DOCUMENTAÇÃO PROJETO GAVIÕES

DESENVOLVEDOR

Matheus Roque Arantes

API UTILIZADA

GOOGLE PLACES API

Objetivo

O objetivo deste projeto é localizar as unidades da academia Gaviões usando a **Google Places API**. Esta **API** permite realizar buscas com base em texto **`STRINGS`**. A linguagem de programação escolhida para a implementação foi **Python**, por sua facilidade de uso e popularidade.

Requisitos

Para interagir com a **API**, foram consideradas duas opções: a biblioteca **`REQUESTS`** ou a galeria **`GOOGLEMAPS`**, desenvolvida especificamente para a **API** do Google. Sendo assim a galeria **`GOOGLEMAPS`** foi a opção escolhida por oferecer maior **legibilidade e facilidade de manutenção**, já que abstrai muitos detalhes das requisições. E para sua instalação, é necessário o seguinte comando:

```
Python 3.x.x
pip install googlemaps
```

Pontos Positivos

A galeria oferece **suporte completo** à documentação da **API**, permitindo o uso de todos os seus serviços e parâmetros.

Oferece uma gama de comandos que possibilita a utilização mais simples e direta como o exemplo **`.places()`** que executa uma pesquisa com certos parâmetros que enviamos, forma esta utilizada neste programa.

A galeria do Google Maps, foi pensada para a utilização da própria **PLACES API.**

Pontos Negativos

Conforme a **PLACES API** é atualizada, é necessário esperar uma atualização da galeria para utilização de novas funções.

Uma diferença que tem entre os **REQUESTS** é que podemos utilizar somente uma **API** utilizando a galeria do Google Maps, dificultando a integração de possíveis novas **APIs**.

Diferente do **REQUESTS**, que é uma das bibliotecas mais utilizadas para comunicação com **APIs**, a galeria **GOOGLEMAPS** é uma **dependência adicional** que precisa ser instalada e gerenciada no projeto. Isso pode aumentar o tamanho do pacote e a complexidade das dependências.

Pontos Positivos e Negativos Encontrados nos Comandos

Comando `.places()`

Pontos Positivos

O comando é muito útil na utilização, já que passando o parâmetro **`carry="Academia Gavioes Brasil"`** ele faz uma pesquisa utilizando uma **STRING**.

Podemos passar também outras instruções para a linguagem escolhida que irá retornar a pesquisa, como o **`language("pt-BR")`**

O comando retorna na pesquisa o **`place_id`** que é um código único para cada estabelecimento. Sendo necessário para nossa utilização

Pontos Negativos

O comando retorna um **número limitado de resultados por página** (até 20), e a busca completa é restrita a um máximo de 3 páginas. Isso exige a implementação de um mecanismo de paginação, o que pode impactar a performance para encontrar todas as unidades.

A busca por texto (Text Search) no comando **`.places()`** pode retornar **resultados indesejados ou imprecisos**. Como a pesquisa se baseia apenas em palavras-chave, uma busca por **`query="Gavioes"`** pode exibir estabelecimentos que não são as academias Gaviões, mas que têm a palavra "Gaviões" no nome ou descrição.

O comando **`.places()`** ele traz certos dados que são desnecessários para nossa utilização, já que não traz os detalhes de cada estabelecimento

Comando `.place()`

Pontos Positivos

Por mais que seja parecido com o **`.places()`** ele tem outra função em nosso programa. Ele retorna dados dos estabelecimentos de forma específica, assim trazendo mais informações uteis que necessitamos, como os Horários de Funcionamento e endereço completo.

Para a realização da pesquisa específica de um estabelecimento é necessário fazer com o **`place_id`**, que o comando **`.places()`** retorna quando faz a pesquisa trazendo os dados necessários que vamos exibir.

Podemos fazer a filtragem dos dados que necessitamos com o parâmetro **`fields=[]`** fazendo que a **API** faça menos requisições, e que torne a sua resposta mais rápida e utilizando menos processamento.

Pontos Negativos

É um comando necessário, mas poderia ser evitado caso o **`.places()`** retornasse os dados do horário de funcionamento, algo que não acontece

e que faz o ser necessário e criando mais requisições na **API** aumentando o custo, já que cada requisição tem um valor a ser pago.

O programa fica lento, por causa de cada requisição com o `place_id` que o **`.place()`** retorna cada um de uma vez, e fora que temos que colocar **`time.sleep (2)`** para que não atinja o limite de **REQUESTS** da **API**.

A qualidade dos dados retornados pelo **`.place()`** **depende diretamente da precisão das informações fornecidas pelo próprio estabelecimento** na plataforma do Google Maps. Se um dado (como horário de funcionamento) não foi preenchido, a **API** não o retornará, resultando em campos vazios na resposta. Essa limitação não é uma falha da **API**, mas sim da fonte dos dados.

Documentações utilizadas:

Documentação da Galeria Python

Documentação Google Developers