SisCoV-BR: Sistema de informação de Casos de COVID-19 no Brasil

Daniel de Oliveira Gonzaga[1], **Luikson da Silva Ferreira**[1], **Kele Teixeira Belloze**[1]

[1]Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET/RJ)
Rio de Janeiro/RJ - Brasil

{daniel.gonzaga,luikson.silva}@aluno.cefet-rj.br

kele.belloze@cefet-rj.br

Abstract. *The world faces a pandemic caused by a virus called SARS-CoV-2, which causes COVID-19, an infectious disease that has already infected 237 million people worldwide, killing nearly 4.8 million patients in October 2021. Variants of the new coronavirus have been detected in different places around the world, being an aggravating factor for the pandemics and the effectiveness of vaccines. Considering the high relevance of the subject today, notifications systems that take into account cases of COVID-19 are being developed to disseminate information to the population. This paper aims at developing a web application called SisCoV-BR. This system numerically presents all the registered cases of infection by a coronavirus in Brazil using an interactive map. This map displays an alert that warns users about regions affected by new coronavirus variants.*

Resumo. *O mundo está enfrentando uma pandemia de um novo vírus chamado SARS-CoV-2, causador da doença infecciosa COVID-19, que já vitimou cerca de 237 milhões de pessoas, sendo aproximadamente 4,8 milhões de casos fatais em outubro de 2021. Variantes do novo Coronavírus foram detectadas em diferentes locais no mundo, sendo este um fator agravante para a pandemia e para a efetividade das vacinas. Devido a alta relevância do assunto atualmente, sistemas de notificação de casos de Coronavírus têm sido desenvolvidos de modo a disseminar informações para a população. Este trabalho visa o desenvolvimento de uma aplicação web chamada SisCoV-BR, que apresenta numericamente os casos registrados de infecção pelo novo Coronavírus no Brasil através de um mapa interativo, e exibe alertas que mostram ao usuário as regiões infectadas por novas variantes do vírus.*

1. Introdução

No final de 2019, o escritório da Organização Mundial de Saúde (OMS) na China recebeu notificações sobre uma infecção pneumonológica de etiologia desconhecida primeiramente detectada em Wuhan, província de Hubei [OMS 2020], sendo este o epicentro inicial da doença que viria a ser identificada como COVID-19, causada pelo vírus SARS-CoV-2. Desde sua descoberta, o vírus se espalhou pelo mundo até ser classificado como uma pandemia em março de 2020 [Ministério da Saúde 2020], perdurando até o momento em que este trabalho é redigido. No mundo já são contabilizados mais de 237 milhões de casos e 4,8 milhões de mortes [Organization et al. 2021], e no Brasil, mais de 21 milhões

de casos e mais de 600 mil mortes [Ministério da Saúde 2021]. Dada a sua elevada taxa de transmissão associada ao grande fluxo de pessoas no mundo, a doença prova ser uma verdadeira ameaça nos tempos atuais.

Sendo um vírus, o SARS-CoV-2 possui uma alta capacidade adaptativa, podendo realizar mutações em meio a seu processo de replicação após infectar as células de um hospedeiro. Neste contexto, faz-se importante oferecer à população soluções capazes de fornecer informações de maneira amigável para o acompanhamento da disseminação do vírus. Sendo assim, este estudo apresenta o desenvolvimento de uma aplicação web capaz de divulgar, de forma interativa ao usuário, informações a respeito da quantidade de casos de COVID-19 pelo território brasileiro, e quais locais estão sendo afetados por variantes do vírus.

Este trabalho está organizado da seguinte maneira: a seção 2 apresenta a metodologia e as tecnologias utilizadas. A seção 3 apresenta a extração e a manipulação dos dados. As seções 4, 5 e 6 descrevem o banco de dados, o *Back End* e o *Front End* da aplicação, respectivamente. A seção 7 apresenta os resultados e discussão. Finalmente, na seção 8 são apresentadas as conclusões e trabalhos futuros.

2. Metodologia e Definição das Tecnologias

A metodologia aplicada no trabalho para o desenvolvimento do SisCoV-BR foi composta por, primeiramente, um profundo estudo acerca do novo Coronavírus e, principalmente, de suas variantes. Essa pesquisa inicial teve como enfoque a ambientação com o tema, visando melhor avaliação crítica sobre a qualidade e coleta dos dados públicos disponíveis sobre o Coronavírus e quais seriam relevantes exibir. O passo seguinte focou na extração e manipulação de dados referentes ao vírus, disponibilizados por fontes confiáveis. Na sequência, foi realizado o projeto do banco de dados, além da construção de *wireframes* e protótipos que auxiliaram na produção do *Front End* da aplicação.

