INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
CAMPUS CAJAZEIRAS
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

API GRAPHQL E FERRAMENTA DE VISUALIZAÇÃO GRÁFICA SOBRE O PORTAL DE DADOS DO IFPB

MOACIR DAVID DE ALMEIDA GONÇALVES

Cajazeiras
2023

MOACIR DAVID DE ALMEIDA GONÇALVES

API GRAPHQL E FERRAMENTA DE VISUALIZAÇÃO GRÁFICA SOBRE O PORTAL DE DADOS DO IFPB

Trabalho de Conclusão de Curso apresentado junto ao Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba - Campus Cajazeiras, como requisito à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador

Prof. Me. Francisco Paulo de Freitas Neto.

Cajazeiras
2023

IFPB / Campus Cajazeiras
Coordenação de Biblioteca
Biblioteca Prof. Ribamar da Silva
Catalogação na fonte: Cícero Luciano Félix CRB-15/750

G635a Gonçalves, Moacir David de Almeida.
API GraphQL e ferramenta de visualização gráfica sobre o portal de dados do IFPB / Moacir David de Almeida Gonçalves.– 2023.

58f. : il.

Trabalho de Conclusão de Curso (Tecnólogo em Análise e Desenvolvimento de Sistemas) - Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, Cajazeiras, 2023.

Orientador(a): Profª. Me. Francisco Paulo de Freitas Neto.

1. Desenvolvimento de sistemas. 2. Gestão de dados. 3. Dados abertos. 4. Lei de acesso à informação. I. Instituto Federal de Educação, Ciência e Tecnologia da Paraíba. II. Título.

IFPB/CZ CDU: 004.4(043.2)

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA

MOACIR DAVID DE ALMEIDA GONÇALVES

API GRAPHQL E FERRAMENTA DE VISUALIZAÇÃO GRÁFICA SOBRE O PORTAL DE DADOS DO IFPB

Trabalho de Conclusão de Curso apresentado junto ao Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba - Campus Cajazeiras, como requisito à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador

Prof. MSc. Francisco Paulo de Freitas Neto.

Aprovada em: **29 de Agosto de 2023.**

Prof. MSc. Francisco Paulo de Freitas Neto - Orientador

Prof. MSc. Paulo Ewerton Gomes Fragoso - Avaliador

IFPB - Campus Cajazeiras

Prof. Esp. Asheley Emmy Lacerda Alves - Avaliadora

IFPB - Campus Cajazeiras

Documento assinado eletronicamente por:

- **Francisco Paulo de Freitas Neto**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 30/08/2023 17:28:03.
- **Asheley Emmy Lacerda Alves**, **PROF ENS BAS TEC TECNOLOGICO-SUBSTITUTO**, em 31/08/2023 07:57:27.
- **Paulo Ewerton Gomes Fragoso**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 31/08/2023 12:20:16.

Este documento foi emitido pelo SUAP em 30/08/2023. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código 468489
Verificador: 5bb69c3d7c
Código de Autenticação:

AGRADECIMENTOS

Agradeço a Deus, o criador dos céus e da terra, que com a sua infinita graça e amor me conduziu até aqui, "porque dele, e por ele, e para ele, são todas as coisas. . . "(Romanos11:36).

Aos meus pais, Joacir e Elizângela, que com amor, sabedoria e perseverança, cuidaram tão bem de mim, desde a infância, me incentivando aos estudos. Em especial à minha mãe, por sempre estar do meu lado e orar por mim.

À minha irmã Kallinny, que está sempre me apoiando e me animando, sendo importante na minha caminhada até aqui.

Ao professor Paulo Freitas, que com paciência e dedicação, me orientou com maestria, repassando o seu conhecimento. E a todos os professores que de maneira direta e indireta, contribuíram para eu chegar até aqui.

RESUMO

Com o avanço dos ideais democráticos, a sociedade busca maior participação nos processos públicos, o que tem impulsionado a divulgação de dados por órgãos governamentais. Nesse contexto, o presente trabalho propõe melhorias no acesso às informações do Portal de Dados do IFPB. Para isso, foi desenvolvida uma aplicação com recursos de busca e filtros mais avançados na consulta aos conjuntos de dados disponíveis, aliada à implementação de uma API GraphQL. Essas melhorias visam proporcionar uma experiência mais eficiente aos cidadãos, permitindo o acesso direto aos dados de interesse, otimizando a busca e reduzindo o tempo de consulta. Como resultados, obteve-se uma aplicação para visualização dos dados e a disponibilidade da API GraphQL, tornando o acesso aos conjuntos de dados do IFPB mais eficaz e adequado às necessidades dos usuários.

Palavras-chave: Dados Abertos. GraphQL. API. Democracia. Tecnologia.

ABSTRACT

With the advancement of democratic ideals, society seeks greater participation in public processes, which has driven the disclosure of data by government agencies. In this context, this work proposes improvements in accessing information from the IFPB Data Portal. To achieve this, an application with advanced search and filtering features was developed, coupled with the implementation of a GraphQL API. These enhancements aim to provide a more efficient experience to citizens, enabling direct access to relevant data, optimizing search capabilities, and reducing query time. As a result, a data visualization application was obtained, along with the availability of the GraphQL API, making access to IFPB datasets more effective and tailored to the users' needs.

Keywords: Open Data. GraphQL. API. Democracy. Technology.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE CÓDIGOS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
CGU	Controladoria Geral da União
CRUD	*Create, Read, Update and Delete*
CSV	*Comma-Separated Values*
GraphQL	*Graph Query Language*
HTTP	*Hypertext Transfer Protocol*
IoT	*Internet of Things*
IFPB	Instituto Federal de Educação, Ciência e Tecnologia da Paraíba
JSON	*JavaScript Object Notation*
LAI	Lei de Acesso à Informação
OGD	*Open Government Data*
OKF	*Open Knowledge Foundation*
REST	*Representational State Transfer*
TCC	Trabalho de Conclusão do Curso
TI	Tecnologia da Informação
UI	*User Interface*
URI	*Uniform Resource Identifier*

SUMÁRIO

1 INTRODUÇÃO

Com a crescente conscientização da população de diversos países nos últimos anos de seus direitos e deveres, bem como o fortalecimento dos ideais democráticos de transparência e participação cidadã nos processos governamentais, a demanda por acesso a informações públicas tem crescido.

Seguindo este processo, conforme Pedroso et al. (2013), diversas nações ao redor do mundo, incluindo o Brasil, vêm adotando medidas que permitem a população ter acesso à dados públicos da administração governamental, proporcionando maior controle social e transparência.

A disponibilização de dados públicos por meio de órgãos dos governos é chamada de dados abertos governamentais. Um dado aberto é um dado que está disponível para o acesso do público geral, sendo possível também o seu compartilhamento e reúso, respeitando a integridade da informação (LOENEN et al., 2018).

Em especial no Brasil, a lei 12.527, chamada de Lei de Acesso à Informação (LAI), sancionada em novembro de 2011, traz orientações sobre como a União, os Estados e os Municípios brasileiros devem garantir o acesso à informação pública a todos que tiverem interesse, seja individual ou coletivamente, exceto em casos de sigilo para a segurança da população e integridade do Estado (BRASIL, 2011).

Esse direito de acesso a informações que estão sob a guarda de órgãos públicos é assegurado pela Constituição Federal Brasileira (BRASIL, 2016), que em seu artigo 216 e parágrafo 2º regulamenta que: "Cabem à administração pública, na forma da lei, a gestão da documentação governamental e as providências para franquear sua consulta a quantos dela necessitem."

É por esta causa que os diversos órgãos da administração pública brasileira devem tornar os seus dados acessíveis, de maneira legível e de fácil compreensão, para fortalecer a transparência e facilitar a participação, bem como a fiscalização cívica nos processos e recursos públicos.

1.1 MOTIVAÇÃO

Para se adequar às exigências da legislação em vigor atualmente no país, em especial, a Lei de Acesso à Informação, os órgãos da administração pública brasileira vêm adotando meios de disponibilizar as informações de interesse público em canais

de comunicação.

No inciso III do artigo 3º da LAI, é determinado que devem ser utilizados todos os meios de comunicação que a Tecnologia da Informação (TI) disponibiliza no que tange a transparência das informações e dados sob posse dos órgãos públicos.

Desde o sancionamento da Lei de Acesso à Informação, diversas plataformas de acesso aos dados públicos vêm sendo implementadas, como os Portais de Transparência do Governo Federal e de outros órgãos públicos, por exemplo.

No âmbito do Instituto Federal da Paraíba (IFPB), instituição pública de educação vinculada ao Governo Federal brasileiro, foi implementado um ambiente *online* chamado de Portal de Dados Abertos do IFPB [1] para fins de transparência e divulgação de informações da autarquia para a comunidade.

A plataforma disponibiliza vários conjuntos de dados sobre a instituição em formatos de *JavaScript Object Notation* (JSON)[2] ou arquivos *Comma-Separated Values* (CSV)[3], estes últimos podendo ser acessados através de aplicativos editores de planilhas.

No Portal de Dados do IFPB, estão disponíveis dados relacionados a:

- Alunos;
- Bolsas;
- Campis;
- Cursos;
- Mapa de Atividades dos Docentes;
- Matriz de Cursos;
- Patrimônio do IFPB;
- Programas de Assistência Estudantil;
- Processos Físicos;

[1] https://dados.ifpb.edu.br/

[2] JSON: é uma estrutura de dados textual, utilizada para compartilhar dados entre aplicações, do tipo chave-valor.

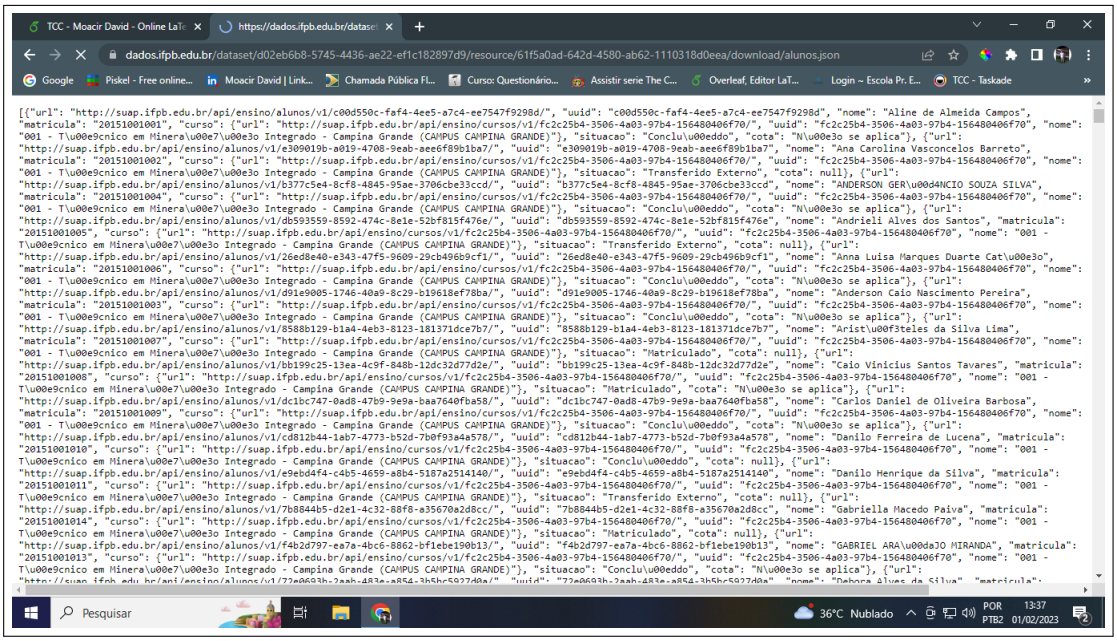
[3] CSV: um formato de arquivo utilizado para salvar dados que podem ser estruturados como uma tabela.

- Projetos de Extensão;
- Projetos de Pesquisa; e
- Versões do SUAP.

O problema apresentado pelo Portal de Dados Abertos do IFPB encontra-se no fato de que os dados que estão sendo exibidos no formato JSON no portal não contam com mecanismos de busca ou filtro, para ajudar na busca pelas informações que realmente são de interesse para o usuário no momento da consulta, podendo ser visto na Figura 1.

Outrossim, os dados que estão no formato CSV contam com mecanismos de busca e filtros, porém de forma limitada, sem permitir que o usuário interessado em acessar esses dados escolha as informações necessárias a serem exibidas. Além disso, eles estão localizados em uma área do portal que não é fácil e intuitiva de encontrar.

