Prêmio MapBiomas 2023

Categoria Políticas Públicas

GEOBNDES: INTELIGÊNCIA TERRITORIAL E CRÉDITO SUSTENTÁVEL

Carlos Vinícius de Castro

Eva Khury

Gabriela Nogueira

Gumersindo Sueiro

Lucas Linhares

Luís Henrique Rosati Rocha

Rio de Janeiro, abril de 2023

1.INTRODUÇÃO

O Banco Nacional de Desenvolvimento Econômico e Social – BNDES tem por propósito promover o desenvolvimento sustentável brasileiro. No âmbito das iniciativas e políticas públicas de financiamento levadas a efeito pelo Banco, busca-se avançar na consolidação de um arcabouço de inteligência territorial para dar suporte informacional à formulação de estratégias de fomento e processos operacionais, impulsionando dessa forma a incorporação de análises socioambientais no fluxo de concessão de crédito.

O GEOBNDES consiste em uma unidade de suporte ao negócio, vinculada atualmente ao Departamento de Estratégia Social e Territorial da Área de Desenvolvimento Social e Gestão Pública do BNDES. Conta com equipe enxuta e multidisciplinar, composta por 6 pessoas: 1 gerente economista, 1 coordenador de geoprocessamento, 1 engenheiro de dados, 2 técnicos de análise de dados e suporte informacional e 1 estagiária de engenharia ambiental. Os trabalhos são realizados de forma matricial, buscando auxiliar as áreas operacionais a construir respostas e soluções para questões de negócio associadas ao impacto territorial e socioambiental de projetos que pleiteiam o apoio financeiro do Banco.

Em linhas gerais, o arcabouço de inteligência territorial GEOBNDES tem por atribuição coletar, processar, consolidar e integrar bases de dados de interesse, permitindo a construção de visões analíticas que possibilitem avaliar aspectos espaciais e mitigar riscos socioambientais potenciais de projetos.

Mais especificamente, descrevendo preliminarmente o escopo do projeto ora candidato ao Prêmio MapBiomas 2023, foi estruturada pelo núcleo GEOBNDES uma plataforma integrada de dados geoespaciais, que une, de um lado, o armazenamento em nuvem de grandes volumes de dados públicos de fontes diversas, repositório que denominamos “Cosmobase Pública GEOBNDES” e, por outro lado, o desenvolvimento de um sistema de informações geográficas que permite realizar consultas a territórios e áreas específicos a partir de coordenadas georreferenciadas ou ainda códigos do Cadastro Ambiental Rural relativos às propriedades, cruzando essas áreas com fontes de informações do MapBiomas para detectar indícios de desmatamento ilegal, perfil do uso do solo, além de cruzar com outras camadas socioambientais, por exemplo, unidades de conservação, áreas úmidas, terras indígenas e quilombolas.

Dessa forma, a plataforma GEOBNDES consubstancia ferramentas de análise de dados geoespaciais que permitem ao Banco um olhar mais acurado sobre os impactos territoriais do apoio financeiro, qualificando a sustentabilidade das suas ações.

2. GEOBNDES: COSMOBASE PÚBLICA, SICAR E MAPBIOMAS

O Brasil conta atualmente de ampla gama de dados públicos disponíveis, oferecendo aos pesquisadores e formuladores de políticas públicas matéria-prima informacional de grande alcance e utilidade. Todavia, usualmente trabalhar com esses dados, principalmente com microdados, não é tarefa simples e traz alguns desafios importantes. Algumas bases possuem tamanhos muito grandes, outras são de difícil obtenção e consolidação; ou ainda necessitam de limpeza ou tratamento; outras variam periodicamente em formato e número de colunas/variáveis; algumas trazem obstáculos que dificultam a sua coleta, tais como *captchas*, ou necessitam de intensa interação humana com o site para a obtenção de pequenas partições da base de cada vez.

No caso específico de bases de dados de natureza espacial, são muitas vezes apresentados sob a forma de imagens raster GeoTIFF, que necessitam de programas especializados para a sua leitura ou mesmo scripts utilizando-se de linguagens de programação para a realização de análises.

A possibilidade de armazenar essas bases em um mesmo repositório com formato compatível tornaria possível realizar de forma funcional cruzamentos entre diferentes conjuntos de dados de interesse, favorecendo a organização de informações importantes para orientar políticas públicas. Com o surgimento recente de novas tecnologias e funcionalidades, particularmente ferramentas de armazenagem cloud e processamento de grandes bancos de dados de altíssima performance como o Google Bigquery, foi possível iniciar o projeto de construção de um *data lake* apto a incorporar toda forma de informação pública disponível no Brasil, mediante a identificação de fontes e construção de scripts de captura, regularização e atualização desses dados.

Esse projeto foi denominado Cosmobase Pública GEOBNDES e, ao longo dos últimos anos, incorporou um grande conjunto de bases de dados, dentre as quais podemos citar IBGE, Portal da Transparência, Receita Federal, STN, BACEN, Datasus (CNES, SIHSUS, SIM), INEP, Incra, INPE, Agência Nacional de Águas, ANATEL, SNIS, FUNAI, DENIT além, é claro, dos dados do MapBiomas e do próprio BNDES.

Na segunda metade de 2022, visando realizar um estudo abrangente dos impactos ambientais em áreas financiadas por crédito rural do BNDES, o núcleo GEOBNDES concentrou seu foco na investigação de imóveis rurais inscritos no Cadastro Ambiental Rural (CAR). A partir da identificação geográfica dos imóveis rurais financiados ou candidatos a apoio financeiro, seria possível verificar aspectos ambientais dos projetos, tais como a existência de indícios de desmatamento ilegal e o

eventual conflito territorial com áreas de preservação ou outros aspectos socioambientais relevantes.

Tendo em vista o objetivo de examinar características específicas de um imóvel rural relacionado a um projeto de investimento que solicite financiamento ao BNDES, foi estruturado mecanismo de cruzamento entre bases de dados com camadas de interesse da política pública com a base de imóveis rurais do SICAR, de modo a responder, para cada um dos 6,5 milhões de imóveis rurais do país, as seguintes perguntas:

a) há alerta de desmatamento MapBiomas ou PRODES?;
b) há interseção com unidades de conservação, áreas indígenas ou quilombolas?;
c) quais são as características do uso do solo desse imóvel?

Assim, foi incorporada à Cosmobase GEOBNDES (Google Bigquery) as bases do SICAR contendo os polígonos dos imóveis rurais e, na sequência, foi realizada a ingestão das bases do MapBiomas (uso do solo e alertas), do PRODES (desmatamento), de unidades de conservação, áreas indígenas, quilombolas e crédito rural (BACEN/SICOR).

A incorporação da base completa do SICAR à Cosmobase Pública GEOBNDES trazia algumas dificuldades. A primeira dificuldade dizia respeito à própria obtenção dos dados, pois o *download* da base de imóveis CAR é disponibilizado por município e para cada um há a necessidade de validação *captcha*. Além disso, a base completa contém cerca de 6,5 milhões de registros, compostos pelo polígono principal do imóvel e por polígonos menores com características do imóvel, em formato shapefile. A incorporação ao Google BigQuery exigia a transformação desses arquivos para o formato csv. Essas duas tarefas de obtenção e conversão dos arquivos de uma base tão extensa não seriam possíveis manualmente.

Assim, foi necessária a programação de um script em Python que automatizasse a tarefa de baixar cada um dos municípios e suprisse automaticamente o *captcha* solicitado pelo site, além de efetuar as conversões de formato necessárias para o armazenamento em formato de tabela relacional na base de dados Cosmobase Pública GEOBNDES (Google Bigquery).

Posteriormente, incorporamos os dados relativos a uso do solo coleção 7 do MapBiomas para o ano de 2021[1]. O arquivo é do tipo GeoTiff com mais de 1Gb, que precisou ser "fatiado" por município e transformado, primeiramente em vetor shapefile e posteriormente em arquivos csv. Esses dados encontram-se armazenados numa tabela do Google Bigquery com a geometria em formato WKT e possuem cerca de 80 milhões de registros (70Gb), o que demandou longo tempo processamento.

Passou-se então à incorporação dos alertas de desmatamento do MapBiomas[2] e dos dados de desmatamento do PRODES[3]. Este último, a exemplo do uso do solo do MapBiomas, trata-se de um arquivo GeoTiff com cerca de 500Mb, contendo os alertas para todo o território brasileiro, ano a ano. Sendo também um arquivo raster muito grande, para ser transformado precisou ser subdividido por municípios e cada um dos municípios processado individualmente, visando seguir a trajetória já descrita de conversão de sua imagem raster primeiro para shapefile e posteriormente planilha csv com geometria em WKT. Uma vez mais, essa tarefa demandou alguns dias de processamento.

Outros dados também incorporados, conforme já citado, foram das unidades de conservação, terras indígenas e quilombolas, nesse caso arquivos pequenos e de fácil assimilação na plataforma, de sorte que não julgamos oportuno detalhar seu processo de inserção na Cosmobase Pública GEOBNDES.

Com os dados armazenados, o cruzamento entre as bases pode ser efetuado utilizando-se a linguagem SQL, que na plataforma Google Bigquery possui comandos nativos para o processamento de dados geográficos como, por exemplo, ST_INTERSECTS que verifica a interseção de duas geometrias (ex. o polígono do imóvel e a área do alerta de desmatamento); ou ST_INTERSECTION, que calcula o polígono de interseção entre duas geometrias (no caso anterior, retorna o polígono do alerta de desmatamento no imóvel em questão); ou ST_AREA que calcula a área de um polígono.

---

[1] Disponível em: https://storage.googleapis.com/mapbiomas-public/brasil/collection-7/lclu/coverage/brasil_coverage_2021.tif. Acesso em 06/04/2023.

[2] Disponível em: https://storage.googleapis.com/alerta-public/dashboard/downloads/dashboard_alerts-shapefile.zip. Acesso em 06/04/2023

[3] Disponível em: http://terrabrasilis.dpi.inpe.br/downloads/. Acesso em 06/04/2023.

A performance do Google Biquery revela-se bastante eficiente, tornando possível, por exemplo, identificar a interseção de 80 milhões de polígonos contendo o uso do solo com 6,5 milhões de polígonos de imóveis rurais, criando-se assim uma tabela resultante, contendo os polígonos de uso do solo para todos os imóveis inscritos no CAR.