Para a implementação da aplicação foram definidas as seguintes tecnologias, levando em conta trabalhos relacionados e a experiência pessoal dos autores: Angular para o desenvolvimento *Front end*, por ser um *framework* do Javascript que facilitaria o desenvolvimento, Python para o tratamento de dados por se tratar de uma linguagem prática, além de ser amplamente utilizada em tarefas de extração e manipulação de dados, PostgreSQL para o sistema gerenciador de banco de dados, e Flask para o *Back end*, que é uma biblioteca do próprio Python, facilitando assim com a criação de *endpoints* que são consumidos pelo *Front end*. A Figura 1 ilustra a visão geral do funcionamento do sistema.

3. Extração e manipulação de dados

Os dados utilizados neste trabalho foram extraídos de *datasets* obtidos nas seguintes fontes: openDATASUS [openDataSus 2021] para dados do Coronavírus e IBGE [IBGE 2021] para dados relativos aos municípios. Sendo, respectivamente, 26 estados mais o Distrito Federal, 5570 municípios e dezenas de milhões de casos colhidos. Ambos os *datasets* são disponibilizados publicamente no formato comma-separated-values (CSV).

Os dados do Coronavírus variam desde a data de notificação e os sintomas apresentados, até o estado e o município de notificação, além de seus respectivos códigos no

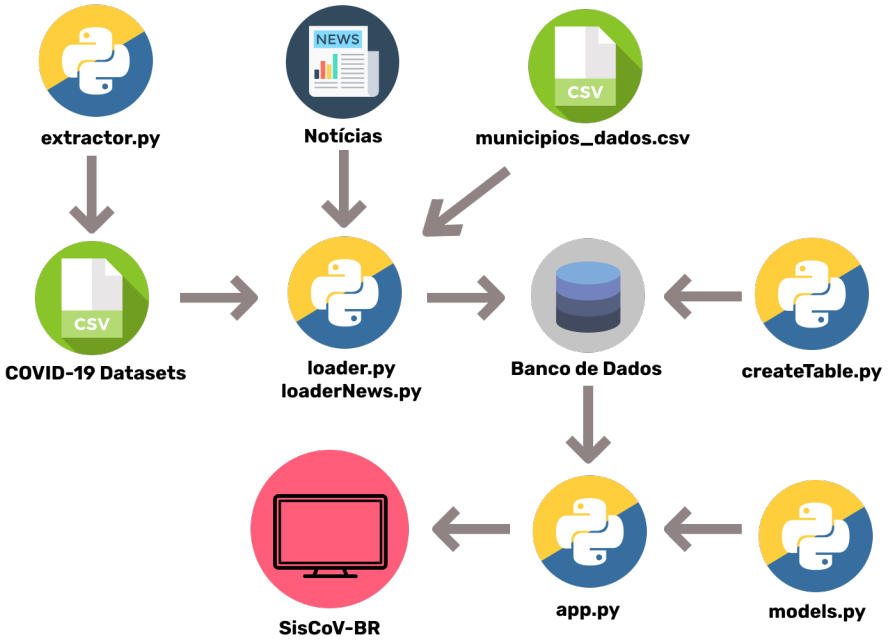

Figura 1. Visão geral do sistema.

IBGE, totalizando 28 colunas. Devido ao elevado número de dados presentes no CSV, optou-se por reduzir a dimensão das informações para somente aquelas necessárias à aplicação. Dessa forma, foram extraídas apenas as informações relevantes para o sistema proposto, que são: identificador, data da notificação, data de início dos sintomas, estado, município da notificação, evolução do caso, e classificação final do caso, totalizando sete colunas das 28 originais.

Na sequência, os dados são inseridos no banco de dados automaticamente através de um *script* desenvolvido para esta finalidade. Os identificadores das tabelas originais do openDATASUS são reutilizados para evitar reinserções de dados já presentes no banco, facilitando o processo de atualização dos casos, que ocorrerá rotineiramente. Em resumo, um processo de ETL.

O ETL (*Extraction-Transformation-Loading*) é um processo de extração, transformação e carregamento de dados de múltiplas fontes, que garante que os dados extraídos de uma base de dados sejam processados, manipulados e, posteriormente, inseridos em outra base de dados.

