**Application Programming Interface**

Hoje em dia é muito comum que diferentes aplicações consumam dados pela internet, muitas vezes dados providenciados por terceiros. Por exemplo, um aplicativo de entrega de alimentos pode usar dados de geolocalização do Google para localizar restaurantes próximos ao usuário e exibir a rota percorrida pelo entregador.

Como as aplicações podem rodar em diferentes plataformas (Windows, Android, MacOS, iOS, um navegador de internet...), é importante estabelecer uma linguagem comum para que todos consigam consumir esses dados.

Essa "linguagem comum" é o que chamamos de API: *Application Programming Interface*. A organização que disponibiliza os dados estabelece algumas "regrinhas" para fazermos requisições, e em contrapartida ela garante que os recursos fornecidos também seguirão certos padrões, facilitando a vida dos programadores.

Portanto, quando decidimos utilizar uma API, a primeira coisa que precisamos fazer é estudar sua documentação. Vejamos alguns dos pontos mais relevantes para procurar.

Todos os exemplos de requisição que mostraremos aqui podem ser colados em seu navegador ou estudados usando um requests.get no Python e imprimindo seu campo *text*.

**URI base**

Várias APIs fornecem um "endereço base". Todas as suas requisições incluirão esse endereço, e ao final dele nós colocamos detalhes específicos para cada um dos recursos disponíveis.

Por exemplo, na *AlphaVantage* (<https://www.alphavantage.co/>), uma API de dados de bolsas de valores e criptomoedas, a URI base é:

https://www.alphavantage.co/query?

Após a interrogação nós colocaremos os campos para nossa consulta. Por exemplo, para fazer uma consulta sem autenticação para valores da IBM, de 5 em 5 minutos, o endereço completo fica:

https://www.alphavantage.co/query?function=TIME\_SERIES\_INTRADAY&symbol=IBM&interval=5min&apikey=demo

Note o formato com &NomeDoCampo=ValorDoCampo. Ele é bastante comum. Outro formato bastante comum é o de "subdiretórios".

Um exemplo é a PokéAPI. A URI base é:

https://pokeapi.co/api/v2/

Para procurar por pokémons, adicionamos pokemon/. Em seguida, podemos colocar números (índices) ou nomes de Pokémon, como:

https://pokeapi.co/api/v2/pokemon/ditto/

https://pokeapi.co/api/v2/pokemon/25

Se ao invés de pokémons estivéssemos interessados em tipos de pokémon, usaríamos *types/* e o nome ou índice do tipo desejado:

https://pokeapi.co/api/v2/type/ghost

**Formato**

**Tipo de dado**

Algumas APIs possuem formatos fixos de dados. Outros permitem que você escolha. É comum, por exemplo, que uma API permita que você escolha entre JSON, XML, CSV e/ou outros formatos.

Caso você tenha entrado no *AlphaVantage* e se registrado para obter uma chave (falaremos mais adiante), você pode especificar, por exemplo, que gostaria de resultados no formato CSV:

https://www.alphavantage.co/query?function=TIME\_SERIES\_INTRADAY&symbol=IBM&interval=5min&dataformat=csv&apikey=demo

Substitua "demo" por sua chave no exemplo acima e ele funcionará.

**Schema**

É bastante comum que as APIs disponibilizem um "modelo" genérico de como será formatado o seu JSON, XML etc para que os desenvolvedores saibam quais campos esperar e quais tipos de dados serão possíveis para cada campo. Por exemplo:

{

'nome':string,

'pontuacao':integer

}

**Autenticação**

Outro aspecto importante é a autenticação. Enquanto algumas APIs são grátis, outras são pagas. Ainda temos algumas híbridas: você pode gratuitamente acessar certos recursos, ou consumir um certo volume de dados, e acima disso você deverá pagar. Os dois modelos mais comuns de autenticação:

* Chave: ao fazer seu registro, você recebe uma chave que será inclusa na requisição, como é o caso do *AlphaVantage*.
* OAuth: um esquema um pouco mais complexo onde são combinados códigos de autorização, identificação do cliente e segredo do cliente em um POST, e o servidor cria uma sessão por um tempo limitado e fornece o ID da mesma. APIs de gigantes da internet (como Google e Facebook) costumam usar esse modelo.

**Rate limiting**

Um dado parcialmente relacionado ao item anterior. As APIs costumam limitar o número de requisições que você pode fazer em um instante de tempo (3 requisições por minuto, 10000 requisições por dia etc). Temos dois motivos:

* Segurança: evitar uma sobrecarga no servidor deles que possa indisponibilizar a API para todos os usuários.
* Venda de planos: várias APIs pagas possuem diferentes planos de pagamento. Os planos mais caros costumam permitir mais requisições do que os mais baratos ou gratuitos.

