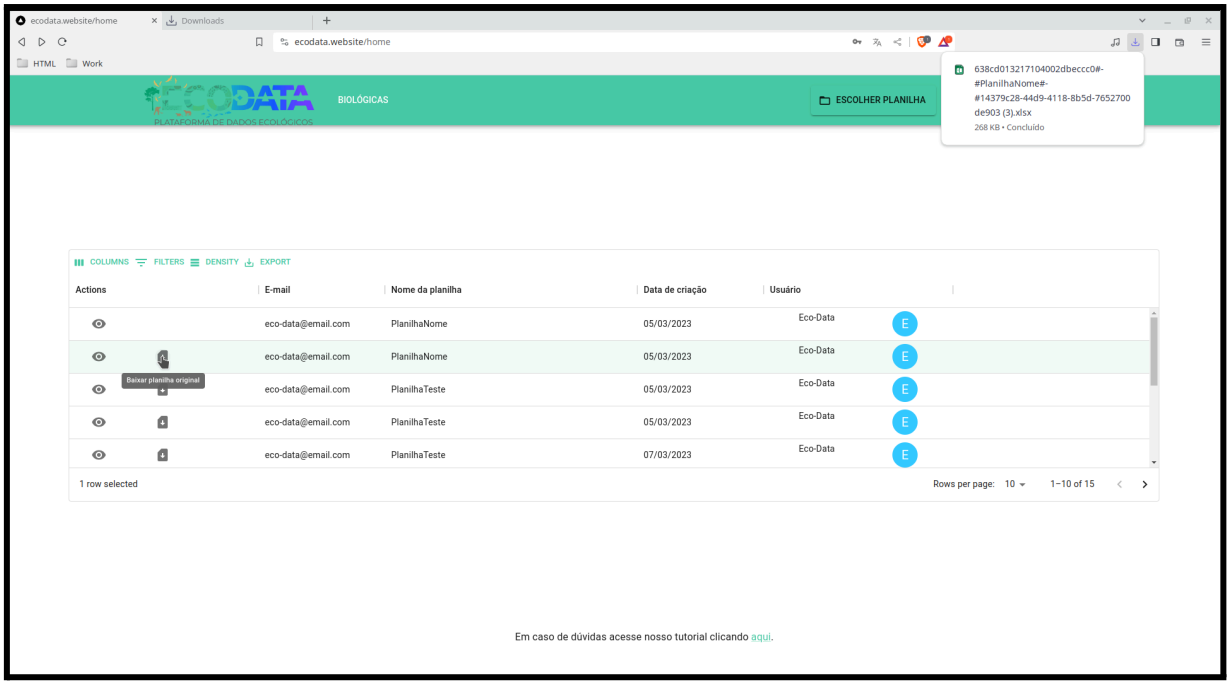
Seguem as etapas em sequência para baixar a planilha original que estiver pública (Figuras 25 e 26).

Figura 25 - Seleção do ícone de baixar planilha original



Fonte - O autor (2023).

Figura 26 - Planilha original baixada com sucesso

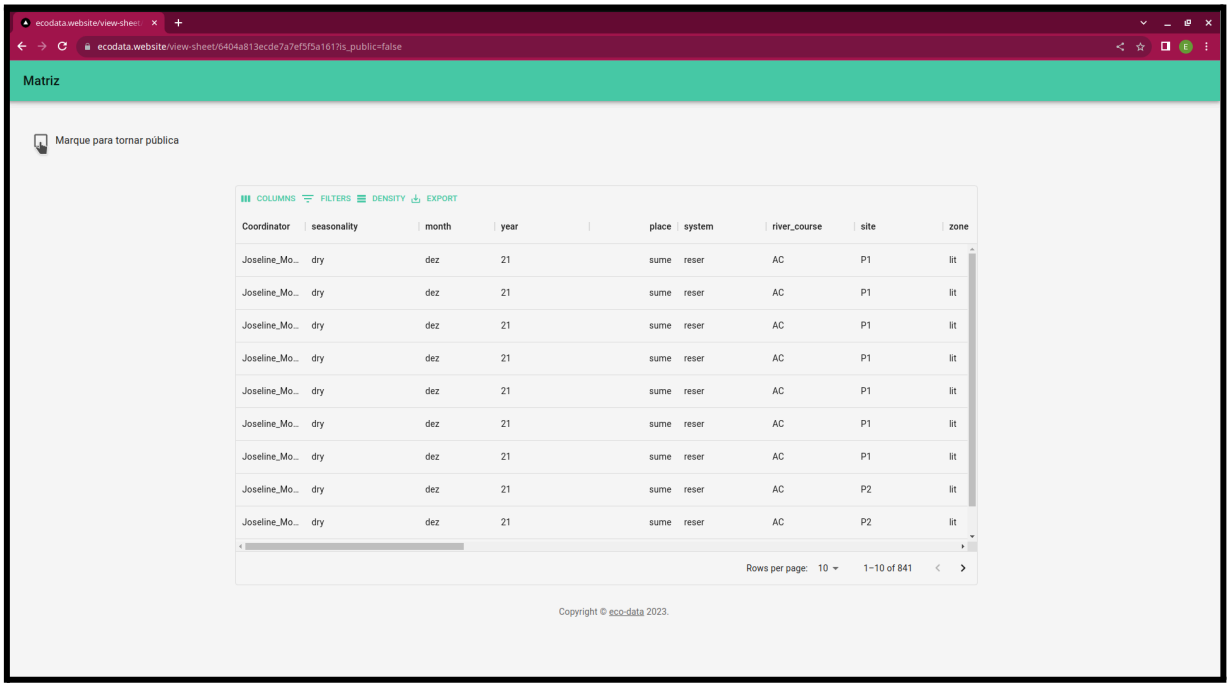


Fonte - O autor (2023).