Um aspecto importante refere-se à ausência (até este momento) de um *dataset* oficial e confiável com relação às variantes do vírus. Desta forma, para abordar esse aspecto que é de suma importância para o sistema, optou-se por criar uma tabela de notícias no banco de dados. A tabela é populada por notícias que identificam áreas que tenham casos de variantes, sejam elas Regiões, estados ou municípios.

As notícias, nesta primeira versão do sistema, são colhidas manualmente em fontes oficiais como o Ministério da Saúde e as Secretarias Estaduais, e em agregadores de notícias confiáveis para o Coronavírus, como o G1[1]. Os *links* dessas notícias são armazenados em um arquivo e posteriormente inseridos via *script* no banco, com as notícias

[1]https://g1.globo.com/saude/coronavirus/

sendo utilizadas para notificar um local que possua casos de variantes. Essa abordagem contorna a escassez de dados referentes às cepas do vírus e indica que variantes estão presentes e ativas em uma região, estado ou município.

4. Implementação do Banco de Dados

O banco de dados criado denonima-se "casos_covid" e seu modelo conceitual é apresentado na Figura 2. A partir desse modelo foram implementadas as seguintes tabelas: `Casos`, responsável por armazenar as informações referentes aos casos de Coronavírus em território nacional; `Municipio`, responsável por fazer a associação entre municípios e casos; `Estado` que se relaciona com a tabela Municipio; `Notícias`, que associa as notícias com relação às variantes com seus respectivos estados e municípios; e as tabelas de associação `noticias_estado` e `noticias_municipio`, que representam o relacionamento muitos para muitos da tabela de notícias com a de estado e da tabela de notícias com a de município, respectivamente.

Com relação às colunas, a tabela `Casos` possui os atributos: "id" (chave primária para identificação do caso), "dataNotificacao" (a data em que foi notificado o caso aos órgãos responsáveis), "dataInicioSintomas" (a data em que o paciente começou a apresentar os sintomas), "evolucaoCaso" (informa se o caso foi cancelado, se o paciente está em tratamento, se foi curado, ou se veio a óbito), "classificacaoFinal" (informa se o paciente era realmente um caso de COVID-19, ou se foi um caso descartado) e "municipio_id", uma chave estrangeira que faz referência à chave primária da tabela município.

A tabela `Estado` possui como colunas seu atributo identificador e o nome do estado em questão. A tabela `Municipio`, de forma análoga, também possui o identificador e o nome do município, com a adição do atributo "estado_id", uma chave estrangeira que relaciona o município com o estado que ele pertence. Por fim, a tabela `Noticias` possui seu identificador, além da URL da notícia em questão.

Todo o processo de criação das tabelas e inserção de valores nas colunas é automatizado através de *scripts* Python. Esse processo será descrito com mais detalhes na seção 5, referente ao processo de *Back End* da aplicação.

5. *Back End*

A definição das tabelas é feita no *script* de modelos *models.py*, desde os atributos até as chaves identificadoras e estrangeiras e o relacionamento entre as tabelas. Nesse mesmo *script* é feita a conexão entre o PostgreSQL e o servidor Flask. A criação de tabelas no banco é feita por meio do *script* de criação de tabelas *createTables.py* que, por sua vez, importa as definições das tabelas e a conexão com o banco do *models.py*. Como resultado, as tabelas são criadas via código e é possível manipulá-las como um objeto, permitindo assim fazer inserções e consultas no banco no próprio código.

Para a extração de dados, inicialmente, foi criado um *script* que faz tanto a extração quanto a manipulação dos dados. Feito o processo de leitura e filtragem dos dados, é realizada a inserção no banco. Para a inserção dos dados nas tabelas e posterior consulta dos mesmos, são utilizadas três bibliotecas ao todo, sendo elas: SQLAlchemy, Flask e a *psycopg2*.