Assim, procedemos à geração de tabelas para cada tipo de dado, contendo o número do CAR de cada imóvel rural. As tabelas descritas foram tornadas públicas na Cosmobase Pública GEOBNDES no Google Bigquery, bastando que o usuário tenha uma conta de acesso à plataforma. São elas:

i. Uso do solo MAPBIOMAS coleção 7, ano de 2021, por CAR

Pode ser acessada em uma consulta SQL na plataforma a partir da referência:
`cosmobase-publica.GEOBNDES.SICAR_MAPBIOMAS_uso_do_solo_colecao_7_2021`

Um exemplo de comando SQL simples para acessar as 1000 primeiras linhas dessa tabela seria :

```
SELECT * FROM `cosmobase-publica.GEOBNDES.SICAR_MAPBIOMAS_uso_do_solo_colecao_7_2021` LIMIT 1000
```

Essa tabela possui cerca de 26 milhões de linhas contendo o uso do solo em 6,5 milhões de imóveis rurais e possui a seguinte estrutura:

Nome do campo	Tipo	Modo
SICAR_car	STRING	NULLABLE
SICAR_area_imovel_ha	STRING	NULLABLE
USO_SOLO_DN	STRING	NULLABLE
USO_SOLO_nome	STRING	NULLABLE
USO_SOLO_name	STRING	NULLABLE
USO_SOLO_color	STRING	NULLABLE
WKT_TIPO_SOLO_NO_IMOVEL	GEOGRAPHY	NULLABLE
AREA_HA_WKT_TIPO_SOLO_NO_IMOVEL	FLOAT	NULLABLE

ii. Alertas de desmatamento do MapBiomas, por CAR

Pode ser acessada em uma consulta SQL na plataforma a partir da referência: `cosmobase-publica.GEOBNDES.SICAR_MAPBIOMAS_uso_do_solo_colecao_7_2021`

Um exemplo de comando SQL simples para acessar as 1000 primeiras linhas dessa tabela seria :

```
SELECT * FROM `cosmobase-publica.GEOBNDES.SICAR_MAPBIOMAS_alerts`
LIMIT 1000
```

Essa tabela possui 550 mil linhas contendo alertas de desmatamento do MapBiomas ocorridos em 256 mil imóveis rurais e possui a seguinte estrutura:

Nome do campo	Tipo	Modo
SICAR_car	STRING	NULLABLE
MAPBIOMAS_ALERTAS_fid	STRING	NULLABLE
MAPBIOMAS_ALERTAS_CodeAlerta	STRING	NULLABLE
MAPBIOMAS_ALERTAS_Fonte	STRING	NULLABLE
MAPBIOMAS_ALERTAS_Bioma	STRING	NULLABLE
MAPBIOMAS_ALERTAS_Estado	STRING	NULLABLE
MAPBIOMAS_ALERTAS_Municipio	STRING	NULLABLE
MAPBIOMAS_ALERTAS_AreaHa	STRING	NULLABLE
MAPBIOMAS_ALERTAS_AnoDetec	STRING	NULLABLE
MAPBIOMAS_ALERTAS_DataDetec	STRING	NULLABLE
MAPBIOMAS_ALERTAS_DtImgAnt	STRING	NULLABLE
MAPBIOMAS_ALERTAS_DtImgDep	STRING	NULLABLE
MAPBIOMAS_ALERTAS_VPressao	STRING	NULLABLE
WKT_MAPBIOMAS_ALERTAS	GEOGRAPHY	NULLABLE
AREA_HA_WKT_MAPBIOMAS_ALERTAS	FLOAT	NULLABLE
WKT_MAPBIOMAS_ALERTAS_NO_IMOVEL	GEOGRAPHY	NULLABLE
AREA_HA_WKT_MAPBIOMAS_ALERTAS_NO_IMOVEL	FLOAT	NULLABLE

iii. Desmatamento do PRODES, por CAR

Pode ser acessada em uma consulta SQL na plataforma a partir da referência: `cosmobase-publica.GEOBNDES.SICAR_PRODES_alerts`

Um exemplo de comando SQL simples para acessar as 1000 primeiras linhas dessa tabela seria:

```
SELECT * FROM `cosmobase-publica.GEOBNDES.SICAR_PRODES_alerts` LIMIT 1000
```

Essa tabela possui a seguinte estrutura:

Nome do campo	Tipo	Modo
SICAR_car	STRING	NULLABLE
ANO_ALERTA_DESMATAMENTO	INTEGER	NULLABLE
WKT_PRODES_ALERTAS_NO_IMOVEL	GEOGRAPHY	NULLABLE
AREA_HA_WKT_PRODES_ALERTAS_NO_IMOVEL	FLOAT	NULLABLE

iv. Interseção de Unidades de Conservação com o imóvel, por CAR

Pode ser acessada em uma consulta SQL na plataforma a partir da referência: `cosmobase-publica.GEOBNDES.SICAR_unidades_de_conservacao`

Um exemplo de comando SQL simples para acessar as 1000 primeiras linhas dessa tabela seria:

```
SELECT * FROM `cosmobase-publica.GEOBNDES.SICAR_unidades_de_conservacao` LIMIT 1000
```

Essa tabela possui a seguinte estrutura:

Nome do campo	Tipo	Modo
SICAR_car	STRING	NULLABLE
nome_uc	STRING	NULLABLE
categoria	STRING	NULLABLE
grupo	STRING	NULLABLE
esfera	STRING	NULLABLE
ato_criaca	STRING	NULLABLE
codigo_uc	STRING	NULLABLE
nome_og	STRING	NULLABLE
nome_abrev	STRING	NULLABLE
WKT_UNIDADE_DE_CONSERVACAO_NO_IMOVEL	GEOGRAPHY	NULLABLE
AREA_HA_UNIDADE_CONSERVACAO	STRING	NULLABLE
AREA_HA_UNIDADE_DE_CONSERVACAO_NO_IMOVEL	FLOAT	NULLABLE
AREA_HA_IMOVEL	FLOAT	NULLABLE

v. Cadastro no SICAR do imóvel

Pode ser acessada em uma consulta SQL na plataforma a partir da referência: `cosmobase-publica.GEOBNDES.SICAR_imoveis`

Um exemplo de comando SQL simples para acessar as 1000 primeiras linhas dessa tabela seria :

```
SELECT * FROM `cosmobase-publica.GEOBNDES.SICAR_imoveis` LIMIT 1000
```

Essa tabela possui a seguinte estrutura:

Nome do campo	Tipo	Modo
SICAR_car	STRING	NULLABLE
SICAR_area_ha	STRING	NULLABLE
SICAR_uf	STRING	NULLABLE
SICAR_municipio	STRING	NULLABLE
SICAR_num_modulos_fiscais	STRING	NULLABLE
SICAR_tipo_imovel	STRING	NULLABLE
SICAR_situacao	STRING	NULLABLE
SICAR_condicao_I	STRING	NULLABLE
SICAR_geo_area_imovel	GEOGRAPHY	NULLABLE

vi. Interseção do imóvel com terras indígenas

Pode ser acessada em uma consulta SQL na plataforma a partir da referência: `cosmobase-publica.GEOBNDES.SICAR_terras_indigenas`

Um exemplo de comando SQL simples para acessar as 1000 primeiras linhas dessa tabela seria :

```
SELECT * FROM `cosmobase-publica.GEOBNDES.SICAR_terras_indigenas` LIMIT 1000
```

Essa tabela possui a seguinte estrutura:

Nome do campo	Tipo	Modo
SICAR_car	STRING	NULLABLE
tipo	STRING	NULLABLE
terrai_cod	STRING	NULLABLE
terrai_nom	STRING	NULLABLE
etnia_nome	STRING	NULLABLE
municipio_	STRING	NULLABLE
uf_sigla	STRING	NULLABLE
superficie	STRING	NULLABLE
fase_ti	STRING	NULLABLE
cr	STRING	NULLABLE
faixa_fron	STRING	NULLABLE
WKT_TERRA_INDIGENA_NO_IMOVEL	GEOGRAPHY	NULLABLE
AREA_HA_TERRA_INDIGENA_NO_IMOVEL	FLOAT	NULLABLE
AREA_HA_TERRA_INDIGENA	FLOAT	NULLABLE
AREA_HA_IMOVEL	FLOAT	NULLABLE

vii. Interseção do imóvel com áreas quilombolas

Pode ser acessada em uma consulta SQL na plataforma a partir da referência: `cosmobase-publica.GEOBNDES.SICAR_quilombolas`

Um exemplo de comando SQL simples para acessar as 1000 primeiras linhas dessa tabela seria :

```
SELECT * FROM `cosmobase-publica.GEOBNDES.SICAR_quilombolas` LIMIT 1000
```

Essa tabela possui a seguinte estrutura:

Nome do campo	Tipo	Modo
SICAR_car	STRING	NULLABLE
cd_quilomb	STRING	NULLABLE
cd_sr	STRING	NULLABLE
nr_process	STRING	NULLABLE
nm_comunid	STRING	NULLABLE
nm_municip	STRING	NULLABLE
cd_uf	STRING	NULLABLE
dt_publica	STRING	NULLABLE
dt_public1	STRING	NULLABLE
nr_familia	STRING	NULLABLE
dt_titulac	STRING	NULLABLE
nr_area_ha	STRING	NULLABLE
nr_perimet	STRING	NULLABLE
cd_sipra	STRING	NULLABLE
ob_descric	STRING	NULLABLE
st_titulad	STRING	NULLABLE
dt_decreto	STRING	NULLABLE
tp_levanta	STRING	NULLABLE
nr_escalao	STRING	NULLABLE
area_calc_	STRING	NULLABLE
perimetro_	STRING	NULLABLE
esfera	STRING	NULLABLE
fase	STRING	NULLABLE
responsave	STRING	NULLABLE
WKT_QUILOMBOLAS_NO_IMOVEL	GEOGRAPHY	NULLABLE
AREA_HA_QUILOMBOLAS_NO_IMOVEL	FLOAT	NULLABLE
AREA_HA_QUILOMBOLAS	FLOAT	NULLABLE
AREA_HA_IMOVEL	FLOAT	NULLABLE

3. EVOLUÇÃO DO PROJETO: DESENVOLVIMENTO DE API

Visando a integração simplificada dos dados processados com sistemas informacionais que necessitem consultar as informações dos imóveis rurais, iniciamos o desenvolvimento de uma API com interface prática. A ideia básica é que seja feita uma chamada a um endereço http, passando-se como parâmetro o número do CAR e recebendo como retorno um vetor geojson contendo as informações desejadas sobre o imóvel rural.

