

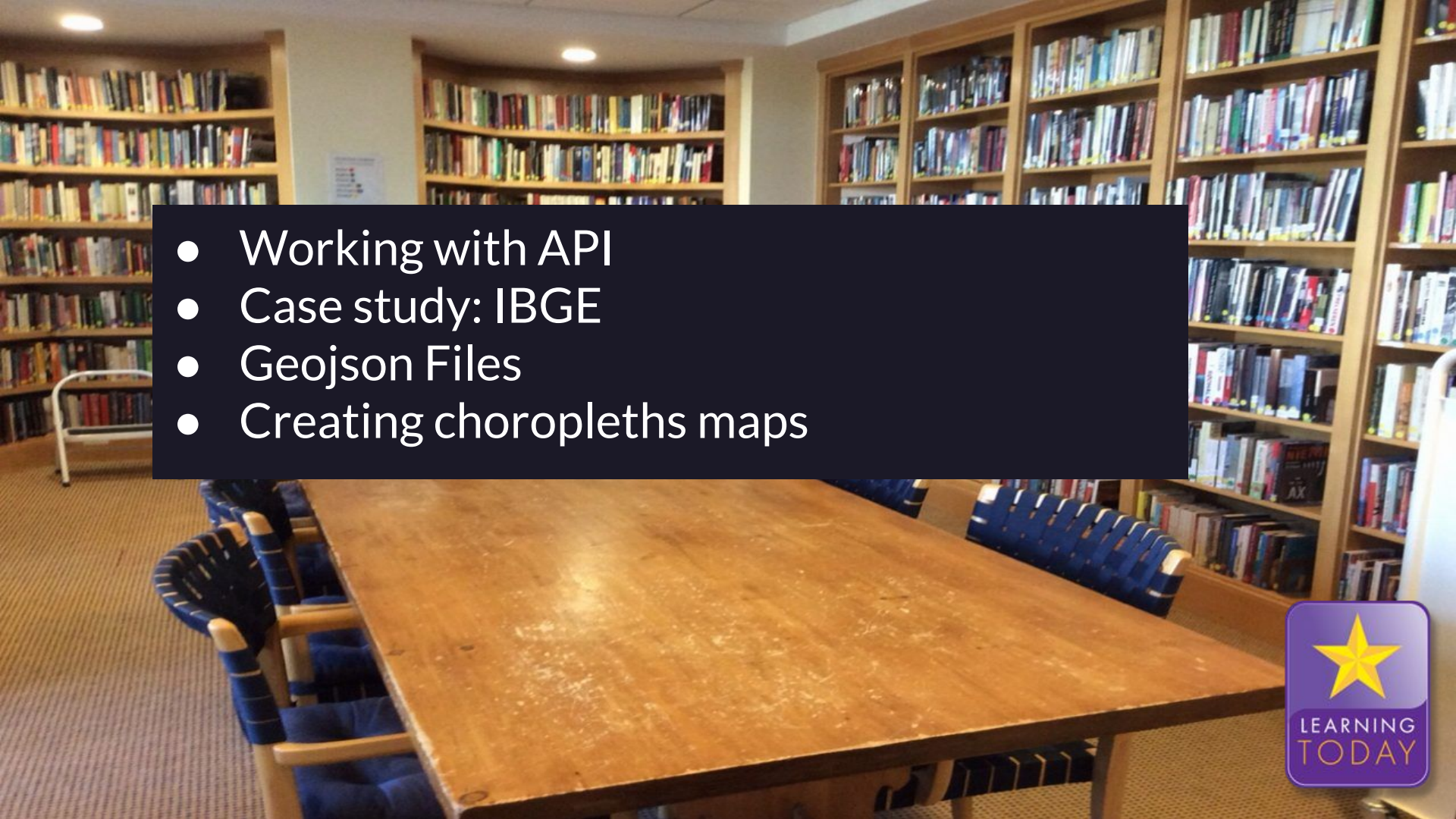


# *Lesson #07*

## *API and*

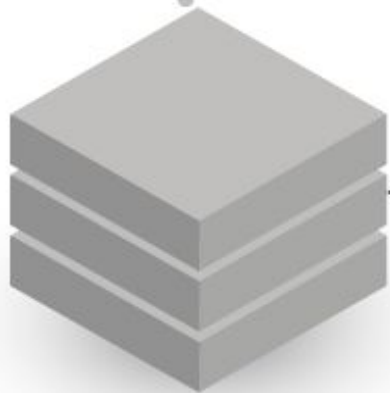
## *Choropleth Maps*

*August 2019*

- 
- Working with API
  - Case study: IBGE
  - Geojson Files
  - Creating choropleths maps

