UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ÍCARO CHAGAS DE ALMEIDA

DESENVOLVIMENTO DE UM SISTEMA PARA DISPONIBILIZAÇÃO DE DADOS DO DATASUS: UMA ABORDAGEM DE ACESSO SIMPLIFICADO

CAMPINA GRANDE - PB

2024

ÍCARO CHAGAS DE ALMEIDA

DESENVOLVIMENTO DE UM SISTEMA PARA DISPONIBILIZAÇÃO DE DADOS DO DATASUS: UMA ABORDAGEM DE ACESSO SIMPLIFICADO

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador : João Arthur Brunet Monteiro

CAMPINA GRANDE - PB

2024

ÍCARO CHAGAS DE ALMEIDA

DESENVOLVIMENTO DE UM SISTEMA PARA DISPONIBILIZAÇÃO DE DADOS DO DATASUS: UMA ABORDAGEM DE ACESSO SIMPLIFICADO

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

João Arthur Brunet Monteiro

Orientador – UASC/CEEI/UFCG

Francilene Procópio Garcia

Examinador – UASC/CEEI/UFCG

Francisco Vilar Brasileiro

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 15 de Maio de 2024.

CAMPINA GRANDE - PB

RESUMO

O Departamento de Informática do Sistema Único de Saúde (DATASUS) disponibiliza dados fundamentais para análises da situação sanitária do Brasil, embasando tomadas de decisão e o desenvolvimento de programas de intervenção em saúde. No entanto, esses dados são fornecidos em um formato não diretamente compatível com ferramentas populares, como Excel ou Google Sheets, dificultando sua acessibilidade para pesquisadores, analistas e profissionais da área médica. Além disso, enfrenta-se o desafio atrelado ao significativo volume dos dados, caracterizando-os a nível de *Big Data*. A escala massiva desses dados demanda abordagens eficientes de engenharia de dados para lidar com processamento e armazenamento, a fim de proporcionar uma estrutura robusta e escalável para a manipulação dos mesmos.

Diante dessa problemática, este trabalho propõe o desenvolvimento de um processo automatizado para extração de 47 tabelas, disponíveis em arquivos no formato dbc e provenientes de 13 bases de dados do DATASUS, o armazenamento em um *data warehouse* e a disponibilização dos dados por meio de uma API. Espera-se que essa iniciativa facilite a análise de dados de saúde no Brasil, oferecendo acesso simplificado e dados prontos para análise a pesquisadores, analistas e profissionais da área médica.

DEVELOPMENT OF A SYSTEM FOR PROVIDING DATASUS DATA: A SIMPLIFIED ACCESS APPROACH

ABSTRACT

The Department of Informatics of the Unified Health System (DATASUS) provides essential data for analyzing the health situation in Brazil, supporting decision-making and the development of health intervention programs. However, this data is provided in a format that is not directly compatible with popular tools such as Excel or Google Sheets, making it less accessible to researchers, analysts, and healthcare professionals. Additionally, there is the challenge associated with the significant volume of data, characterized as Big Data. The massive scale of this data demands efficient data engineering approaches for processing and storage to provide a robust and scalable system for handling it.

In light of this problem, this work proposes the development of an automated process for extracting 47 tables available in dbc format files from 13 DATASUS databases, storing them in a data warehouse, and making the data available through an API. This initiative is expected to facilitate health data analysis in Brazil by providing simplified access and ready-to-analyze data to researchers, analysts, and healthcare professionals.

Desenvolvimento de Um Sistema Para Disponibilização de Dados do DATASUS: Uma Abordagem de Acesso Simplificado

Ícaro Chagas de Almeida
Universidade Federal de Campina Grande
icaro.almeida@ccc.ufcg.edu.br

João Arthur Brunet Monteiro
Universidade Federal de Campina Grande
joao.arthur@computacao.ufcg.edu.br

RESUMO

O Departamento de Informática do Sistema Único de Saúde (DATASUS) disponibiliza dados fundamentais para análises da situação sanitária do Brasil, embasando tomadas de decisão e o desenvolvimento de programas de intervenção em saúde. No entanto, esses dados são fornecidos em um formato não diretamente compatível com ferramentas populares, como Excel ou Google Sheets, dificultando sua acessibilidade para pesquisadores, analistas e profissionais da área médica. Além disso, enfrenta-se o desafio atrelado ao significativo volume dos dados, caracterizando-os a nível de *Big Data*. A escala massiva desses dados demanda abordagens eficientes de engenharia de dados para lidar com processamento e armazenamento, a fim de proporcionar uma estrutura robusta e escalável para a manipulação dos mesmos.

Diante dessa problemática, este trabalho propõe o desenvolvimento de um processo automatizado para extração de 47 tabelas, disponíveis em arquivos no formato dbc e provenientes de 13 bases de dados do DATASUS, o armazenamento em um *data warehouse*, e a disponibilização dos dados por meio de uma API. Espera-se que essa iniciativa contribua com a análise de dados de saúde no Brasil, oferecendo acesso simplificado e dados prontos para análise a pesquisadores, analistas e profissionais da área médica.