Para efeitos de performance e de economia (estamos falando por vezes de dezenas de milhões de registros e 6,5 milhões de imóveis rurais), optamos por pré-processar as informações para cada imóvel e disponibilizar a consulta por intermédio de um micro serviço web que utiliza a tecnologia Google Functions para responder a uma chamada web com parâmetros determinados. Até o momento já foram implantadas as seguintes chamadas:

i. MAPBIOMAS USO DO SOLO 2021 – COLEÇÃO 7

https://map-imoveis-dqymna3bzq-rj.a.run.app/?car=AL-2703304-EDECD398D9484A4F810F1EDEB9FE1AA2&tipo=mapbiomas_solo&formato=geojson

Essa chamada retorna o geojson:

{"type": "FeatureCollection", "features": [{"type": "Feature", "id": 0, "geometry": {"type": "MultiPolygon", "coordinates": [[[[-37.6772294780694, -9.2566000136812], [-37.6774989726546, -9.2566000136812], [-37.6774989726546, -9.25686950826643], [-37.6780379618251, -9.25686950826643], [-37.6780379618251, -9.25713900285167], [-37.6772294780694, -9.25713900285167], [-37.6772294780694, -9.25686950826643], [-37.6769599834842, -9.25686950826643], [-37.6769599834841, -9.2566000136812], [-37.6766904888989, -9.2566000136812], [-37.6766904888989, -9.25633051909596], [-37.6772294780694, -9.25633051909596], [-37.6772294780694, -9.2566000136812]]], [[[-37.6834278535298, -9.24952214120944], [-37.6834278535298, -9.24959315446507], [-37.6833571490679, -9.24959315448452], [-37.6834278535298, -9.24952214120944]]], [[[-37.6748040268023, -9.26764929167587], [-37.6742650376318, -9.26764929167587], [-37.6742650376318, -9.26684080792016], [-37.6748040268023, -9.26684080792016], [-37.6748040268023, -9.26764929167587]]], [[[-37.6764209943137, -9.26603232416445], [-37.6761514997284, -9.26603232416445], [-37.6761514997284, -9.26657131333493],

[-37.6758820051432, -9.26657131333493], [-37.6758820051432, -9.26603232416445], [-37.675612510558, -9.26603232416445], [-37.675612510558, -9.26576282957922], [-37.6764209943137, -9.26576282957922], [-37.6764209943137, -9.26603232416445]]], [[[-37.6766904888989, -9.2633373783121], [-37.6764209943137, -9.2633373783121], [-37.6764209943137, -9.26306788372686], [-37.6766904888989, -9.26306788372686], [-37.6766904888989, -9.2633373783121]]], [[[-37.6772294780694, -9.26384697482247], [-37.6772294780694, -9.26387636748257], [-37.6769599834841, -9.26387636748257], [-37.6769599834841, -9.26360687289733], [-37.6770794706393, -9.26360687292218], [-37.6772294780694, -9.26384697482247]]], [[[-37.6769599834841, -9.26341562149608], [-37.6769599834841, -9.26360687289733], [-37.6766904888989, -9.26360687289733], [-37.6766904888989, -9.2633373783121], [-37.6769110999525, -9.26333737832705], [-37.6769599834841, -9.26341562149608]]], [[[-37.6764209943137, -9.26255291106123], [-37.6764209943137, -9.26306788372686], [-37.6761514997284, -9.26306788372686], [-37.6761514997284, -9.26252889455639], [-37.675217570688, -9.26252889474289], [-37.6762568415612, -9.26229016583336], [-37.6764209943137, -9.26255291106123]]], [[[-37.6731870592909, -9.2663703320639], [-37.6731870592908, -9.26576282957922], [-37.6737260484613, -9.26576282957922], [-37.6737260484613, -9.26549333499398], [-37.6742650376318, -9.26549333499398], [-37.6742650376318, -9.26522384040875], [-37.6753430159727, -9.26522384040875], [-37.6753430159727, -9.26549333499398], [-37.675612510558, -9.26549333499398], [-37.675612510558, -9.26576282957922], [-37.6753430159727, -9.26576282957922], [-37.6753430159727, -9.26603232416445], [-37.6748040268023, -9.26603232416445], [-37.6748040268023, -9.26630181874969], [-37.6742650376318, -9.26630181874969], [-37.6742650376318, -9.26657131333493], [-37.6737260484613, -9.26657131333493], [-37.6737260484613, -9.26684080792016], [-37.6734565538761, -9.26684080792016], [-37.6734565538761, -9.26657131333493], [-37.6732759993642, -9.26657131335719], [-37.6731870592909, -9.2663703320639]]]]}, "properties": {"sicar_car": "AL2703304EDECD398D9484A4F810F1EDEB9FE1AA2", "sicar_area_imovel_ha": "96.3622987546675", "uso_solo_dn": "21", "ano": "2021", "num_colecao": "7", "uso_solo_nome": "3.4. Mosaico de Usos", "uso_solo_name": "3.4. Mosaic of Uses", "uso_solo_color": "#fff3bf", "area_ha_wkt_tipo_solo_no_imovel": "4.837996193741248"}}]}

ii. MAPBIOMAS - ALERTAS DE DESMATAMENTO NO IMÓVEL

https://map-imoveis-dqymna3bzq-rj.a.run.app/?car=AC-1200013-01F4B8AC769C426585FEC9C11371DBAB&tipo=mapbiomas_alerts&formato=geojson

iii. PRODES –DESMATAMENTO NO IMÓVEL

https://map-imoveis-dqymna3bzq-rj.a.run.app/?car=AC-1200013-01F4B8AC769C426585FEC9C11371DBAB&tipo=prodes_alerts&formato=geojson

iv. TERRAS INDÍGENAS NO IMÓVEL

https://map-imoveis-dqymna3bzq-rj.a.run.app/?car=GO-5205521-F740EF6662F64C2DAA3C98D887F983F4&tipo=terras_indigenas&formato=geojson

v. UNIDADES DE CONSERVAÇÃO NO IMÓVEL

https://map-imoveis-dqymna3bzq-rj.a.run.app/?car=SP-3519600-2D683FAE64464436BF349EEAFF506199&tipo=unidades_conservacao&formato=geojson

vi. ÁREA DO IMÓVEL DECLARADA NO SICAR E SITUAÇÃO

https://map-imoveis-dqymna3bzq-rj.a.run.app/?car=AC-1200013-01F4B8AC769C426585FEC9C11371DBAB&tipo=sicar&formato=geojson

viii. PRESENÇA DE COMUNIDADES QUILOMBOLAS NO IMÓVEL

https://map-imoveis-dqymna3bzq-rj.a.run.app/?car=AL-2700706-0270C93B39134379B8D0F284A74CAE75&tipo=quilombolas&formato=geojson

4. CONSIDERAÇÕES FINAIS

Com a estrutura integrada de dados geoespaciais acima descrita, foi possível internalizar um arcabouço de inteligência territorial bastante útil à realização de estudos e análises socioambientais sobre as políticas públicas de fomento e concessão de crédito do BNDES, trazendo impactos diretos no fluxo operacional e na tomada de decisões do Banco, com destaque para a classificação socioambiental de projetos; análise de contexto territorial de empreendimentos de infraestrutura, avaliação e mitigação de riscos nas áreas de entorno; aprimoramento de mecanismos de compliance socioambiental e veto a operações em áreas com indício de desmatamento ilegal.

Conforme já mencionado, as tabelas da Cosmobase Pública GEOBNDES estruturadas na plataforma Google Bigquery encontram-se disponíveis para acesso público, de modo que podem ser amplamente consultadas. Assim, é possível identificar a situação de um imóvel rural mediante simples consultas SQL aos dados, bem como efetuar cruzamentos com outras fontes de dados do projeto GEOBNDES ou mesmo externas.

Como próximos passos, pretende-se disponibilizar a API também para consulta pública. Para viabilizar essa empreitada, será necessário implementar mecanismos de controle de acesso e limitação ao número de chamadas, a fim de evitar que sejam geradas cobranças pecuniárias elevadas ao projeto pelo Google.

Finalmente, esperamos com esse projeto facilitar o acesso a dados públicos relativos ao território brasileiro e seu cruzamento com camadas socioambientais, possibilitando o acesso simplificado a informações de desmatamento e uso do solo, dentre outros, que favoreçam o trabalho de agentes públicos e privados em estudos e iniciativas que contribuam para a formulação e implementação de políticas públicas mais efetivas e sustentáveis, contribuindo para o desenvolvimento do país de forma compatível com a preservação do planeta para as futuras gerações.