APÊNDICE L - TORNAR PLANILHA PÚBLICA PARA DOWNLOAD

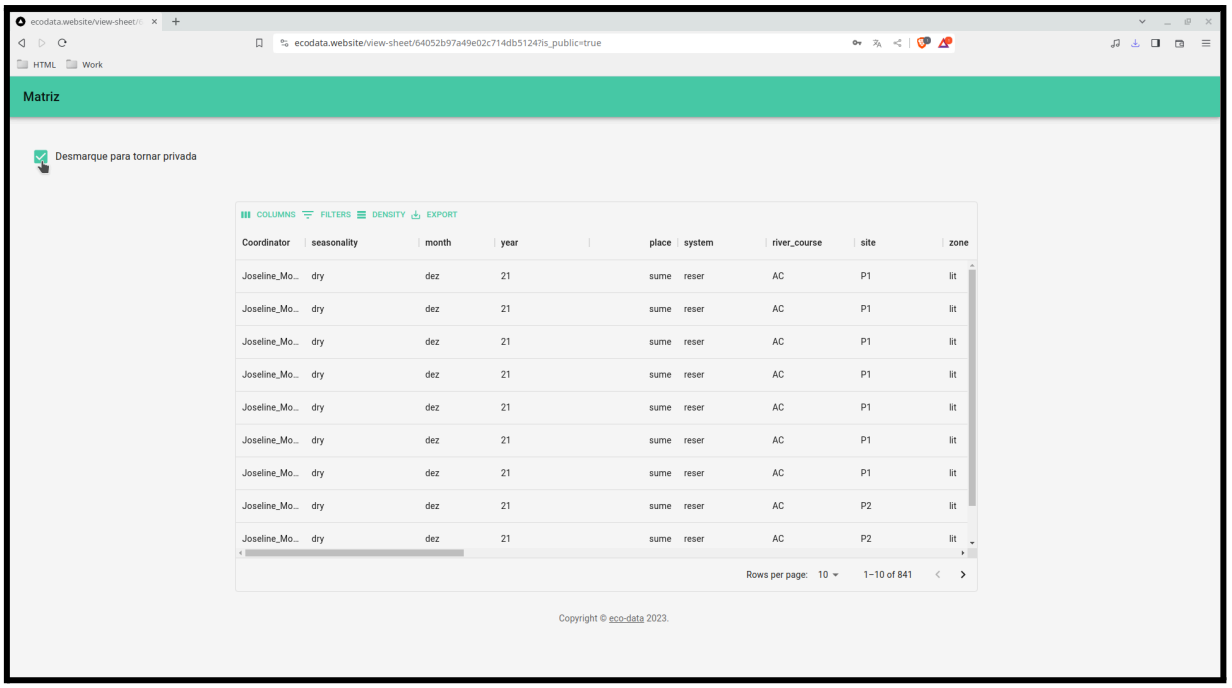
Seguem as etapas em sequência para tornar a planilha enviada pública para download (Figuras 27 e 28).

Figura 27 - Caixa de marcação para tornar planilha pública para download



Fonte - O autor (2023).

Figura 28 - Caixa de marcação para tornar planilha pública para download preenchida

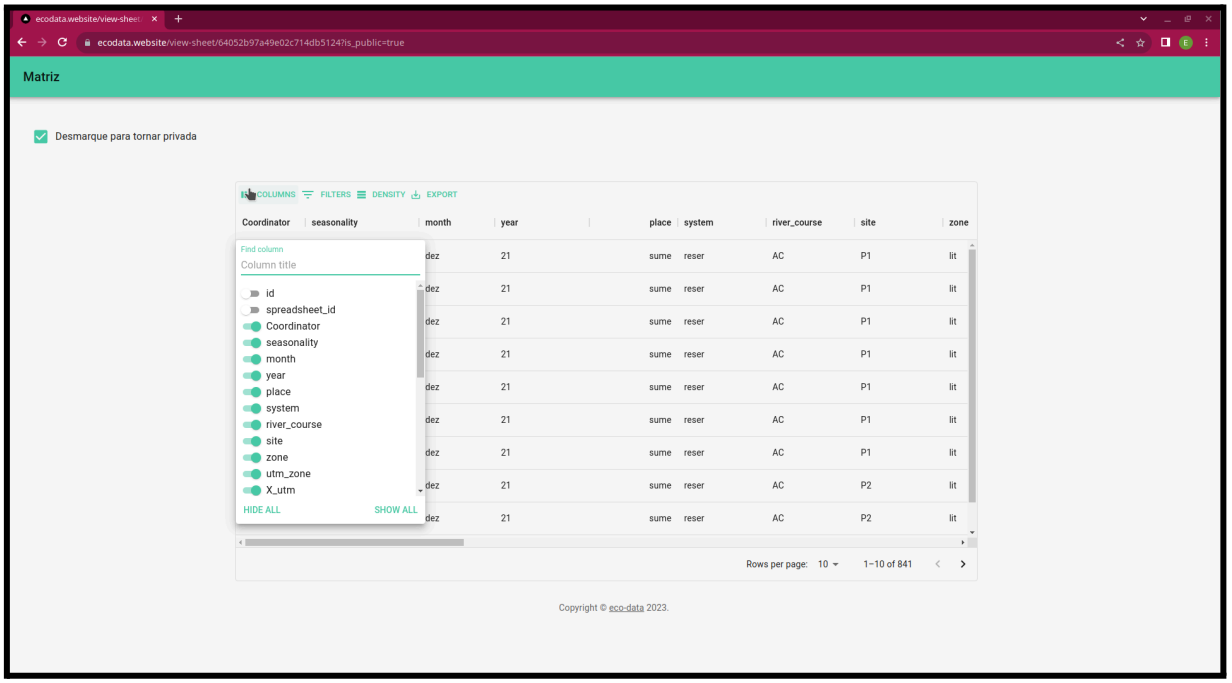


Fonte - O autor (2023).

APÊNDICE M - EDITAR COLUNAS VISÍVEIS DOS DADOS PROCESSADOS

Segue a Figura 29 demonstrando a edição das colunas visíveis nos dados processados da planilha.

Figura 29 - Edição de colunas visíveis dos dados processados

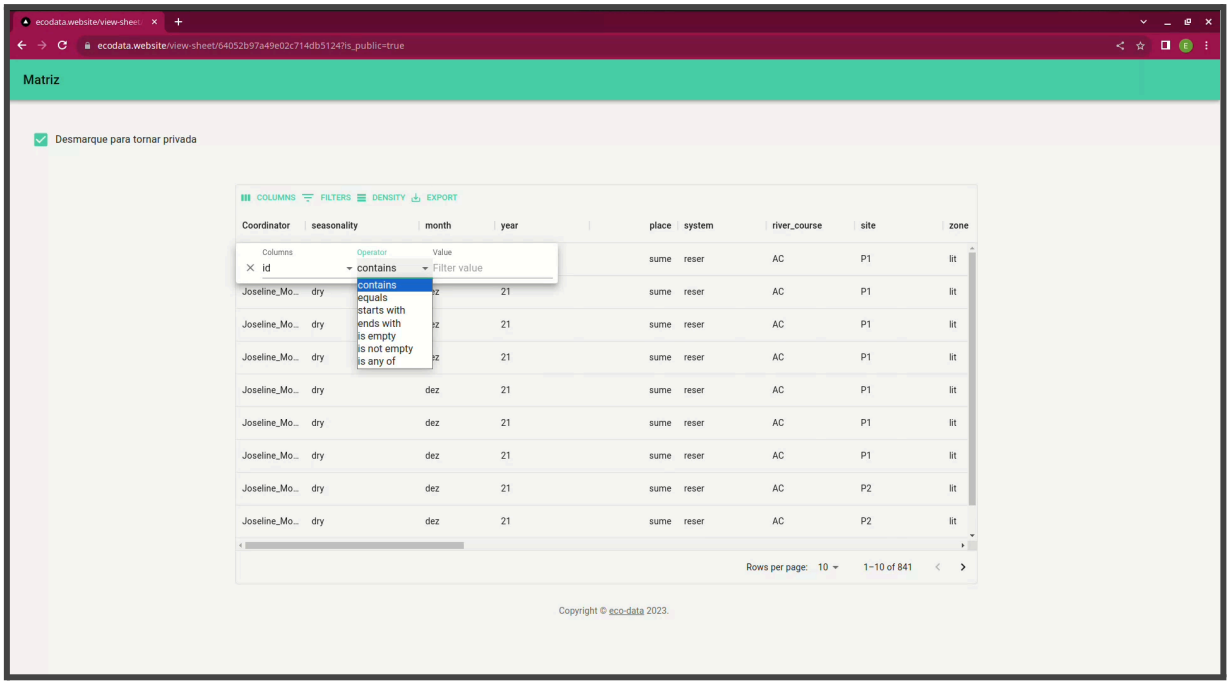


Fonte - O autor (2023).

