Vaga Cientista de Dados Jr. SolarView

Você está trabalhando em um projeto que tem o objetivo de prever os níveis de irradiação solar dentro do territorio brasileiro. Para conseguir os dados você irá utilizar bases de dados geradas a partir da API da Nasa.

Portanto esse desáfio consiste em:

1 - Modelar um banco de dados (MySQL) para armazenar os dados que você irá trazer chamando a API.

2 - Criar um script que irá fazer a leitura da API da Nasa.

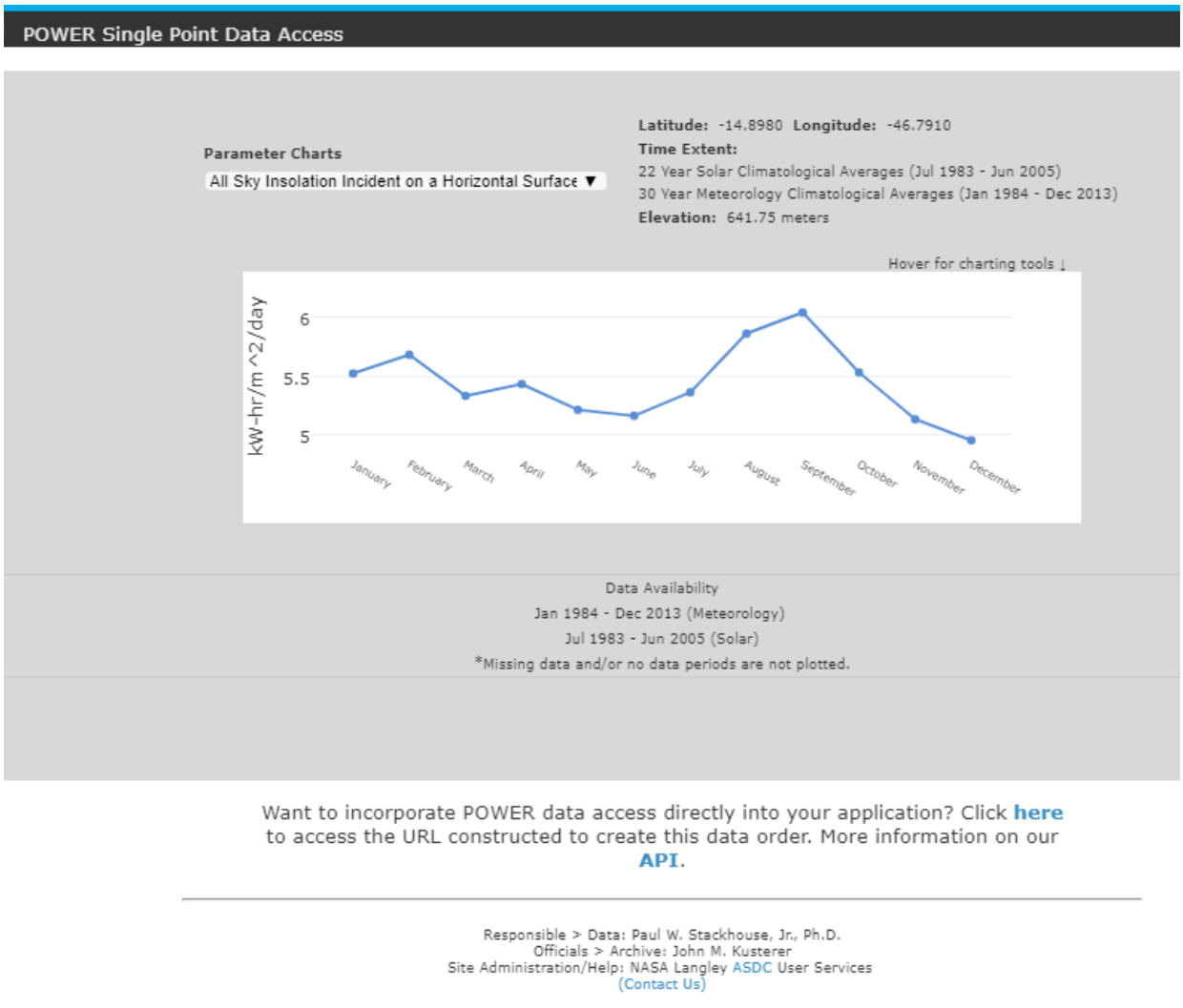
3 - Criar uma função que irá criar um banco de dados com base nos dados retirados da API (enviar também o SQL para gerar o banco).