ANEXOS - SCRIPTS

```
# -*- coding: utf-8 -*-
"""API_MAPBIOMAS_alerts.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1idCFumAWeNXUUYnJ8u6aVHz-7L2luyom

## Importa os alertas do Mapbiomas
"""

from google.cloud import bigquery
# autentica
from google.colab import auth
auth.authenticate_user()
from google.colab import drive
drive.mount('/content/drive')

# Commented out IPython magic to ensure Python compatibility.
# %cd /content/drive/MyDrive/cosmobase
# %ls

# Commented out IPython magic to ensure Python compatibility.
# %cd dados

# Commented out IPython magic to ensure Python compatibility.
# %ls

!wget https://storage.googleapis.com/alerta-public/dashboard/downloads/dashboard_alerts-shapefile.zip

!sudo apt install unzip
!sudo apt install zip
import pandas as pd  # First, we'll import Pandas, a data processing and CSV file I/O library
import numpy as np
from datetime import datetime


!apt-get update
!pip install --upgrade pip
!pip install urllib3


import io
import os
import urllib3
```

```
ENCODING = 'UTF-8'

# instalando outros pacotes
#Installations
!apt install gdal-bin python-gdal python3-gdal
!pip install rasterio
!apt install python3-rtree
#!pip install git+git://github.com/geopandas/geopandas.git
!pip install geopandas
!pip install descartes

!unzip dashboard_alerts-shapefile.zip

# Commented out IPython magic to ensure Python compatibility.
# %rm -r *.txt

# Commented out IPython magic to ensure Python compatibility.
# %ls

# salva todos os arquivos shp de um diretório como csv
from os import listdir
from os.path import isfile, join
import geopandas
import numpy as np
myDir = './'
def listaDir(txtDIR):
    # função que retorna uma lista de arquivos de um diretório
    try:
        mypath = txtDIR
        onlyfiles = [f for f in listdir(mypath) if isfile(join(mypath, f)) and f[-3:] == 'shp']
        return onlyfiles
    except:
        return ''

for myFile in listaDir(myDir):
    myFileInput = myDir + '/' + myFile
    myFileOutput = myDir +  myFile[:-3]+'csv'
    print(myFileInput)
    df = geopandas.read_file(myFileInput)
    df.fillna('', inplace = True)
    df.to_csv(myFileOutput, index=False, sep = ';')

# copia do google drive para o bucket
from google.colab import auth
auth.authenticate_user()
!gsutil -m cp -r ./dashboard_published_alerts_download_shpPolygon.csv
gs://cosmobase-publica-bucket/dashboard_published_alerts_download_shpPolygon.csv
```

```
!gsutil -m cp -r gs://cosmobase-publica-
bucket/dashboard_published_alerts_download_shpPolygon.csv
./dashboard_published_alerts_download_shpPolygon.csv

# Função genérica para atualizar tabelas
from google.cloud import bigquery
# Construct a BigQuery client object.
client = bigquery.Client('cosmobase-publica')

def atualiza_tabela(txt_arquivo, txt_tabela, dic_campos):
    # INÍCIO monta a string do schema
    txt_schema = '['
    for key in dic_campos:
      txt_schema = txt_schema + 'bigquery.SchemaField("' + key +'","' +
dic_campos[key] + '"),'
      #print(key, '->', dic_campos[key])
    txt_schema = txt_schema+']'
    # FIM monta a string do schema
    # Set table_id to the ID of the table to create.
    table_id = txt_tabela

    job_config = bigquery.LoadJobConfig(
        schema=eval(txt_schema),
        skip_leading_rows=1,
        max_bad_records=0,
        encoding='UTF-8',
        # The source format defaults to CSV, so the line below is optional.
        source_format=bigquery.SourceFormat.CSV,
        write_disposition=bigquery.WriteDisposition.WRITE_TRUNCATE,
    )
    job_config.field_delimiter = ';'
    uri = txt_arquivo

    load_job = client.load_table_from_uri(
        uri, table_id, job_config=job_config
    )  # Make an API request.

    load_job.result()  # Waits for the job to complete.

    destination_table = client.get_table(table_id)  # Make an API request.
    print("Loaded {} rows.".format(destination_table.num_rows))

# Processa todas as tabelas
from os import listdir

lista_arq = ['dashboard_published_alerts_download_shpPolygon.csv']


for arquivo in lista_arq:
  with open(arquivo, 'r') as e:
```

```
    try:
        cabecalho = e.readline()
        cabecalho = cabecalho.replace('\n','').replace('\r','')
        txt_dic_campos = cabecalho.replace(';',':'STRING','")
        txt_dic_campos = txt_dic_campos.replace(':', '":")
        txt_dic_campos = txt_dic_campos.replace(';', '",'")
        txt_dic_campos = '{"'+txt_dic_campos
        txt_dic_campos = txt_dic_campos + '":'STRING'}'

        txt_arquivo = 'gs://cosmobase-publica-bucket/'+ arquivo
        txt_tabela = 'cosmobase-
publica.DESMATAMENTO_ADIG.MAPBIOMAS_alerts'
        dic_campos = eval(txt_dic_campos)
        print(cabecalho)
        print(txt_arquivo, txt_tabela, dic_campos) # processa o job do bigquery
        atualiza_tabela(txt_arquivo, txt_tabela, dic_campos) # processa o job do bigquery
        print(txt_tabela, ' processada')
    except Exception as f:
        print(f)
        print(arquivo, ' não processado')
        pass
    e.close()

# rodar a consulta gera_MAPBIOMAS_alerts em DESMATAMENTO_SICAR

from google.cloud import storage
from google.cloud import bigquery

client = bigquery.Client('cosmobase-publica')

query = """
    with mapbiomas as (
    select
        sicar_car,
        mapbiomas_alertas_fonte alertas_fonte,
        mapbiomas_alertas_anodetec alertas_anodetec,
        wkt_mapbiomas_alertas_no_imovel wkt_alerta_no_imovel,
        area_ha_wkt_mapbiomas_alertas_no_imovel area_ha_alerta_no_imovel,
        area_ha_wkt_mapbiomas_alertas area_ha_alerta,
        MAPBIOMAS_ALERTAS_DataDetec data_detec,
        MAPBIOMAS_ALERTAS_CodeAlerta code_alerta,
        row_number() over() linha

        from `cosmobase-publica.DESMATAMENTO_SICAR.MAPBIOMAS_alerts`

        )

        select * from mapbiomas
        order by linha asc
"""
```

```
results = client.query(query)

# Commented out IPython magic to ensure Python compatibility.
# posiciona no diretório desejado
# %cd dados
# %ls

"""## Preprocessa os arquivos de desmatamento, 1 por CAR"""

# salva em dados 1 arquivo por CAR

def write_file(blob_name, txt_conteudo):
        with open(blob_name, 'a') as f:
            f.write(txt_conteudo+'\r\n')
        f.close()
i=0
for row in results:
        sicar_car = row['sicar_car']
        alertas_fonte = row['alertas_fonte']
        alertas_anodetec = row['alertas_anodetec']
        wkt_alerta_no_imovel = row['wkt_alerta_no_imovel']
        area_ha_alerta_no_imovel = row['area_ha_alerta_no_imovel']
        area_ha_alerta = row['area_ha_alerta']
        data_detec = row['data_detec']
        code_alerta = row['code_alerta']

        write_file(f'{sicar_car}.txt',
f'{sicar_car}|{alertas_fonte}|{alertas_anodetec}|{wkt_alerta_no_imovel}|{area_ha_aler
ta_no_imovel}|{area_ha_alerta}|{data_detec}|{code_alerta}')
        if i%10000==0:
           print(row['linha'])
        i+=1
print('arquivos locais salvos', i)

# lista todos os arquivos do diretório dados, para copiá-los
from os import listdir
from os.path import isfile, join

def listaDir(txtDIR):
   # função que retorna uma lista de arquivos de um diretório
   try:
      mypath = txtDIR
      onlyfiles = [f for f in listdir(mypath) if isfile(join(mypath, f))]
      return onlyfiles
   except:
      return ''
lista_arq = listaDir('./dados')

i=0
for myFile in lista_arq:
```

```
    if i%10000==0:
      print(i)
    i+=1
print('numero de arquivos:',i)


# -*- coding: utf-8 -*-
"""API_QUILOMBOLAS.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1Ccwh4pP174v3eBX1nCWxqqrQBFbL_7j6

## IMPORTA A TABELA DE QUILOMBOLAS DO INCRA
https://certificacao.incra.gov.br/csv_shp/export_shp.py
"""

from google.cloud import bigquery
# autentica
from google.colab import auth
auth.authenticate_user()
from google.colab import drive
drive.mount('/content/drive')

# Commented out IPython magic to ensure Python compatibility.
# %cd /content/drive/MyDrive/cosmobase
# %ls

!unzip "Áreas de Quilombolas.zip"

# Commented out IPython magic to ensure Python compatibility.
# %ls

"""# Importa as bibliotecas necessárias"""

import pandas as pd  # First, we'll import Pandas, a data processing and CSV file I/O library
import numpy as np
from datetime import datetime

# instalando o Selenium
!apt-get update
!pip install --upgrade pip
!pip install urllib3


import io
import os
import urllib3
```

```
ENCODING = 'UTF-8'

# instalando outros pacotes
#Installations
!apt install gdal-bin python-gdal python3-gdal
!pip install rasterio
!apt install python3-rtree
#!pip install git+git://github.com/geopandas/geopandas.git
!pip install geopandas
!pip install descartes
!sudo apt install unzip

"""Salva todos os arquivos shp de um diretório do google drive como csv"""

from os import listdir
from os.path import isfile, join
import geopandas
import numpy as np
lista_arq = [' ├Ɓreas de Quilombolas.shp']



for myFile in lista_arq:
    myFileInput = myFile
    myFileOutput = myFile[:-3]+'csv'
    print(myFileInput)
    df = geopandas.read_file(myFileInput)
    df.fillna('', inplace = True)
    df.to_csv(myFileOutput, index=False, sep = '|', encoding='UTF-8')

# repara o arquivo csv,  que tem quebras de página dentro das linhas

from os import listdir
from os.path import isfile, join
#corrige o arquivo do finbra
import sys
import csv
import re


myFile = r'Areas de Quilombolas.csv'
cabecalho =
['cd_quilomb','cd_sr','nr_process','nm_comunid','nm_municip','cd_uf','dt_publica','dt_pu
blic1','nr_familia','dt_titulac','nr_area_ha','nr_perimet','cd_sipra','ob_descric','st_titulad','
dt_decreto','tp_levanta','nr_escalao','area_calc_','perimetro_','esfera','fase','responsave','g
eometry']

def repara(cabecalho, myFile):
# tira o caracteres \r e \n de linhas defeituosas de arquivos texto csv
```

```
    myFileInput = myFile