APÊNDICE N - FILTRAR DADOS PROCESSADOS VISÍVEIS

Segue a Figura 30 demonstrando como aplicar filtros nos dados processados visíveis na página atual.

Figura 30 - Filtros dos dados processados visíveis na tela

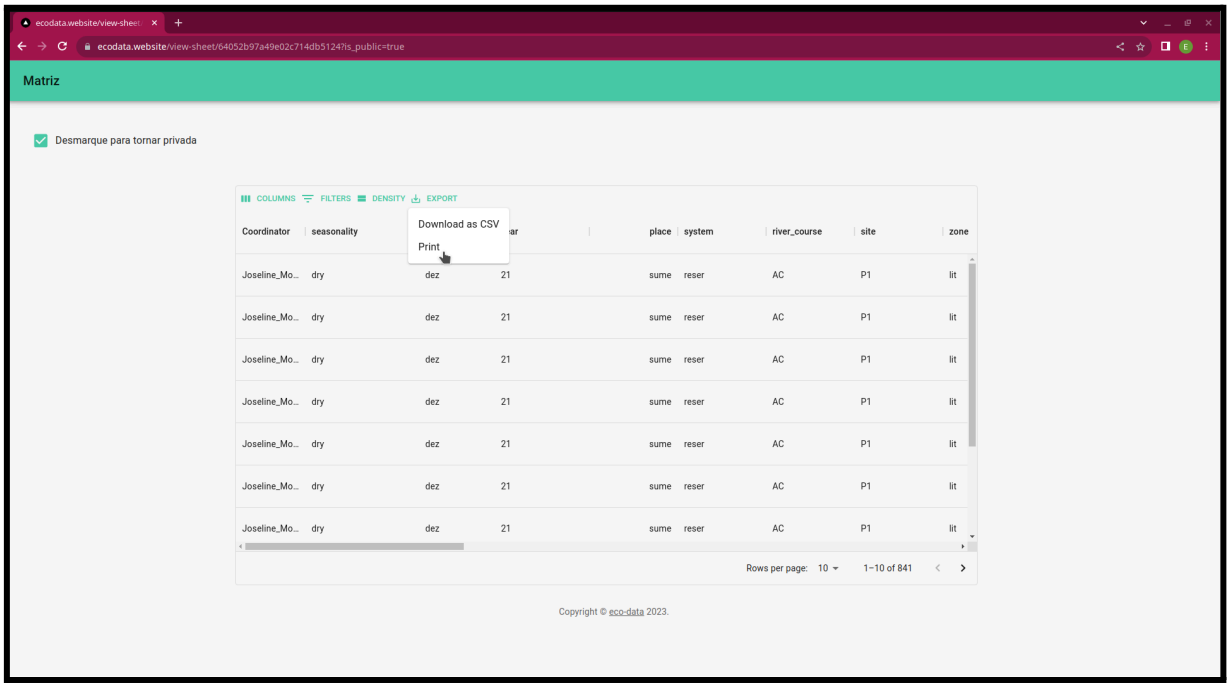


Fonte - O autor (2023).

APÊNDICE O - EXPORTAR PDF DOS DADOS PROCESSADOS VISÍVEIS

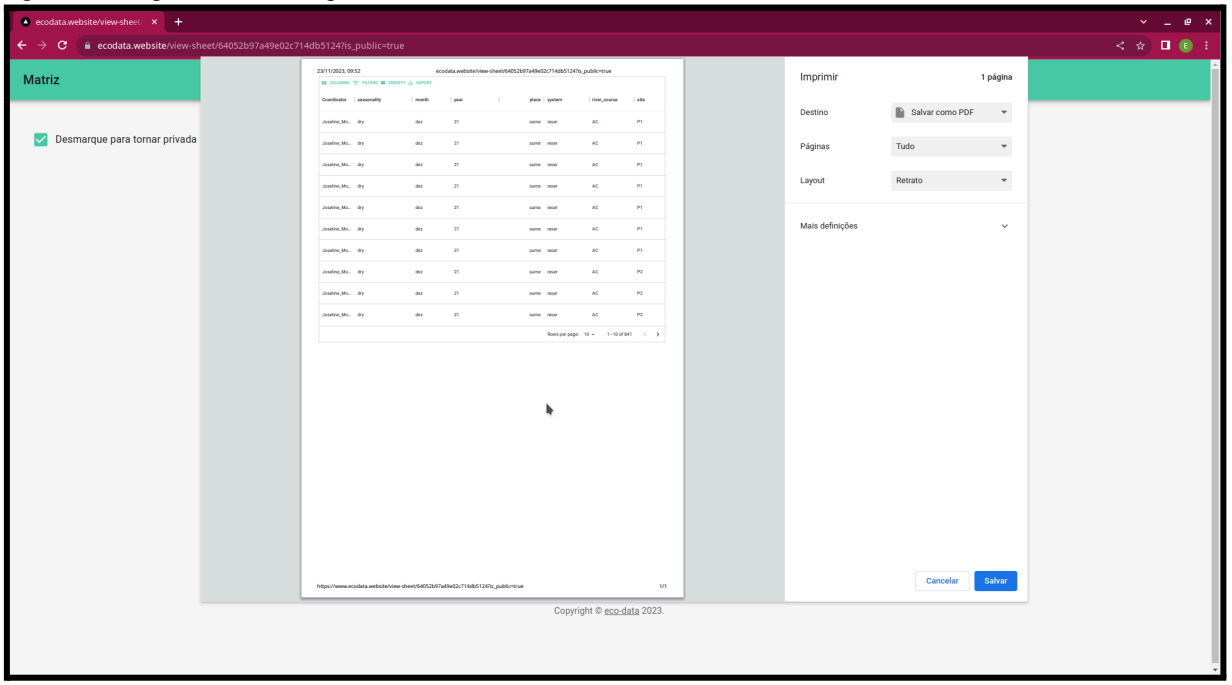
Segue figura demonstrando as etapas em sequência para se exportar um print dos dados processados visíveis na pagina atual para serem salvos no formato de *Portable Document Format* (PDF) (Figuras 31 e 32).

Figura 31 - Exportar dados visíveis para uma captura



Fonte - O autor (2023).