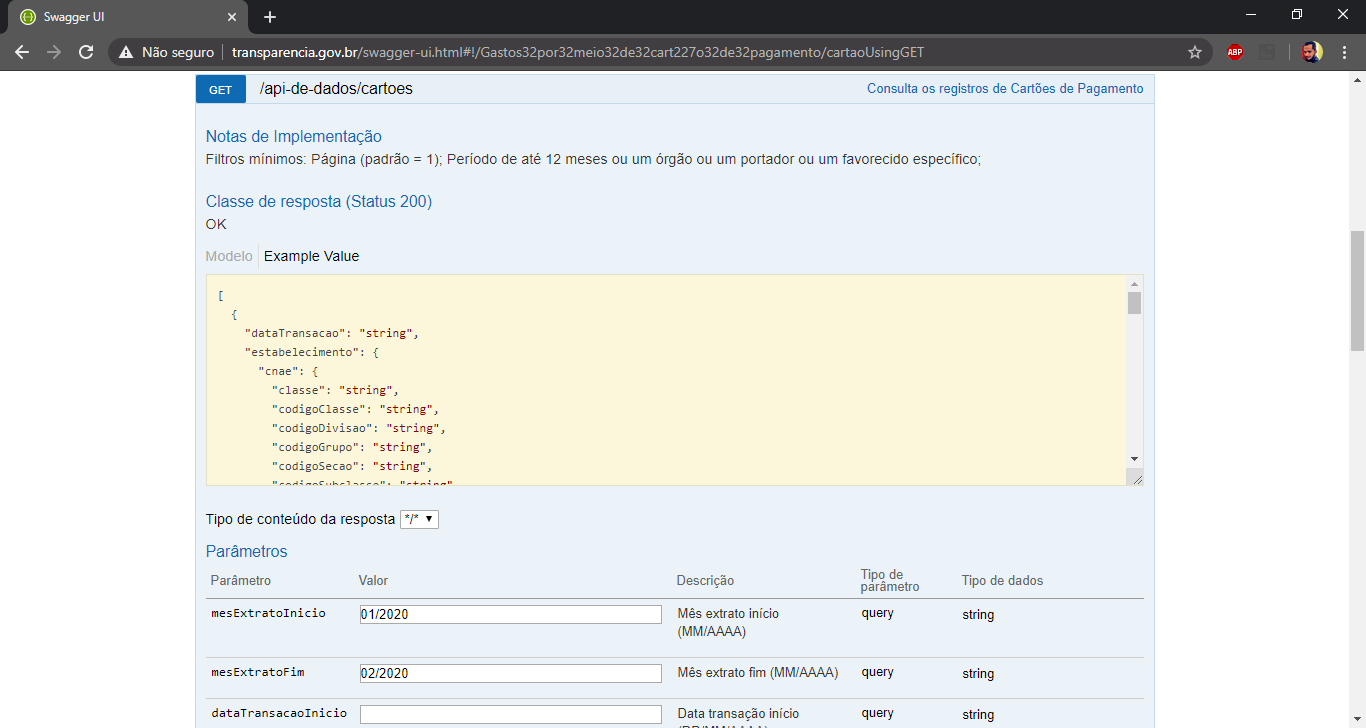
**Wrappers**

Algumas APIs possuem tantas buscas diferentes e os resultados podem ser tão complexos que mesmo vindo em formatos simples como JSON pode ser um pouco trabalhoso montar as requisições e isolar os dados que queremos. Por conta disso, frequentemente são fornecidas *wrapper libraries*: bibliotecas escritas em linguagens de programação específicas que já trazem classes e funções prontas para fazer requisições automaticamente e já quebrar o resultado em objetos fáceis de serem utilizados. Elas também costumam oferecer alguns benefícios adicionais, como *caching*: de tempos em tempos a base de dados é totalmente ou parcialmente baixada por completo e salva localmente, o que ajuda a economizar requisições e, consequentemente, uso de dados (bastante útil considerando em usuários de dispositivos móveis, por exemplo).