01

Database



02

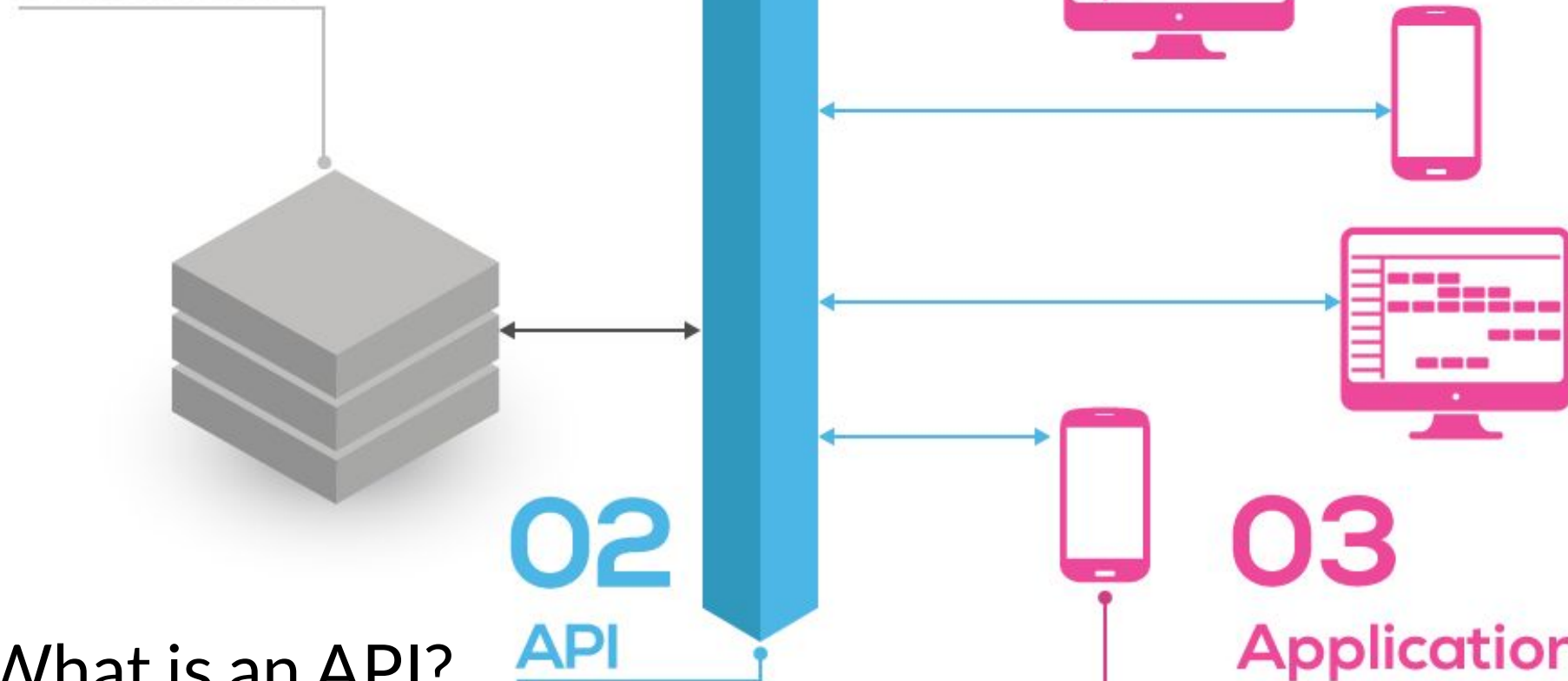
API

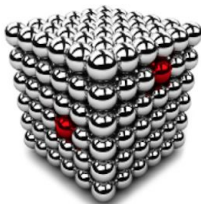


03

Applications

What is an API?





### Agregados

Análise multidimensional ao seu alcance.

2.0 | 3.0



### Calendário

Conheça o cronograma de ações e publicações do IBGE

3.0



### CNAE

API referente à Classificação Nacional de Atividades Econômicas

2.0



### Localidades

Obtenha os dados sobre as divisões administrativas do Brasil

1.0



### Malhas geográficas

Malhas com as mais diversas resoluções e formatos para você liberar a sua imaginação

2.0



### Metadados

Metadados associados às pesquisas

2.0

Tip: view in any  
JSON Online

**População  
Estimada**



Produto Interno Bruto dos Municípios 2002-2016

14/12/2018



Registro Civil 2017

31/10/2018



Abate

Leite

Couro

POG

LSPA

Estoques

IPCA

INPC

IPCA15

Sinapi

PIM-PF/BR

PIM-PF/RG

IPP

PNADC/M

PNADC/T

PMC

**PMS**

CNT

Pesquisa Mensal de Serviços - janeiro 2019



Dados divulgados na sexta-feira, 15 de março de 2019 - 09:00:00

Local:



Índice e variação do volume de serviços, segundo as atividades de serviços (Brasil - janeiro 2019)



# Consulta API Sidra

A API Sidra permite a extração de dados do Banco Sidra, para utilização em programas e aplicativos web, através da utilização de uma linguagem de programação (JavaScript, C#, Java etc).

Para que o desenvolvedor possa testar o acesso à API antes de implementá-lo em seu programa, foi desenvolvida a facilidade a seguir, que parte do princípio de que já se conhece a tabela Sidra de onde os dados devem ser extraídos.

---

Clique [para obter ajuda sobre a API](#), seus parâmetros para extração e para formatação de dados de uma tabela Sidra.