Figura 32 - Captura dos dados processados visíveis

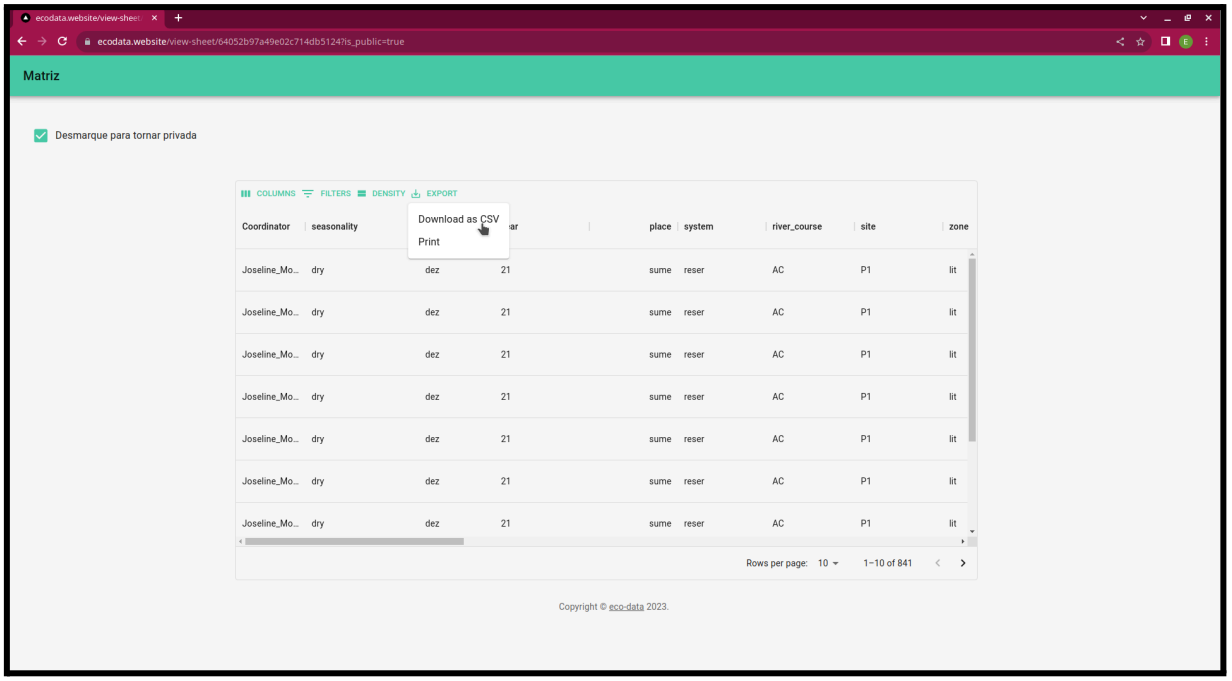


Fonte - O autor (2023).

APÊNDICE P - EXPORTAR CSV DOS DADOS PROCESSADOS VISÍVEIS

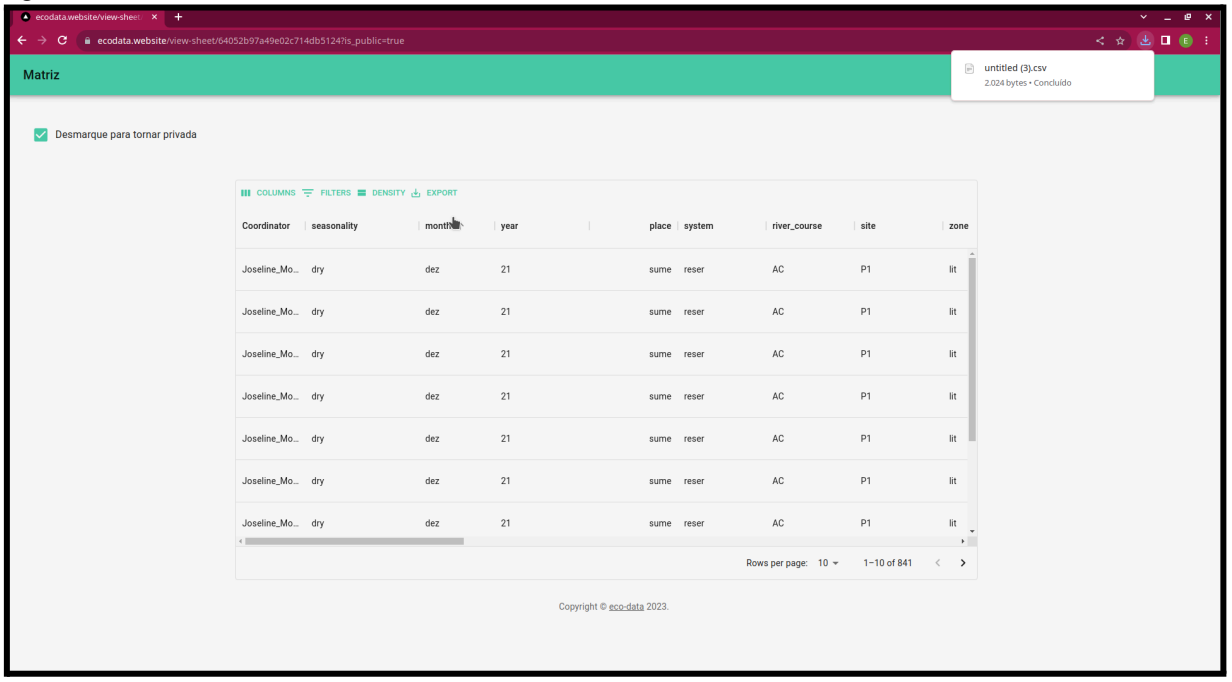
Segue figura demonstrando as etapas em sequência para se exportar um csv dos dados processados visíveis na pagina atual (Figuras 33 e 34).

Figura 33 - Realizar download dos dados visíveis como CSV



Fonte - O autor (2023).

Figura 34 - Download dos dados visíveis como CSV concluído

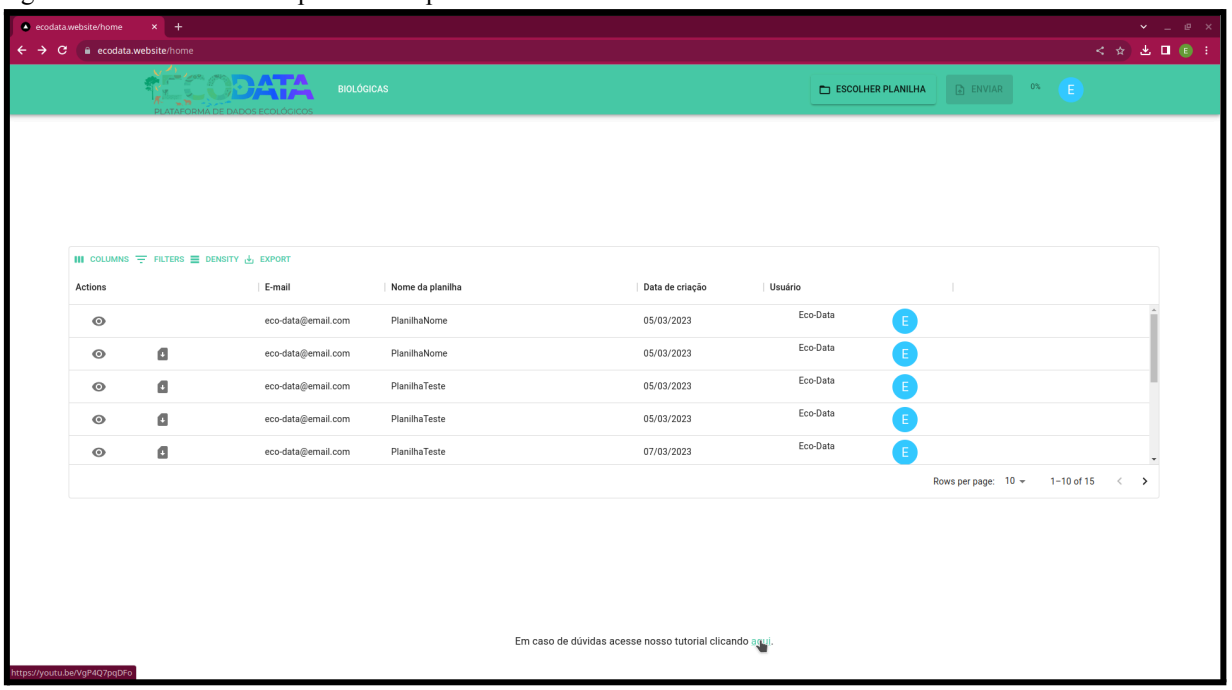


Fonte - O autor (2023).