Essa inserção no banco é feita pelo *script loader.py*, da seguinte forma: é lido um arquivo CSV, que possui todos os municípios do país, seu estado e sua população,

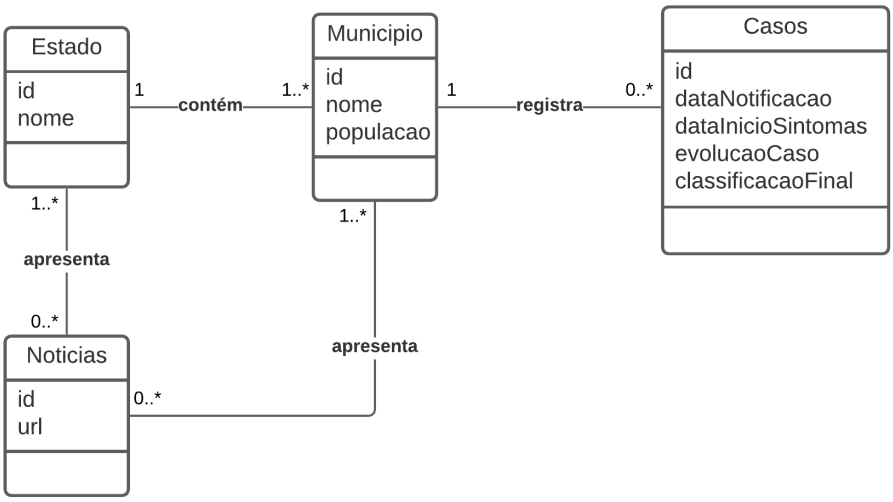

Figura 2. Modelo Conceitual do Banco de Dados do SisCoV-BR

montado com base nas informações do IBGE [IBGE 2021]. Iterando esse arquivo são inseridos os municípios e estados em suas respectivas tabelas, fazendo as devidas referências aos relacionamentos entre tabelas no processo. Para a tabela de `Casos`, por se tratar de um volume massivo de dados, o processo de inserção é relativamente demorado. Visando mitigar essa situação, é feita uma cópia do arquivo CSV transformado de cada estado para uma tabela temporária no banco. Posteriormente, são realizadas as inserções na tabela de `Casos` utilizando os valores dessa tabela temporária, fazendo as devidas junções necessárias para inserção das referências. Ao fim do processo, a tabela temporária é excluída.

Há também a inserção de notícias que, analogamente às demais tabelas, é declarada no *script* de modelos. Foi criado um arquivo local chamado *news.py*, com notícias para cada estado, pesquisadas em fontes governamentais de cada um dos estados e no G1. Esse arquivo possui informações como o estado ao qual a notícia pertence, sua URL e os municípios que possuem casos registrados daquela notícia em questão. A inserção dessas notícias é feita pelo *script newsLoader.py*, que lê o arquivo de notícias e, para cada uma, verifica se ela existe no banco, e em caso negativo, insere a mesma na tabela de `Noticias`, assim como as referências das tabelas associativas. Por fim, é executado o servidor do Flask através do arquivo *app.py*, que possui as rotas que serão utilizadas para as consultas e o retorno dos dados.

Com o uso do *framework* SQLAlchemy, são feitas as consultas correspondentes, utilizando os identificadores que são passados na consulta do *endpoint*. A resposta do servidor é um objeto em JSON, com os dados provindos do banco. Dependendo da consulta, o processo de selecionar os dados no banco e retornar como resposta da rota levava um tempo considerável, para isso adotou-se uma política de cache local. Um arquivo de cache executa as consultas previamente e armazena os resultados localmente. Quando algumas consultas são realizadas, ele verifica se já existe o arquivo local e reaproveita o resultado, caso contrário ele realiza a consulta e cria o arquivo de cache com a resposta da consulta. Essa estratégia do cache só foi possível de ser adotada pois não há inserção

ou atualização do banco via *Front End* da aplicação, ou seja, o usuário só faz requisições *GET* para consultar o banco através dos *endpoints*. Assim sendo, os arquivos de cache podem ser atualizados a cada atualização dos CSV da aplicação.

6. *Front End*

Antes de iniciar o desenvolvimento *Front End* da aplicação, foi construído um protótipo, como mostrado na Figura 3, utilizando a aplicação *Figma*[2]. Este protótipo possibilitou um desenvolvimento mais fluido da aplicação e alguns ajustes em relação aos conceitos inicialmente levantados.