    myFileOutput = myFile[:-4]+'_corrigido.csv'
    #print(myFileInput)


    with open(myFileInput, 'r', newline='', encoding='UTF8') as csvfileInput,
open(myFileOutput, 'w', newline='', encoding='UTF8') as csvfileOutput:
        writer = csv.writer(csvfileOutput)
        i = 1
        linhaErradaNova = True
        for line in csvfileInput:
            listline = line.split('|')
            if (len(cabecalho)>len(listline)):
                if linhaErradaNova:
                    listParte1 = line.replace('\r','').replace('\n','').replace('"','')
                    linhaErradaNova = False
                    print(i, line)
                else:

                    listParte2 = line.replace('\r','').replace('\n','').replace('"','')
                    linhaNova = listParte1+listParte2
                    #print(i,'erro', linhaNova)
                    print(i,line)
                    linhaErradaNova = True
                    minhaListalinha = list(linhaNova.split('|'))
                    #print(minhaListalinha)
                    writer.writerow(minhaListalinha)
                    linhaNova=''
            elif (len(cabecalho)<len(listline)):
                linhaErradaNova = True
                txtline = re.sub(r'(\")(.*)(\;+)(.*)(\")', '\g<2>\g<4>',
line.replace(r'"000000000000,"',r'000000000000').replace(r'"00000000000,,"',r'0000000
0000').replace('"',''))
                minhaListalinha = list(txtline.replace('\r','').replace('\n','').replace('"','').split('|'))
                writer.writerow(minhaListalinha)
                linhaNova=''
                print(i, 'erro', line)
            else:
                linhaErradaNova = True
                minhaListalinha = list(line.replace('\r','').replace('\n','').replace('"','').split('|'))
                writer.writerow(minhaListalinha)
                linhaNova=''
            i+=1

    csvfileInput.close()
    csvfileOutput.close()


print(myFile)
repara(cabecalho,myFile)
```

```
# copia do google drive para o bucket
from google.colab import auth
auth.authenticate_user()
!gsutil -m cp -r "./Areas de Quilombolas_corrigido.csv" "gs://cosmobase-publica-
bucket/Areas de Quilombolas.csv"

"""## Função genérica para atualizar tabelas"""

# Função genérica para atualizar tabelas
from google.cloud import bigquery
# Construct a BigQuery client object.
client = bigquery.Client('cosmobase-publica')

def atualiza_tabela(txt_arquivo, txt_tabela, dic_campos):
    # INÍCIO monta a string do schema
    txt_schema = '['
    for key in dic_campos:
      txt_schema = txt_schema + 'bigquery.SchemaField("' + key +'","' +
dic_campos[key] + '"),'
      #print(key, '->', dic_campos[key])
    txt_schema = txt_schema+']'
    # FIM monta a string do schema
    # Set table_id to the ID of the table to create.
    table_id = txt_tabela

    job_config = bigquery.LoadJobConfig(
        schema=eval(txt_schema),
        skip_leading_rows=1,
        max_bad_records=0,
        encoding='UTF-8',
        # The source format defaults to CSV, so the line below is optional.
        source_format=bigquery.SourceFormat.CSV,
        write_disposition=bigquery.WriteDisposition.WRITE_TRUNCATE,
    )
    job_config.field_delimiter = ','
    uri = txt_arquivo

    load_job = client.load_table_from_uri(
        uri, table_id, job_config=job_config
    )  # Make an API request.

    load_job.result()  # Waits for the job to complete.

    destination_table = client.get_table(table_id)  # Make an API request.
    print("Loaded {} rows.".format(destination_table.num_rows))

"""## Processa todas as tabelas"""

# Processa todas as tabelas
```

```
from os import listdir

lista_arq = ['Areas de Quilombolas.csv']


for arquivo in lista_arq:
  with open(arquivo, 'r') as e:
    try:
        cabecalho = e.readline()
        cabecalho = cabecalho.replace('\n','').replace('\r','')
        txt_dic_campos = cabecalho.replace('|','':'STRING','')
        txt_dic_campos = txt_dic_campos.replace('|', '':'')
        txt_dic_campos = txt_dic_campos.replace('|', '','')
        txt_dic_campos = "{''+txt_dic_campos
        txt_dic_campos = txt_dic_campos + "':'STRING'}"
        #print(txt_dic_campos)
        txt_arquivo = 'gs://cosmobase-publica-bucket/'+ arquivo
        txt_tabela = 'cosmobase-publica.INCRA.Areas_Quilombolas'
        dic_campos = eval(txt_dic_campos)
        #print(dic_campos)
        #print(cabecalho)
        print(txt_arquivo, txt_tabela, dic_campos) # processa o job do bigquery
        atualiza_tabela(txt_arquivo, txt_tabela, dic_campos) # processa o job do bigquery
        print(txt_tabela, ' processada')
    except Exception as f:
        print(f)
        print(arquivo, ' não processado')
        pass
    e.close()

# rodar a consulta gera_sicar_QUILOMBOLAS em INCRA

"""## Preprocessa as tabelas"""

from google.cloud import storage
from google.cloud import bigquery

client = bigquery.Client('cosmobase-publica')

query = """
with quilombolas as (
    select
        SICAR_car,
        nm_comunid nome_comunidade,
        nm_municip municipio,
        cd_uf uf,
        nr_familia nr_familia,
        esfera,
        AREA_HA_QUILOMBOLAS_NO_IMOVEL area_ha_quilombola_no_imovel,
        AREA_HA_QUILOMBOLAS area_ha_quilombola,
```

```
        AREA_HA_IMOVEL area_ha_imovel,
        WKT_QUILOMBOLAS_NO_IMOVEL wkt_quilombola_no_imovel,
        row_number() over() linha

        from `cosmobase-publica.INCRA.sicar_QUILOMBOLAS`

        )

        select * from quilombolas
        order by linha asc
"""
results = client.query(query)

"""## Preprocessa os arquivos, 1 por CAR"""

# salva em dados 1 arquivo por CAR

def write_storage(blob_name, txt_conteudo):
    """Write and read a blob from GCS using file-like IO"""
    with open(blob_name, 'a') as f:
      f.write(txt_conteudo+'\r\n')
    f.close()
i=0
for row in results:
    var_SICAR_car = row['SICAR_car']
    var_nome_comunidade = row['nome_comunidade']
    var_municipio = row['municipio']
    var_uf = row['uf']
    var_nr_familia = row['nr_familia']
    var_esfera = row['esfera']
    var_AREA_HA_QUILOMBOLA_NO_IMOVEL =
row['area_ha_quilombola_no_imovel']
    var_AREA_HA_QUILOMBOLA = row['area_ha_quilombola']
    var_AREA_HA_IMOVEL = row['area_ha_imovel']
    var_WKT_QUILOMBOLA_NO_IMOVEL = row['wkt_quilombola_no_imovel']

    write_storage(f'{var_SICAR_car}.txt',
f'{var_SICAR_car}|{var_nome_comunidade}|{var_municipio}|{var_uf}|{var_nr_famili
a}|{var_esfera}|{var_AREA_HA_QUILOMBOLA_NO_IMOVEL}|{var_AREA_HA_
QUILOMBOLA}|{var_AREA_HA_IMOVEL}|{var_WKT_QUILOMBOLA_NO_IM
OVEL}')
    if i%10000==0:
        print(row['linha'])
    i+=1
print('fim', i)

# -*- coding: utf-8 -*-
"""API_TERRAS_INDIGENAS.ipynb

Automatically generated by Colaboratory.
```

```
Original file is located at
    https://colab.research.google.com/drive/1nIJ2dzJsq0tMmyBQ-hS3ZKuyG5jHFgCs

## IMPORTA A TABELA DE TERRAS INDIGENAS DA FUNAI
https://metadados.snirh.gov.br/geonetwork/srv/api/records/3fa8cc38-79b4-4aa1-8179-bba315baea4b
"""

from google.cloud import bigquery
# autentica
from google.colab import auth
auth.authenticate_user()
from google.colab import drive
drive.mount('/content/drive')

# Commented out IPython magic to ensure Python compatibility.
# %cd /content/drive/MyDrive/cosmobase
# %ls

# Commented out IPython magic to ensure Python compatibility.
# %rm -r *

"""# Importa as bibliotecas necessárias"""

import pandas as pd  # First, we'll import Pandas, a data processing and CSV file I/O library
import numpy as np
from datetime import datetime

# instalando o Selenium
!apt-get update
!pip install --upgrade pip
!pip install urllib3


import io
import os
import urllib3

ENCODING = 'UTF-8'

# instalando outros pacotes
#Installations
!apt install gdal-bin python-gdal python3-gdal
!pip install rasterio
!apt install python3-rtree
#!pip install git+git://github.com/geopandas/geopandas.git
!pip install geopandas
!pip install descartes
```

```
!sudo apt install unzip

# copia o arquivo para o drive. Tem que ser manualmente

# fazer download do arquivo tis_poligonais.zip
# de
https://geoserver.funai.gov.br/geoserver/Funai/ows?service=WFS&version=1.0.0&requ
est=GetFeature&typeName=Funai:tis_poligonais&CQL_FILTER=dominio_uniao=%27
t%27&outputFormat=SHAPE-ZIP

#!unzip tis_poligonais.zip

"""Salva todos os arquivos shp de um diretório do google drive como csv"""

from os import listdir
from os.path import isfile, join
import geopandas
import numpy as np
lista_arq = ['./tis_poligonais_areas_sob_interdicao/tis_poligonais.shp',
'./tis_poligonais_homologada/tis_poligonais.shp',
'./tis_poligonais_nao_homologada/tis_poligonais.shp',
'./tis_poligonais_reserva_indigena/tis_poligonais.shp','./tis_poligonais_terras_dominiais
_indigenas/tis_poligonais.shp','./tis_pontos_terra_indigena_em_estudo/tis_pontos.shp']



for myFile in lista_arq:
    myFileInput = myFile
    myFileOutput = myFile[:-3]+'csv'
    print(myFileInput)
    df = geopandas.read_file(myFileInput)
    df.fillna('', inplace = True)
    df.to_csv(myFileOutput, index=False, sep = '|', encoding='UTF-8')

# copia do google drive para o bucket
from google.colab import auth
auth.authenticate_user()
!gsutil -m cp -r ./tis_poligonais_areas_sob_interdicao/tis_poligonais.csv
gs://cosmobase-publica-bucket/FUNAI/tis_poligonais_areas_sob_interdicao.csv

"""## Função genérica para atualizar tabelas"""

from google.cloud import bigquery
# Construct a BigQuery client object.
client = bigquery.Client('cosmobase-publica')

def atualiza_tabela(txt_arquivo, txt_tabela, dic_campos):
    # INÍCIO monta a string do schema
    txt_schema = '['
    for key in dic_campos:
```

```
      txt_schema = txt_schema + 'bigquery.SchemaField("' + key +'","' +
dic_campos[key] + '"),'
      #print(key, '->', dic_campos[key])