Palavras-Chave

DATASUS, Data Warehouse, Big Data, Saúde Pública, Automação de Processos, Engenharia de Dados, Desenvolvimento de Software

1. INTRODUÇÃO

O acesso a dados confiáveis e abrangentes desempenha um papel fundamental na compreensão da situação sanitária de um país, na formulação de políticas públicas eficazes e na implementação de programas de intervenção em saúde. No contexto brasileiro, o Departamento de Informática do Sistema Único de Saúde (DATASUS) desempenha um papel crucial ao fornecer uma vasta gama de dados relacionados à saúde, abrangendo desde estatísticas de mortalidade e natalidade até informações sobre procedimentos médicos e internações hospitalares.

Contudo, apesar da riqueza desses dados, sua utilização eficaz tem sido limitada por diversos desafios, destacando-se a complexidade do formato em que são disponibilizados e o volume massivo dos mesmos, caracterizando-os como *Big Data*. Os dados do DATASUS são comumente disponibilizados em formato dbc, o que dificulta sua interpretação e análise direta por ferramentas convencionais como Excel ou Google Sheets. Isso dificulta significativamente o acesso e a interpretação dos dados, demandando conhecimento técnico, e fazendo com que usuários muitas vezes precisem recorrer a processos manuais e demorados para converter e preparar os dados para análise.

Com a quantidade massiva de dados gerados diariamente pelo sistema de saúde brasileiro, o processamento e armazenamento para análise por terceiros tornam-se desafiadores devido ao volume crescente de informações. Lidar com esse desafio requer abordagens escaláveis e robustas; neste cenário os *data warehouses* desempenham um papel fundamental no contexto do *Big Data*.

Um *data warehouse* é um sistema de armazenamento de dados que permite consolidar e organizar grandes quantidades de dados de diversas fontes em um único local. Tal ferramenta oferece recursos para consultas analíticas complexas, garantindo a integridade, segurança e disponibilidade dos dados [1]. Além disso, um *data warehouse* proporciona um ambiente otimizado para análises de negócios e tomada de decisões estratégicas, fornecendo *insights* valiosos a partir dos dados armazenados [2]. No caso específico dos dados do DATASUS, o *data warehouse* se apresenta como uma solução adequada para lidar com o volume e a complexidade desses dados, facilitando o acesso, armazenamento e análise das informações disponibilizadas.

Neste contexto, este trabalho propõe o desenvolvimento de um processo automatizado para tornar os dados do DATASUS mais acessíveis e utilizáveis. Através da aplicação de técnicas de engenharia de dados, possibilita-se que o usuário extraia dados em formato dbc, os armazene em um *data warehouse*, e possa fazer consultas às informações disponibilizadas, de modo que todo esse processamento esteja abstraído por uma API. Essa abordagem não apenas facilita o acesso e a análise dos dados, mas também proporciona uma estrutura robusta e escalável para lidar com o crescente volume de informações produzidas pelo sistema de saúde brasileiro.

2. FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta conceitos teóricos fundamentais que embasam o desenvolvimento e a implementação da API proposta neste trabalho. O objetivo é fornecer uma visão geral dos fundamentos teóricos subjacentes à estrutura e ao funcionamento da API, e prover uma compreensão inicial dos conceitos relevantes para o contexto do sistema desenvolvido.

2.1 Data Warehouse

Um *data warehouse* é um sistema de armazenamento de dados projetado para suportar a análise e geração de relatórios a partir de dados provenientes de múltiplas fontes. É considerado um componente central no contexto de *Business Intelligence* (BI), pois viabiliza acesso facilitado a uma grande variedade de dados. Ao oferecer uma visão consolidada e histórica dos dados, o *data warehouse* possibilita análises complexas e a geração de *insights* valiosos para qualquer organização [1].

2.2 Processamento ETL

O processo ETL (Extração, Transformação e Carga) é uma etapa crucial no fluxo de dados de um sistema, especialmente quando lidamos com grandes volumes de informações e fontes de dados heterogêneas [1]. A etapa de extração envolve a coleta de dados de várias fontes, que podem incluir bancos de dados, arquivos, APIs e outros sistemas externos [3]. Uma vez extraídos, os dados passam por uma fase de transformação, onde são limpos, normalizados, enriquecidos e reestruturados conforme necessário para atender às necessidades de análise e armazenamento. Essa fase é essencial para garantir a consistência e a integridade dos dados, além de prepará-los para a etapa final do processo.

Na fase de carga, os dados transformados são carregados no destino final, que geralmente é um data warehouse, um banco de dados relacional ou outro sistema de armazenamento. Essa etapa deve possibilitar uma inserção eficiente e escalável, garantindo que os dados estejam prontos para análises posteriores. O processo ETL é fundamental para assegurar a qualidade e a confiabilidade dos dados em todo o ciclo de vida de um sistema de informação, permitindo que as organizações tomem decisões baseadas em dados precisos e atualizados [1].

2.3 Big Data

Com o avanço da tecnologia, a quantidade de dados gerados cresceu exponencialmente, impulsionada por uma variedade de fontes, como transações online, registros de sensores, mídias sociais, dispositivos móveis e muito mais. Essa explosão de dados resulta em desafios significativos para sua captura, armazenamento, processamento, análise e interpretação. No entanto, também apresenta oportunidades sem precedentes para extrair *insights* valiosos e tomar decisões [4].

Geralmente, o *Big Data* é caracterizado em relação a volume, velocidade e variedade. O volume se refere à imensa quantidade de dados gerados por diversas fontes. A velocidade diz respeito à taxa rápida com que esses dados são gerados e precisam ser processados. Por fim, a variedade engloba a diversidade de tipos de dados, que podem ser estruturados, semi estruturados ou não estruturados [5].