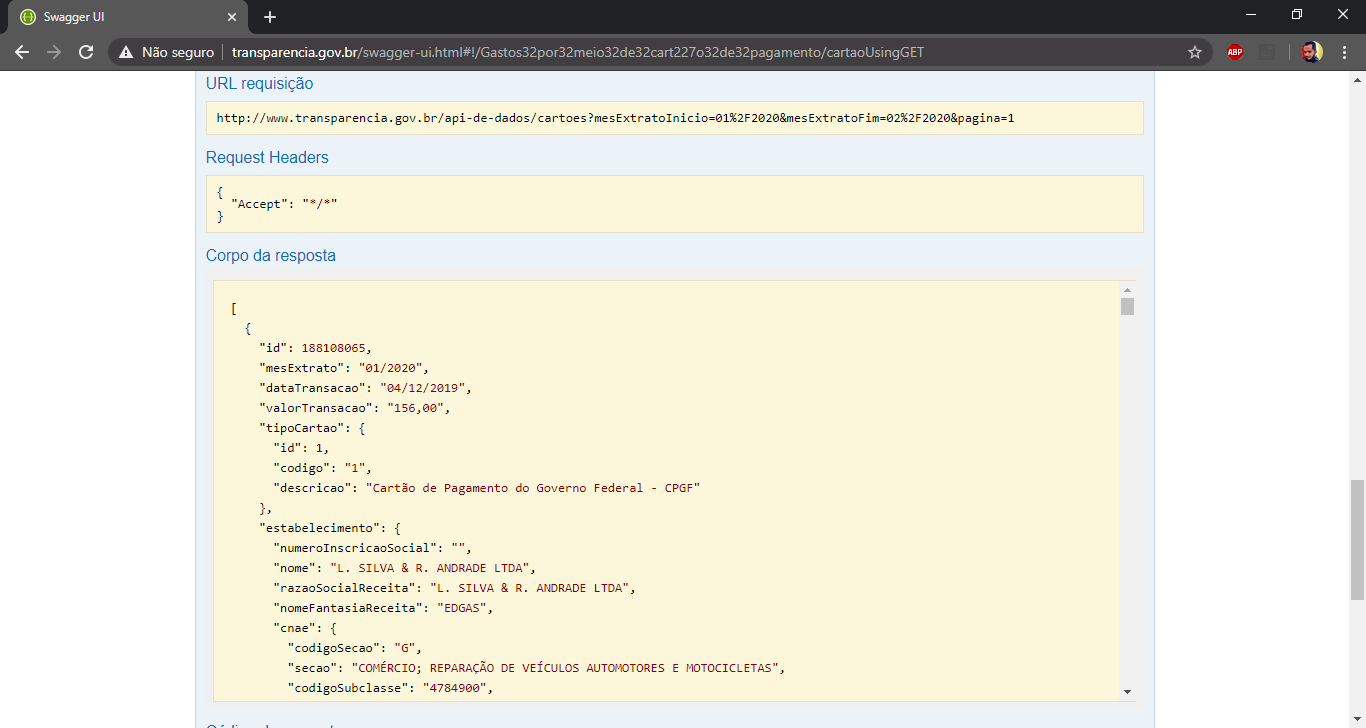
**Sandbox**

Várias APIs possuem no mesmo site de sua documentação uma área conhecida como *sandbox*, onde você pode simular requisições no próprio navegador e ver não só a resposta formatada, como informações sobre como montar aquela requisição em software.

Aqui podemos observar a área de *sandbox* para fazer consultas de gastos por meio de cartão de pagamento do Portal da Transparência do governo federal. Note que ele mostra o *schema* e apresenta campos para preenchermos as buscas.



E aqui, o resultado: ele gera o endereço completo para a busca baseado nos valores preenchidos nos campos e mostra a resposta formatada.



**Consumindo APIs em Python**

As APIs são meios de nos conectarmos a recursos na internet. Portanto, já possuímos as ferramentas na mão desde os capítulos anteriores. Você irá construir a lógica para decidir o que você irá buscar/consultar, montará uma *string* seguindo o formato indicado pela documentação da API (como todos os exemplos deste capítulo). Em seguida você tratará a resposta de acordo:

* Se for JSON, utilize o método *json* da própria *requests*.
* Se for CSV, utilize o módulo *CSV* estudado anteriormente.
* Se for XML, podemos utilizar o módulo *BeautifulSoup*, que não será estudado aqui.
* Para outros formatos, provavelmente a solução mais fácil será baixar um módulo preparado para lidar com eles.

**Descobrindo APIs:** tem boas ideias e gostaria de saber se existe uma boa API para ajudar? Confira alguns bons repositórios de API organizados por categoria:

<https://github.com/n0shake/public-apis>

<https://github.com/public-apis/public-apis>

<https://any-api.com/>

Sites de governos costumam ter uma grande riqueza de dados também. Segue abaixo algumas sugestões (oficiais ou mantidas por voluntários) com dados do Brasil como um todo. Experimente buscar por bases de dados de sua cidade ou estado!

<http://www.transparencia.gov.br/swagger-ui.html>

<http://www.dados.gov.br/>

<https://brasil.io/home/>