A tela é dividida em duas seções: na primeira metade da tela há a exibição do mapa interativo, onde o usuário pode interagir clicando nos locais do mapa; na segunda metade, há a listagem dos locais que estão sendo representados no mapa. Cada local listado, ao clicar, se expande e mostra as informações do local (população total, casos totais e número de mortes). Nessa seção expandida, também pode ser exibido um botão clicável caso o local esteja sendo afetado por variantes da COVID-19, que traz para o usuário um modal com notícias de portais oficiais ou confiáveis em que comprovam a ocorrência de casos de variantes no local. Caso o mapa esteja em um modo de visualização que não seja o de municípios, aparecerá também um botão para visualizar um mapa mais específico, isto é, caso a visualização vigente seja a de Regiões do Brasil, qualquer local expandido exibirá um botão para visualizar o mapa de estados do Brasil. Caso o mapa já esteja mostrando os estados, qualquer estado expandido na listagem mostrará um botão para visualizar o mapa daquele estado em questão, com cada um de seus municípios.

Há uma interação completa entre o mapa e a listagem de regiões. Se o usuário clicar em uma região do mapa, este local é expandido na lista, e a região no mapa fica selecionada. Caso o local seja expandido através de um clique na listagem, o local no mapa é selecionado da mesma forma. A listagem foi implementada pensando na usabilidade da aplicação para mapas que contenham muitos municípios, pois seria difícil a utilização da aplicação, já que o usuário seria obrigado a saber o posicionamento geográfico do município que quisesse consultar, além de exigir precisão para posicionar o cursor do *mouse* em municípios muito pequenos.

A concepção deste protótipo apoiou na definição da tecnologia a ser utilizada para a exibição dos mapas. Restrições com o uso de API de terceiros, como o *Google Maps* e outras vistas nos trabalhos relacionados estudados, principalmente por terem limitações de uso em suas versões gratuitas, fez com que a estratégia utilizada fosse a de mapas vetorizados no formato SVG. O completo suporte do *Figma* a este tipo de imagem facilitou a criação de cada um dos mapas neste formato, sendo de fácil exportação para a utilização no código HTML da página.

Para o desenvolvimento da aplicação em si, foi utilizado o framework Angular[3]. Também foi utilizado o Angular CLI[4], que facilita o desenvolvimento com seus comandos para a criação automática de componentes, módulos e serviços, além de possibilitar subir a aplicação localmente de maneira mais simples. Durante o desenvolvimento do

[2]Página oficial do Software: https://www.figma.com/
[3]https://angular.io/
[4]https://angular.io/cli

projeto, foram utilizadas bibliotecas complementares que auxiliaram na criação de funcionalidades do sistema: o PrimeNG, que é uma biblioteca de componentes para o Angular, e o *lodash*, que é uma biblioteca com diversos algoritmos úteis para o desenvolvimento, como o de busca em listas e os de formatação de *strings*.

7. Discussão e Resultados

A aplicação alcançou resultados satisfatórios de funcionamento. Todos os artefatos produzidos para a aplicação estão disponíveis em repositório do Github[5]. A modelagem do banco de dados foi suficiente para realizar as consultas necessárias para o funcionamento da aplicação. O arquivo *extractor.py* faz o *download* de todos os *datasets* utilizados sem erros, além do tratamento necessário, e o arquivo *loader.py* é capaz de popular o banco corretamente. Todavia, a escolha da biblioteca SQLAlchemy, que utiliza o padrão ORM, mostrou-se não recomendada para a inserção de grandes volumes de dados. Para contornar esse problema, optou-se pela estratégia de copiar o CSV completo para o banco de dados em uma tabela temporária, e posteriormente realizar a inserção na tabela `Casos`.

Ademais, será necessária uma revisão sobre métodos para melhorar o desempenho das consultas. A demora na resposta em consultas que deveriam trazer dados de múltiplos estados ou municípios se mostrou um empecilho, uma vez que o tempo de resposta para o usuário seria muito lento. A alternativa encontrada foi realizar as consultas mais demoradas previamente, antes de levantar a aplicação *Front End* e colocar o resultado em cache, que é um arquivo no formato JSON, escrito localmente. Quando o *endpoint* for chamado, ele irá verificar se consulta está em cache, e em caso positivo ele utiliza o arquivo local, retornando o resultado em um tempo muito mais aceitável para o usuário.

O arquivo *app.py*, que é o principal arquivo de funcionamento do Flask, possui todas as consultas funcionais e se comunicando corretamente com o *Front End*. Inicialmente, a chamada aos *endpoints* pela aplicação Angular apresentava uma demora considerável por conta da serialização e desserialização que o Flask realiza ao processar as consultas. Contudo, após a solução da utilização de cache para as consultas com maior volume de dados, houve uma melhora significativa no desempenho. Sendo assim, todas as telas inicialmente planejadas estão funcionais, listando todas as Regiões, estados ou municípios, pintadas de acordo com a métrica utilizada para o cálculo das cores, mostrando as informações necessárias e as notícias, caso existam. A Figura 3 mostra a aplicação no modo de visualização de mapa de estados do Brasil.