APÊNDICE Q - ACESSO AO TUTORIAL DA PLATAFORMA

Segue figura demonstrando as etapas em sequência para realizar o acesso ao link do tutorial da plataforma (Figuras 35 e 36).

Figura 35 - Link do tutorial para uso da plataforma



Fonte - O autor (2023).

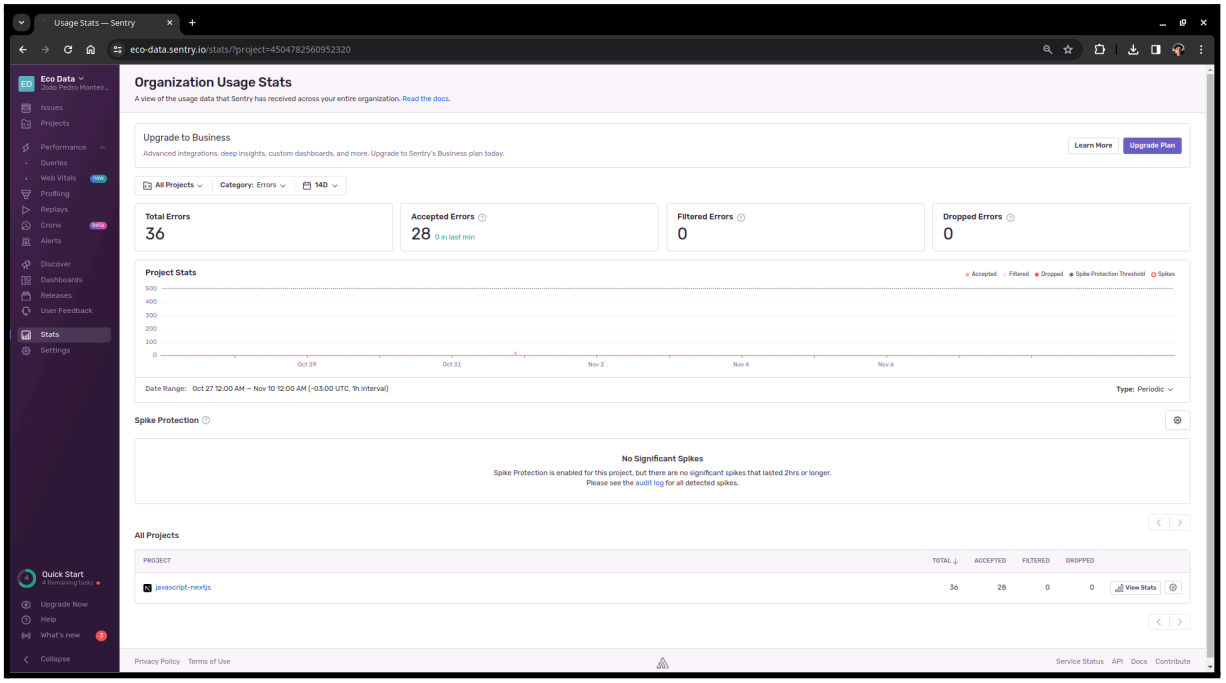
Figura 36 - Tutorial para utilização da plataforma

Fonte - O autor (2023).

APÊNDICE R - MONITORAMENTO E OBSERVABILIDADE DO FRONT-END

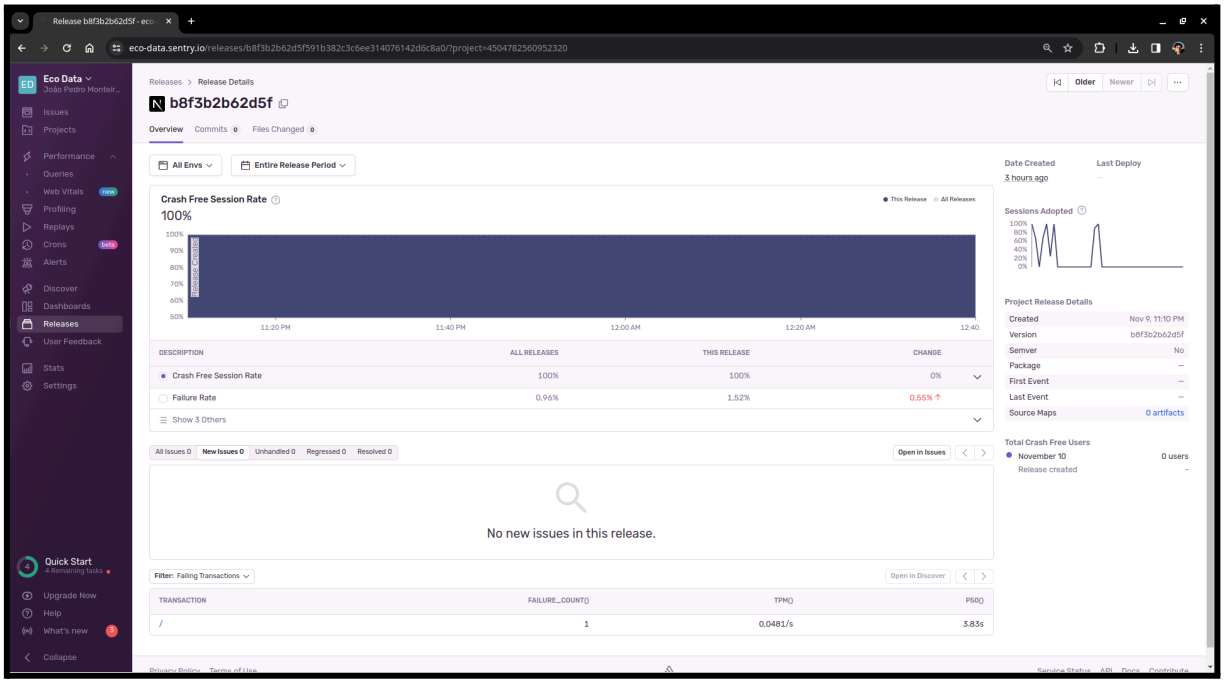
Seguem as páginas de monitoramento e observabilidade do *front-end* (Figuras 37 a 40) disponíveis na ferramenta do sentry que foi integrado ao projeto para este fim.