Figura 1 – Exemplo do retorno dos dados relacionados a alunos no formato JSON no Portal de Dados do IFPB



Fonte: Elaborado pelo autor

Ademais, embora os formatos JSON e CSV sejam indicados para disponibilização de dados públicos na internet pela facilidade de reuso e compartilhamento, o formato de exibição das informações não é de fácil visibilidade para todas as pessoas, principalmente se o usuário da plataforma não conhece a estrutura de dados JSON.

A falta de uma melhor qualidade na disponibilidade dos dados que estão no Portal de Dados do IFPB vai de encontro com o que é disposto pela LAI (BRASIL, 2011) sobre a disponibilidade das informações. Segundo a lei, a disponibilidade dos dados é a "[...] qualidade da informação que pode ser conhecida e utilizada por indivíduos, equipamentos ou sistemas autorizados"(inciso VI, artigo 4º).

Diante disso, é preciso criar um mecanismo para que os dados públicos do IFPB sejam retornados em um formato mais legível, tornando-os acessíveis universalmente, pois, como diz o Manual da Lei de Acesso à Informação para Estados e Municípios da Controladoria Geral da União (CGU), o acesso à informação é um direito intrínseco à democracia, sendo dever dos órgãos públicos facilitar o acesso aos dados para todos os cidadãos (CGU, 2013).

Este Trabalho de Conclusão de Curso (TCC) propõe a implementação de uma *Application Programming Interface* (API)[4] utilizando GraphQL sobre os dados disponibilizados pelo IFPB e uma ferramenta complementar que permita uma melhor visualização das informações do Portal de Dados do IFPB, com recursos de busca e filtro nos dados mais avançados.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

O objetivo geral deste Trabalho de Conclusão de Curso é facilitar o acesso e busca aos dados disponibilizados pelo Portal de Dados Abertos do IFPB.

1.2.2 Objetivos Específicos

Os objetivos específicos deste trabalho são:

- Disponibilizar uma API utilizando a tecnologia GraphQL sobre os dados disponíveis no Portal de Dados do IFPB;
- Permitir a consulta dos dados utilizando filtros conforme o desejo do usuário;
- Construir uma interface gráfica para que os dados sejam exibidos de maneira a facilitar a leitura e interpretação das informações;
- Implementar um mecanismo de busca para que os dados sejam consultados com mais rapidez.

[4] API: conjunto de regras que possibilita a comunicação entre sistemas e aplicações diferentes.

1.3 TRABALHOS RELACIONADOS

A proposta deste Trabalho de Conclusão de Curso (TCC) está alinhada a outros estudos acadêmicos, como o trabalho realizado por Junior (2022). O autor destaca deficiências na disponibilidade e qualidade dos dados públicos no Brasil e no estado do Ceará, incluindo a forma de disponibilização, qualidade da informação, mecanismos de busca e integração dos dados. As atividades realizadas pelo autor foram a coleta de dados relevantes, como vacinação a nível nacional durante a pandemia de coronavírus e segurança pública do Ceará, e utilização de técnicas de mineração de dados para extrai-los e, depois, carregar o banco de dados. Em seguida, o autor gerou uma API GraphQL e REST para análise desses dados, visando a identificação de soluções para os problemas encontrados.

Um dos resultados foi a disponibilização de uma API unificada para os dados do Governo Federal e do Estado do Ceará, proporcionando acesso integrado e centralizado às informações para todos os cidadãos. De forma similar ao trabalho de Junior (2022), este TCC também prioriza a disponibilidade e qualidade dos dados públicos, implementando uma API GraphQL no Portal de Dados do IFPB e criando uma ferramenta que facilite a visualização das informações, com recursos avançados de busca e filtros. Essas melhorias buscam promover maior participação cívica e conhecimento dos dados públicos pela população brasileira.

Já no trabalho de Silva et al. (2020), foi realizada uma análise do Portal Brasileiro de Dados Abertos com o objetivo de atestar se o ambiente de divulgação dos dados públicos da administração pública federal brasileira no ano de 2018 estavam atendendo aos critérios de publicação de dados abertos governamentais, segundo as principais organizações do movimento de dados abertos governamentais, como a *Open Government Data* (OGD).

Pode-se citar alguns exemplos dos critérios utilizados para analisar o portal que no trabalho foi citado, sendo alguns deles a disponibilidade dos dados, acessibilidade, atualidade, credibilidade e velocidade de acesso às informações.

Os autores chegaram à conclusão que o repositório de dados abertos do governo brasileiro atende a todos os princípios de qualidade na publicação de dados abertos governamentais, embora apresente alguns problemas que merecem atenção, como a falta de atualizações constantes nos dados (atualidade), relevância dos conjuntos de dados (alguns conjuntos de dados importantes faltando, como os relacionados a programas sociais) e a velocidade de acesso às informações, considerada satisfatória.

A velocidade de acesso aos dados é algo discutido na proposta deste TCC, que utilizará GraphQL como alternativa ao padrão arquitetural REST na implementação de uma API sobre o Portal de Dados do IFPB, visando mais agilidade no acesso às informações públicas da instituição acadêmica federal, bem como reduzir o consumo de largura de banda na parte dos usuários.

1.4 METODOLOGIA

Para o desenvolvimento deste trabalho, foram realizadas as seguintes atividades:

- Estudo da tecnologia GraphQL, através da documentação oficial e vídeos;
- Implementação dos códigos necessários para a criação da API com GraphQL sobre os dados disponibilizados pelo IFPB;
- Levantamento teórico para justificar a implementação da proposta deste trabalho;
- Escrita do documento contendo as informações relevantes para o trabalho, como levantamento teórico, metodologia, protótipos e outras;
- Criação dos protótipos da ferramenta utilizando a ferramenta Figma;
- Implementação da ferramenta de visualização gráfica dos dados.

1.5 ESTRUTURA E ORGANIZAÇÃO DO DOCUMENTO

Este documento está dividido da seguinte forma:

No capítulo 2, a fundamentação teórica utilizada neste trabalho é exposta, abordando os conceitos e tecnologias utilizadas para se chegar ao objetivo deste TCC. No capítulo 3 os resultados alcançados são expostos. Por fim, o capítulo 4 contém as consideração finais e sugestões de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, serão abordados os conceitos discutidos neste trabalho e a tecnologia utilizada. Primeiramente, será discorrido sobre a importância dos Dados Abertos. Segundo, será apresentado a definição da arquitetura *Representational State Transfer* (REST) e suas características, e, logo em seguida, o GraphQL e seus recursos. Por fim, na última seção deste capítulo, o processo de desenvolvimento da aplicação proposta é apresentado, juntamente com as adaptações que foram necessárias para a realização do objetivo proposto.

2.1 DADOS ABERTOS

Segundo Isotani e Bittencourt (2015), um dado aberto é um dado que pode ser utilizado ou reutilizado e compartilhado por qualquer pessoa, respeitando somente a integridade das informações, bem como a citação da fonte e o mantimento da licença deste dado.

Como enfatiza Segundo (2015), a demanda por acesso a informações tem crescido no século XXI, fazendo os órgãos governamentais e até mesmo as empresas privadas buscarem meios de estruturar, dar sentido e disponibilizar os seus dados, visando a participação cidadã na tomada de decisões e maior transparência.

Diversas nações estão se adaptando à nova realidade de disponibilizar os dados de interesse público na *internet*, respondendo ao anseio da sociedade por transparência e maior controle social.

Esse movimento pela busca de maior transparência e disponibilização das informações dos governos, empresas privadas, universidades e outros órgãos de interesse público é inspirado pelos ideais da *Open Knowledge Foundation* (OKF), instituição criada em 2004, que dá suporte ao movimento de dados abertos ao redor do mundo.

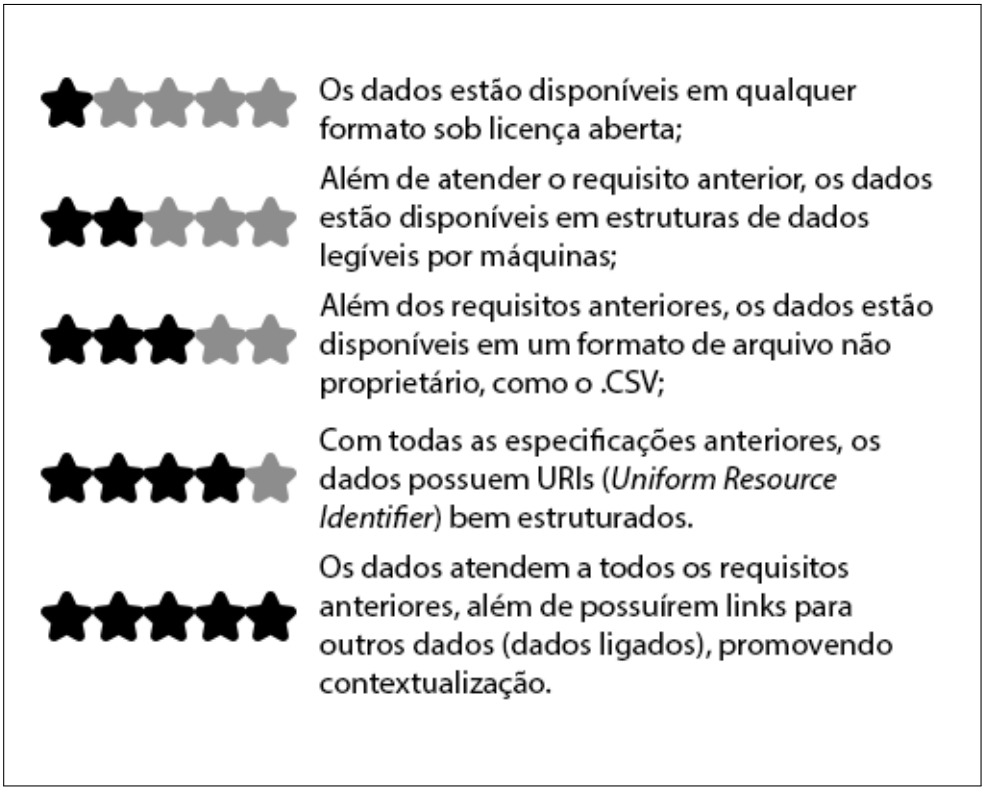
Segundo a Open Knowledge Foundation (2019), quatro etapas devem ser seguidas para a publicação de dados abertos: 1) escolha do conjunto de dados a ser publicado; 2) aplicação de licenças abertas para o conjunto de dados a ser disponibilizado; 3) disponibilização dos dados, em formato simples e passível de ser reutilizado, se possível através de alguma API; e 4) facilidade na descoberta destes dados, através da publicação na internet e meios de chegar facilmente aos mesmos.

Tendo em vista estes princípios, não basta somente deixar os dados publicados em um site. É preciso torná-los acessíveis, legíveis e permitir que qualquer pessoa, sem acepção qualquer, acesse, entenda, utilize, reutilize e compartilhe estes dados, não se restringindo somente a humanos, mas também ao processamento das informações por máquinas.

Por isso, Tim Berners-Lee, criador da Web, idealizou o conceito de *Linked Data*[1], em que os dados são ligados semanticamente, por um sistema denominado de "Cinco Estrelas dos Dados Abertos", em que quanto maior o nível de estrelas (de 1 a 5), melhor é a qualidade da publicação e disponibilidade dos dados.

Este sistema é organizado da seguinte forma, apresentado na Figura 2:

Figura 2 – Classificação de um dado aberto proposto por Tim Berners-Lee



Fonte: Adaptado de W3C (2011)

Seguindo o método de classificação dos dados abertos proposto por Tim Berners-Lee, os conjuntos de dados disponibilizados pelo IFPB em seu portal está no nível 5 estrelas, pois, atende a todos os requisitos propostos pelo sistema de classificação de qualidade dos dados abertos, sendo estes:

[1] https://www.w3.org/DesignIssues/LinkedData.html

- Uso de licença aberta na publicação dos dados - este requisito é atendido pelo Portal de Dados Abertos do IFPB;
- Publicação dos dados em formatos de arquivo não proprietários e legíveis por máquina - o Portal de Dados do IFPB disponibiliza os dados nos formatos JSON e CSV;
- Os dados possuem URIs[2] bem definidas e estruturadas; e
- Os conjuntos de dados no Portal de Dados do IFPB possuem *links* para outros dados relacionados, além de estarem contextualizados com metadados.

Embora o Portal de Dados do IFPB esteja no nível mais alto de classificação dos dados abertos pelo sistema proposto por Tim Berners-Lee, a ideia deste Trabalho de Conclusão de Curso visa melhorar ainda mais a qualidade do ambiente de divulgação dos dados públicos da instituição, contribuindo com um mecanismo de busca, sistema de filtros nas pesquisas e uma interface gráfica que deixe os dados mais legíveis.