Clique [para ver um exemplo de uso em JavaScript](#) (após o carregamento da página visualize o seu código fonte).

Clique [para baixar um exemplo de uso em C#](#) (projeto em Visual Studio 2010).

---

Para saber quais parâmetros e valores utilizar na API para extrair dados de uma tabela Sidra, informe abaixo o nro. da tabela e clique em Descrever.

Tabela:

---

Para testar a extração de dados de uma tabela pela API Sidra, informe na linha abaixo os parâmetros e seus valores (Ex: /t/1612/n2/all/v/all/p/last/c81/2702/f/u). Ao final, clique em Consultar.

Parâmetros/valores da API:

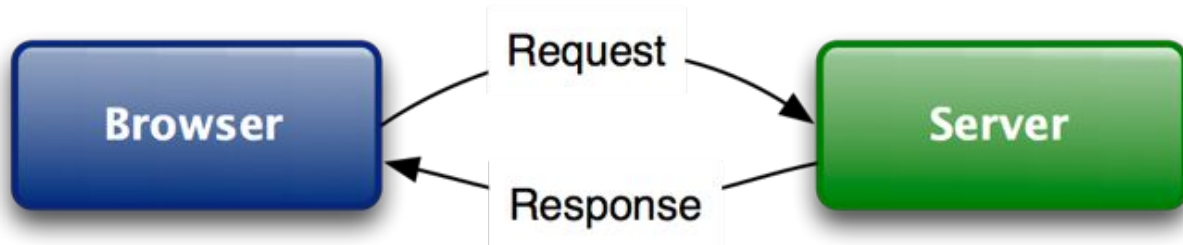
---

<http://api.sidra.ibge.gov.br/>

# HyperText Transfer Protocol (HTTP)

---

- Foundation of data communication for the web
- HTTP is the protocol that is used by web servers and browsers to communicate.
- HTTP is based on a request and a response.



# GET requests using a higher-level request lib.



- One of the most downloaded Python packages

**Used by:** Twitter, Spotify, Microsoft, Amazon, Lyft, BuzzFeed, Reddit, The NSA, Her Majesty's Government, Google, Twilio, Runscope, Mozilla, Heroku, PayPal, NPR, Obama for America, Transifex, Native Instruments, The Washington Post, SoundCloud, Kippt, Sony, and Federal U.S. Institutions that prefer to be unnamed claim to use Requests internally.



```
import requests
# configure a generical header
headers = {
    'Content-Type': 'application/json;charset=UTF-8',
    'User-Agent': 'google-colab',
    'Accept': 'application/json, text/plain, */*',
    'Accept-Encoding': 'gzip, deflate, br',
    'Accept-Language': 'pt-BR,pt;q=0.9,en-US;q=0.8,en;q=0.7',
    'Connection': 'keep-alive',
}
response = requests.get("https://servicodados.ibge.gov.br/api/v1/localidades/estados",
                        headers=headers)
```

# Understanding status code

---

`response.status_code`

[https://en.wikipedia.org/wiki/List\\_of\\_HTTP\\_status\\_codes](https://en.wikipedia.org/wiki/List_of_HTTP_status_codes)

- **200** - Everything went okay, and the server returned a result (if any).
- **301** - The server is redirecting you to a different endpoint. This can happen when a company switches domain names, or an endpoint's name has changed.
- **401** - The server thinks you're not authenticated. This happens when you don't send the right credentials to access an API (we'll talk about this in a later mission).
- **400** - The server thinks you made a bad request. This can happen when you don't send the information the API requires to process your request, among other things.
- **403** - The resource you're trying to access is forbidden; you don't have the right permissions to see it.
- **404** - The server didn't find the resource you tried to access.

```
https://servicodados.ibge.gov.br/api/v1/localidades/estados
```

## Parâmetros

## Respostas

Status: 200 - Array de Unidades da Federação

### Schema

```
▼ [
  ▼ {
    id: ▼ number
      Identificador da Unidade da Federação

    nome: ▼ string
      Nome da Unidade da Federação

    sigla: ▼ string
      Sigla da Unidade da Federação

    regioao: ▼ {
      id: ► number
      nome: ► string
      sigla: ► string
    }
  }
]
```

```
import json
response.json()
```

```
response.content
```

```
b' [
  {
    "id":11,
    "sigla":"RO",
    "nome":"Rond\xc3\xba4nia",
    "regiao":{
      "id":1,
      "sigla":"N",
      "nome":"Norte"
    }
  },{
    "id":12,
    "sigla":"AC",
    "nome":"Acre",
    "regiao":{
      "id":1,
      "sigla":"N",
      "nome":"Norte"
    }
  },
  ...
  ...
  ...
]
```

# Python JSON

## JavaScript Object Notation

`dumps()` -- Takes in a Python object, and converts it to a string  
`loads()` -- Takes a JSON string, and converts it to a Python object

```
# Make a dictionary
```

```
cities_population = {  
    "Currais Novos": 44000,  
    "Caico": 67000,  
    "Acari": 11000  
}
```

```
# We can also dump a dictionary to a string
```

```
cities_population_string = json.dumps(cities_population)  
cities_population_string
```

```
'{"Currais Novos": 44000, "Caico": 67000, "Acari": 11000}'
```

```
json.loads(cities_population_string)
```

```
{"Currais Novos": 44000, "Caico": 67000, "Acari": 11000}
```



## Perfil

**Consulte o perfil  
da sua localidade**

estado

região metropolitana

município

unidade de desenvolvimento humano

Busca





# Lesson #07 - Choropleth Maps.ipynb

## Section #01 Working with APIs

# Geojson

---

GeoJSON is a format for encoding a variety of geographic data structures.