O *Big Data* não é apenas um desafio técnico, mas também uma oportunidade estratégica para organizações de todos os setores. A capacidade de coletar, armazenar, processar e analisar grandes volumes de dados pode implicar em *insights* valiosos que impulsionam a inovação, a eficiência operacional e a tomada de decisões informadas. No entanto, para aproveitar ao máximo o potencial do *Big Data*, é fundamental adotar abordagens holísticas e ferramentas apropriadas que possibilitem lidar com a complexidade e a escala de um ambiente de dados em constante expansão [6].

2.4 Application Programming Interface

APIs (*Application Programming Interfaces*) desempenham um papel fundamental na interação entre sistemas de *software*, permitindo a comunicação e o compartilhamento de dados, e funcionalidades entre diferentes aplicações. APIs fornecem uma interface padronizada que permite que desenvolvedores integrem facilmente seus aplicativos com serviços externos, acessando recursos específicos de forma eficiente e segura.

No contexto da *Web*, as APIs são frequentemente utilizadas para criar serviços web RESTful (*Representational State Transfer*), que seguem os princípios arquiteturais do REST. Esses serviços fornecem *endpoints* bem definidos que podem ser acessados por clientes por meio de solicitações HTTP, como GET, POST, PUT e DELETE, para realizar operações específicas nos dados.

As APIs são cruciais na construção de aplicações robustas e escaláveis, além de serem essenciais para a criação de ecossistemas de desenvolvimento de *software*, onde diferentes partes podem colaborar e se integrar de forma eficaz [7].

3. METODOLOGIA

Esta seção descreve detalhadamente cada etapa da metodologia adotada neste trabalho, incluindo a escolha das ferramentas, as tecnologias utilizadas, as estratégias de desenvolvimento e as abordagens adotadas para garantir a eficiência e a qualidade do sistema.

3.1 Extração e Conversão dos Dados

Os dados do DATASUS encontram-se disponíveis no site indicado na seção de Anexos, podendo também ser baixados através de ferramentas que utilizam *File Transfer Protocol* (FTP) como o FileZilla. No entanto, é importante ressaltar que nem todos os dados, ainda que listados na fonte, estão disponíveis para extração.

O sistema desenvolvido possibilita a extração de 47 tabelas provenientes de 13 bases de dados distintas. O volume dos dados varia entre arquivos com aproximadamente 30 a 200 colunas, e algumas dezenas de milhares a alguns milhões de linhas.

A grande maioria das tabelas são compostas por arquivos separados por estado, atualizados mensalmente, resultando em pelo menos 27 arquivos disponíveis para cada tabela a cada mês (alguns arquivos são divididos em partes a, b e c para evitar arquivos excessivamente grandes). Além disso, há arquivos disponibilizados anualmente e outros que abrangem todos os estados da federação.

Para realizar a extração dos dados, foi desenvolvido um script que automatiza os processos de *download* e conversão dos arquivos de .dbc para o formato .csv. Nesta etapa fez-se uso das bibliotecas ftplib para listar os arquivos disponíveis, urllib para efetuar os respectivos *downloads*, e pysus para converter os arquivos para .csv.

3.2 Tratamento e Carregamento dos Dados

Os processos de tratamento e carregamento dos dados foram implementados em outro script dedicado, que permite realizar transformações nos dados para garantir sua conformidade com o esquema do banco de dados e, em seguida, carregá-los no PostgreSQL. Para realizar o tratamento dos dados fez-se uso da biblioteca pandas, que oferece uma ampla gama de funcionalidades para manipulação e limpeza de dados. O processo

de inserção no SGBD de destino foi implementado utilizando a biblioteca psycopg2, que fornece uma interface Python para manipular o PostgreSQL.

Com o objetivo de evitar inconsistências durante o processo de inserção, faz-se uso de um mecanismo que registra em uma tabela de metadados os arquivos já inseridos no banco de dados. Esse mecanismo garante a integridade dos dados, impedindo a inserção de dados duplicados em cada tabela do banco de dados.

Durante todo o processo ETL, mensagens de log são exibidas para informar ao usuário da API sobre o andamento dos processos e possíveis erros. Esse monitoramento contínuo tem o propósito de garantir transparência e confiabilidade ao método desenvolvido.

3.3 Desenvolvimento da API com Django REST Framework

Esta etapa descreve o processo de desenvolvimento da API utilizando o Django REST *Framework* (DRF). Inicialmente, foram definidos os modelos de dados necessários para representar as tabelas do DATASUS que seriam disponibilizadas pela API. Tais modelos foram criados de acordo com a estrutura dos dados disponibilizada pela documentação do DATASUS.

Após a definição dos modelos, foram desenvolvidas as views da API. As *views* são responsáveis por mapear as solicitações HTTP para as operações CRUD correspondentes nos modelos de dados. Com o intuito de assegurar um fluxo consistente, foram implementadas validações nas *views* em relação aos parâmetros fornecidos durante as requisições, de modo que mensagens de erro são retornadas quando parâmetros mandatórios não são fornecidos, ou são fornecidos de forma errada.