4 - Realizar uma análise explorátoria dos dados e gerar uma visualização adequada da irradiação solar em territorio nacional.

- Dado da API que vai te ajudar na tarefa: ALLSKY_SFC_SW_DWN
- Parâmetro para analisar pelo POWER Data Access Viewer: All Sky Insolation Incident on a Horizontal Surface.
- Inputs de dados: Latitude e Longitude

Tarefa 4 parte 1

Plotar a média histórica da irradiação mensal para uma localidade no território brasileiro. Um gráfico onde no eixo X teremos meses (jan-dez) e no eixo Y irradiação média mensal. Segue um exemplo:



Links principais:

POWER Data Acess Viewer (https://power.larc.nasa.gov/data-access-viewer/)

NASA Prediction of Worldwide Energy Resources (https://power.larc.nasa.gov/)

Documentação API NASA (https://power.larc.nasa.gov/docs/v1/index.html)

Orientações:

- Você poderá usar tanto Python como R para realizar o desáfio.
- Uma das partes do desáfio é o entendimento de um problema com base em informações de terceiros (API da NASA).
- A forma como você vai apresentar o desáfio pode variar muito. Por exemplo, se você trabalha com R, você pode usar uma dashboard em Shiny para apresentar o resultado. Lembre, uma tarefa de data science é a gestão colaborativa de dados.
- Seu codigo deverá ser claro e conciso, e o processo deverá estar documentado.
- Você pode (e deve) compartilhar seu desafio no GitHub.

Perguntas:

O que significa enérgia fotovoltaica?

- Energia fotovoltaica é a energia obtida a partir da conversão direta da luz solar, atravéz das células fotovoltaicas. A célula fotovoltaica é um dispositivo fabricado de materiais semicondutores que possibilita o aproveitamento do efeito fotoelétrico.
- O efeito fotoelétrico foi explicado por Albert Einstein (o que deu a ele o Premio Nobel de Fisica). Esse efeito consiste na emissão de elétrons por um material, quando este é exposto a uma radiação eletromagnética (luz, por exemplo).
- A celula fotovoltaica apresenta a estrutura semelhante a de um diodo, com duas camadas (P e N) sobrepostas. A luz incide sobre a camada N, que emite eletrôns para a camada P, gerando uma diferença de potêncial.

O que são séries temporais?

- Series temporais são sequências de numeros indexados por sua ordem cronológica. Alguns exemplos de series temporais são: Incidencia mensal de chuvas, preços de criptomoedas, registros de eletroencefalograma.
- As series temporais são importantes em diversas areas do conhecimento e a partir disso desenvolveu-se o campo de análise de series temporais, que desenvolve métodos para obtenção de informação util a partir das series temporais.

Que algoritimo de machine learning você pensou em utilizar?

- Poderiam ser realizados algumas formas de preprocessamento, como decomposiçao en
- Para gerar predições da serie temporal em questão poderiamos utilizar um método tradicional, como o ARIMA (ou Seasonal ARIMA adaptado para series com componentes de sasonalidade) ou então tecnicas que utilizam redes CNN ou LSTM. As CNNs para series temporais realizam a convolução do sinal com filtros que são aprendidos ao longo do treinamento, e as LSTMs, que foram projetadas para series temporais, contêm componentes de memória em suas unidades, fazendo-as adequadas para reconhecer padrões temporais complexos.

Como a SolarView utiliza dados para gerar valor para seus clientes?

- A SolarView coleta dados de unidades de geração de energia solar, utilizando dataloggers e medidores (com hardware próprio que pode ser integrado a uma ampla gama de inversores de frequência), e os disponibiliza de forma agregada em plataformas Web/Mobile.
- Dentre as principais funcionalidades oferecidas pelo produto estão a gestão centralizada de todo sistema de geração de energia, independente do fabricante dos equipamentos, a auditoria de valores de leitura e consumo antes mesmo da chegada da fatura mensal e gestão inteligente da manutenção.
- Outras entregas interessantes poderiam ser realizadas cruzando informações de predições por demanda de energia, predições de quantidade de energia produzida com os indices de produçao dos clientes e também no auxilio a escolha de local para futuras instalações de geração de eergia solar.