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  },
  "properties": {
    "name": "Dinagat Islands"
  }
}
```

<http://geojson.org/>

GeoJSON supports the following geometry types:

- Point
- LineString
- Polygon
- MultiPoint
- MultiLineString
- MultiPolygon

# Malhas

## API e documentação

Versão: 2.0.0



Malhas referentes às unidades administrativas do Brasil. O formato renderizado será de acordo com o parâmetro Accept do cabeçalho HTTP

```
image/svg+xml      - SVG
application/vnd.geo+json - Geo JSON
application/json    - TopoJSON
```

obs 1: consulte os identificadores das localidades por meio da [API de Localidades](#)

## Malha por identificador

Obtém a malha referente ao identificador da localidade.

GET

```
http://servicodados.ibge.gov.br/api/v2/malhas/{id}
```

```
response = requests.get("https://servicodados.ibge.gov.br/api/v2/malhas/24/"+  
                        "?formato=application/vnd.geo+json&resolucao=5",  
                        headers=headers)
```

```
data_json = response.json()
```

```
{'crs': {'properties': {'name': 'urn:ogc:def:crs:EPSG::4674'}, 'type': 'name'},  
  'features': [{'geometry': {'coordinates': [[[-36.6045, -6.2903],  
        [-36.5649, -6.3122],  
        [-36.5479, -6.3443],  
        [-36.5424, -6.3729],  
        [-36.5043, -6.3864],  
        ...  
        ...  
        'type': 'Polygon'],  
        'properties': {'centroide': [-36.650141304369136, -6.409156513938322],  
        'codarea': '2400109'},  
        'type': 'Feature'},  
    {'geometry': {'coordinates': [[[-37.0512, -5.294],  
        [-37.0294, -5.3256],  
        [-36.9284, -5.3537],  
        [-36.8564, -5.4008],  
        [-36.8384, -5.4037],  
        ...  
        ...
```

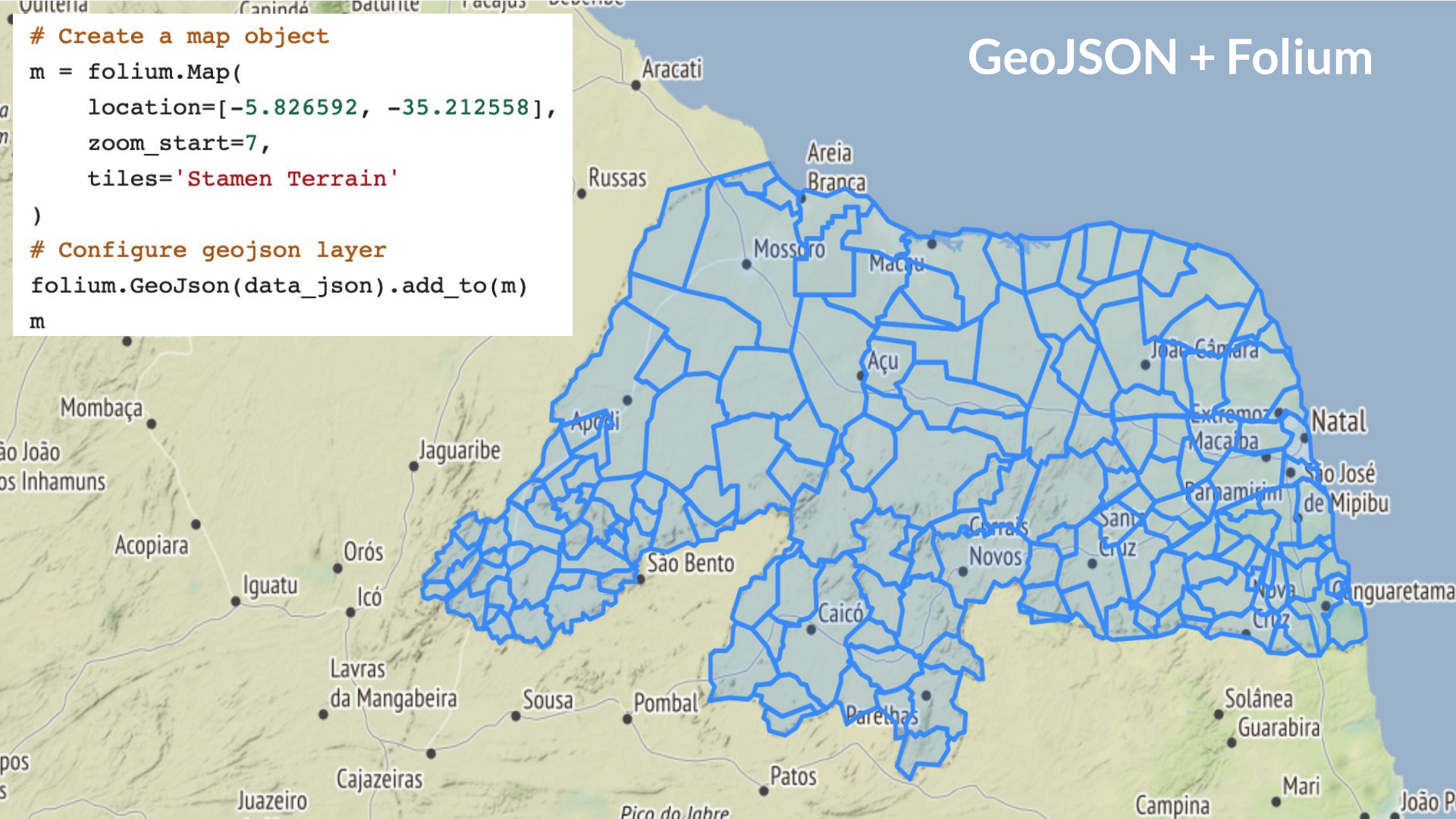
# GeoJSON + Folium