Considerando a importância de garantir a eficiência na manipulação de grandes volumes de dados, a arquitetura da API foi projetada levando em consideração o *data warehouse* subjacente. Isso inclui a implementação de *indexes* nos modelos de dados para otimizar consultas e o uso de paginação para reduzir a carga em memória durante as requisições GET.

Por fim, foram configuradas as rotas da API utilizando o arquivo urls.py do DRF, o qual mapeia *endpoints* para as *views* correspondentes. Após a conclusão das etapas descritas anteriormente, a API passou a expor métodos HTTP GET, POST e DELETE permitindo a disponibilização e manipulação dos dados do DATASUS.

3.4 Documentação dos dados disponibilizados

Para garantir a compreensão dos dados disponibilizados pela API e facilitar sua utilização por parte dos usuários, foi desenvolvida uma documentação detalhada dos dados disponibilizados. Essa documentação abrange informações essenciais sobre os diferentes conjuntos de dados, incluindo base de dados de origem, nomes das colunas, descrição das colunas, tipos dos dados e frequência de atualização.

A documentação dos dados foi elaborada em formato de texto e json, e organizada de forma clara e intuitiva para permitir análises eficazes e informadas pelos usuários. Tal documentação pode ser encontrada na seção de Anexos do presente documento, e no *endpoint* referenciado na mesma seção. É importante ressaltar que nem todos os dicionários de dados estão presentes na fonte, e consequentemente não estão disponibilizados na API.

3.5 Documentação da API

A documentação da API foi elaborada utilizando o Swagger UI, uma ferramenta que permite aos usuários visualizar e interagir com os recursos da API sem exigir conhecimento detalhado sobre sua implementação. Tal documentação fornece uma visão detalhada das rotas, listando todos os *endpoints* disponíveis juntamente com os respectivos status HTTP de resposta. Para acessar a documentação, basta acessar a rota /swagger.

Devido ao volume de dados permitido via paginação (200.000 linhas), o Swagger pode não ser capaz de lidar com algumas requisições GET. Nesses casos sugere-se utilizar ferramentas como Postman ou Insomnia para realizar tais requisições.

Para garantir a precisão e atualização da documentação, foi aplicado um processo de revisão contínua. Qualquer alteração nos *endpoints* da API foi refletida na documentação correspondente, garantindo que os usuários tenham acesso a informações precisas e atualizadas sobre o funcionamento da API.

3.6 Desenvolvimento de Containers Docker

Considerando que a biblioteca pysus foi desenvolvida para ser executada apenas em ambiente Linux, e com o intuito facilitar a implantação da API, assim como garantir a consistência do ambiente de execução, foi elaborado um arquivo Dockerfile que contém as instruções necessárias para construir a imagem Docker da aplicação.

O Dockerfile utiliza uma imagem base do Python 3.8 e instala o cliente PostgreSQL, bem como define as variáveis de ambiente necessárias para configurar o ambiente Python. Em seguida, o Dockerfile copia o arquivo requirements.txt, responsável por listar todas as dependências da aplicação, e instala essas dependências utilizando o comando pip. Por fim, o Dockerfile copia todos os arquivos do diretório atual para o diretório /app dentro do container e expõe a porta 8000 para que a aplicação Django possa ser acessada externamente.

Adicionalmente, foi desenvolvido um arquivo docker-compose.yml para simplificar o processo de execução da aplicação em um ambiente de desenvolvimento local. O docker-compose.yml define dois serviços: um serviço *web*, que constrói a imagem da aplicação a partir do Dockerfile e executa o servidor de desenvolvimento do Django, e um serviço de banco de dados PostgreSQL. O docker-compose.yaml também define volumes para persistir os dados do banco de dados entre as reinicializações dos containers.

Ao concluir esta etapa, a API tornou-se pronta para ser implantada e disponibilizada aos usuários finais, oferecendo uma solução robusta e escalável para o acesso aos dados do DATASUS. A utilização de containers Docker proporciona uma infraestrutura consistente e flexível, permitindo uma implantação mais eficiente e confiável da aplicação, e garantindo uma análise facilitada e abrangente das informações de saúde disponibilizadas pela API.

4. RESULTADOS E DISCUSSÕES

4.1 Sistema Resultante

A arquitetura da API desenvolvida encontra-se disponível na Figura 1. Quando um *request* é feito a API, o fluxo de processamento é direcionado para um URL *Pattern* específico, que é responsável por mapear o *request* para a *View* correspondente. Se o *request* for do tipo POST, o processo segue para o script de extração, o qual é acionado para buscar e extrair os dados do DATASUS. Em seguida, os dados extraídos são processados pelo *script* de transformação e carregamento, que os converte para um formato adequado e os carrega no SGBD PostgreSQL. Após o processamento ETL referido anteriormente, um response vazio com o código 201 é retornado.

Por outro lado, se o *request* for do tipo GET ou DELETE, o processo é um pouco diferente. Nessas situações, mediante ao mapeamento com os respectivos *models*, os dados são buscados diretamente no PostgreSQL. Em seguida, no caso do request GET, os dados são serializados e retornados no *response* apropriado. Para a requisição DELETE, após a remoção dos dados correspondentes aos parâmetros fornecidos, um *response* vazio é retornado com o código HTTP 204.

Independentemente do tipo de *request*, a API está preparada para lidar com possíveis erros que possam ocorrer durante o processamento. Caso ocorram erros devido a parâmetros inválidos ou problemas internos, uma mensagem de erro informativa é anexada ao *response* da requisição. Essa abordagem é fundamental para garantir uma experiência amigável para os usuários da API, facilitando a identificação e resolução de problemas.