Ademais, mesmo sendo uma tecnologia emergente, a utilização do GraphQL na implementação da API permite uma maior gama de possibilidade no uso dos dados, como a busca pelos dados exatos conforme a necessidade no momento da pesquisa, além de possuir um ecossistema em constante evolução e demais especificidades que atendem ao objetivo deste trabalho.

2.2 ARQUITETURA REST

Como diz Oliveira (2015), o padrão de arquitetura *Representational State Transfer* (REST) foi criado por Roy Fielding, que também foi um dos criadores do protocolo *Hypertext Transfer Protocol* (HTTP)[3] nos anos 2000.

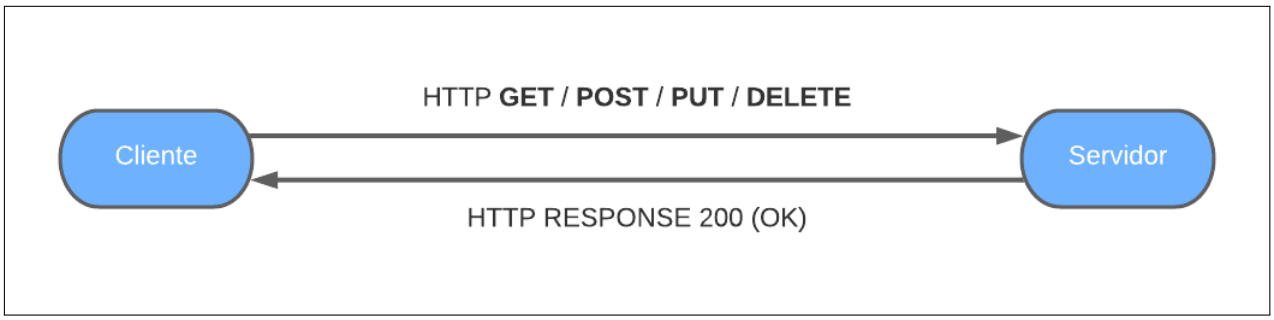
Essa arquitetura define um conjunto de padrões que devem ser observados pelos programadores para desenvolverem APIs, padronizando a maneira de comunicação entre os diversos sistemas e dispositivos que trafegam dados entre si. O padrão REST é o mais utilizado atualmente ao redor do mundo para construção de APIs (BITTENCOURT, 2021).

França e Silva (2020) descreve a arquitetura REST como uma solução para construção de APIs no modelo cliente-servidor, que define bem a maneira como os

[2] URI: uma sequência de caracteres utilizada para identificar um recurso, seja na internet ou não.
[3] HTTP: protocolo de comunicação utilizado na internet, em que os clientes solicitam dados dos servidores, que respondem a estes conforme o solicitado.

dados são trafegados pela rede e onde estão localizados, através de *Uniform Resource Indentifiers* (URIs), bem como a utilização de métodos HTTP para a realização de requisições.

Figura 3 – Diagrama de uma API REST



Fonte: Elaborado pelo autor

Como visto na Figura 3, um cliente (que pode ser um dispositivo móvel ou IoT[4], computador ou até mesmo outra aplicação ou sistema) faz requisições ao servidor, que podem ser do tipo *GET*, leitura de dados; *POST*, inserção de dados; *PUT*, atualização; e *DELETE*, exclusão de alguma informação do servidor. O processamento ocorre no lado do servidor, que retorna uma resposta para o cliente conforme a operação que foi realizada.

O padrão arquitetônico REST apresenta benefícios como alta escalabilidade e portabilidade, fruto das características deste padrão descritas a seguir, como enfatizado, por (GONÇALVES, 2022):

- Ausência de Estado: cada requisição feita em uma API REST é única, não sendo armazenadas informações sobre o cliente, nem as informações sobre as sessões que são encerradas;
- Cache: a arquitetura REST define que um recurso deve ser armazenado em *cache* caso seja requisitado múltiplas vezes, evitando processamentos desnecessários;
- Cliente-Servidor: com base nas premissas do protocolo HTTP, a arquitetura REST implementa o conceito de cliente-servidor, em que o servidor aguarda por requisições (*requests*) feitas pelo cliente e em seguida concede respostas (*responses*) de acordo com o pedido;

[4] IoT: https://www.oracle.com/br/internet-of-things/what-is-iot/

- Interface uniforme: todos os recursos (*resources*) disponibilizados em uma API REST devem ter uma interface uniforme, o que corresponde a terem URIs bem definidas e mensagens autodescritivas para ajudar na compreensão dos dados; e
- Sistema em camadas: a API é construída em camadas, tornando a aplicação mais modular e escalável.

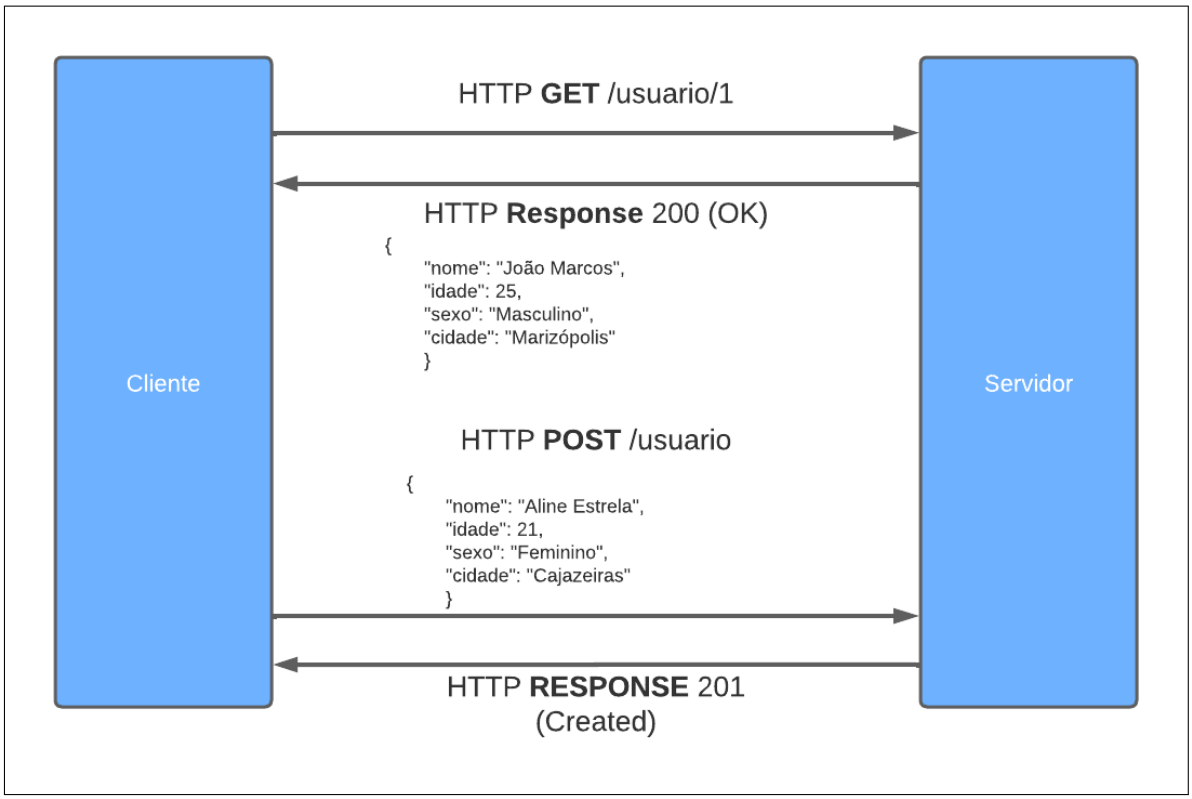
Apesar dos inúmeros benefícios apresentados, algumas características do padrão REST podem apresentar desvantagens dependendo do contexto e das regras de negócio que uma aplicação possui.

Conforme Neto (2020), a arquitetura REST apresenta as seguintes características: a) *overfetching*, ou, sobrecarga de dados desnecessários na rede; b) *underfetching*, sendo o inverso do *overfetching*, resultando em outra particularidade do padrão: c) a necessidade de criação de vários *endpoints* para realizar requisições sobre um conjunto de informações.

O funcionamento de uma API REST pode ser visto na Figura 4. Conforme representado, para acessar os dados do usuário com o *id* 1, deve-se fazer uma requisição no *endpoint*[5] "usuario", passando o número de identificação como parâmetro. A resposta retornará todas as informações sobre o usuário, sem a possibilidade de escolher quais campos devem ser retornados na resposta.

[5] Endpoint: é um ponto de acesso específico de uma aplicação Web, geralmente uma URL, em que o cliente pode realizar requisições e obter respostas específicas.

Figura 4 – Consulta de dados em uma API REST utilizando mais de um endpoint



Fonte: Elaborado pelo autor

Caso também seja necessário solicitar outros dados e realizar outras operações, como atualização de informações, criação e deleção de dados, outros *endpoints* devem ser acessados.

Essas características mencionadas podem resultar em menor desempenho da aplicação e maior consumo de largura de banda, já que, para cada consulta realizada na API, é necessário acessar vários *endpoints*, e todos os dados sobre uma informação vêm na resposta, mesmo que não sejam essenciais. Isso pode ser uma desvantagem em relação ao acesso aos dados do Portal de Dados do IFPB, que contém conjuntos de dados de grande tamanho.

2.3 GRAPHQL

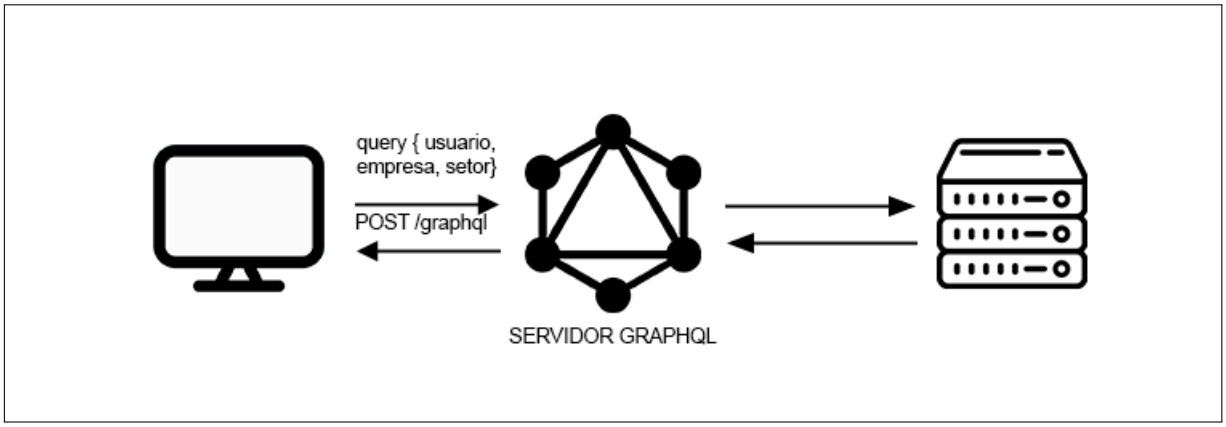
O GraphQL (*Graph Query Language*) é uma linguagem de consulta, que também permite a especificação de dados para APIs independentemente da linguagem de programação, criado em 2012 pelo Facebook e lançada ao público em 2015, como uma alternativa a arquitetura REST. O objetivo da equipe da empresa era resolver os problemas de *over fetching*, ou seja, a sobrecarga de dados desnecessários na rede

interna da organização, oriundas das requisições de APIs REST (BRITO; VALENTE, 2020).

Conforme Tavares e Fernandes (2022), o GraphQL permite uma descrição exata dos dados a serem requisitados, solucionando assim o problema de respostas (*responses*) do servidor com dados que não são necessários à consulta, reduzindo o consumo de largura de banda.

A Figura 5 descreve como funciona uma API implementada com GraphQL. Como pode ser visto na figura, um cliente requisita ao servidor vários dados diferentes (no exemplo da figura, informações sobre usuários, setores e empresas) em um único *endpoint*, bastando apenas uma requisição. O servidor retorna os dados no formato JSON.

Figura 5 – Esquema de uma API GraphQL



Fonte: Elaborado pelo autor

A possibilidade de realizar várias operações básicas de CRUD (criação, leitura, atualização e deleção dos dados) em uma única requisição e *endpoint*, utilizando o método POST do protocolo HTTP, do GraphQL, torna-se um fator positivo para a melhoria de desempenho da aplicação (QUIÑA-MERA et al., 2022).

Essas diferenças entre uma API GraphQL e REST são possíveis devido ao fato que a linguagem de consulta GraphQL possui uma sintaxe própria, que permite a descrição exata dos dados, através da declaração de campos e valores, exemplificado no Algoritmo 1.