```
# Create a map object
```

```
m = folium.Map(  
    location=[-5.826592, -35.212558],  
    zoom_start=7,  
    tiles='Stamen Terrain'  
)
```

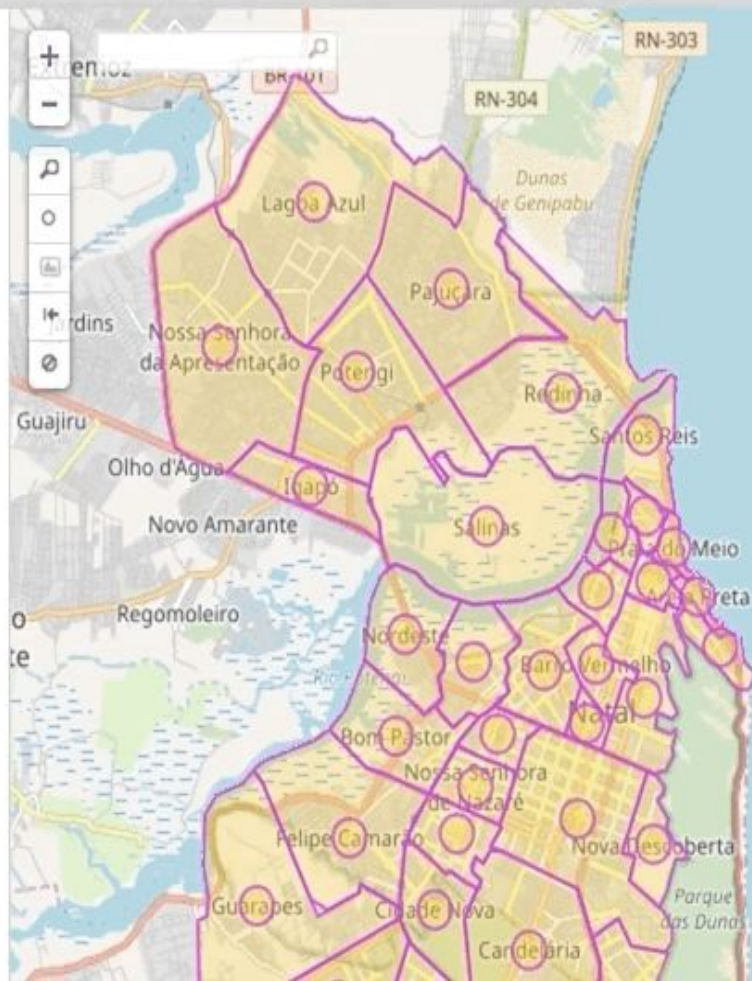
```
# Configure geojson layer
```