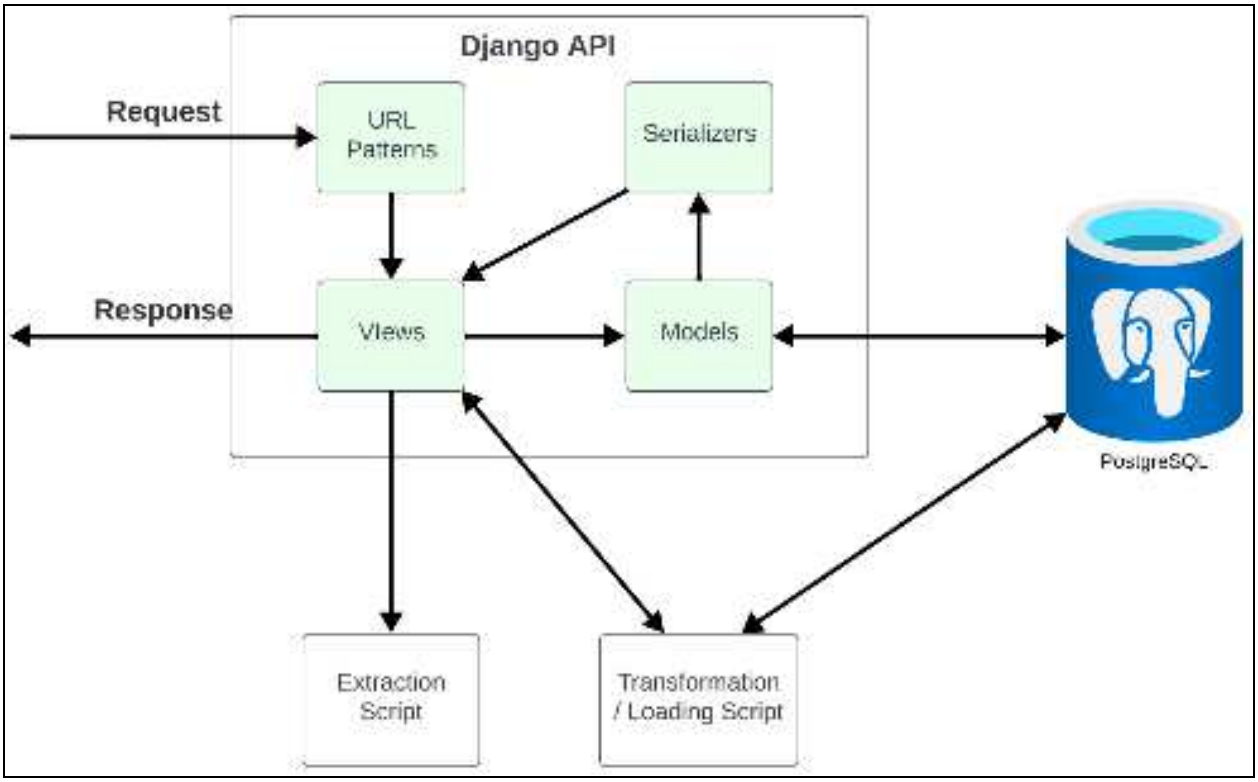

Figura 1. Arquitetura da API que disponibiliza dados do DATASUS.

Em suma, a arquitetura da API implementada demonstra uma abordagem eficiente e escalável para lidar com diferentes tipos de requisições e fluxos de dados. Ao seguir um fluxo de processamento claro e consistente, desde a recepção do request até a entrega do *response* final, a API garante uma manipulação precisa e eficaz dos dados do DATASUS. Além disso, a inclusão de mecanismos de tratamento de erros contribui para a robustez e confiabilidade do sistema, assegurando uma experiência positiva para os usuários finais.

4.2 Acessibilidade e Utilização dos Dados do DATASUS via API

A implementação da API desenvolvida neste trabalho representa um avanço na acessibilidade e utilização dos dados do DATASUS. Conforme pode-se averiguar na Figura 2, e na fonte dos dados, originalmente os dados do DATASUS encontram-se

disponíveis apenas na extensão .dbc, o que os torna de difícil acesso e demanda conhecimento técnico específico para manipulação. No entanto, com o desenvolvimento da API, esses dados tornam-se mais acessíveis aos usuários finais devido a simplificação e acessibilidade atreladas ao sistema desenvolvido.

As etapas de extração, conversão e inserção dos dados no SGBD, são encapsuladas pela API, permitindo que os usuários se concentrem na obtenção e manipulação dos dados, sem a necessidade de lidar diretamente com os detalhes técnicos subjacentes. Isso simplifica significativamente o processo de trabalho com os dados do DATASUS, mesmo para usuários com pouco conhecimento técnico.

Adicionalmente, o grande volume e a complexidade dos dados do DATASUS implica em desafios para sua análise. A API aborda esses desafios através de endpoints bem definidos, que permitem a realização de consultas específicas com paginação, e, consequentemente, sem a necessidade de carregar conjuntos completos de dados em memória. Como exemplo, a Figura 3 demonstra (em Postman) uma requisição que é feita para a API com objetivo de obter os dados de Leitos (LT) para o mês de Janeiro, e estado do Rio Grande do Norte, em relação a todos os anos disponíveis para os dados armazenados no SGBD.