Algoritmo 1 – Exemplo da estrutura de dado na sintaxe GraphQL que descreve um aluno

```
1     type Aluno {
2         nome: String,
3         matricula: String,
4         idade: Int
5     }
```

Fonte: Elaborado pelo autor

A palavra-chave *type* no GraphQL permite definir tipos personalizados de objetos. No algoritmo acima, o tipo "aluno"é definido, com seus campos e valores de retorno, conforme especificados na estrutura (*schema*) da API.

Além disso, o GraphQL possui mecanismos de buscas que permitem consultas personalizadas conforme a necessidade do usuário. A *query* é um tipo de estrutura de dado especial do GraphQL utilizado para realizar consultas (ver Algoritmo 2), que são resolvidas por funções específicas chamadas *resolvers*, responsáveis por buscar a fonte dos dados (ver Algoritmo 3).

Algoritmo 2 – Exemplo da sintaxe de descrição de uma consulta disponível em uma API GraphQL

```
1     type Query {
2         alunos: [Aluno],
3         alunoPorMatricula(matricula: String): Aluno,
4     }
```

Fonte: Elaborado pelo autor

Dentro do objeto *Query*, duas consultas foram definidas dentro da API GraphQL. Estas receberam um nome, "alunos"e "alunosPorMatricula", respectivamente, em que a segunda consulta recebe um código de matrícula por parâmetro. Ambas as consultas retornam dados do tipo Aluno, como foi implementado no Algoritmo 1.

Por fim, as funções que são responsáveis por tratar as consultas e buscar os dados (*resolvers*) são definidas. Utilizando *JavaScript*[6], dentro de um objeto chamado Query, foram definidas funções com os mesmos nomes das consultas implementadas no Algoritmo 2. Essas funções assíncronas retornam os dados de um banco de dados, resolvendo a consulta.

Esses recursos apresentados pelo GraphQL atendem aos propósitos deste TCC, que tem como objetivo permitir que o usuário consulte somente os dados necessários no momento da pesquisa, além de diminuir o consumo de largura de banda, pois

[6] https://developer.mozilla.org/pt-BR/docs/Web/JavaScript

Algoritmo 3 – Exemplo de resolvers para atender as consultas disponíveis em uma API GraphQL com JavaScript

```
module.exports = {
    Query: {
        alunos: async () => {
            return await Aluno.find({});
        },
        alunoPorMatricula: async (_, args) => {
             return await Aluno.findOne({matricula: args.
                matricula});
        }
    }
}
```

Fonte: Elaborado pelo autor

permitir a especificação exata de quais dados retornar da API faz com que comportamentos como o *overfetching* e *underfetching* não ocorram.

2.4 PROCESSO DE DESENVOLVIMENTO

Um *software*, segundo Pressman (2011), pode ser definido como um conjunto de instruções de computador que processa e transforma informações. O desenvolvimento de um *software* pode seguir processos formais definidos na Engenharia de Software, conforme a especificidade de um projeto.

Para Pressman (2011), um processo pode ser entendido como um roteiro a ser seguido para o desenvolvimento de um projeto, com o objetivo de entregar um *software* com qualidade e dentro do prazo, selecionando as atividades, ações e tarefas específicas para tal.

Ainda conforme o autor, um modelo genérico de processo de desenvolvimento de *software* segue as seguintes etapas:

- Comunicação: fase de compreensão das necessidades das partes interessadas (cliente);
- Planejamento: etapa de definição das técnicas que serão utilizadas para desenvolver o projeto;
- Modelagem: durante esta etapa, é feito um esboço do projeto para melhor entendimento das necessidades levantadas pelas partes interessadas;

- Construção: é a fase da própria implementação do projeto em si, da criação dos códigos e testes necessários;
- Emprego: o *software* é entregue ao cliente, que retorna uma avaliação para a equipe de desenvolvimento.

Dentre os processos de desenvolvimento de *softwares*, existem vários modelos, variando de processos tradicionais (menos flexíveis), iterativos e mais flexíveis, até os processos que utilizam os conceitos e filosofias dos modelos ágeis.

Na categoria dos processos tradicionais, pode-se citar o modelo de desenvolvimento em cascata, por exemplo. Neste processo, todo o projeto é executado de maneira sequencial e linear, não começando outra etapa até que a atual seja concluída. Embora este processo apresente um modelo bem definido e estruturado para desenvolver um *software*, ele não apresenta a característica de ser flexível e fácil de adaptar-se quando há requisitos voláteis (VALENTE, 2020).

Já os processos iterativos são mais flexíveis, fáceis de adaptar quando há novos requisitos e são divididos em ciclos curtos de desenvolvimento, entregando partes do *software* em construção para recolher o *feedback* dos usuários. Um exemplo de processo iterativo é o modelo espiral, que entrega artefatos do *software* buscando sempre a evolução do mesmo (PRESSMAN, 2011).

Por fim, os processos baseados nos modelos ágeis têm suas premissas baseadas nos fundamentos do Manifesto Ágil[7], emergidas por pessoas que entenderam o processo de desenvolvimento de *software* com os seguintes princípios, resumidamente (OLIVEIRA; PEDRON, 2021):

- Colaboração: tanto os desenvolvedores como os clientes (partes interessadas) devem colaborar uns com os outros durante o projeto;
- Flexibilidade: novos requisitos são bem-vindos e, com adaptação ao projeto, são incluídos no processo;
- Melhoria contínua: a equipe de desenvolvimento do projeto faz inspeções regularmente visando corrigir erros e melhorar pontos do processo de desenvolvimento;
- Satisfação do cliente: a entrega contínua de artefatos e partes do *software* prontos ao cliente é feita visando a satisfação do mesmo.

[7] https://agilemanifesto.org/iso/ptbr/manifesto.html

Embora não tenha sido utilizado de maneira formal algum processo de desenvolvimento de *software* na implementação da ideia desse TCC, de maneira adaptada, os conceitos e ideias da metodologia ágil Scrum foram seguidos.

2.4.1 SCRUM

O Scrum é um *framework* ágil que ajuda empresas, equipes e indivíduos no desenvolvimento de projetos, com foco na agregação de valor e entrega periódica de artefatos funcionais, além do compromisso com as pessoas. (SCHWABER; SUTHERLAND, 2020)

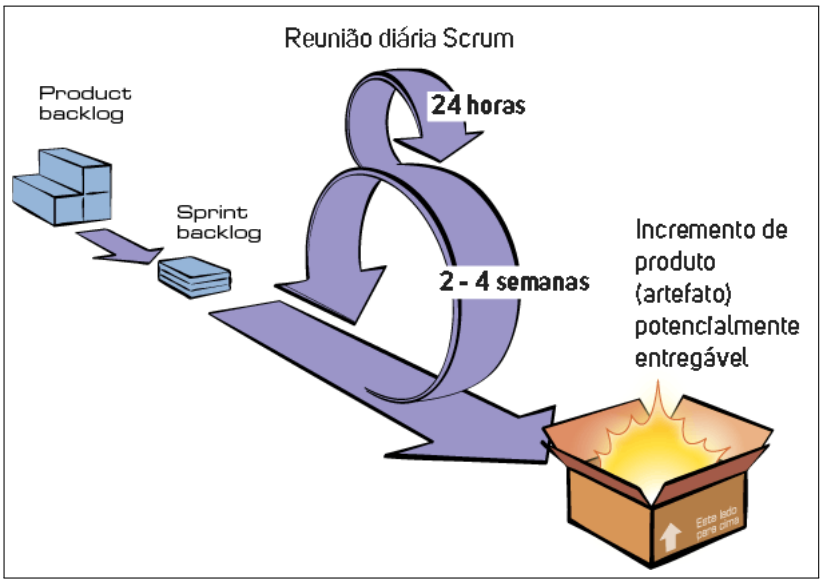
Conforme os autores, o Scrum tem como pilares a transparência: todos envolvidos no projeto, tanto quem executa, quanto quem recebe o produto final, devem conhecer sobre todos os processos de um projeto, para ajudar na tomada de decisões; a inspeção, que é a análise periódica do processo de desenvolvimento do projeto, para decidir o que deve ser mantido e o que pode melhorar ou ser descontinuado; e, por último, a adaptação, que permite a inserção de novos elementos no projeto ou a correção de algo que esteja incorreto, conforme as necessidades dos *stakeholders* e surgimento de melhores práticas no desenvolvimento do projeto.

O gerenciamento de projetos com o Scrum é dividido em etapas, chamadas de *sprints*, que têm duração máxima de um mês. As *sprints* servem para a implementação de novas funcionalidades, inspeção do que já está em execução, correções e atividades cujo objetivo é atingir a meta definida para aquele ciclo (BASTOS; BASTOS, 2021; SCHWABER; SUTHERLAND, 2020).

As atividades realizadas durante uma *sprint* seguem o que foi decidido pela equipe em uma reunião de planejamento, constituindo o que é chamado de *product backlog*, ou, lista do que deve ser feito para alcançar os objetivos. A prioridade dos itens dessa lista são decididas em equipe (BASTOS; BASTOS, 2021).

No final de cada *sprint*, há a entrega de um artefato funcional e outro ciclo se inicia, de maneira iterativa. O ciclo de uma *sprint* pode ser melhor visualizado na Figura 6.

Figura 6 – Ciclo de uma Sprint no framework Scrum



Fonte: A daptado de Metodologia Ágil (s.d.)

Para que as etapas de um projeto gerenciado pelo Scrum ocorra de maneira a atender as expectativas do cliente e haja eficiência durante o desenvolvimento, há uma separação clara de papéis na equipe Scrum, embora não exista hierarquia (DANTAS, 2021), podendo ser visto no Quadro 1.

Quadro 1 – Funções dentro de uma equipe Scrum

Função	**Descrição**
Product Owner	Pessoa proprietária do produto. É responsável por comunicar, de maneira transparente, os requisitos do produto. Pode solicitar novos itens e alterações, além de aprovar ou reprovar o lançamento de uma *sprint*.
Scrum Master	Pessoa responsável pela equipe Scrum. Lidera o grupo, coordena as atividades, promove ajuda aos membros da equipe para que alcancem o objetivo do projeto, além de buscar a eficiência e produtividade.
Developers	Pessoas desenvolvedoras dentro da equipe Scrum, responsáveis por produzir artefatos que agregam valor ao projeto, além de resolver as pendências do *Product Backlog*.

Fonte: Adaptado de Schwaber e Sutherland (2020).

No contexto deste Trabalho de Conclusão de Curso, o processo de desenvolvimento da ideia proposta seguiu, de maneira adaptada, as orientações do Scrum. Os papéis de *Product Owner* e *Scrum Master* foram desempenhados pelo professor

orientador deste trabalho, que orientou a ideia do projeto, fez requisitos que compôs o *Product Backlog* e aprovou e ou desaprovou os artefatos desenvolvidos.

Por sua vez, o papel de *Developer* foi de responsabilidade do orientando deste TCC. As *sprints* foram adaptadas, ocorrendo a cada semana, com a entrega dos requisitos levantados pelo professor orientador, listadas na ferramenta gerenciamento de atividades e fluxo de trabalho, Taskade[8].

[8] https://www.taskade.com/

3 API GRAPHQL E FERRAMENTA DE VISUALIZAÇÃO GRÁFICA

Neste capítulo, os resultados da implementação do projeto proposto neste Trabalho de Conclusão de Curso serão apresentados. A primeira seção aborda questões relacionadas à etapa de análise, descrevendo os *stakeholders*, requisitos funcionais, casos de uso e protótipos. Na segunda, a arquitetura da aplicação é descrita. Em seguida, o processo de importação dos dados para o uso na ferramenta é exibido. Na quarta seção, a API GraphQL implementada é exposta, findando, na última seção, com a exibição das telas do sistema.

3.1 ANÁLISE

3.1.1 STAKEHOLDERS

Em um projeto, os *stakeholders* são todas as pessoas ou empresas envolvidas, que estão sendo ou serão beneficiadas ou afetadas de diversas maneiras com o resultado do projeto (CAMARGO; VALLE, 2022).

A API e a ferramenta de visualização gráfica implementada neste trabalho têm como público-alvo a sociedade brasileira em geral, pois, os dados públicos oriundos do IFPB podem ser acessados por todos que almejarem ter mais conhecimento sobre a gestão dos recursos públicos e exercer sua função nos processos cívicos democráticos.

3.1.2 REQUISITOS FUNCIONAIS

Segundo Machado (2018), os requisitos de um *software* podem ser descritos como as características ou exigências por parte dos clientes e usuários que um sistema deve possuir para a resolução de um problema, bem como chegar aos resultados esperados pelos *stakeholders*.

Por sua vez, os requisitos funcionais estão relacionados às funcionalidades que um sistema ou aplicativo deve ter, como, por exemplo, a possibilidade de realizar autenticação, o processamento de dados em uma aplicação, dentre outras ações.