```
folium.GeoJson(data_json).add_to(m)  
m
```

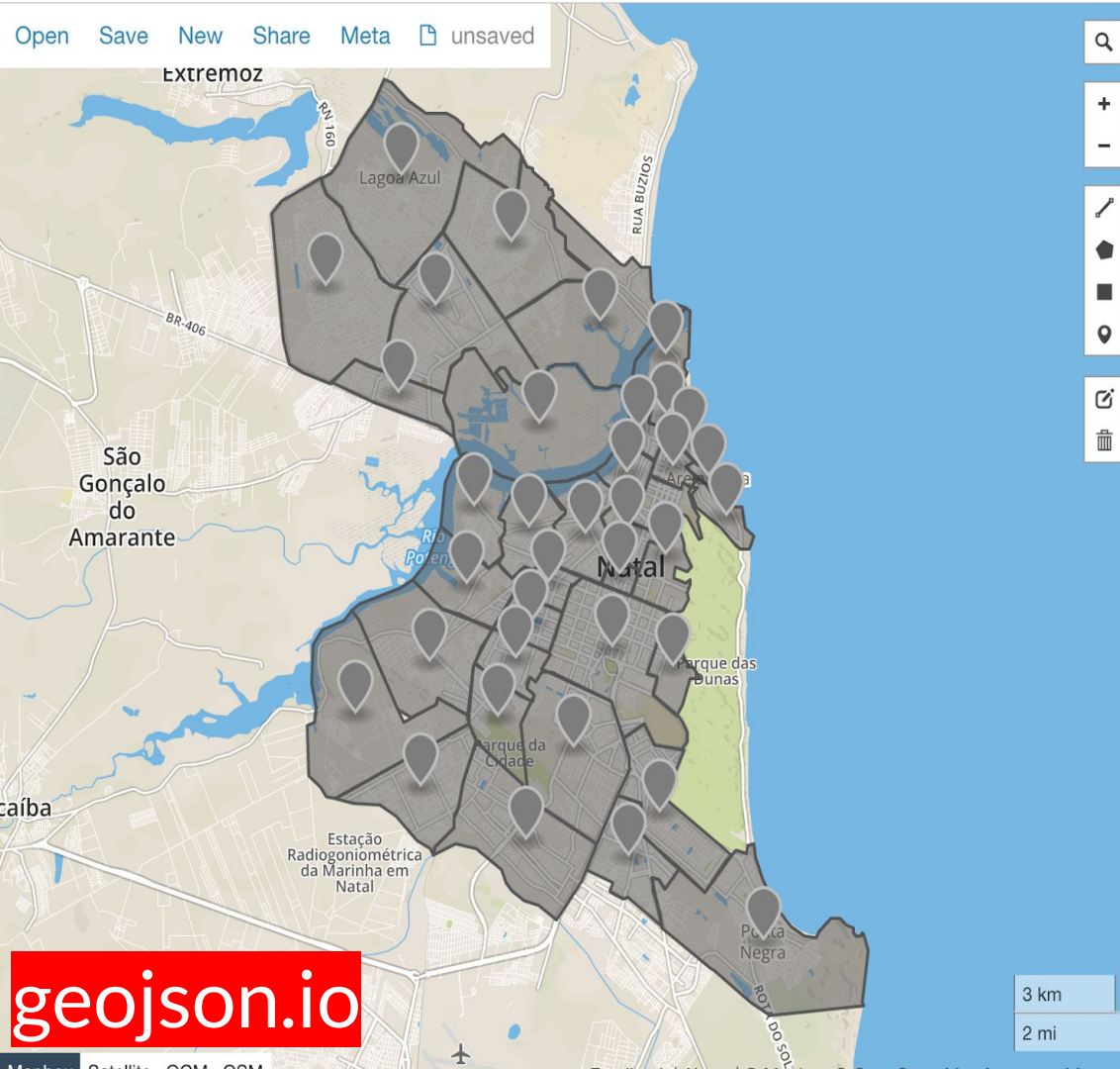




```
1 [out:json][timeout:25];
2 {{geocodeArea:Natal RN Brasil}}->.searchArea;
3 (
4   relation["admin_level"="10"](area.searchArea);
5 );
6 out body;
7 >;
8 out skel qt;
```







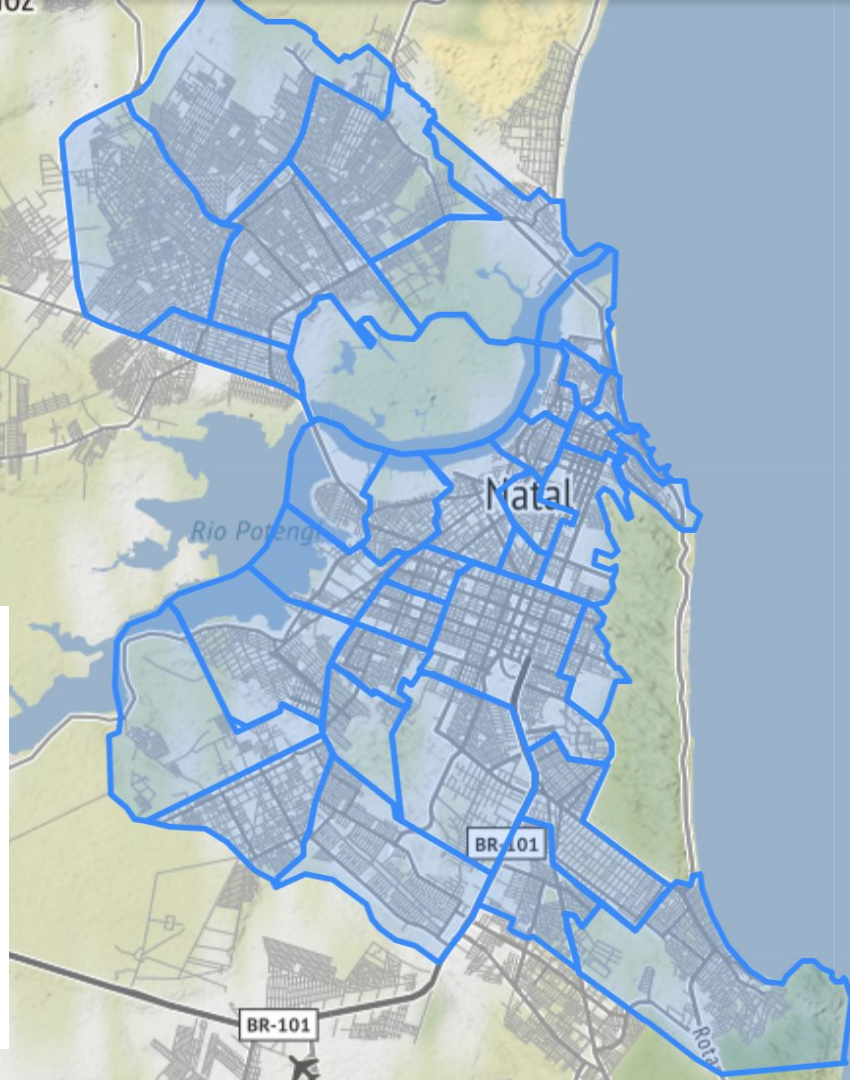
```
1 {
2   "type": "FeatureCollection",
3   "generator": "overpass-side",
4   "copyright": "The data included in this document",
5   "timestamp": "2019-03-17T18:18:02Z",
6   "features": [
7     {
8       "type": "Feature",
9       "properties": {
10        "@id": "relation/388146",
11        "admin_level": "10",
12        "boundary": "administrative",
13        "is_in": "Natal",
14        "name": "Pitimbu",
15        "place": "suburb",
16        "type": "boundary"
17      },
18      "geometry": {
19        "type": "Polygon",
20        "coordinates": [
21          [
22            [
23              -35.2417636,
24              -5.8441346
25            ],
26            [
27              -35.2422443,
28              -5.8437674
```