Em resumo, ao simplificar o processo de acesso e manipulação de dados, a API capacita os usuários a realizar análises mais profundas e obter *insights* valiosos para realizar tomadas de decisão, e impulsionar a pesquisa e a inovação no campo da saúde pública.

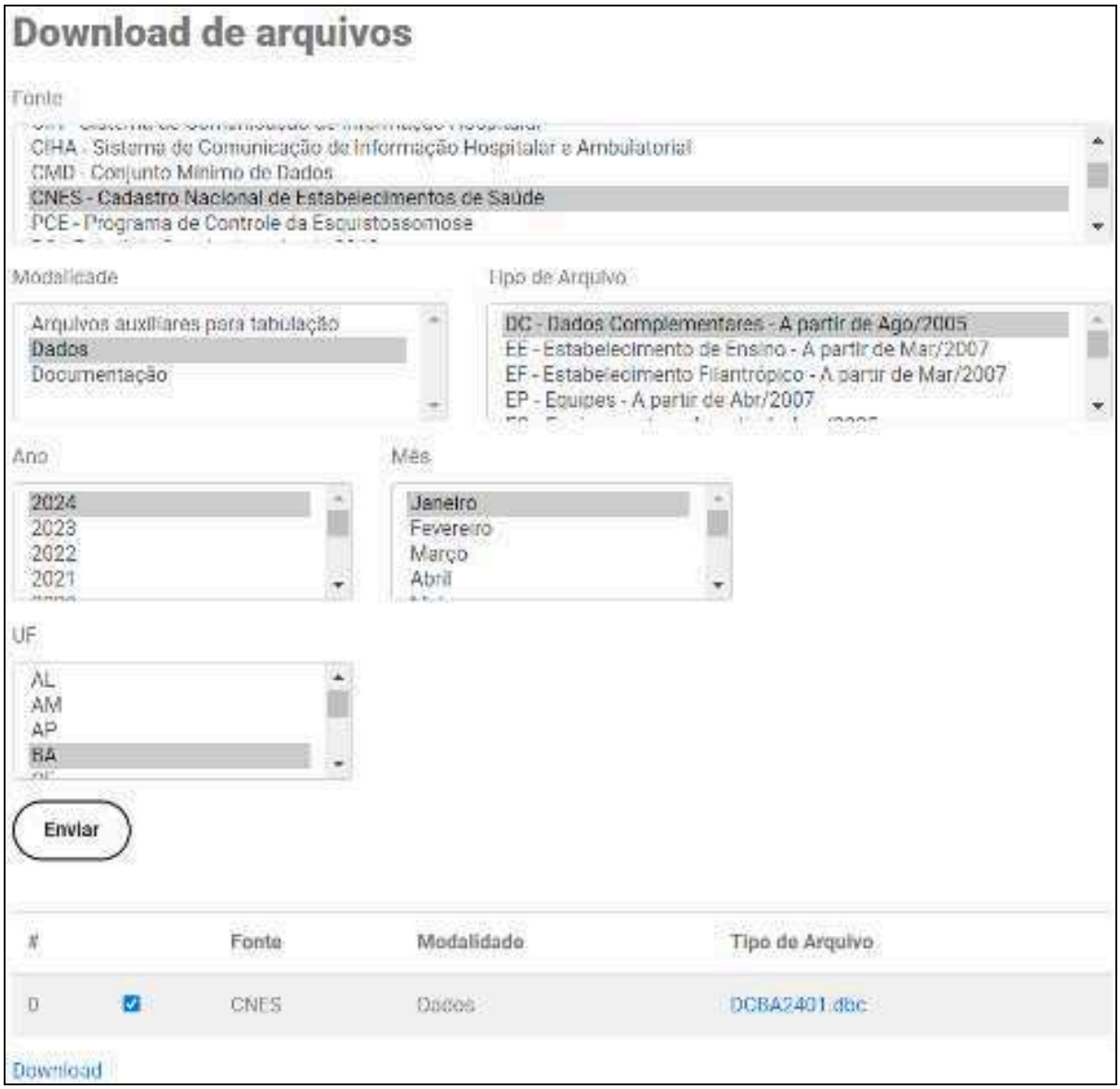

Figura 2. Processo para obtenção do arquivo DCBA2401.dbc do CNES.

Figura 3. Exemplo de requisição para obtenção dos dados do LT do CNES para o mês de Janeiro e estado do Rio Grande do Norte.

4.3 Avaliação de Desempenho

A análise do desempenho da API desenvolvida revelou resultados promissores em relação à capacidade de lidar com o grande volume de dados do DATASUS de forma eficiente e escalável. A implementação de *indexes* e paginação demonstrou-se crucial para otimizar o acesso e manipulação dos dados, permitindo uma resposta rápida mesmo diante de conjuntos de dados extensos.

Além disso, os testes realizados evidenciaram que o uso do Pandas para carregar e manipular os arquivos do DATASUS, antes da inserção no PostgreSQL, é eficaz e não requer a adoção de ferramentas mais complexas, como o PySpark. Devido ao tamanho dos arquivos não ser demasiadamente grande, os resultados do testes de leitura demonstraram que a diferença no tempo de carregamento entre Pandas e PySpark é irrelevante para as necessidades deste projeto.