    txt_schema = txt_schema+']'
    # FIM monta a string do schema
    # Set table_id to the ID of the table to create.
    table_id = txt_tabela

    job_config = bigquery.LoadJobConfig(
        schema=eval(txt_schema),
        skip_leading_rows=1,
        maxBadRecords=0,
        encoding='UTF-8',
        # The source format defaults to CSV, so the line below is optional.
        source_format=bigquery.SourceFormat.CSV,
        write_disposition=bigquery.WriteDisposition.WRITE_TRUNCATE,
    )
    job_config.fieldDelimiter = '|'
    uri = txt_arquivo

    load_job = client.load_table_from_uri(
        uri, table_id, job_config=job_config
    )  # Make an API request.

    load_job.result()  # Waits for the job to complete.

    destination_table = client.get_table(table_id)  # Make an API request.
    print("Loaded {} rows.".format(destination_table.num_rows))

"""## Processa todas as tabelas"""

from os import listdir

lista_arq = ['./tis_poligonais_areas_sob_interdicao/tis_poligonais.csv',
'./tis_poligonais_homologada/tis_poligonais.csv',
'./tis_poligonais_nao_homologada/tis_poligonais.csv',
'./tis_poligonais_reserva_indigena/tis_poligonais.csv','./tis_poligonais_terras_dominiais
_indigenas/tis_poligonais.csv','./tis_pontos_terra_indigena_em_estudo/tis_pontos.csv']

for arquivo in lista_arq:
  with open(arquivo, 'r') as e:
    try:
        cabecalho = e.readline()
        txt_dic_campos = cabecalho.replace('|',":'STRING',")
        txt_dic_campos = txt_dic_campos.replace(':', "':")
        txt_dic_campos = txt_dic_campos.replace('|', ",'")
        txt_dic_campos = "{'"+txt_dic_campos
        txt_dic_campos = txt_dic_campos + "':'STRING'}"
        if arquivo =='./tis_pontos_terra_indigena_em_estudo/tis_pontos.csv':
```

```
            txt_arquivo = 'gs://cosmobase-publica-
bucket/FUNAI/tis_pontos_terra_indigena_em_estudo.csv'
            txt_tabela = 'cosmobase-publica.FUNAI.tis_pontos_terra_indigena_em_estudo'
        else:
            txt_arquivo = 'gs://cosmobase-publica-bucket/FUNAI/'+arquivo[2:-19]+'.csv'
            txt_tabela = 'cosmobase-publica.FUNAI.'+arquivo[2:-19]
        atualiza_tabela(txt_arquivo, txt_tabela, dic_campos) # processa o job do bigquery
        print(txt_tabela, ' processada')
    except Exception as f:
        print(f)
        print(arquivo, ' não processado')
        pass
    e.close()

lista_arq = ['./tis_poligonais_areas_sob_interdicao/tis_poligonais.csv',
'./tis_poligonais_homologada/tis_poligonais.csv',
'./tis_poligonais_nao_homologada/tis_poligonais.csv',
'./tis_poligonais_reserva_indigena/tis_poligonais.csv','./tis_poligonais_terras_dominiais
_indigenas/tis_poligonais.csv','./tis_pontos_terra_indigena_em_estudo/tis_pontos.csv']
for arquivo in lista_arq:
  with open(arquivo, 'r') as e:
    cabecalho = e.readline()
    txt_dic_campos = cabecalho.replace('|',",")

    print(txt_dic_campos)
    if arquivo =='./tis_pontos_terra_indigena_em_estudo/tis_pontos.csv':
        txt_arquivo = 'gs://cosmobase-publica-
bucket/FUNAI/tis_pontos_terra_indigena_em_estudo.csv'
        txt_tabela = 'cosmobase-publica.FUNAI.tis_pontos_terra_indigena_em_estudo'
    else:
        txt_arquivo = 'gs://cosmobase-publica-bucket/FUNAI/'+arquivo[2:-19]+'.csv'
        txt_tabela = 'cosmobase-publica.FUNAI.'+arquivo[2:-19]
    print(txt_arquivo, txt_tabela)

"""## Preprocessa as tabelas"""

from google.cloud import storage
from google.cloud import bigquery

client = bigquery.Client('cosmobase-publica')

query = """
with indigenas as (
select
    T0.SICAR_car,
    tipo,
    terrai_cod,
    terrai_nom,
    etnia_nome,
    municipio_,
```

```
    uf_sigla,
    superficie,
    fase_ti,
    cr,
    faixa_fron,
    WKT_TERRA_INDIGENA_NO_IMOVEL,
    AREA_HA_TERRA_INDIGENA_NO_IMOVEL,
    AREA_HA_TERRA_INDIGENA,
    AREA_HA_IMOVEL,
    row_number() over() linha

from `cosmobase-publica.DESMATAMENTO_SICAR.SICAR_biomas` T0
inner join `cosmobase-publica.FUNAI.sicar_TERRAS_INDIGENAS` T1
on  T0.SICAR_car = regexp_replace(T1.SICAR_car,'-','')
)
select * from indigenas
order by linha asc
"""
results = client.query(query)

"""## Preprocessa os arquivos, 1 por CAR"""

# salva em dados 1 arquivo por CAR

def write_storage(blob_name, txt_conteudo):
    """Write and read a blob from GCS using file-like IO"""
    with open(blob_name, 'a') as f:
      f.write(txt_conteudo+'\r\n')
    f.close()
i=0
for row in results:
    var_SICAR_car = row['SICAR_car']
    var_tipo = row['tipo']
    var_terrai_cod = row['terrai_cod']
    var_terrai_nom = row['terrai_nom']
    var_etnia_nome = row['etnia_nome']
    var_municipio_ = row['municipio_']
    var_uf_sigla = row['uf_sigla']
    var_superficie = row['superficie']
    var_fase_ti = row['fase_ti']
    var_cr = row['cr']
    var_faixa_fron = row['faixa_fron']
    var_WKT_TERRA_INDIGENA_NO_IMOVEL =
row['WKT_TERRA_INDIGENA_NO_IMOVEL']
    var_AREA_HA_TERRA_INDIGENA_NO_IMOVEL =
row['AREA_HA_TERRA_INDIGENA_NO_IMOVEL']
    var_AREA_HA_TERRA_INDIGENA = row['AREA_HA_TERRA_INDIGENA']
    var_AREA_HA_IMOVEL = row['AREA_HA_IMOVEL']
```

```
    write_storage(f'{var_SICAR_car}.txt',
f'{var_SICAR_car}|{var_tipo}|{var_terrai_cod}|{var_terrai_nom}|{var_etnia_nome}|{
var_municipio_}|{var_uf_sigla}|{var_superficie}|{var_fase_ti}|{var_cr}|{var_faixa_fro
n}|{var_AREA_HA_TERRA_INDIGENA_NO_IMOVEL}|{var_AREA_HA_TERRA
_INDIGENA}|{var_AREA_HA_IMOVEL}|{var_WKT_TERRA_INDIGENA_NO_IM
OVEL}')
    if i%10000==0:
        print(row['linha'])
    i+=1
print('fim', i)

# apaga todos os arquivos
!gsutil -m rm -r gs://cosmobase-functions-
bucket/consulta_CAR/dados/terras_indigenas/*.txt

# copia todos os arquivos. cosmobase-functions-bucket/consulta_CAR/dados
!gsutil -m cp -r -n ./ gs://cosmobase-functions-
bucket/consulta_CAR/dados/terras_indigenas/

# -*- coding: utf-8 -*-
"""API uso do solo MAPBIOMAS.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1radAKr6VBGz5T4fcRiKvw0rQCEopaRdL

## Acessa o google Drive
"""

from google.colab import drive
drive.mount('/content/drive')

"""## Troca o diretório para o local que contém o diretório com os shapefiles"""

# Commented out IPython magic to ensure Python compatibility.
# %cd /content/drive
# %cd MyDrive
# %cd cosmobase
# %ls

"""## Instalação de pacotes"""

#Installations
!apt install gdal-bin python-gdal python3-gdal
!pip install rasterio
!apt install python3-rtree
!pip install geopandas
!pip install descartes
!pip install unzip
```

```
!wget 'https://production.alerta.mapbiomas.org/geoserver/mapbiomas-
alertas/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=mapbiomas
-alertas%3Adashboard_cities-static-layer&outputFormat=SHAPE-ZIP' -O
municipios.zip

# pega a coleção 7 do mapbiomas
!wget https://storage.googleapis.com/mapbiomas-public/brasil/collection-
7/lclu/coverage/brasil_coverage_2021.tif -O brasil_coverage_2021_col7.tif

!unzip municipios.zip

"""## Clipa o raster da coleção 7 pelo shape de cada município
https://code.earthengine.google.com/?accept_repo=users/mapbiomas/user-toolkit
"""

import fiona
import rasterio
import rasterio.mask

def clipa_raster(raster_input, shapes, raster_clipped):


  #with fiona.open(shape_input, "r") as shapefile:
  #    shapes = [feature["geometry"] for feature in shapefile]

  with rasterio.open(raster_input) as src:
      out_image, out_transform = rasterio.mask.mask(src, shapes, crop=True)
      out_meta = src.meta

  out_meta.update({"driver": "GTiff",
                "height": out_image.shape[1],
                "width": out_image.shape[2],
                "transform": out_transform})

  with rasterio.open(raster_clipped, "w", **out_meta) as dest:
      dest.write(out_image)

import geopandas
df = geopandas.read_file("dashboard_cities-static-layer.shp")

import shapely


def getFeatures(gdf):
    """Function to parse features from GeoDataFrame in such a manner that rasterio
wants them"""
    import json
    return [json.loads(gdf.to_json())['features'][0]['geometry']]
```

```
df.geometry = shapely.force_2d(df.geometry)
df.fillna('', inplace = True)
try:
  df.to_crs("EPSG:3857")
except Exception as e:
  print(e)
df.head()

#src_raster_path = r'brasil_coverage_2021_col7.tif'
src_raster_path = r'prodes_brasil_2021.tif'

print(df.head())

for item, row in df[:].iterrows():
    output_raster_path ='./tif/' + row['name']+'-'+str(row['id'])+'.tif'
    id = row['id']
    df_aux = df[df['id'] == id]
    df_exploded = df_aux.explode(ignore_index=True)
    coords = getFeatures(df_exploded)
    try:
      clipa_raster(src_raster_path, coords, output_raster_path)
    except Exception as e:
      print(e)
    print(output_raster_path)

"""## Salva todos os arquivos raster como shp"""

# tem que estar posicionado no diretorio cosmobase
from os import listdir
from os.path import isfile, join

import geopandas as gpd
import numpy as np
import matplotlib.pyplot as plt
import rasterio as rio
from rasterio.plot import show

from osgeo import gdal, ogr
from osgeo import osr
import sys

# this allows GDAL to throw Python Exceptions
gdal.UseExceptions()

myDir = r'../cosmobase/'
def listaDir(txtDIR):
    # função que retorna uma lista de arquivos de um diretório
    try:
        mypath = txtDIR
```

```
        onlyfiles = [f for f in listdir(mypath) if isfile(join(mypath, f)) and f[-3:] == 'tif']
        return onlyfiles
    except:
        return ''

for myFile in listaDir(myDir+'tif/'):
    myFileInput = myDir + 'tif/' + myFile
    myFileOutput = myDir + 'shp/' + myFile[:-3]+'shp'
    print(myFileInput)

    #
    #  get raster datasource
    #
    srs=osr.SpatialReference()
    srs.ImportFromEPSG(3857) # define a projeção
    raster = gdal.Open(myFileInput)
    band = raster.GetRasterBand(1)


    drv = ogr.GetDriverByName('ESRI Shapefile')
    outfile = drv.CreateDataSource(myFileOutput)