```
# Create a map object
```

```
m = folium.Map(  
    location=[-5.826592, -35.212558],  
    zoom_start=11,  
    tiles='Stamen Terrain'  
)
```

```
# Configure geojson layer
```

```
folium.GeoJson(geo_json_natal).add_to(m)  
m
```





# Lesson #07 - Choropleth Maps.ipynb

## Section #02 Working with GeoJSON



+

-



Extremoz

331

11,527

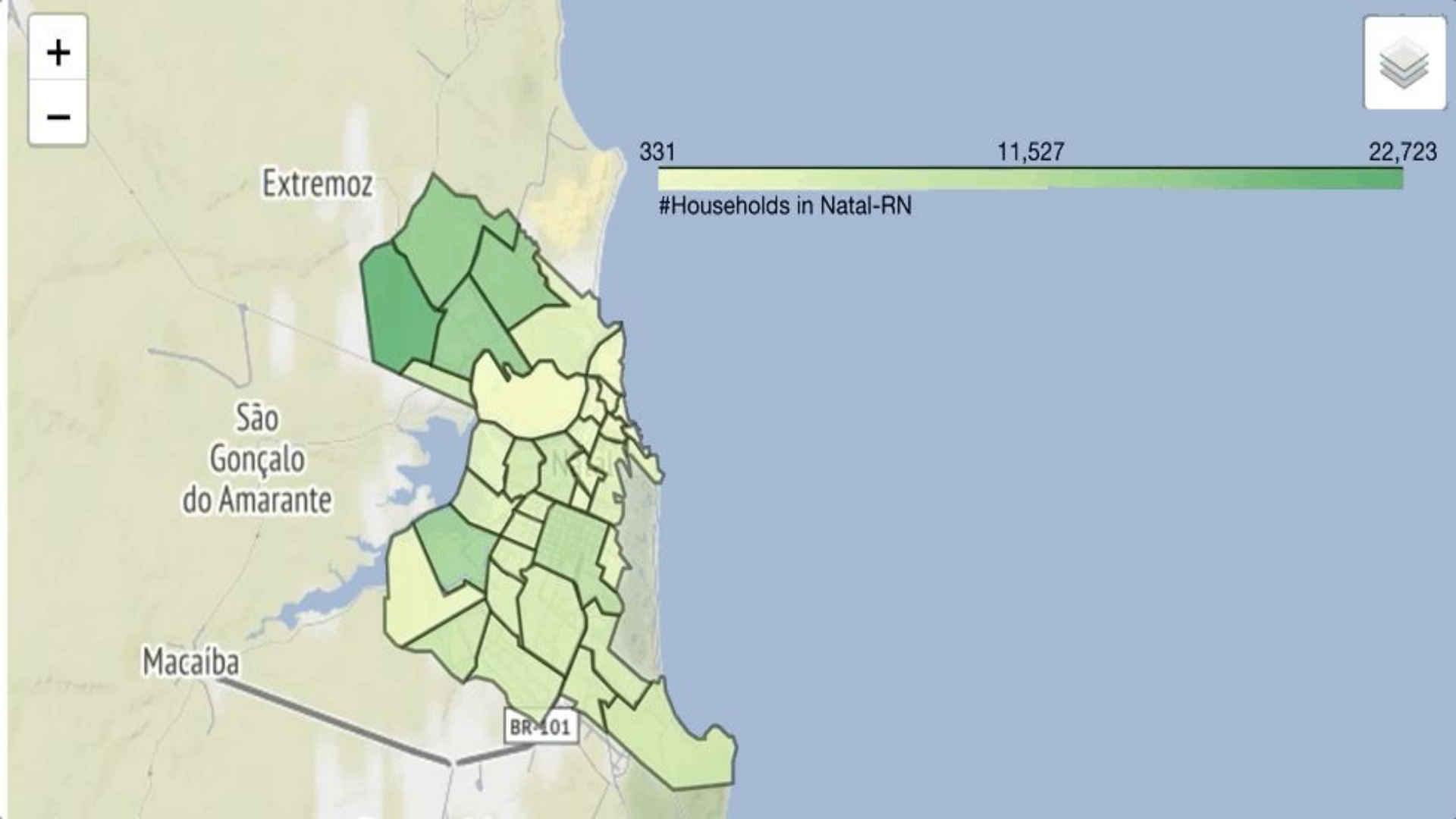
22,723

#Households in Natal-RN

São  
Gonçalo  
do Amarante

Macaíba

BR-401



# #01 Preparing the Data