Extração de dados

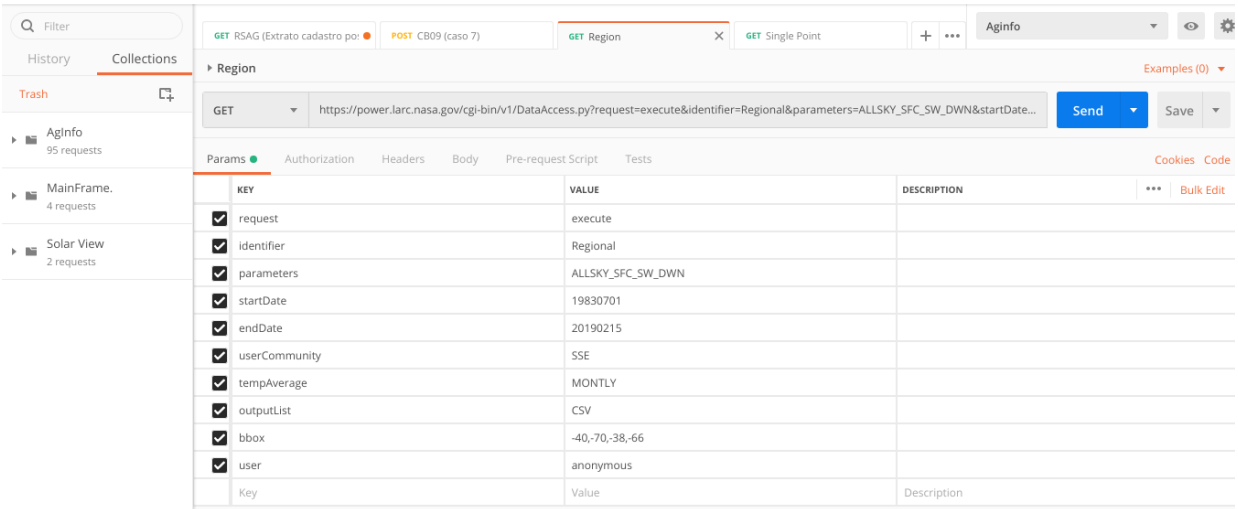
Consulta a API da NASA

Devido a uma indisponibilidade tecnica dn API da NASA durante o final de semana, as propostas do desafio foram cumpridas com dados arbitrarios, obtidos durante testes preliminares na API da NASA, sem levar em conta os critérios de localização (eram necessários dados do territorio nacional, porém estou utilizando dados de uma área da Argentina) e nem de amostragem (poderiam ser utilizados dados amostrados mensalmente, porém, utilizei dados amostrados diáriamente).

Utilizei o Postman para explorar a API, e lá obtive os dados que foram utilizados no desafio. Não criei um script para realizar as consultas devido a indisponibilidade da API no fim de semana (16 e 17/02/2019), mas explicarei o fluxo geral da consulta:

Na consulta especificamos a região requisitada, o intervalo de tempo, a taxa de amostragem (Ex.: DAILY, MONTLY), o formato de retorno dos dados (Ex.: CSV, JSON, ASCII) e as variáveis necessárias.

Após um tempo de processamento do servidor era retornada uma resposta, contendo um link para o arquivo de dados criado, contêndo os dados selecionados na consulta (o arquivo é apagado do servidor da NASA 48h após a criação).



Criação do banco de dados

Foi utilizado um banco de dados mySQL e a interface de administração phpMyAdmin. Mais detalhes sobre o ambiente ver o arquivo create_env.sh (create_env.sh), que sobe os containers necessários. Segue print do banco de dados modelado no phpMyAdmin:

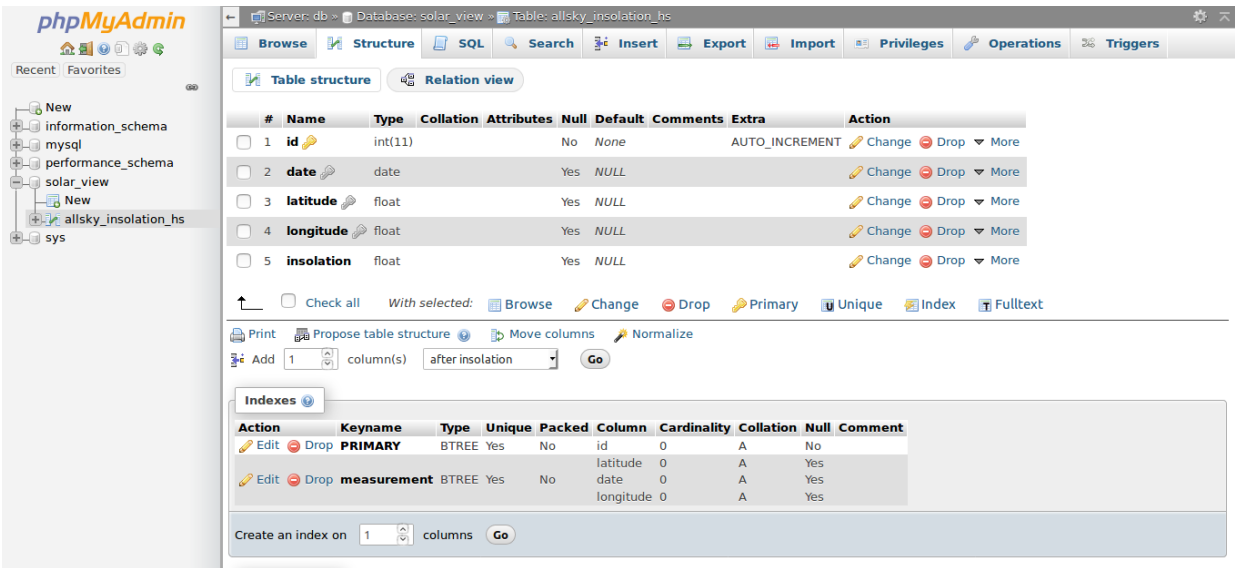
In [ ]:
```
# COMANDO SQL usado para criar o banco
CREATE TABLE `solar_view`.`allsky_insolation_hs` (
    `id` INT NOT NULL AUTO_INCREMENT ,
    `date` DATE NULL DEFAULT NULL ,
    `latitude` FLOAT NULL DEFAULT NULL ,
    `longitude` FLOAT NULL DEFAULT NULL ,
    `insolation` FLOAT NULL DEFAULT NULL ,
    PRIMARY KEY (`id`),
    UNIQUE `measurement` (`latitude`, `date`, `longitude`))
ENGINE = InnoDB;
```

Observações

A modelagem utiliza uma chave primaria chamada `id` com incremento automático, e uma chave composta unica chamada `meassurement` que assegura que assegura que só haja uma entrada de `insolation` no banco para cada uma das combinações dos campos `latitude` , `date` , `longitude` .

Inserção dos dados no banco

Com o arquivo retornado pela consulta a API da NASA, será realizada a adaptação de formatos para o banco e seu conteúdo será inserido no mesmo.



In [15]:
```
# importação de bibliotecas necessarias
from datetime import date
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from pathos.multiprocessing import ProcessPool
from mysql.connector import connection
import gmaps
sns.set_style('whitegrid')
```

In [39]:
```
# leitura do arquivo CSV obtido. São omitidas as 10 primeiras linhas do arqu
ivo, que contêm metadados.
file = 'POWER_Regional_Daily_19830701_20190210_e983fe72.csv'
data = pd.read_csv(file, skiprows=10)
# Transforma o formato da informação de data
data['date'] = list(map(date, data.pop('YEAR'), data.pop('MO'), data.pop('DY
')))
data[:5]
```

Out[39]:

	LAT	LON	ALLSKY_SFC_SW_DWN	date
0	-39.75	-69.75	1.71	1983-07-01
1	-39.75	-69.25	1.71	1983-07-01
2	-39.75	-68.75	1.57	1983-07-01
3	-39.75	-68.25	1.57	1983-07-01
4	-39.75	-67.75	1.31	1983-07-01

Criação da query e da conexão com o banco

A função `insert_data` abre e fecha uma conexão com o banco para cada entrada a ser inserida. Foi feita dessa forma pois a conexão com o banco não é thread-safe, o que gera problemas quando o objeto é utilizado por mais de um processo.