Contudo, é importante ressaltar que o tamanho de alguns arquivos, como os arquivos SP do SIHSUS e o PA do SIASUS (maior base do DATASUS em termos de espaço requerido para armazenamento), impacta no processo ETL como um todo, e pode implicar em mais tempo para conclusão. A título de exemplificação, a Tabela 1 detalha o volume e o tempo atrelado ao processamento ETL para alguns arquivos do DATASUS. O ambiente utilizado para quantificar o tempo consumido para o processamento de cada arquivo foi um notebook Linux Ubuntu, com 16 GB de memória RAM, 512 GB de armazenamento SSD, e processador Intel® Core™ i7-11390H.

A etapa de extração inclui apenas o *download* dos dados, a etapa de transformação compreende os processos de conversão de .dbc para .csv, ajuste de colunas, geração de colunas adicionais, e transformação do *data frame* em uma lista de tuplas. A etapa de carregamento é composta pela inserção da lista de tuplas no SGBD de destino. A Tabela 1, evidencia que, no geral, o processo de carregamento dos dados consome mais tempo, seguido pela etapa de transformação e por último a extração.

Ainda que alguns arquivos demandem mais tempo de processamento, conforme pode-se constatar ao analisar a Tabela 1, o processo ETL ocorre de forma consistente e sem erros, garantindo a integridade e confiabilidade nos dados manipulados pela API

Os resultados obtidos ressaltam a eficácia e robustez da abordagem adotada na implementação da API, a qual fornece uma infraestrutura sólida para o processamento e disponibilização dos dados do DATASUS, mesmo em face a desafios relacionados ao tamanho e complexidade dos conjuntos de dados.

Nome do arquivo	Número de linhas	Número de colunas	Tempo para Extração (min)	Tempo para Transformação (min)	Tempo para Carregamento (min)	Tempo Total (min)
LTSP2401	8.405	28	0,028	0,007	0,026	0,061
POBR2024	62.981	23	0,041	0,047	0,155	0,243
PFSP2401	1.340.037	40	1,174	1,492	5,141	7,807
SPSP2401	3.604.262	36	2,915	3,791	13,299	20,005
PASP2401a	3.086.333	60	2,080	5,283	13,761	21,124

Tabela 1 - Detalhamento do processamento ETL para arquivos do DATASUS.

5. CONCLUSÃO

Neste trabalho, foi desenvolvida uma API utilizando Django REST *Framework* com o intuito de disponibilizar os dados do DATASUS de forma mais acessível e utilizável. Ao longo do processo de desenvolvimento, diversas etapas foram abordadas, desde a extração até a disponibilização dos dados via rotas da API.

A elaboração da arquitetura da API levou em consideração o *data warehouse* subjacente, garantindo uma estrutura eficiente para lidar com o grande volume de dados e otimizar consultas ao SGBD PostgreSQL.

Com o objetivo de facilitar a usabilidade da API, elaborou-se uma documentação, com o Swagger UI, que proporciona aos usuários uma visão clara dos recursos disponíveis, facilitando a interação e compreensão dos *endpoints* e parâmetros necessários para cada requisição. Por meio do processo de revisão contínua, foi garantida a precisão e atualização da documentação, refletindo qualquer alteração nos *endpoints* da API.

A implementação desta API, possibilita que pesquisadores, profissionais de saúde e desenvolvedores possam utilizar essas informações de forma mais eficiente em seus projetos e análises.

Em suma, este trabalho contribui para a democratização do acesso aos dados de saúde pública no Brasil, fornecendo uma ferramenta robusta e bem documentada para a obtenção e manipulação de informações do DATASUS.

6. SUGESTÕES PARA TRABALHOS FUTUROS

Para garantir a evolução contínua do sistema, algumas melhorias podem ser implementadas. Primeiramente, é importante avaliar a necessidade de aprimoramento da performance do sistema, e caso positivo, refatorar partes do código para lidar de forma mais eficiente com o grande volume de dados. Em segundo lugar, a implementação de testes automatizados é essencial para garantir a estabilidade e a qualidade do sistema ao longo do tempo, permitindo a detecção precoce de possíveis problemas e facilitando a manutenção do código. Por fim, sugere-se a implementação do suporte a múltiplos bancos de dados, ampliando a compatibilidade do sistema, e permitindo que este seja utilizado em diferentes ambientes e cenários, além do PostgreSQL. Tais melhorias podem contribuir significativamente para a eficiência, confiabilidade e adaptabilidade do sistema.

7. ANEXOS

Fonte dos dados

Os dados disponibilizados pelo sistema desenvolvido neste trabalho podem ser originalmente encontrados na seguinte url: https://datasus.saude.gov.br/transferencia-de-arquivos/

Código do sistema desenvolvido

O código fonte do sistema desenvolvido encontra-se no seguinte repositório:

https://github.com/icaro-chagas/datasus-api

Documentação dos dados do DATASUS

Esta seção fornece um detalhamento das bases de dados do DATASUS, especificando tabelas e frequência de atualização. Todo o detalhamento acerca de nomes de colunas, descrição de colunas, e tipos de dados é disponibilizado via um *endpoint* da API. A estrutura da requisição para obter os dicionários de dados é a seguinte: /datasus-doc?db_name={param1}&table_name={param2}. Por exemplo, para obtenção do dicionário de dados da tabela LT do CNES, a requisição deve ser feita desta forma: /datasus-doc?db_name=CNES&table_name=LT

CIH - Sistema de Comunicação de Informação Hospitalar

Esta base de dados possui uma única tabela denominada CR (Comunicação de Internação Hospitalar), e a frequência de atualização dos dados é mensal.