A ferramenta proposta neste trabalho deve atender aos seguintes requisitos funcionais:

- **RF01 - Selecionar Conjunto de Dados**: O sistema deve permitir a seleção de um Conjunto de Dados específico, conforme o interesse do usuário.

- **RF02 - Escolher filtros**: O sistema deve permitir pesquisas por dados utilizando filtros.
- **RF03 - Escolher campos**: O sistema deve permitir a consulta somente de campos necessários dos dados retornados pela API conforme o desejo do usuário.
- **RF04 - Realizar busca**: O sistema deve permitir a pesquisa por dados específicos em um conjunto de dados utilizando um motor de buscas.
- **RF05 - Redefinir consulta**: O sistema deve permitir a redefinição das consultas, retornando ao estado inicial após clicar em um botão.
- **RF06 - Buscar Conjuntos de Dados**: O sistema deve permitir a busca por Conjuntos de Dados específicos, utilizando termos-chave e palavras.

3.1.3 CASO DE USO

Como descreve Cockburn (2005), um caso de uso é uma descrição de como um sistema reage, em diversas circunstâncias, à interação de um usuário, chamado de ator primário, que inicia o processo de interação visando atingir um objetivo.

No caso da aplicação proposta neste TCC, que consiste em uma ferramenta de visualização gráfica para exibir os dados consumidos por uma API GraphQL, existem atividades relacionadas à escolha do conjunto de dados que será consultado e aos possíveis campos dos dados e filtros a serem utilizados, além da realização de buscas por dados específicos.

Um dos casos de uso da ferramenta proposta neste trabalho representa a ação de selecionar um conjunto de dados pelo usuário e pode ser visto no Quadro 2. Os demais casos de uso estão no Apêndice A.

Quadro 2 – Caso de uso: Selecionar Conjunto de Dados

Nome do Caso de Uso	**Selecionar Conjunto de Dados**
Ator principal	Usuário
Atores secundários	Nenhum
Resumo	Neste caso de uso, as etapas seguidas por um usuário para selecionar um Conjunto de Dados e, posteriormente, consultar os dados, são descritas.
Pré-condições	O usuário está na página principal da aplicação.
Pós-condições	O Conjunto de Dados selecionado é exibido na página da aplicação.
Ações do Ator	Ações do Sistema
1. O usuário acessa a página principal da aplicação.	2. O sistema exibe as categorias de Conjunto de Dados existentes.
3. O usuário seleciona uma categoria de conjunto de dados.	4. O sistema exibe os conjuntos de dados pertencentes à categoria que o usuário escolheu.
5. O usuário escolhe o conjunto de dados que deseja acessar.	6. O sistema exibe o conjunto de dados selecionado em uma página.

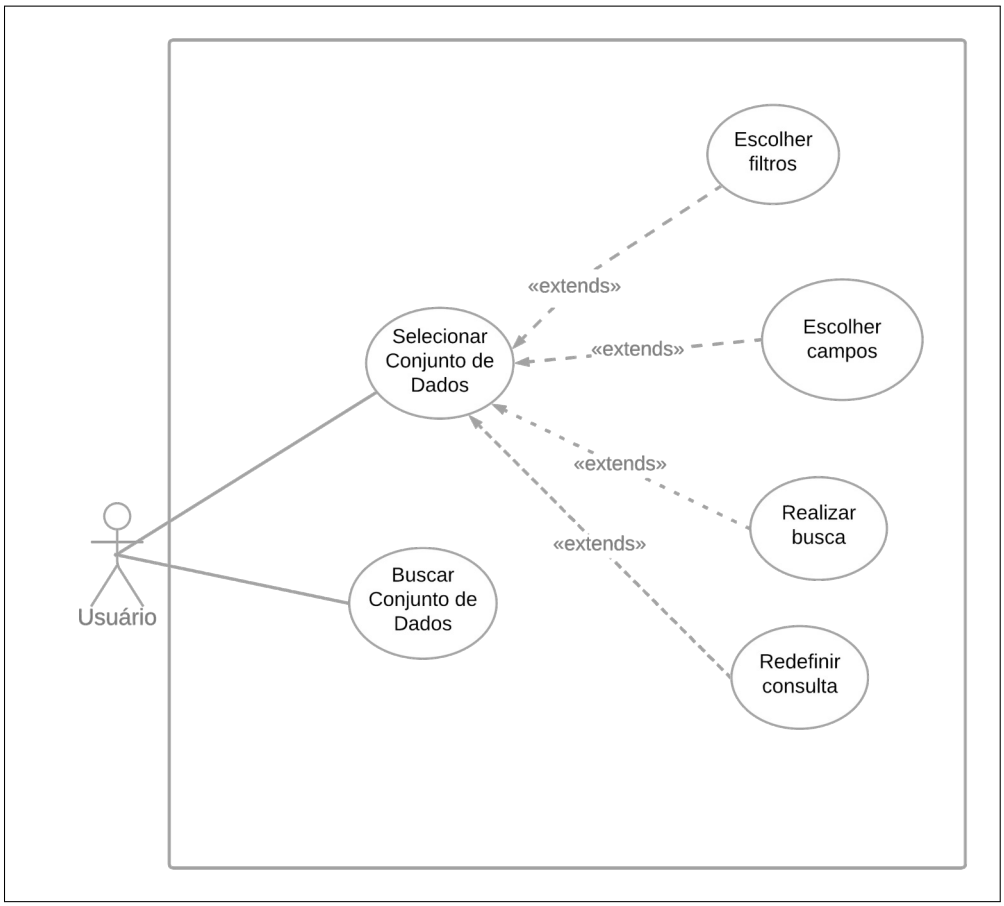
Fonte: Elaborado pelo autor.

Já um diagrama de caso de uso é uma ferramenta visual que pode ser utilizada para mostrar, em detalhes gerais, as principais funcionalidades de um sistema e quem executará as mesmas. Os executores das tarefas são denominados de atores (Lucidchart, c2023).

Na Figura 7, é possível ver o diagrama de casos de uso da aplicação proposta neste trabalho. Como demonstrado na figura do diagrama de casos de uso, o usuário, que é o ator principal, seleciona um conjunto de dados no qual deseja acessar as informações. Após isso, como ações que podem ocorrer ou não, ele pode escolher aplicar filtros na consulta, escolher os campos que devem ser retornados, realizar buscas específicas por dados e redefinir a consulta. Em seguida, a aplicação retornará os dados conforme a consulta que o usuário deseja realizar no momento.

Da mesma forma, o usuário pode realizar buscas por conjuntos de dados, que retornará os conjuntos existentes que correspondem aos termos utilizados na busca.

Figura 7 – Diagrama de casos de uso da aplicação



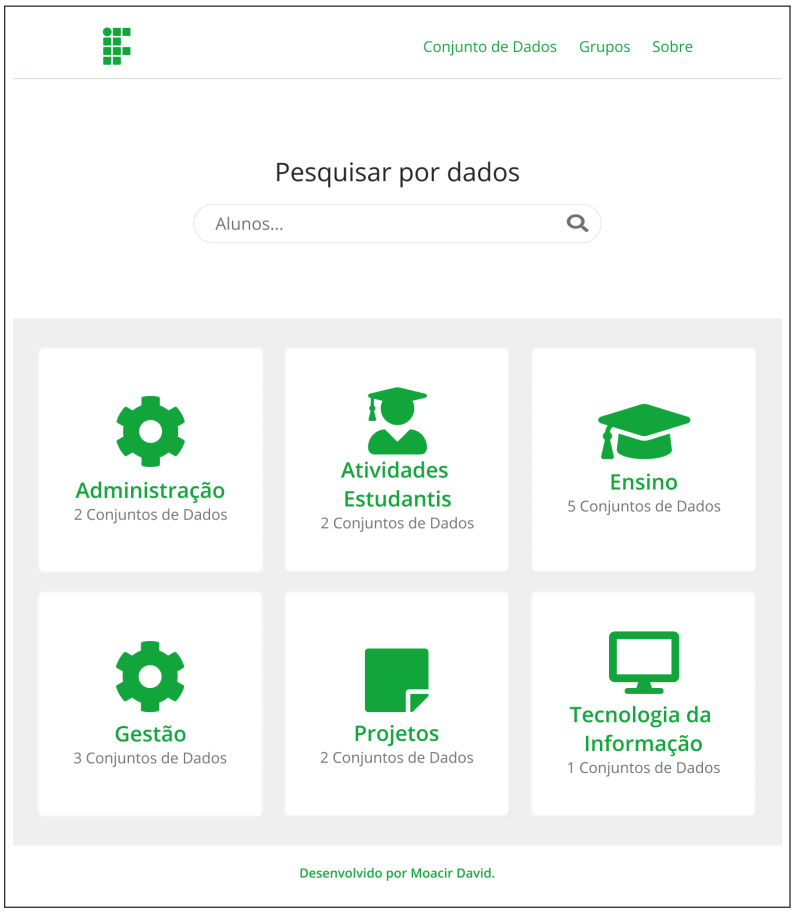
Fonte: Elaborado pelo autor

3.1.4 Protótipos

Um protótipo é uma espécie de rascunho de um sistema a ser desenvolvido, que permite a exibição do planejamento de como será a Interface de Usuário, as funcionalidades e outros recursos de um projeto, para validação com o cliente (FARIAS, 2022).

O protótipo da tela inicial da ferramenta Web pode ser visto na Figura 8.

Figura 8 – Protótipo da tela inicial da ferramenta



Fonte: Elaborado pelo autor

Na tela inicial, o usuário verá uma área que contém um *input* para realizar pesquisas por Conjunto de Dados. Também, uma área com *cards* que representam o agrupamento de Conjunto de Dados por categoria.

Os protótipos das demais telas podem ser vistos na ferramenta Figma[1], disponibilizados neste link: <https://www.figma.com/community/file/1264321776226660624>.

3.2 ARQUITETURA DA APLICAÇÃO

O projeto trata-se de uma simples aplicação para visualização de dados consumidos por uma API GraphQL, construída com tecnologias atuais bastante utilizadas no Desenvolvimento Web, como a biblioteca para a criação de interfaces de usuário (UI) em JavaScript, chamada *ReactJS*[2], na ferramenta de visualização dos dados em uma página Web.

[1] https://www.figma.com/
[2] https://react.dev/

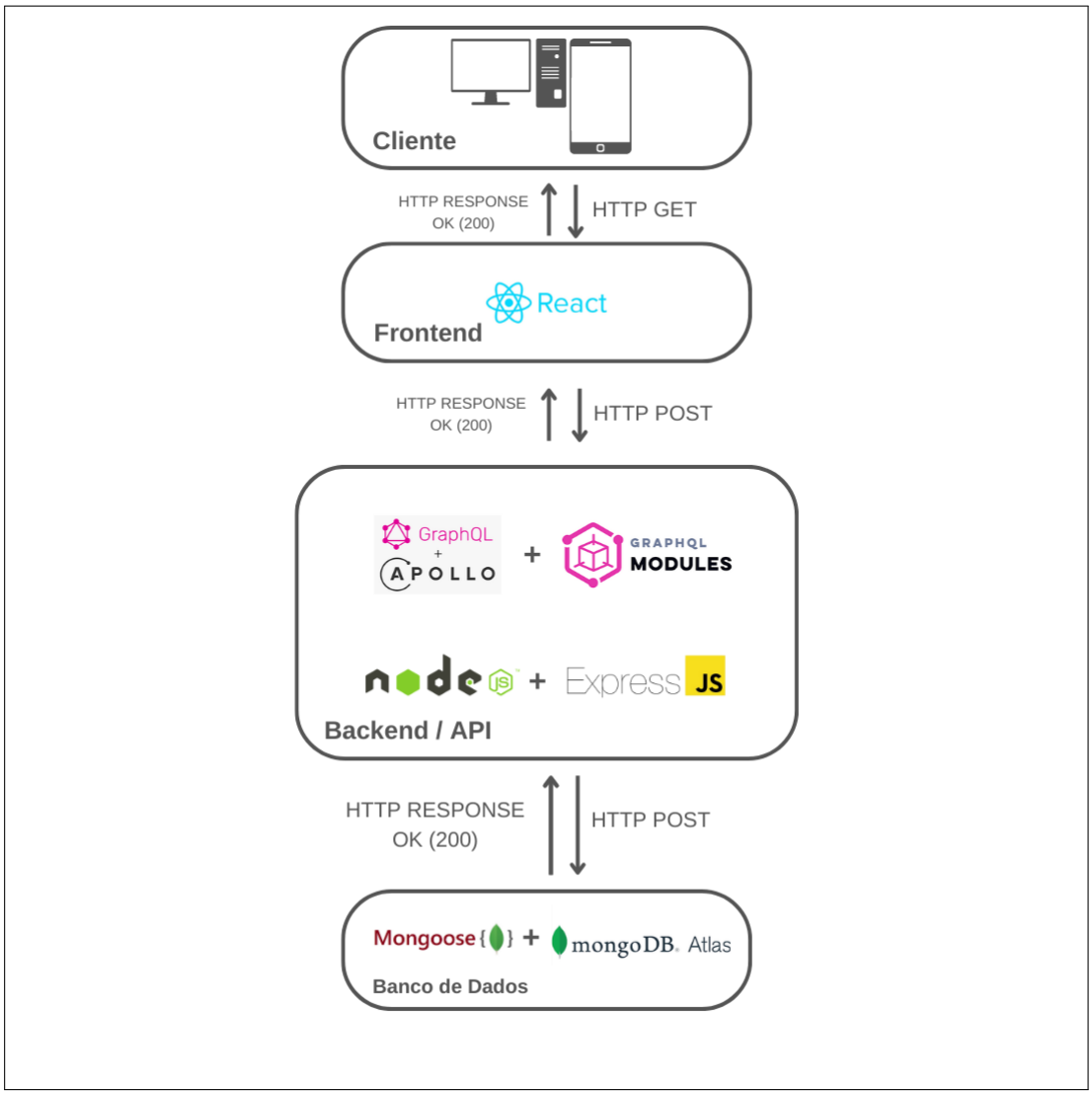
Para o servidor, foi utilizado o *Node.js*[3], um ambiente de execução de JavaScript do lado do servidor, juntamente com o *Express*[4], um *framework* utilizado para criar servidores web com JavaScript. A API implementada foi desenvolvida utilizando a tecnologia GraphQL, com o auxílio do Apollo Server[5], uma biblioteca *open source* em JavaScript para criação de servidores GraphQL.