In [2]:
```
add_entry = ('INSERT INTO `allsky_insolation_hs` '
             '(`date`, `latitude`, `longitude`, `insolation`)'
             ' VALUES (%(date)s, %(LAT)s, %(LON)s, %(ALLSKY_SFC_SW_DWN)s)')


def connect():
    return connection.MySQLConnection(
        user='root',
        password='admin',
        database='solar_view'
    )


def insert_data(data):
    cnx = connect()
    cursor = cnx.cursor()
    data = {k: float(v) for k, v in dict(data[1]).items()}
    try:
        cursor.execute(add_entry, data)
    except mysql.connector.IntegrityError as err:
        pass
    cnx.commit()
    cursor.close()
    cnx.close()
```

Inserção paralelizada dos dados no banco

Utilizando a biblioteca `pathos` foi realizada a inserção paralelizada dos dados no banco.

In [ ]:
```
pool = ProcessPool(nodes=4)
pool.map(insert_data, data.iterrows())
```

Após esse processo estamos com o banco de dados preenchido com o histórico diário de incidencia solar (desde 1983) para uma área retangular compreendida pela Argentina:

A mostrar registos de 0 - 24 (540962 total, A consulta demorou 0.0002 segundos.)

```
SELECT * FROM `allsky_insolation_hs`
```

Perfil [Ed

1 > >> | Número de registos: 25 Filtrar registos: Pesquisar esta tabe

+ Opções

	id	date	latitude	longitude	insolation
Edita Copiar Apagar	27853	1983-09-20	-37.75	-69.25	3.18
Edita Copiar Apagar	27854	1983-09-20	-37.75	-68.75	3.68
Edita Copiar Apagar	27855	1983-09-20	-37.75	-69.75	3.18
Edita Copiar Apagar	27856	1983-09-20	-37.75	-67.75	3.84
Edita Copiar Apagar	27857	1983-09-20	-37.75	-68.25	3.68
Edita Copiar Apagar	27858	1983-09-20	-37.75	-67.25	3.84
Edita Copiar Apagar	27859	1983-09-20	-37.75	-66.75	4.54

Análise exploratória dos dados

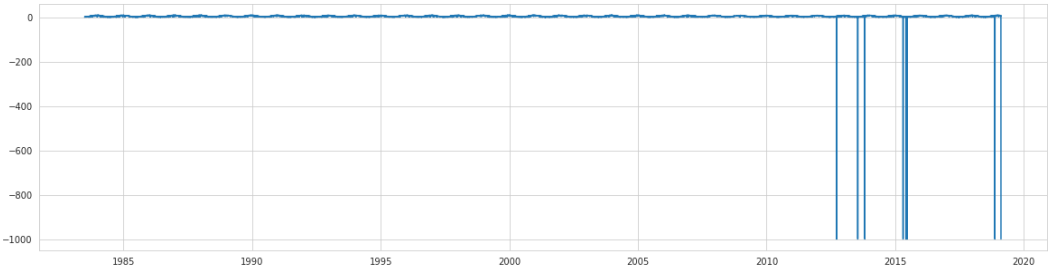
Seleção dos dados históricos de um lugar especifico

Foi selecionado um ponto arbitrário dos dados colhidos e para ele será realizada a plotagem do gráfico de media histórica de insolação por mês:

In [44]:
```
# executa a consulta ao banco
query = ('SELECT * FROM `allsky_insolation_hs` WHERE `latitude` = -38.25 AND
`longitude` = -65.75')
cnx = connect()
cursor = cnx.cursor()
cursor.execute(query)

# coloca os resultados em um DataFrame
col = ['id', 'date', 'latitude', 'longitude', 'insolation']
df = pd.DataFrame(list(cursor), columns=col)
df.index = list(map(pd.Timestamp, df.pop('date')))
df = df.sort_index()
```

In [45]:
```
fig, ax = plt.subplots(figsize=(20, 5))
df['insolation'].plot(ax=ax);
```

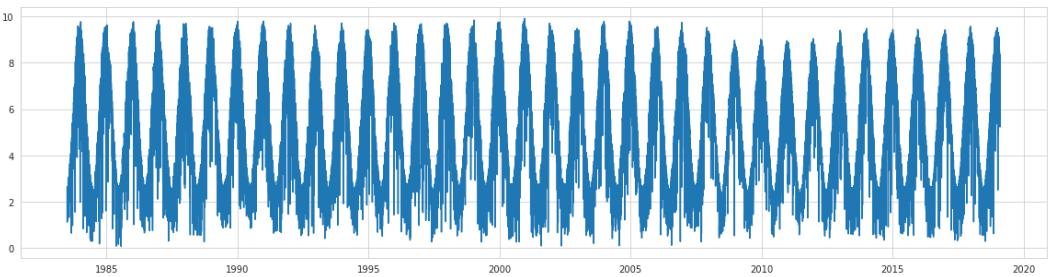


Observação

Na figura vemos que os dados aparentam ter outliers. Na verdade esse outliers são dias onde não foram realizadas observações (NA) e o valor é setado como -999. A seguir é realizada a substituição desses valores.