CIHA - Comunicação de Informação de Hospitalar e Ambulatorial

Esta base de dados possui uma única tabela de mesmo nome, e a frequência de atualização dos dados é mensal.

CNES - Cadastro Nacional de Estabelecimentos do SUS.

As informações desta base de dados referem-se aos dados cadastrais dos estabelecimentos de saúde cadastrados no Sistema de Cadastro Nacional de Estabelecimentos do SUS. Os dados do CNES possuem frequência de atualização mensal. As tabelas disponíveis para o CNES são as seguintes: DC (Dados Complementares), EE (Estabelecimento de Ensino), EF (Estabelecimento Filantrópico), EP (Equipes), EQ (Equipamentos), GM (Gestão e Metas), HB (Habilitação), IN

(Incentivos), LT (Leitos), PF (Profissional), RC (Regra Contratual), SR (Serviço Especializado).

PCE - Programa de Controle da Esquistossomose

Esta base de dados possui uma única tabela de mesmo nome, e a frequência de atualização dos dados é anual.

PO - Painel de Oncologia

Esta base de dados possui uma única tabela de mesmo nome, e a frequência de atualização dos dados é anual.

RESP - Notificações de casos suspeitos de SCZ

Esta base de dados possui uma única tabela de mesmo nome, e a frequência de atualização dos dados é anual.

SIASUS - Sistema de Informações Ambulatoriais do SUS

Os dados do SIASUS possuem frequência de atualização mensal. A maior parte das tabelas desta base estão vinculadas a Autorização de Procedimento Ambulatorial (Alta complexidade/custo) - APAC. As tabelas disponíveis são as seguintes: ABO (APAC Acompanhamento Pós Cirurgia Bariátrica), ACF (APAC Confecção de Fístula Arteriovenosa), AD (APAC de Laudos Diversos), AM (APAC de Medicamentos), AN (APAC de Nefrologia), AQ (APAC de Quimioterapia), AR (APAC de Radioterapia), ATD (APAC Tratamento Dialítico), PA (Produção Ambulatorial), PS (Psicossocial) e SAD (Atenção Domiciliar).

SIHSUS - Sistema de Informações Ambulatoriais do SUS

Os dados do SIHSUS possuem frequência de atualização mensal. As tabelas disponíveis para esta base de dados são as seguintes: ER (AIH Rejeitadas com código de erro), RD (AIH Reduzida), RJ (AIH Rejeitadas), SP (Serviços Profissionais).

SIM - Sistema de informações de Mortalidade

Os dados do SIM possuem frequência de atualização anual. As tabelas disponíveis para esta base de dados são as seguintes: DO (Declarações de Óbito), DOEXT (Declarações de Óbitos por causas externas), DOFET (Declarações de Óbitos fetais), DOINF (Declarações de Óbitos infantis), DOMAT (Declarações de Óbitos maternos), DOREXT (Mortalidade de residentes no exterior).

SINASC - Sistema de informação de Nascidos Vivos

Os dados do SINASC possuem frequência de atualização anual. As tabelas disponíveis para esta base de dados são as seguintes: DN (Declarações de nascidos vivos), DNEX (Declarações de nascidos vivos residentes no exterior).

SISCOLO - Sistema de Informações de Cânceres de Colo de Útero

Os dados do SISCOLO possuem frequência de atualização mensal. As tabelas disponíveis para esta base de dados são as seguintes: CC (Citopatológico de Colo de Útero), HC (Histopatológico de Colo de Útero).

SISMAMA - Sistema de Informações de Cânceres de Mama

Os dados do SISMAMA possuem frequência de atualização mensal. As tabelas disponíveis para esta base de dados são as seguintes: CM (Citopatológico de Mama), HC (Histopatológico de Mama).

SISPRENATAL - Sistema de Monitoramento e Avaliação do Pré-Natal, Parto, Puerpério e Criança

Esta base de dados possui uma única tabela denominada PN (Pré-Natal), e a frequência de atualização dos dados é mensal.

REFERÊNCIAS

[1] Kimball, R., Ross, M. The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. 3. Ed. John Wiley & Sons, 2013..

[2] Flesca, S., Greco, S., Masciari, E., Saccà, D. A Comprehensive Guide Through the Italian Database Research Over the Last 25 Years. 1. Ed. Springer, 2013.

[3] Krishnan, K. Data Warehousing in the Age of Big Data. 1. Ed. Morgan Kaufmann, 2013.

[4] Mayer-Schönberger, V., Cukier, K. Big Data: A Revolution That Will Transform How We Live, Work, and Think. 2. Ed. Harper Business, 2014.

[5] Gandomi, A., Haider, M. Beyond the hype: Big data concepts, methods, and analytics. International Journal of Information Management, v. 35, p. 137-144, 2015.

[6] Zikopoulos, P., Eaton, C. Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data. 1. Ed. McGraw-Hill Osborne Media, 2012.

[7] Bass, L., Clements, P., Kazman, R. Software Architecture in Practice. 3. Ed. Addison-Wesley Professional, 2012