Os dados da API são armazenados no serviço em nuvem do banco de dados MongoDB[6], o MongoDB Atlas[7]. Para a modelagem dos dados e conexão com o banco, foi utilizada a biblioteca do *Node.js* chamada *Mongoose*[8].

A Figura 9 mostra a arquitetura da aplicação. O usuário (cliente) carrega as páginas da aplicação, que são renderizadas pelo *React*. Em seguida, em uma página de um Conjunto de Dados, por exemplo, são feitas requisições à API GraphQL, que consulta o banco de dados através da biblioteca *Mongoose*. A resposta (os dados) são retornados para o usuário e renderizado pelo React na interface de usuário.

[3] https://nodejs.org/en
[4] https://expressjs.com/pt-br/
[5] https://www.apollographql.com/docs/apollo-server/
[6] https://www.mongodb.com/
[7] https://www.mongodb.com/pt-br/atlas/database
[8] https://mongoosejs.com/

Figura 9 – Arquitetura da aplicação



Fonte: Elaborado pelo autor

3.3 IMPORTAÇÃO DOS DADOS

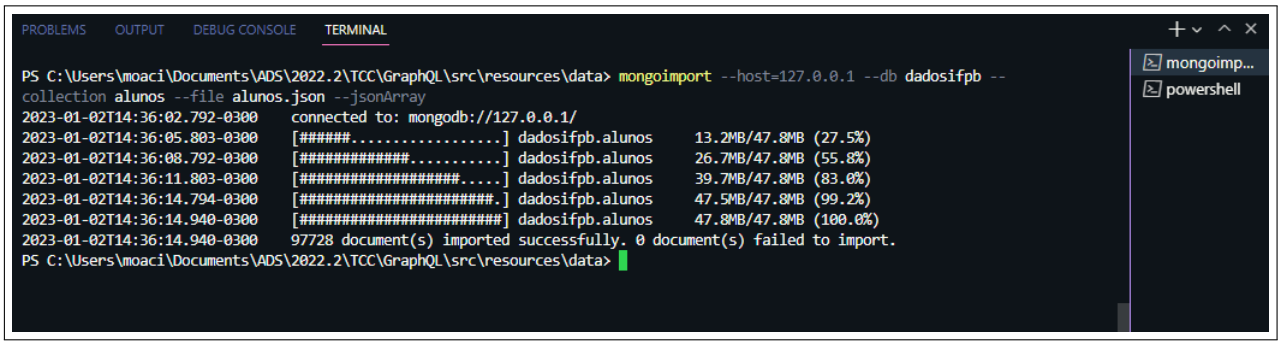
Para disponibilizar os dados do IFPB para consulta na API proposta, é necessário baixar os arquivos JSON dos conjuntos de dados disponíveis no Portal de Dados do IFPB e importá-los para o banco de dados MongoDB.

Primeiramente, acessa-se um conjunto de dados no formato JSON no Portal de Dados do IFPB, e então é feito o download clicando no botão 'Baixar', salvando o arquivo no diretório de recursos do projeto.

Em seguida, por meio de um terminal, o comando 'mongoimport' da ferramenta MongoDB Shell é utilizado para importar os dados do arquivo JSON para uma coleção

no banco de dados MongoDB. São passadas informações como o local e a porta em que o MongoDB está em execução, o nome do banco de dados, o nome da coleção no banco de dados e o caminho em que o arquivo JSON está armazenado. Também é indicado que o arquivo é um array de JSON usando a opção '- -jsonArray'.

Figura 10 – Exemplo de utilização do comando mongoimport



Fonte: Elaborado pelo autor

3.4 API GRAPHQL

Foi implementada a API GraphQL sob os dados disponibilizados pelo IFPB, em formato JSON. Para a implementação, foi utilizada a biblioteca *graphql-modules*[9] para deixar a aplicação modularizada e o *Apollo Server*[10], um servidor GraphQL que permite a realização das requisições e obtenção das respostas da API.

No Algoritmo 4, pode ser visto o *schema* do conjunto de dados relativo aos *campus* do IFPB, ou, em termos gerais, a "forma"que descreve como os dados estão estruturados, bem como as consultas disponíveis e seus valores de retorno.

[9] https://www.npmjs.com/package/graphql-modules
[10] https://www.apollographql.com/docs/apollo-server/

Algoritmo 4 – Schema do Conjunto de Dados dos Campus implementado na API GraphQL com JavaScript

```
   const { gql } = require('graphql-modules');

module.exports = gql`

    type Query {
        campi (filtros: InputsCampi): [Campi],
    }

    type Campi {
        url: String!,
        uuid: ID!,
        nome: String!,
        sigla: String!,
        cep: String!,
        telefone: String,
        cnpj: String!,
        endereco: String!,
        municipio: MunicipioCampi
    }

    type MunicipioCampi {
        nome: String,
        uf: String
    }

    input FiltrosCampi {
        offset: Int,
        limit: Int,
        nome: String,
        sigla: String
    }

    input InputsCampi {
        filtro: FiltrosCampi
    }

`;

`;
```

Fonte: Elaborado pelo autor

Da mesma forma, pode-se ver os *resolvers* no Algoritmo 5. Estes são funções responsáveis pela resolução e retorno dos dados que são solicitados pelas consultas.

Algoritmo 5 – Resolver do Conjunto de Dados dos Campus implementado na API GraphQL com JavaScript

```
    const Campi = require('../models/campi')

module.exports = {
    Query: {
        campi: async (_, args) => {
            const {filtro} = args.filtros;

            const query = {};

            if(filtro.nome !== null) {
                const nome = new RegExp(filtro.nome);
                query.nome = { $regex: nome, $options: 'i' };
            }

            if(filtro.sigla !== null) {
                const sigla = new RegExp(filtro.sigla);
                query.sigla = { $regex: sigla, $options: 'i' };
            }

            try {

                const campi = await Campi.find(query)
                .skip(filtro.offset)
                .limit(filtro.limit);

                return campi;
            } catch(e) {
                console.log(`Ocorreu um erro: ${e}`);
            }

        }
    }
};
```

Fonte: Elaborado pelo autor

Já no Algoritmo 6, o arquivo responsável por agrupar e criar o módulo, contendo o *schema* e o *resolver* do conjunto de dados dos *campus*, exportado para implementação no servidor GraphQL.

Algoritmo 6 – Módulo do Conjunto de Dados dos Campus implementado na API GraphQL com JavaScript

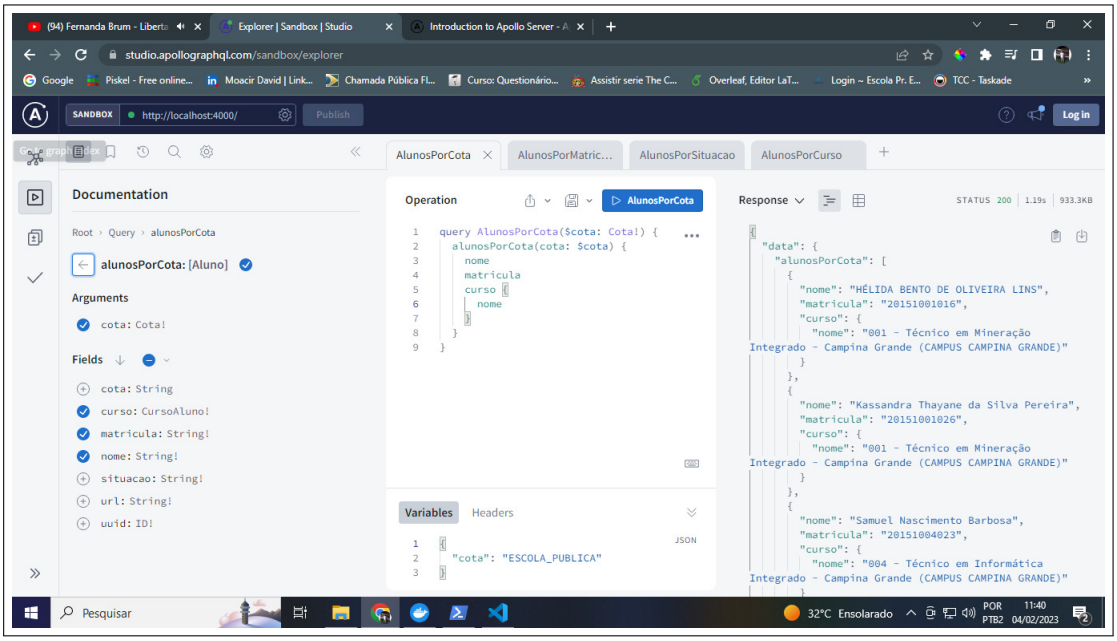
```
    const { createModule } = require('graphql-modules');
const Campi = require('../schemas/campi.type.graphql');
const CampiResolvers = require('../resolvers/campi.resolver.graphql.
   js');

module.exports = createModule({
    id: 'campi-modulo',
    dirname: __dirname,
    typeDefs: [Campi],
    resolvers: [CampiResolvers]
  });
```

Fonte: Elaborado pelo autor

Por fim, a API em execução na rede local com o *Apollo Server* (um servidor compatível com GraphQL) é demonstrada na Figura 11, realizando a consulta de alunos do IFPB que pertencem a cota de oriundos de escola pública, requisitando somente os campos nome, matrícula e o nome do curso dos estudantes.

Figura 11 – API GraphQL executada no servidor local com o servidor Apollo Server



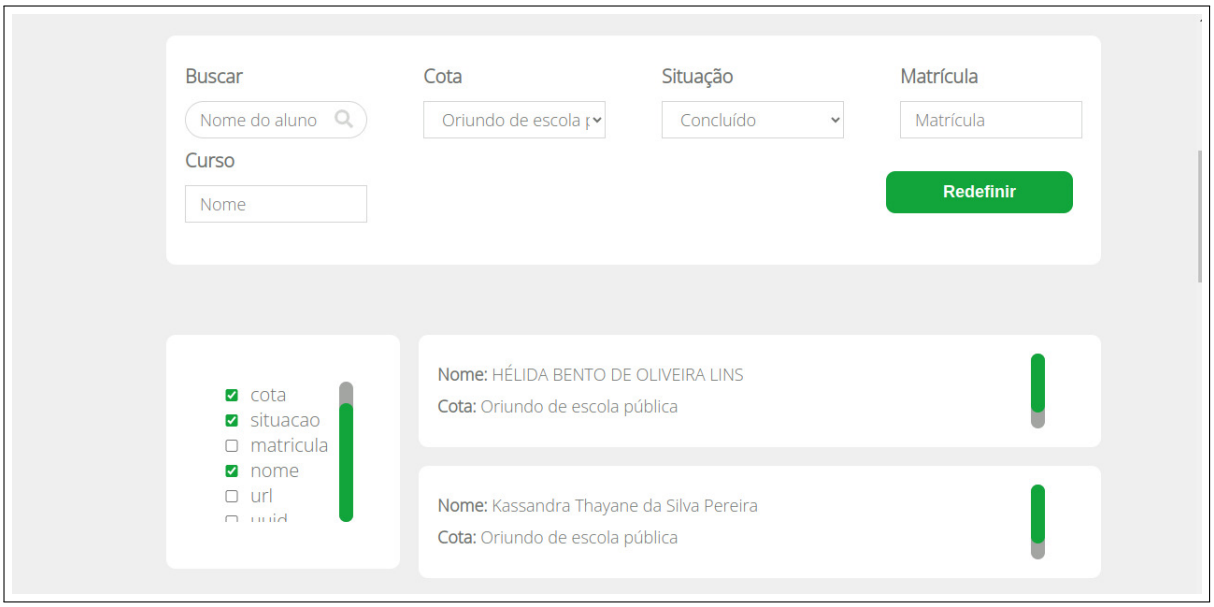
Fonte: Elaborado pelo autor

A API implementada em modo produção pode ser acessada neste link: <https://backend-tcc-bd47.onrender.com/>. Já a ferramenta de visualização dos dados encontra-se neste endereço: <https://frontend-tcc-vf2i.onrender.com/>. Por fim, o repositório público do projeto no GitHub pode ser acesso através dessa URL: <https://github.com/moacirdavidag/tcc-ads-api-graphql>.