```
# some neighborhoods names are different in IBGE API and GeoJSON file
neigh_df.loc[neigh_df.name == "Pitimbú", 'name'] = "Pitimbu"
neigh_df.loc[neigh_df.name == "Mãe Luíza", 'name'] = "Mãe Luiza"
neigh_df.loc[neigh_df.name == "Filipe Camarão", 'name'] = "Felipe Camarão"
neigh_df.loc[neigh_df.name == "Guarapés", 'name'] = "Guarapes"
```

```
neigh_df.set_index('name', inplace=True)
neigh_df.index.name = None
neigh_df["name"] = neigh_df.index
neigh_df.head()
```

	households	neighborhood_id	name
<b>Santos Reis</b>	1531	2408102001	Santos Reis
<b>Praia do Meio</b>	1620	2408102002	Praia do Meio
<b>Rocas</b>	3067	2408102003	Rocas
<b>Ribeira</b>	764	2408102004	Ribeira
<b>Petrópolis</b>	1733	2408102005	Petrópolis

## #02 Create a Colormap Bar

```
from branca.colormap import linear

# colormap yellow and green (YlGn)
colormap = linear.YlGn_03.scale(
    neigh_df.households.min(),
    neigh_df.households.max())

colormap.caption="#Households in Natal-RN"
```





## #03 Personalize GeoJSON file/function

```
# Insert additional information ('households') into GeoJSON file
for neigh in geo_json_natal['features']:
    name_aux = neigh['properties']['name']
    neigh['properties']['households'] = str(neigh_df.loc[name_aux, "households"])

# Create a Choropleth using folium.GeoJson()
folium.GeoJson(geo_json_natal,
               name='Households',
               style_function=lambda x: {'fillColor': colormap(neigh_df.loc[x['properties']['name'],
                                                                           "households"]),
                                          'color': 'black', 'weight': 2, 'fillOpacity': 0.8},
               tooltip=folium.GeoJsonTooltip(fields=['name', "households"],
                                             aliases=['Name:', "Households:"],
                                             localize=True)
               ).add_to(m)
```



```
index.js
import React, { useState } from 'react';
import './index.css';
import './index.html';
import './index.js';

function App() {
  const [contacts, setContacts] = useState([]);
  const [name, setName] = useState('');
  const [phone, setPhone] = useState('');
  const [email, setEmail] = useState('');

  const handleSubmit = (e) => {
    e.preventDefault();
    setContacts([...contacts, { name, phone, email }]);
    setName('');
    setPhone('');
    setEmail('');
  };

  return (
    <div>
      <h1>React Form</h1>
      <form>
        <input type="text" value={name} onChange={e => setName(e.target.value)} />
        <input type="text" value={phone} onChange={e => setPhone(e.target.value)} />
        <input type="text" value={email} onChange={e => setEmail(e.target.value)} />
        <button type="submit" value="Submit" />
      </form>
      <table>
        <caption>Contacts</caption>
        <thead>
          <tr>
            <th>Name</th>
            <th>Phone</th>
            <th>Email</th>
          </tr>
        </thead>
        <tbody>
          {contacts.map((contact) => (
            <tr>
              <td>{contact.name}</td>
              <td>{contact.phone}</td>
              <td>{contact.email}</td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  );
}

export default App;
```

```
index.html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>React Form</title>
  </head>
  <body>
    <div>
      <h1>React Form</h1>
      <form>
        <input type="text" value="" />
        <input type="text" value="" />
        <input type="text" value="" />
        <button type="submit" value="Submit" />
      </form>
      <table>
        <caption>Contacts</caption>
        <thead>
          <tr>
            <th>Name</th>
            <th>Phone</th>
            <th>Email</th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td></td>
            <td></td>
            <td></td>
          </tr>
        </tbody>
      </table>
    </div>
  </body>
</html>
```