Figura 37 - Status de uso da organização no sentry



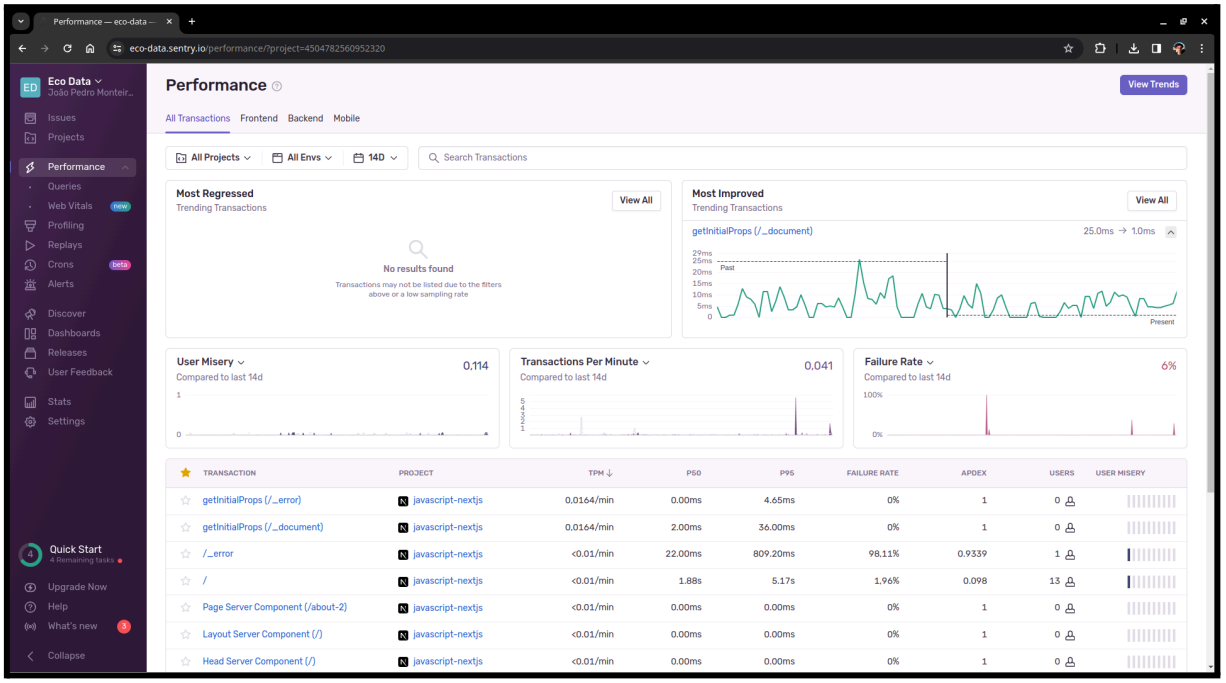
Fonte - O autor (2023).

Figura 38 - Dados de releases efetuados



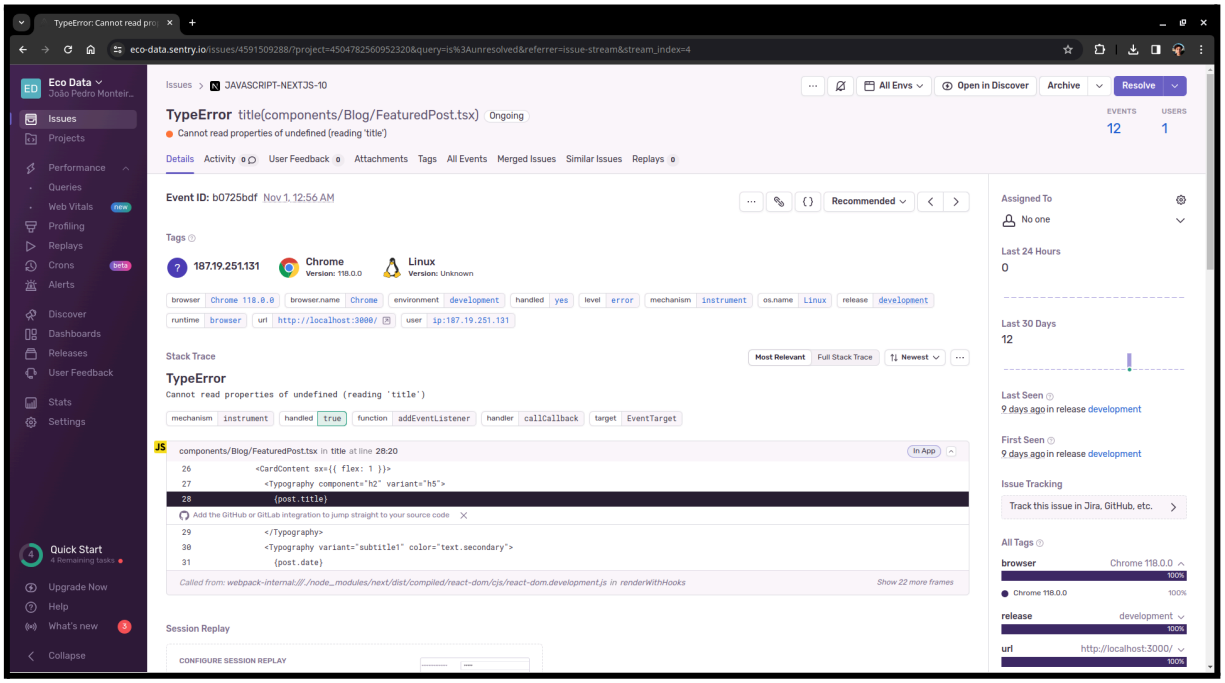
Fonte - O autor (2023).

Figura 39 - Dados de performance do front-end



Fonte - O autor (2023).