3.5 TELAS DO SISTEMA

A Figura 12 mostra a página do conjunto de dados de alunos quando é realizada uma consulta à API. É possível visualizar as partes da aplicação que contêm os campos de busca e filtros, os campos de dados disponíveis para a consulta e o próprio retorno dos dados, que são renderizados na página.

Figura 12 – Tela do Conjunto de Dados de Alunos



Fonte: Elaborado pelo autor

As demais telas do sistema encontram-se no Apêndice B.

4 CONSIDERAÇÕES FINAIS

O acesso às informações públicas oriundas da esfera governamental por parte dos cidadãos é um processo muito importante em uma democracia. A tomada de decisões por parte dos governantes, que utilizam o erário, deve ter a opinião da sociedade civil considerada, para uma melhor gestão de recursos e desenvolvimento social.

As tecnologias da informação e comunicação, alinhadas aos propósitos democráticos, contribuem para que cada vez mais a sociedade seja atuante nos processos públicos e agente fiscalizador dos recursos utilizados pelos órgãos governamentais.

Um dos desafios a ser superados, por parte da instituição (IFPB), é a atualização dos conjuntos de dados, visto que os mesmos datam de 2019. É preciso manter as informações públicas atualizadas para garantir a transparência e relevância dos dados.

A API e a ferramenta desenvolvida neste trabalho são mais uma das contribuições para que os cidadãos interessados possam fiscalizar a utilização dos recursos públicos oriundos do Governo Federal ao Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, promovendo maior transparência e controle cívico.

4.1 TRABALHOS FUTUROS

Como sugestão para trabalhos futuros, pode ser desenvolvido um meio de atualização dos dados de forma automática (o que, atualmente, é feito através de importação utilizando o comando mongoimport no terminal). Além disso, pode ser desenvolvido um meio de gerenciar melhor a ferramenta de visualização dos dados, a partir de um *dashboard*, que seria acessado via autenticação, por servidores autorizados do IFPB que precisem fazer alterações nas estruturas das páginas ou adicionar novos conjuntos de dados para a visualização pública das informações.

REFERÊNCIAS

BASTOS, A. M.; BASTOS, A. M. Uso do scrum como método para otimização na elaboração de projetos. **Research, Society and Development**, v. 10, n. 9, p. e40610918329–e40610918329, 2021.

BITTENCOURT, A. L. d. M. R. Uma comparação de performance entre arquitetura graphql e rest. 2021.

BRASIL. Regula o acesso a informações previsto no inciso xxxiii do art. 5º , no inciso ii do § 3º do art. 37 e no § 2º do art. 216 da constituição federal; altera a lei nº 8.112, de 11 de dezembro de 1990; revoga a lei nº 11.111, de 5 de maio de 2005, e dispositivos da lei nº 8.159, de 8 de janeiro de 1991; e dá outras providências. **Brasília: LEI No 12.527, DE 18 DE NOVEMBRO DE 2011**, 2011.

____. [constituição (1988)]. constituição da república federativa do brasil. **Brasília, DF: Senado Federal**, 2016.

BRITO, G.; VALENTE, M. T. Rest vs graphql: A controlled experiment. In: IEEE. **2020 IEEE international conference on software architecture (ICSA)**. [S.l.], 2020. p. 81–91.

CAMARGO, Á. A. B. de; VALLE, A. B. do. **Gerenciamento de mudanças e stakeholders em projetos**. [S.l.]: Editora FGV, 2022.

CGU. Manual da lei de acesso à informação para estados e municípios. **Controladoria Geral da União, Brasília - DF**, 2013.

COCKBURN, A. **Escrevendo Casos de Usos Eficazes: Um guia prático para desenvolvedores de software**. [S.l.]: Bookman Editora, 2005.

DANTAS, J. C. Contribuições da implantação do scrum como metodologia ágil para a otimização da gestão de projetos nas organizações. **RECIMA21-Revista Científica Multidisciplinar-ISSN 2675-6218**, v. 2, n. 7, p. e27541–e27541, 2021.

FARIAS, T. **O que é prototipagem de software?** 2022. Disponível em : <https://softeo.com.br/ferramentas-pt/entenda-o-que-e-prototipagem-de-software/>.

FRANÇA, M. D. C.; SILVA, E. da. Performance evaluation of rest and graphql apis searching nested objects. **Anais do Computer on the Beach**, v. 11, n. 1, p. 237–244, 2020.

GONÇALVES, A. d. S. **Concepção e desenvolvimento de uma API REST com incorporação de mecanismos de segurança aplicacional**. Tese (Doutorado), 2022.

ISOTANI, S.; BITTENCOURT, I. I. **Dados abertos conectados: em busca da web do conhecimento**. [S.l.]: Novatec Editora, 2015.

JUNIOR, F. E. d. O. Da coleta à análise: um estudo de caso usando dados públicos de saúde, segurança e desenvolvimento social. 2022.

LOENEN, B. van; VANCAUWENBERGHE, G.; CROMPVOETS, J. **Open Data Exposed**. T.M.C. Asser Press, 2018. (Information Technology and Law Series). ISBN 9789462652613. Disponível em: <https://books.google.com.br/books?id=nPZ0DwAAQBAJ>.

Lucidchart. **Diagrama de caso de uso UML: O que é, como fazer e exemplos**. c2023. <https://www.lucidchart.com/pages/pt/diagrama-de-caso-de-uso-uml>, Acesso em: 04 maio 2023.

MACHADO, F. N. R. **Análise e Gestão de Requisitos de Software Onde nascem os sistemas**. [S.l.]: Saraiva Educação SA, 2018.

Metodologia Ágil. **Scrum**. s.d. Disponível em : <https://metodologiaagil.com/scrum/>, Acesso em: 20 jul. 2023.

NETO, N. A. S. Graphql uma abordagem a lternativa ao padrão rest. **Sistemas de Informação-Pedra Branca**, 2020.

OLIVEIRA, P. H. C. Desenvolvimento de um gerador de api rest seguindo os principais padrões da arquitetura. 2015.

OLIVEIRA, R. L. F.; PEDRON, C. D. Métodos ágeis: Uma revisão sistemática sobre benefícios e limitações. **Brazilian Journal of Development**, v. 7, n. 1, p. 4520–4534, 2021.

Open Knowledge Foundation. **How to open data**. 2019. <https://okfn.org/opendata/how-to-open-data/>.

PEDROSO, L.; TANAKA, A.; CAPPELLI, C. A lei de acesso à informação brasileira e os desafios tecnológicos dos dados abertos governamentais. In: SBC. **Anais do IX Simpósio Brasileiro de Sistemas de Informação**. [S.l.], 2013. p. 523–528.

PRESSMAN, R. S. **Engenharia de Software: uma abordagem profissional**. 7ª ed.. ed. São Paulo: McGraw-Hill, 2011.

QUIÑA-MERA, A.; FERNANDEZ, P.; GARCÍA, J. M.; RUIZ-CORTÉS, A. Graphql: A systematic mapping study. **ACM Computing Surveys (CSUR)**, ACM New York, NY, 2022.

SCHWABER, K.; SUTHERLAND, J. **Scrum Guide**. 2020. Acesso em: 19 07 2023. Disponível em: <https://scrumguides.org/scrum-guide.html>.

SEGUNDO, J. E. S. Web semântica, dados ligados e dados abertos: uma visão dos desafios do brasil frente às iniciativas internacionais. In: **XVI Encontro Nacional de Pesquisa em Pós-Graduação em Ciência da Informação**. [S.l.: s.n.], 2015.

SILVA, A. d. A. P.; MONTEIRO, D. A. A.; REIS, A. de O. Qualidade da informação dos dados governamentais abertos: análise do portal de dados abertos brasileiro. **Revista Gestão em Análise**, v. 9, n. 1, p. 31–47, 2020.

TAVARES, R. O.; FERNANDES, S. R. Graphql-rest in peace: Uma análise comparativa. **Seminários de Trabalho de Conclusão de Curso do Bacharelado em Sistemas de Informação**, v. 8, n. 1, 2022.

VALENTE, M. T. Engenharia de software moderna. **Princípios e Práticas para Desenvolvimento de Software com Produtividade**, v. 1, p. 24, 2020.

W3C. **5 Star Linked Data**. 2011. <https://www.w3.org/2011/gld/wiki/5_Star_Linked_Data>, Last accessed on 2023-01-09.

APÊNDICE A – CASOS DE USO DA FERRAMENTA DE VISUALIZAÇÃO GRÁFICA DOS DADOS

Quadro 3 – Caso de uso: Escolher filtros

Nome do Caso de Uso	**Escolher filtros**
Ator principal	Usuário
Atores secundários	Nenhum
Resumo	Neste caso de uso, as etapas seguidas por um usuário para realizar uma consulta aos dados utilizando filtros são descritas.
Pré-condições	O usuário está na página de um Conjunto de Dados e escolheu pelo menos um dos campos do dado disponível para consulta.
Pós-condições	É realizada uma consulta aos dados utilizando os filtros escolhidos pelo usuário e o resultado é renderizado para visualização.
Ações do Ator	Ações do Sistema
1. O usuário acessa a página de um Conjunto de Dados.	2. O sistema exibe a página referente ao Conjunto de Dados escolhido.
3. O usuário seleciona um dos filtros disponíveis para o Conjunto de Dados.	4. É realizada uma consulta à API e o resultado é renderizado na página para visualização.

Fonte: Elaborado pelo autor.

Quadro 4 – Caso de uso: Escolher campos

Nome do Caso de Uso	**Escolher campos**
Ator principal	Usuário
Atores secundários	Nenhum
Resumo	Neste caso de uso, as etapas seguidas por um usuário para selecionar os campos disponíveis para consulta do Conjunto de Dados selecionado são descritas.
Pré-condições	O usuário está na página de um Conjunto de Dados.
Pós-condições	O (s) campo (s) disponíveis para consulta no Conjunto de Dados escolhido são exibidos.
Ações do Ator	Ações do Sistema
1. O usuário acessa a página de um Conjunto de Dados.	2. O sistema exibe a página do Conjunto de Dados selecionado.
3. O usuário seleciona um ou mais campos disponíveis que devem ser consultados no Conjunto de Dados.	4. O sistema retorna os campos selecionados pelo usuário e exibe na página.

Fonte: Elaborado pelo autor.

Quadro 5 – Caso de uso: Realizar busca

Nome do Caso de Uso	**Realizar busca**
Ator principal	Usuário
Atores secundários	Nenhum
Resumo	Neste caso de uso, as etapas seguidas por um usuário para realizar buscas por um dado específico são descritas.
Pré-condições	O usuário está na página de um Conjunto de Dados e escolheu pelo menos um dos campos disponíveis para consulta.
Pós-condições	A informação, caso existente, é retornada ao usuário e exibida na página. Do contrário, uma mensagem de informação não encontrada é exibida.
Ações do Ator	Ações do Sistema
1. O usuário acessa a página de um Conjunto de Dados.	2. O sistema exibe a página do Conjunto de Dados selecionado.
3. O usuário seleciona um ou mais campos disponíveis que devem ser consultados no Conjunto de Dados.	4. O sistema retorna os campos selecionados pelo usuário e exibe na página.
5. O usuário seleciona um dos campos de busca disponíveis e digita a informação que deseja buscar.	6. O sistema consulta a API e retorna as informações buscadas, caso existam. 7. As informações (ou mensagem de erro, caso os dados buscados não existam) são exibidas na página.