8. Considerações Finais

Neste trabalho foi desenvolvido um sistema de informações a respeito do novo Coronavírus, apresentando sua propagação, número de casos e variantes através de mapas interativos. Foi desenvolvido o *Back End* da aplicação, com a extração, tratamento e inserção dos *datasets* utilizados, alimentando a aplicação através dos *endpoints* criados com o Flask. Também foi desenvolvida a estratégia do fornecimento de notícias de variantes para os locais atingidos, resolvendo o problema da falta de dados específicos sobre as variantes do vírus.

Destaca-se os trabalhos futuros, que englobam, em curto prazo, a disponibilização pública do sistema, além da avaliação experimental da usabilidade utilizando *System Usa-*

[5]https://github.com/danielgonzaga/siscov-br

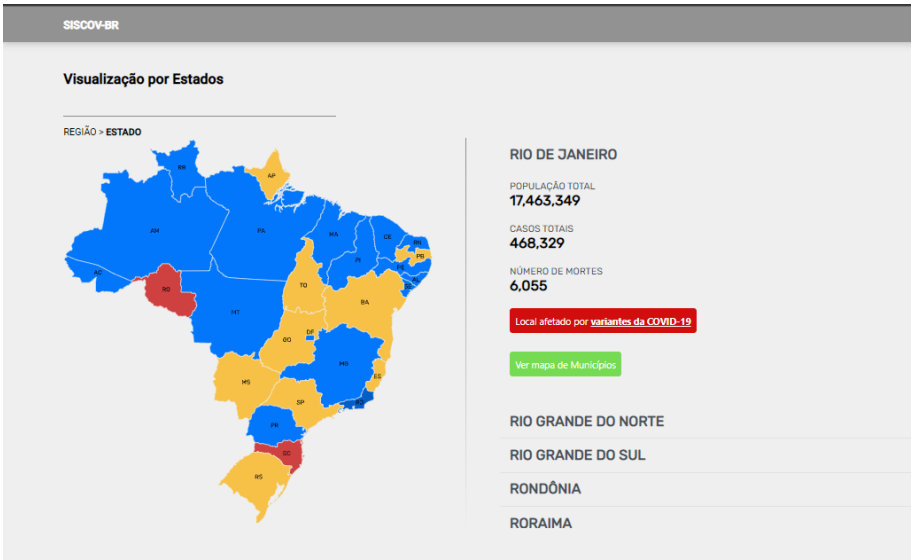

Figura 3. Aplicação apresentada na visualização de mapa de estados do Brasil.

bility Scale (SUS). Outros trabalhos futuros compreendem: extração e utilização dos *datasets* específicos de cada estado como Rio de Janeiro e Rio Grande do Sul, que se mostram mais atualizados em relação do *dataset* do openDATASUS; criação de *bots* para verificar a atualização dos arquivos CSVs nas fontes de origem e posterior atualização dos dados na aplicação e; extração automática das notícias sobre as variantes.

Referências

IBGE (2021). ESTIMATIVAS DA POPULAÇÃO RESIDENTE NO BRASIL. Acessado em 10/11/2021 de `https://ftp.ibge.gov.br/Estimativas_de_Populacao/Estimativas_2021/estimativa_dou_2021.pdf`.

Ministério da Saúde (2020). Coronavirus classificado como pandemia. `https://www.gov.br/pt-br/noticias/saude-e-vigilancia-sanitaria/2020/03/oms-classifica-coronavirus-como-pandemia`. Acessado em 28/02/2021.

Ministério da Saúde (2021). Painel coronavírus. `https://covid.saude.gov.br/`. Acessado em 12/05/2021.

OMS (2020). Pneumonia of unknown cause – china. `https://www.who.int/emergencies/disease-outbreak-news/item/2020-DON229`. Acessado em 27/02/2021.

openDataSus (2021). Notificações de Síndrome Gripal. Acessado em 12/05/2021 de `https://opendatasus.saude.gov.br/gl/dataset/casos-nacionais`.

Organization, W. H. et al. (2021). Covid-19 weekly epidemiological update, 13 october 2021.