Figura 40 - Detalhes de erros ocorridos durante a utilização da plataforma

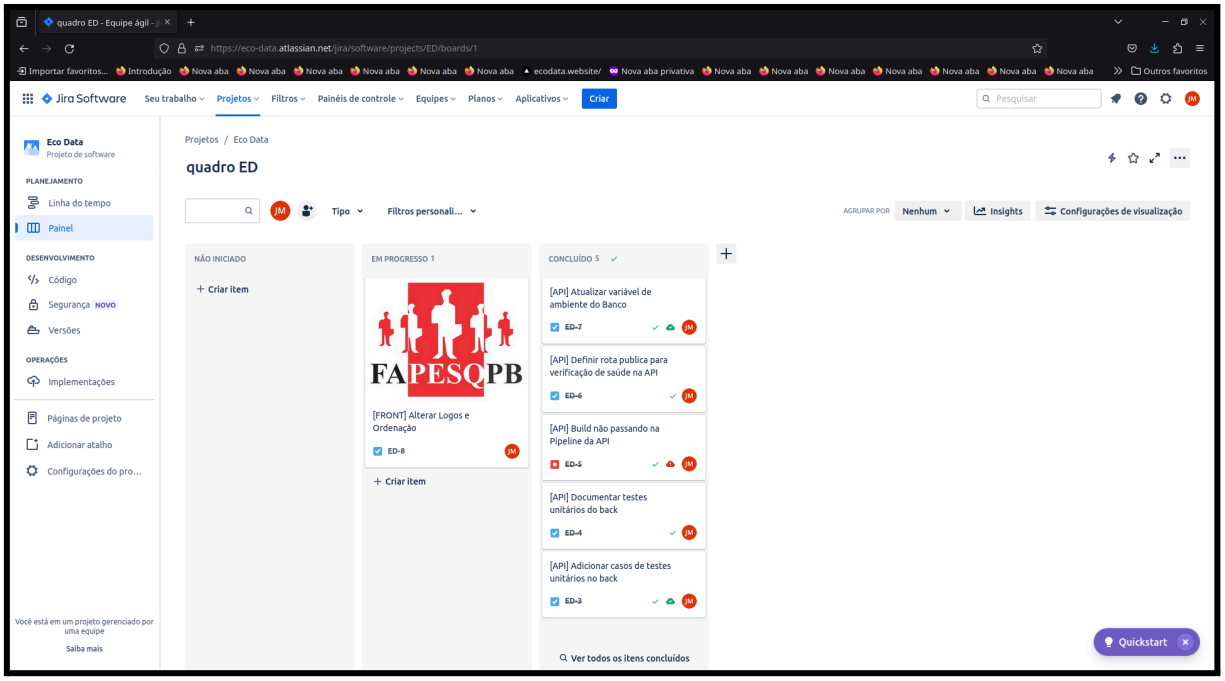


Fonte - O autor (2023).

APÊNDICE S - GESTÃO DO PROJETO NO JIRA

Seguem as páginas da ferramenta do Jira (Figuras 41 a 45) que foi utilizado para gerir o projeto do desenvolvimento da plataforma utilizando a metodologia do kanban[19], além de demonstrar a integração aplicada entre o github e o Jira para acompanhamento do desenvolvimento e deploy do código diretamente no painel do jira.

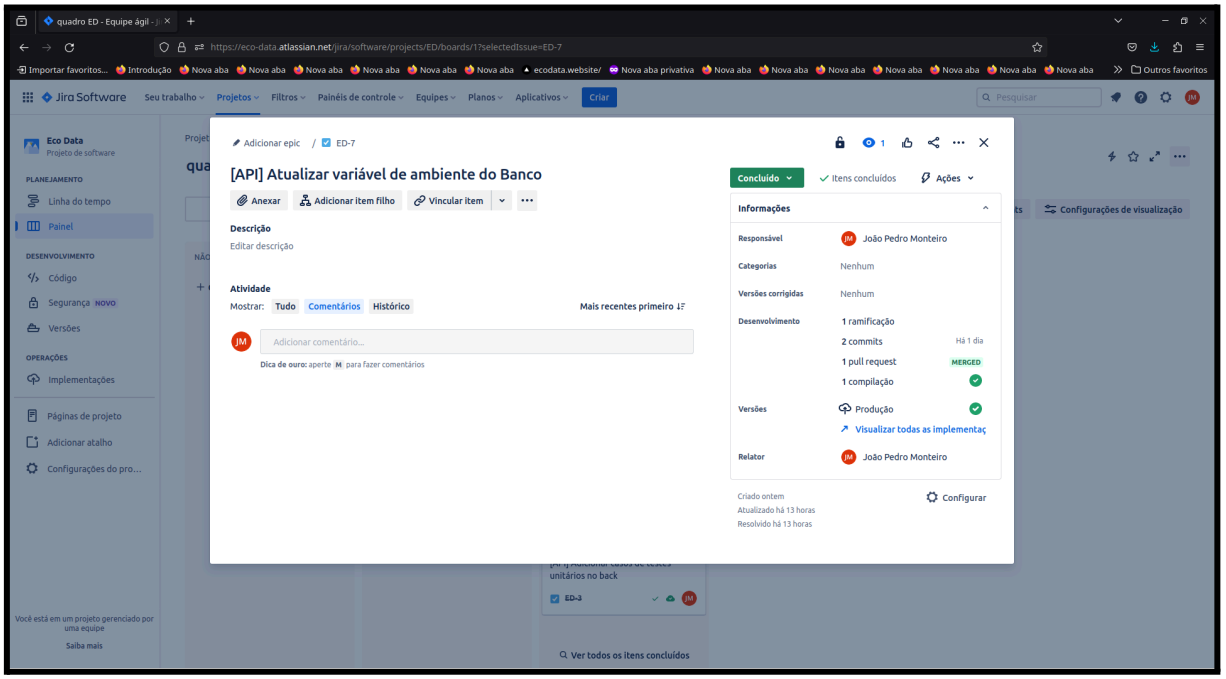
Figura 41 - Painel kanban do Jira



Fonte - O autor (2023).

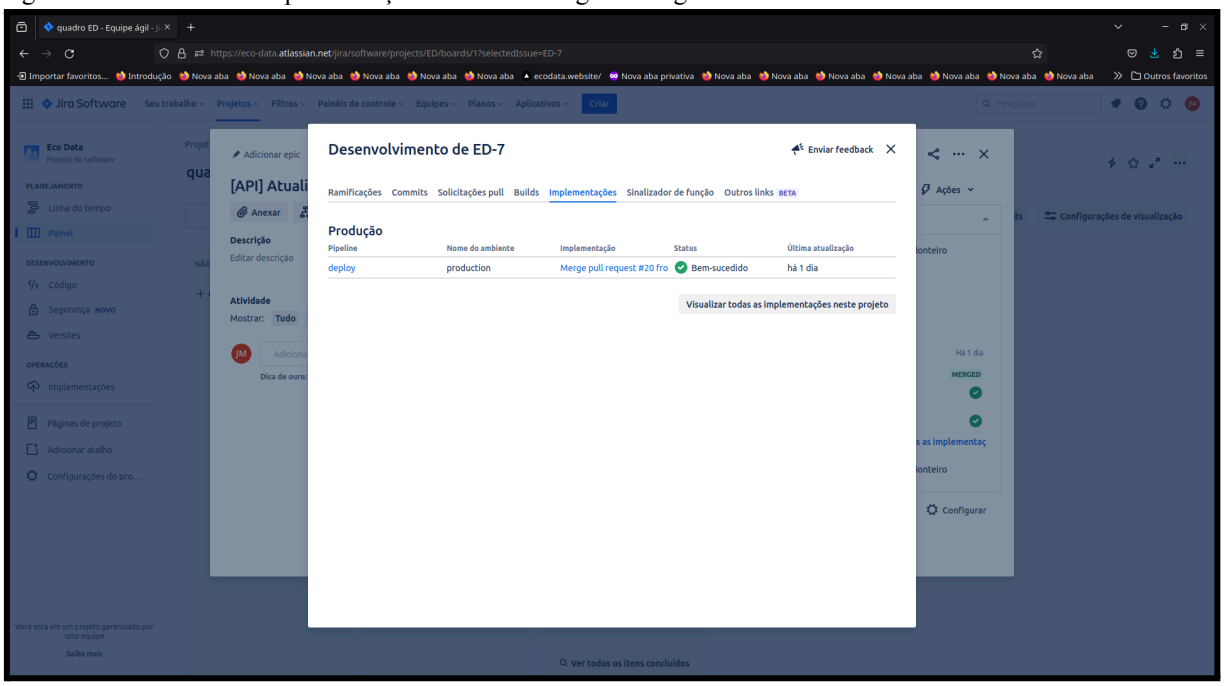
[19] Kanban é uma metodologia ágil de gestão de projetos e fluxos de trabalho, originária do sistema de produção da Toyota. Caracteriza-se pelo uso de um quadro visual (Kanban board) para mapear e acompanhar as tarefas em diferentes estágios de execução (como "A Fazer", "Em Andamento", "Concluído"). O foco do Kanban está na otimização do fluxo de trabalho e na eficiência, permitindo uma visão clara do progresso e facilitando a identificação de gargalos no processo.