In [46]:
```
df.loc[df.query('insolation < 0').index, 'insolation'] = np.nan
```

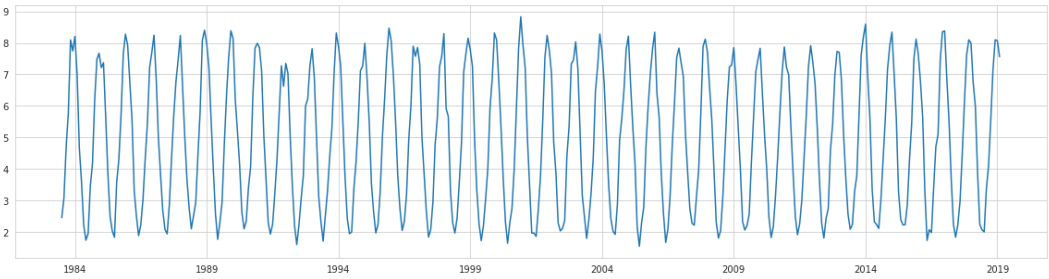
In [47]:
```
fig, ax = plt.subplots(figsize=(20, 5))
df['insolation'].plot(ax=ax);
```



Observações

Vemos uma sasonalidade nos dados com periodo de cerca de um ano (visualmente). Nos inicios e finais de ano vemos um aumento expressivo na insolação enquanto em meados do ano a insolação chega a próximo de zero.

In [48]:
```
# tirando a média mensal dos dados
fig, ax = plt.subplots(figsize=(20, 5))
montly = df.resample('M')['insolation'].mean()
montly.plot(ax=ax);
```

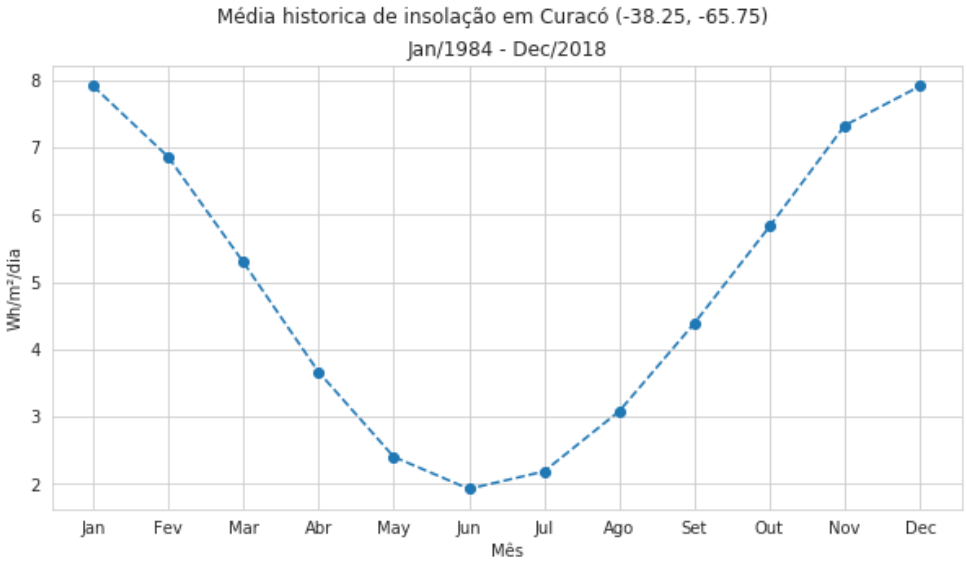


In [49]:
```
#alinhando as pontas (tirando meses quebrados do inicio e do final do regist
ro)
montly = montly[6:-2]
montly
```