Fonte: Elaborado pelo autor.

Quadro 6 – Caso de uso: Redefinir consulta

Nome do Caso de Uso	**Redefinir consulta**
Ator principal	Usuário
Atores secundários	Nenhum
Resumo	Neste caso de uso, as etapas seguidas por um usuário para redefinir uma consulta são apresentadas.
Pré-condições	O usuário está na página de um Conjunto de Dados e pelo menos um campo disponível para consulta ou filtro foi selecionado.
Pós-condições	A consulta é redefinida ao estado inicial, com todos os campos selecionados retirados, filtros e campos de busca voltando para o valor padrão.
Ações do Ator	Ações do Sistema
1. O usuário acessa a página de um Conjunto de Dados.	2. O sistema exibe a página do Conjunto de Dados selecionado.
3. O usuário seleciona um ou mais campos disponíveis, filtros ou realiza buscas no Conjunto de Dados.	4. O sistema retorna a consulta feita pelo usuário.
5. O usuário quer começar uma nova consulta e, para isso, clica no botão "Redefinir".	6. Todos os campos selecionados são retirados, os filtros e campos de busca voltam para o valor padrão. 7. A página do Conjunto de Dados volta a ficar vazia, pedindo para o usuário selecionar pelo menos um campo para consulta.

Fonte: Elaborado pelo autor.

Quadro 7 – Caso de uso: Buscar Conjunto de Dados

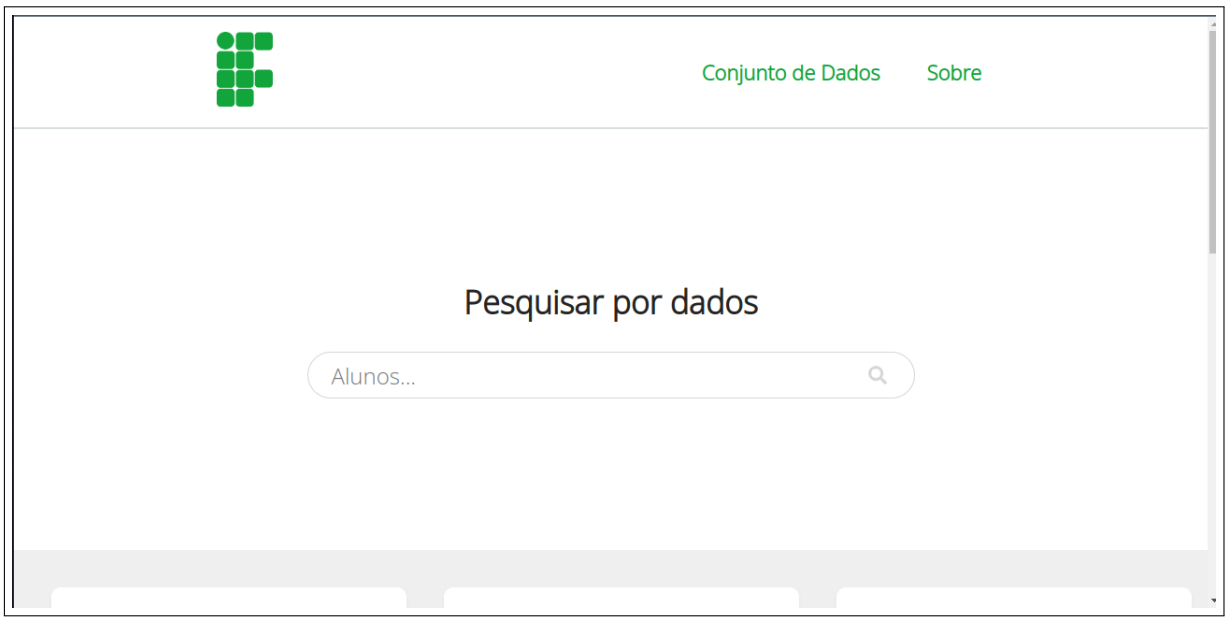
Nome do Caso de Uso	**Buscar Conjunto de Dados**
Ator principal	Usuário
Atores secundários	Nenhum
Resumo	Neste caso de uso, as etapas seguidas por um usuário para realizar buscas por conjuntos de dados são descritas.
Pré-condições	O usuário está na página inicial da aplicação.
Pós-condições	Os conjuntos de dados correspondentes aos termos utilizados na busca são exibidos como resultados. Caso não exista nenhum resultado para a consulta, uma mensagem informando que nenhum resultado foi encontrado é exibida na página.
Ações do Ator	Ações do Sistema
1. O usuário acessa a página inicial da aplicação.	2. O sistema exibe a página inicial da aplicação.
3. O usuário digita o termo que corresponde ao conjunto de dados que ele deseja procurar no *input* de pesquisa, apertando a tecla 'Enter' posteriormente.	4. O sistema faz a busca pelos conjuntos de dado de acordo com o valor passado na busca, renderizando em outra página os resultados. 5. Caso existam conjuntos de dados relacionados aos termos utilizados na busca, o sistema renderiza uma lista com os mesmos na página de resultados da busca. 6. Caso não exista nenhum conjunto de dados correspondente aos termos da busca, uma mensagem informando a inexistência de resultados é exibida na página de resultados da busca.

Fonte: Elaborado pelo autor.

APÊNDICE B – TELAS DA FERRAMENTA DE VISUALIZAÇÃO GRÁFICA DOS DADOS

A Figura 13 mostra a tela inicial da aplicação de exibição dos dados. Nela, um *input* permite ao usuário buscar por conjuntos de dados através de termos-chaves, como "servidores", por exemplo.

Figura 13 – Tela inicial da ferramenta de visualização gráfica dos dados



Fonte: Elaborado pelo autor

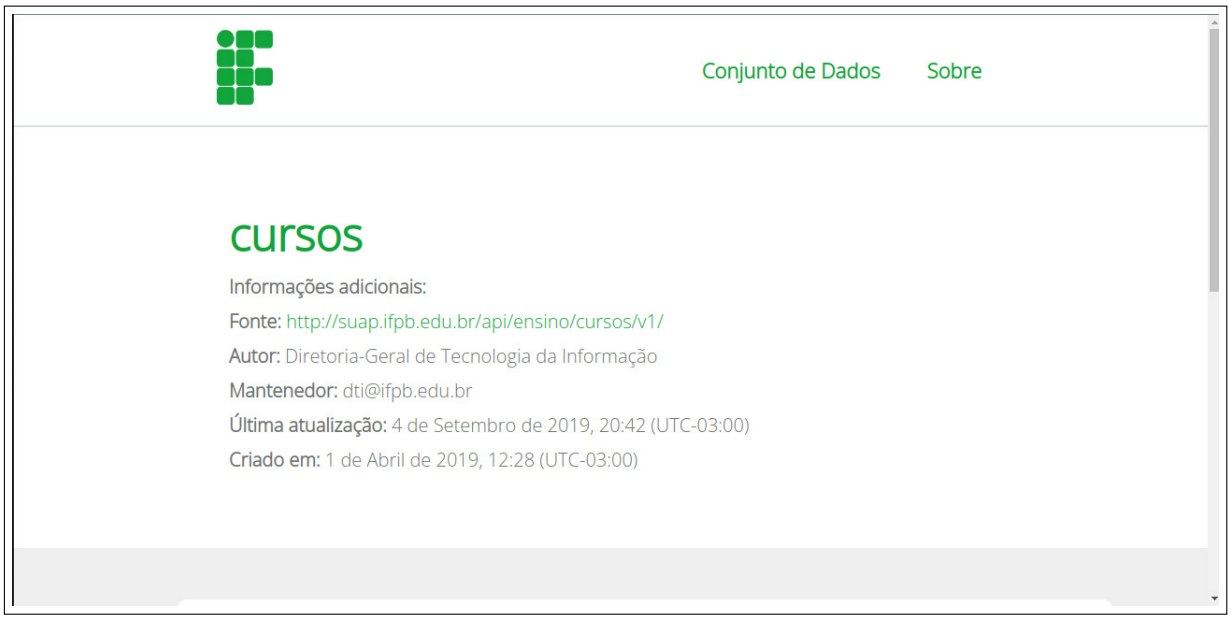
A Figura 14 mostra a página "Conjunto de Dados", em que todos os conjuntos de dados disponíveis para consulta são listados.

Figura 14 – Página contendo todos os Conjuntos de Dados disponíveis

Fonte: Elaborado pelo autor

Na Figura 15, a página individual de um conjunto de dados é exibida. No exemplo da figura, foi selecionada a página do conjunto de dados dos cursos do IFPB.

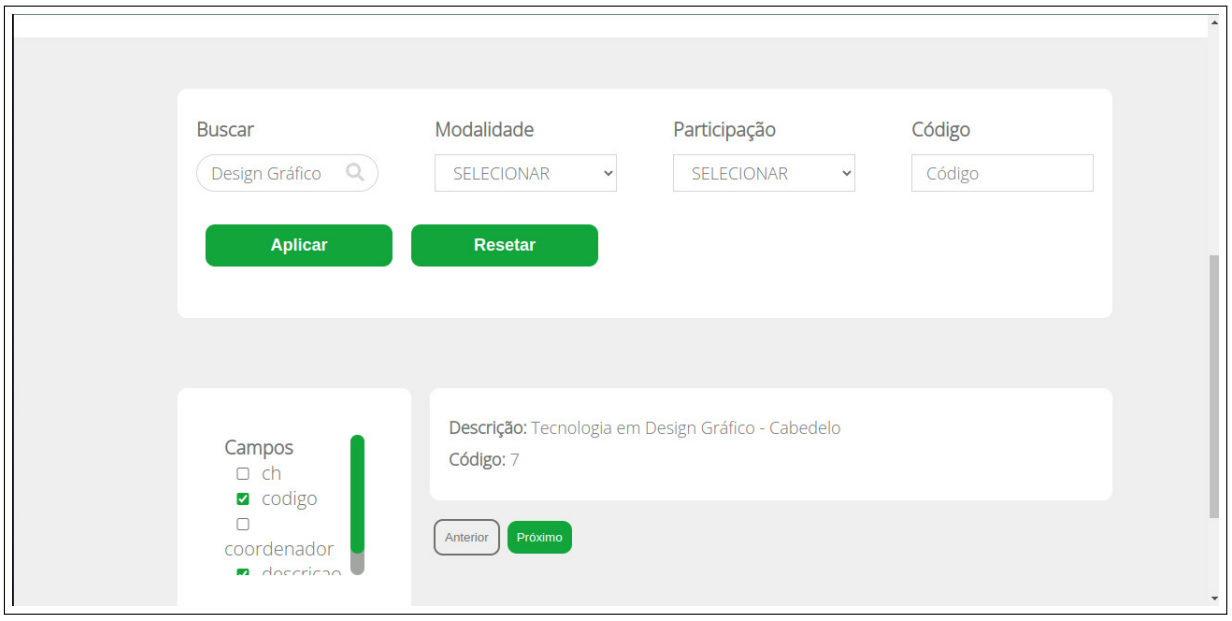
Figura 15 – Página de um Conjunto de Dados



Fonte: Elaborado pelo autor

A Figura 16 mostra o mecanismo de busca por dados específicos em funcionamento. No exemplo, na página do conjunto de dados sobre os cursos do IFPB foi pesquisado o curso que tem nome "Design Gráfico"no campo de busca por nome. Como resultado, foi renderizado o título (descrição) e código do curso, pois estes foram os campos escolhidos para consulta.

Figura 16 – Busca por dados específicos



Fonte: Elaborado pelo autor

A Figura 17 mostra a busca por conjuntos de dados. A palavra "projetos"foi o termo-chave para a pesquisa, que retornou dois resultados: o conjunto de dados de projetos de pesquisa e de extensão.

Figura 17 – Busca por conjuntos de dados

Fonte: Elaborado pelo autor

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus Cajazeiras
Rua José Antônio da Silva, 300, Jardim Oásis, CEP 58.900-000, Cajazeiras (PB)
CNPJ: 10.783.898/0005-07 - Telefone: (83) 3532-4100

Documento Digitalizado Ostensivo (Público)

Documento final - TCC Moacir David

Assunto: Documento final - TCC Moacir David
Assinado por: Moacir David
Tipo do Documento: Anexo
Situação: Finalizado
Nível de Acesso: Ostensivo (Público)
Tipo do Conferência: Cópia Simples

Documento assinado eletronicamente por:

- **Moacir David de Almeida Gonçalves, ALUNO (202012010020) DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - CAJAZEIRAS**, em 13/09/2023 13:05:06.

Este documento foi armazenado no SUAP em 13/09/2023. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 941093
Código de Autenticação: 0ffddfa7da