Figura 42 - Detalhes da tarefa do Jira



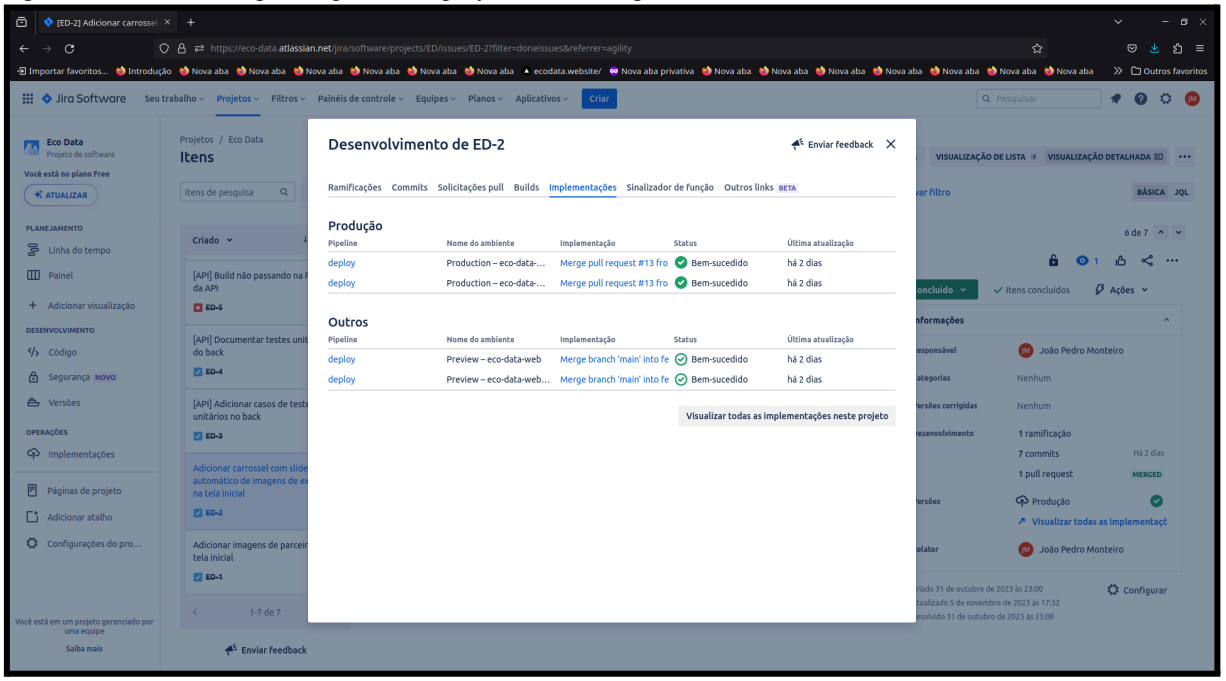
Fonte - O autor (2023).

Figura 43 - Detalhes de implementações da tarefa integrada ao github



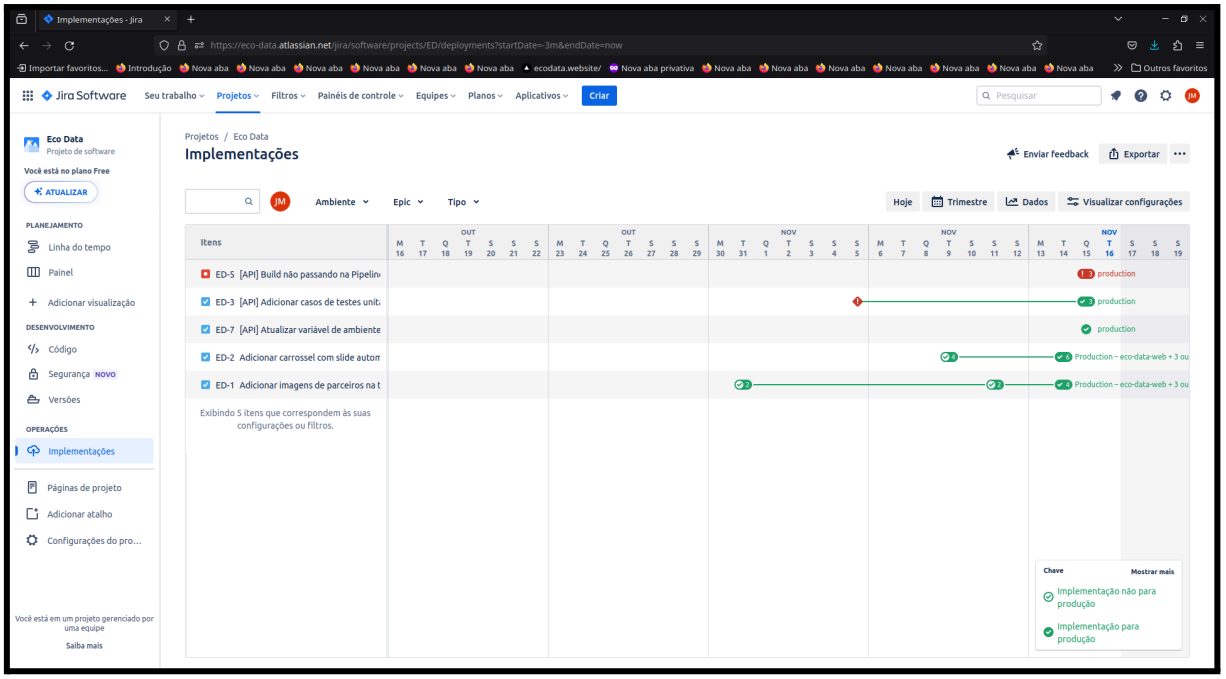
Fonte - O autor (2023).

Figura 44 - Detalhes de pull requests e deploys efetuados a partir da tarefa do Jira



Fonte - O autor (2023).

Figura 45 - Detalhes de deploys efetuados no Jira



Fonte - O autor (2023).

APÊNDICE T - LINK DO TUTORIAL DE USO DA PLATAFORMA

https://bit.ly/tutorial-plataforma-ecodata

APÊNDICE U - LINK DO PROTÓTIPO DA PLATAFORMA

https://bit.ly/prototipo-eco-data

APÊNDICE V - LINK DO REPOSITÓRIO FRONT END DA PLATAFORMA

https://bit.ly/repositorio-front-end-plataforma-eco-data

APÊNDICE W - LINK DO REPOSITÓRIO BACK END DA PLATAFORMA

https://bit.ly/repositorio-back-end-plataforma-eco-data

APÊNDICE X - LINK DO REPOSITÓRIO DO PROCESSAMENTO DE PLANILHA DA PLATAFORMA

https://bit.ly/repositorio-processamento-da-planilha-plataforma-eco-data

APÊNDICE Y - LINK DE ACESSO A PLATAFORMA

https://bit.ly/plataforma-eco-data