    outlayer = outfile.CreateLayer('polygonized raster', srs )
    newField = ogr.FieldDefn('DN', ogr.OFTReal)
    outlayer.CreateField(newField)
    gdal.Polygonize(band, None, outlayer, 0, [])
    outfile = None
    !rm "{myFileInput}"

"""## Converte POLYGON Z e MULTIPOLYGON Z para 2 dimensões"""

## Converte POLYGON Z e MULTIPOLYGON Z para 2 dimensões
from shapely.geometry import *

def remove_third_dimension(geom):
    if geom.is_empty:
        return geom

    if isinstance(geom, Polygon):
        exterior = geom.exterior
        new_exterior = remove_third_dimension(exterior)

        interiors = geom.interiors
        new_interiors = []
        for int in interiors:
            new_interiors.append(remove_third_dimension(int))

        return Polygon(new_exterior, new_interiors)

    elif isinstance(geom, LinearRing):
        return LinearRing([xy[0:2] for xy in list(geom.coords)])
```

```
    elif isinstance(geom, LineString):
        return LineString([xy[0:2] for xy in list(geom.coords)])

    elif isinstance(geom, Point):
        return Point([xy[0:2] for xy in list(geom.coords)])

    elif isinstance(geom, MultiPoint):
        points = list(geom.geoms)
        new_points = []
        for point in points:
            new_points.append(remove_third_dimension(point))

        return MultiPoint(new_points)

    elif isinstance(geom, MultiLineString):
        lines = list(geom.geoms)
        new_lines = []
        for line in lines:
            new_lines.append(remove_third_dimension(line))

        return MultiLineString(new_lines)

    elif isinstance(geom, MultiPolygon):
        pols = list(geom.geoms)

        new_pols = []
        for pol in pols:
            new_pols.append(remove_third_dimension(pol))

        return MultiPolygon(new_pols)

    elif isinstance(geom, GeometryCollection):
        geoms = list(geom.geoms)

        new_geoms = []
        for geom in geoms:
            new_geoms.append(remove_third_dimension(geom))

        return GeometryCollection(new_geoms)

    else:
        raise RuntimeError("Currently this type of geometry is not supported:
{}".format(type(geom)))

"""# Salva todos os arquivos shp de um diretório do google drive como csv"""

import shapely
from os import listdir
from os.path import isfile, join
```

```
import geopandas
import numpy as np
myDir = r'../cosmobase/'
def listaDir(txtDIR):
    # função que retorna uma lista de arquivos de um diretório
    try:
        mypath = txtDIR
        onlyfiles = [f for f in listdir(mypath) if isfile(join(mypath, f)) and f[-3:] == 'shp']
        return onlyfiles
    except:
        return ''

for myFile in listaDir(myDir+'shp/'):
  try:
    myFileInput = myDir + 'shp/' + myFile
    myFileOutput = myDir + 'csv/' + myFile[:-3]+'csv'
    df = geopandas.read_file(myFileInput)
    df.geometry = shapely.force_2d(df.geometry)
    df.fillna('', inplace = True)
    try:
      df.to_crs("EPSG:4326")
    except Exception as e:
      print(e)
    df.to_csv(myFileOutput, index=False, sep = ';')
    !rm "{myFileInput[:-4]}"*
    print(myFileInput, ' processado')
  except:
    print(myFileInput, ' não processado')

from google.colab import auth
auth.authenticate_user()

# Commented out IPython magic to ensure Python compatibility.
# %cd csv

from os import listdir
from os.path import isfile, join

myDir = r'./'
def listaDir(txtDIR):
    # função que retorna uma lista de arquivos de um diretório
    try:
        mypath = txtDIR
        onlyfiles = [f for f in listdir(mypath) if isfile(join(mypath, f))]
        return onlyfiles
    except:
        return ''

for myFile in listaDir(myDir):
  print(myFile)
```

```
  !gsutil -m cp -r -n "./{myFile}" gs://cosmobase-publica-bucket/PRODES/
  !rm "{myFile}"

# apaga o diretório
!perl -e 'for(<*>){((stat)[9]<(unlink))}'

./tif/ITAUBAL-19441.tif
WARNING:rasterio._env:CPLE_AppDefined in TIFFFetchNormalTag:IO error during
reading of "GeoPixelScale"; tag ignored
WARNING:rasterio._env:CPLE_AppDefined in TIFFFetchNormalTag:IO error during
reading of "GeoTiePoints"; tag ignored
WARNING:rasterio._env:CPLE_AppDefined in TIFFFetchNormalTag:IO error during
reading of "GeoKeyDirectory"; tag ignored
WARNING:rasterio._env:CPLE_AppDefined in TIFFFetchNormalTag:IO error during
reading of "GeoDoubleParams"; tag ignored
WARNING:rasterio._env:CPLE_AppDefined in TIFFFetchNormalTag:IO error during
reading of "GeoASCIIParams"; tag ignored
./tif/MAFRA-19442.tif

"""## Salva no bigquery"""

# Função genérica para atualizar tabelas
from google.cloud import bigquery
from google.cloud import storage
# Construct a BigQuery client object.
client = bigquery.Client('cosmobase-publica')

def atualiza_tabela(txt_arquivo, txt_tabela, dic_campos):
   # INÍCIO monta a string do schema
   txt_schema = '['
   for key in dic_campos:
     txt_schema = txt_schema + 'bigquery.SchemaField("' + key +'","' +
dic_campos[key] + '"),'
     #print(key, '->', dic_campos[key])
   txt_schema = txt_schema+']'
   # FIM monta a string do schema
   # Set table_id to the ID of the table to create.
   table_id = txt_tabela

   job_config = bigquery.LoadJobConfig(
       schema=eval(txt_schema),
       skip_leading_rows=1,
       max_bad_records=0,
       encoding='UTF-8',
       # The source format defaults to CSV, so the line below is optional.
       source_format=bigquery.SourceFormat.CSV,
       write_disposition=bigquery.WriteDisposition.WRITE_TRUNCATE,
   )
   job_config.field_delimiter = ';'
   uri = txt_arquivo
```

```
    load_job = client.load_table_from_uri(
        uri, table_id, job_config=job_config
    )  # Make an API request.

    load_job.result()  # Waits for the job to complete.

    destination_table = client.get_table(table_id)  # Make an API request.
    print("Loaded {} rows.".format(destination_table.num_rows))

# Processa todas as tabelas
from os import listdir

lista_arq = ['*.csv']


for arquivo in lista_arq:
    try:
        txt_dic_campos = """{'DN': 'STRING', 'geometry':'STRING'}"""
        txt_arquivo = 'gs://cosmobase-publica-bucket/MAPBIOMAS/'+ arquivo
        txt_tabela = 'cosmobase-publica.MAP_BIOMAS.uso_do_solo_colecao_7_2021'
        dic_campos = eval(txt_dic_campos)
        print(txt_arquivo, txt_tabela, dic_campos) # processa o job do bigquery
        atualiza_tabela(txt_arquivo, txt_tabela, dic_campos) # processa o job do bigquery
        print(txt_tabela, ' processada')
    except Exception as f:
        print(f)
        print(arquivo, ' não processado')
        pass



#!/usr/bin/env python

# coding: utf-8


# ## Scrapping da base de downloads do CAR - Cadastro Ambiental Rural

# ## https://www.car.gov.br/publico/municipios/downloads


# In[26]:



# importação de bibliotecas

import pandas as pd

import io
```

```
import os
from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.keys import Keys
from selenium.common.exceptions import WebDriverException
from selenium.webdriver.support.ui import Select
from selenium.webdriver.common.action_chains import ActionChains

import time
from datetime import datetime
from bs4 import BeautifulSoup
import random
from google.cloud import bigquery

# instancia o cliente bigquery com o json com a chave do bigquery
bigquery_client =
bigquery.Client.from_service_account_json(r'/Users/luishenriquerosatirocha/Desktop/c
osmobase-publica/cosmobase-publica-2c01b4830eca.json')

ENCODING = 'UTF-8'

import datetime
filename = r'/Users/luishenriquerosatirocha/Desktop/cosmobase-publica/chromedriver'
#driver = webdriver.Chrome(filename)



from selenium.webdriver.chrome.options import Options
chrome_options = Options()
chrome_options.add_argument('--ignore-certificate-errors')
```

```
driver = webdriver.Chrome(filename, chrome_options=chrome_options)




wait = WebDriverWait(driver, 2)
url = """https://www.car.gov.br/publico/municipios/downloads?sigla="""
driver.get(url+'RJ')
time.sleep(4)
driver.get(url+'RJ')


# In[27]:


# pega os municípios já importados
sql_shape = r'select distinct COD_ESTADO, NOM_MUNICI from cosmobase-publica.CAR.AREA_IMOVEL_corrigido order by COD_ESTADO asc, NOM_MUNICI asc'
sql_mun = r'select distinct uf COD_ESTADO, mun NOM_MUNICI from cosmobase-publica.CAR.IMOVEIS order by uf asc, mun asc'
df_shape = bigquery_client.query(sql_shape).to_dataframe()
df_mun = bigquery_client.query(sql_mun).to_dataframe()

def checaMun(txtUF,txtMun,txtTipo):
    # verifica se já importou o município de uma UF nos tipos planilha ou shape e retorna False se já importou
    NUM_REGISTROS = 0
    if txtTipo == 'shape':
        NUM_REGISTROS = eval(r'len(df_shape[(df_shape["COD_ESTADO"] == "' + txtUF + r'") & (df_shape["NOM_MUNICI"] == "' + txtMun + r'")].index)')
    else:
```

```
    NUM_REGISTROS = eval(r'len(df_mun[(df_mun["COD_ESTADO"] == "' + txtUF + r'") & (df_mun["NOM_MUNICI"] == "' + txtMun + r'")].index)')
    print('          ' + txtTipo)

    if NUM_REGISTROS>0:
        return(True) # troque se quiser checar o que já importou
        #return(False)
    else:
        return(True)



def captcha():

    input = wait.until(EC.element_to_be_clickable((By.ID,'form-email-download-base')))

    input.clear()
    input.send_keys('xxxxxxxxxx' + Keys.TAB)
    #div_mae = wait.until(EC.element_to_be_clickable((By.CLASS_NAME,"text-center")))
    #refresh = div_mae.find_elements_by_tag_name("button")[0]

    #refresh.click()




    from bestcaptchasolverapi3.bestcaptchasolverapi import BestCaptchaSolverAPI

    # Set access_token for authentication
    access_token = 'xxxxxxx'
    # get your access token from https://bestcaptchasolver.com/account
    bcs = BestCaptchaSolverAPI(access_token)
    img = wait.until(EC.element_to_be_clickable((By.XPATH,"//*[@id='img-captcha-base-downloads']")))
```

```
    import pyautogui
    # Improting Image class from PIL module