Out[49]:
```
1984-01-31    8.206129
1984-02-29    6.981034
1984-03-31    4.674839
1984-04-30    3.583667
1984-05-31    2.218710
1984-06-30    1.742333
1984-07-31    1.950968
1984-08-31    3.450323
1984-09-30    4.187000
1984-10-31    6.193226
1984-11-30    7.478333
1984-12-31    7.672903
1985-01-31    7.214839
1985-02-28    7.371071
1985-03-31    5.671290
1985-04-30    3.859000
1985-05-31    2.497419
1985-06-30    2.036667
1985-07-31    1.836774
1985-08-31    3.574516
1985-09-30    4.299667
1985-10-31    5.719355
1985-11-30    7.627000
1985-12-31    8.277419
1986-01-31    7.930968
1986-02-28    6.736786
1986-03-31    5.552258
1986-04-30    3.295667
1986-05-31    2.497742
1986-06-30    1.885667
                  ...
2016-07-31    1.983871
2016-08-31    3.418710
2016-09-30    4.705667
2016-10-31    5.100645
2016-11-30    7.461000
2016-12-31    8.349355
2017-01-31    8.378710
2017-02-28    6.856786
2017-03-31    5.518387
2017-04-30    3.686667
2017-05-31    2.243548
2017-06-30    1.837000
2017-07-31    2.236129
2017-08-31    2.970645
2017-09-30    4.448667
2017-10-31    6.098387
2017-11-30    7.609667
2017-12-31    8.097097
2018-01-31    7.991290
2018-02-28    6.731071
2018-03-31    5.977742
2018-04-30    3.871333
2018-05-31    2.243548
2018-06-30    2.072333
2018-07-31    2.005484
2018-08-31    3.362581
2018-09-30    4.107000
2018-10-31    5.560000
2018-11-30    7.138621
2018-12-31    8.097097
Freq: M, Name: insolation, Length: 420, dtype: float64
```

In [50]:
```
years = [[montly[k:(k + 12)]] for k in range(0, len(montly), 12)]
months = index=['Jan', 'Fev', 'Mar', 'Abr', 'May', 'Jun', 'Jul', 'Ago', 'Set
', 'Out', 'Nov', 'Dec']
# Calculo da media por mes
m_means = np.mean(years, axis=0)[0]
```

In [51]:
```
fig, ax = plt.subplots(figsize=(10, 5))
ax.plot(m_means, '--o')
plt.title('Jan/1984 - Dec/2018')
plt.suptitle('Média historica de insolação em Curacó (-38.25, -65.75)')
plt.xlabel('Mês')
plt.ylabel('Wh/m²/dia')
plt.xticks(range(12), months);
```



Observações

O grafico acima mostra uma clara distição entre o inicio e o meio do ano, o que faz sentido, dado que o local em questão se encontra na zona temperada. Zonas estas que têm como caracteristica estações bem definidas, com verões quentes e invernos frios.

Plotar mapa com padrão de irradiação

Nessa parte do desafio foi escolhida uma data arbitrária para análise e os dados de toda a região escolhida foi plotado.

In [36]:
```
query = ('SELECT * FROM `allsky_insolation_hs` WHERE `date` = \'2014-07-05\'
')
cnx = connect()
cursor = cnx.cursor()
cursor.execute(query)

# coloca os resultados em um DataFrame
col = ['id', 'date', 'latitude', 'longitude', 'insolation']
df2 = pd.DataFrame(list(cursor), columns=col)
df2.index = list(map(pd.Timestamp, df2.pop('date')))
df2 = df2.sort_index()
df2[:5]
```

Out[36]:

	id	latitude	longitude	insolation
2014-07-05	533887	-39.75	-69.75	2.36
2014-07-05	533888	-39.75	-69.25	2.36
2014-07-05	533889	-39.75	-68.75	1.61
2014-07-05	533890	-39.75	-68.25	1.61
2014-07-05	533891	-39.75	-67.75	1.01

In [37]:
```
gmaps.configure(api_key='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX') # Tirei a minha
chave desse arquivo

fig = gmaps.figure()
heatmap_layer = gmaps.heatmap_layer(
    df2[['latitude', 'longitude']], weights=df2['insolation'],
    max_intensity=df2['insolation'].max()/3,
    point_radius=18
)
heatmap_layer.opacity = 0.3
fig.add_layer(heatmap_layer)
fig
```

Observações

O gráfico acima foi inserida como imagem pois este depende de widgets de terceiros (que podem não ter sido instalados em todos os ambientes)