    from PIL import Image
    im2 = pyautogui.screenshot(r'/Users/luishenriquerosatirocha/Desktop/cosmobase-publica/CAR/captcha.png')
    # Opens a image in RGB mode
    im = Image.open(r'/Users/luishenriquerosatirocha/Desktop/cosmobase-publica/CAR/captcha.png')

    # Size of the image in pixels (size of orginal image)
    # (This is not mandatory)
    #width, height = im.size
    #print(width, height)

    # Setting the points for cropped image
    left = 1000
    top = 700
    right = 1500
    bottom = 900

    # Cropped image of above dimension
    # (It will not change orginal image)
    im1 = im.crop((left, top, right, bottom))
    #im1.show()
    im1.save(r'/Users/luishenriquerosatirocha/Desktop/cosmobase-publica/CAR/captcha2.png')
    #time.sleep(1)
    data = {}
    data['image'] = r'/Users/luishenriquerosatirocha/Desktop/cosmobase-publica/CAR/captcha2.png'

    # optional parameters
    data['is_case'] = True
    data['is_phrase'] = False
```

```
    data['is_math'] = False
    data['alphanumeric'] = all
    data['minlength'] = 4
    data['maxlength'] = 5

    captcha_id = bcs.submit_image_captcha(data)

    image_text = bcs.retrieve(captcha_id)['text']
    input = wait.until(EC.element_to_be_clickable((By.ID,'form-captcha-download-base')))
    input.clear()
    input.send_keys(image_text + Keys.TAB)
    botao = wait.until(EC.element_to_be_clickable((By.XPATH,"//*[@id='btn-baixar-dados']")))
    botao.click()
    #print('oi')
    try:
        deuErro = wait.until(EC.visibility_of_element_located((By.XPATH,"//*[@id='alert-download-error']")))
        if deuErro.get_attribute('class') == 'alert alert-warning':
            deuErro = True
            time.sleep(7)
        else:
            deuErro = False
    except e as exception:
        deuErro = False
    print('    ', deuErro, e)

    return(deuErro)


# In[76]:
```

```
# Improting Image class from PIL module
from PIL import Image
im2 = pyautogui.screenshot(r'/Users/luishenriquerosatirocha/Desktop/cosmobase-publica/CAR/captcha.png')
# Opens a image in RGB mode
im = Image.open(r'/Users/luishenriquerosatirocha/Desktop/cosmobase-publica/CAR/captcha.png')

# Size of the image in pixels (size of orginal image)
# (This is not mandatory)
#width, height = im.size
#print(width, height)

# Setting the points for cropped image
left = 1000
top = 700
right = 1500
bottom = 900

# Cropped image of above dimension
# (It will not change orginal image)
im1 = im.crop((left, top, right, bottom))
im1.show()
im1.save(r'/Users/luishenriquerosatirocha/Desktop/cosmobase-publica/CAR/captcha2.png')


# In[28]:


def pegaUF(txtUF):

    # faz scrapping de um estado
```

```
    url = """https://www.car.gov.br/publico/municipios/downloads?sigla="""
    driver.get(url+txtUF)
    # conecta no bigquery e pega a relação de UFs
    print(txtUF)
    window_before = driver.window_handles[0]
    #time.sleep(2)
    divPai = wait.until(EC.element_to_be_clickable((By.CLASS_NAME,'col-md-12')))
    childs = driver.find_elements_by_tag_name("div")
    for child in childs:
        if child.get_attribute("class")[:15] == 'lista-municipio':
            txtMun = child.get_attribute("data-municipio")
            print(txtMun)
            espera = wait.until(EC.element_to_be_clickable((By.CLASS_NAME,'col-md-12')))

        if child.get_attribute("class")[:9] == 'btn-group':
            buttons = child.find_elements_by_tag_name("button")
            for button in buttons:
                if ((button.get_attribute("title")[:16] == 'Baixar Shapefile' and checaMun(txtUF,txtMun,'shape')) or (button.get_attribute("title")[:10] == 'Baixar CSV') and checaMun(txtUF,txtMun,'planilha')):
                    try:
                        button.click()
                        input = wait.until(EC.element_to_be_clickable((By.ID,'form-email-download-base')))

                        for i in range(5):
                            try:
                                if not captcha():
                                    break
                            except:
                                pass
                        try:
```

```
                    #close                                                    =
wait.until(EC.element_to_be_clickable((By.XPATH,"//*[@id='modal-download-
base']/div/div/div[1]/button")))
                    close                                                     =
wait.until(EC.element_to_be_clickable((By.XPATH,"/html/body/div[1]/div/div[2]/div[
3]/div/div/div[1]/button")))
                    close.click()



                except:
                    try:
                        close                                                 =
wait.until(EC.element_to_be_clickable((By.XPATH,"//*[@id='modal-download-
base']/div/div/div[1]/button")))
                        close.click()
                    except:
                        pass

                    #time.sleep(1)
                except Exception as e:
                    print('erro na importação!', e)
                    #time.sleep(1)
                    pass

pegaUF('AC')


# In[29]:


txtUF = 'AC'
url = """https://www.car.gov.br/publico/municipios/downloads?sigla="""
driver.get(url+txtUF)
print(txtUF)
```

```
# conecta no bigquery e pega a relação de UFs
window_before = driver.window_handles[0]
divPai = wait.until(EC.element_to_be_clickable((By.CLASS_NAME,'col-md-12')))
childs = driver.find_elements_by_tag_name("div")
for child in childs:
    if child.get_attribute("class")[:15] == 'lista-municipio':
        txtMun = child.get_attribute("data-municipio")
        print(txtMun)
        if child.get_attribute("class")[:9] == 'btn-group':
            buttons = child.find_elements_by_tag_name("button")
            for button in buttons:
                if (button.get_attribute("title")[:16] == 'Baixar Shapefile' or
button.get_attribute("title")[:10] == 'Baixar CSV'):
                    print(button.get_attribute("title"))


# In[20]:


captcha()


# ### captcha()

# In[8]:


# pega subdiretorios de um diretorio e unzipa
from glob import glob

#myDir = """/Users/luishenriquerosatirocha/Desktop/cosmobase-publica/CAR/AC/*"""

def unzipa(txtPath):
    import os
```

```
    import zipfile
    from glob import glob
    for path, dir_list, file_list in os.walk(txtPath):
        for file_name in file_list:
            if file_name.endswith(".zip"):
                abs_file_path = os.path.join(path, file_name)

                # The following three lines of code are only useful if
                # a. the zip file is to unzipped in it's parent folder and
                # b. inside the folder of the same name as the file

                parent_path = os.path.split(abs_file_path)[0]
                output_folder_name = os.path.splitext(abs_file_path)[0]
                output_path = os.path.join(parent_path, output_folder_name)

                zip_obj = zipfile.ZipFile(abs_file_path, 'r')
                zip_obj.extractall(output_path)
                zip_obj.close()
    # deleta os arquivos zip
    files = glob(txtPath+'/*.zip')
    for f in files:
        os.remove(f)

def unzipaUF(txtUF):
    unzipa("""/Users/luishenriquerosatirocha/Desktop/cosmobase-publica/CAR/"""+
txtUF)
    diretorios          =          glob("""/Users/luishenriquerosatirocha/Desktop/cosmobase-
publica/CAR/""" + txtUF + "/*""")
    for diretorio in diretorios:
        print(diretorio)
        unzipa(diretorio)



unzipaUF('RN')
```

```
# In[9]:


def converteshp_csv(txtUF):
    import geopandas
    from glob import glob
    # lista os shapefiles do diretório
    # pega a lista de sub-diretoorios

    myDir    =    """/Users/luishenriquerosatirocha/Desktop/cosmobase-publica/CAR/"""+
txtUF + """/*/"""
    listSubDir = glob(myDir)

    #print(listSubDir)


    for itemDir in listSubDir[:]:
    #    print(itemDir+'*/')
        listSubDir2 = glob(itemDir+'*/')
    #    print(listSubDir2)


        for itemSubDir in listSubDir2:
            #print(itemSubDir[:-1])
            listfilesshp = glob(itemSubDir+'*.shp')
            for fileshp in listfilesshp:
                print(fileshp)
                myFileOutput = fileshp[:-3]+'csv'
                try:
                    df = geopandas.read_file(fileshp)
                    df[df.columns[:-1]].fillna('', inplace = True)
```

```
                df.to_csv(myFileOutput, index=False)
            except:
                pass
converteshp_csv('RN')


# In[10]:


def deletashp(txtUF):

    from glob import glob
    import os

    # lista os shapefiles do diretório
    # pega a lista de sub-diretoorios

    myDir    =    """/Users/luishenriquerosatirocha/Desktop/cosmobase-publica/CAR/"""+
txtUF + """/*/"""
    listSubDir = glob(myDir)

    #print(listSubDir)


    for itemDir in listSubDir[:]:
    #    print(itemDir+'*/')
        listSubDir2 = glob(itemDir+'*/')
    #    print(listSubDir2)


        for itemSubDir in listSubDir2:
            #print(itemSubDir[:-1])
            # deleta os arquivos zip
            files = glob(itemSubDir+'/*.shp')
```

```
        for f in files:
            os.remove(f)
        files = glob(itemSubDir+'/*.shx')
        for f in files:
            os.remove(f)
        files = glob(itemSubDir+'/*.prj')
        for f in files:
            os.remove(f)
        files = glob(itemSubDir+'/*.dbf')
        for f in files:
            os.remove(f)

deletashp('RN')


# In[86]:


sql = """ select distinct UF from cosmobase-publica.IBGE.populacao_estimativa_DOU
where _PARTITIONDATE = '2020-04-08' order by UF asc   """
df = bigquery_client.query(sql).to_dataframe()


# In[91]:




# In[10]:
```

```
close = wait.until(EC.element_to_be_clickable((By.XPATH,"/html/body/div[1]/div/div[2]/div[3]/div/div/div[1]/button")))
close.click()


# In[ ]:
```