Hacker News API Guide

For Junior Developers

Zenahr Barzani

August 4, 2020

Contents

Disclaimer
This is an unofficial guide to the HackerNews API. The author is not affiliated with HN in any official function.

Getting Started

This tutorial is meant as a quick step-by-step guide to get started using the HN API whilst also teaching some fundamental things about APIs and introducing useful tools for testing APIs. This tutorial is mainly geared towards junior developers.

Hacker News

Hacker News is a social news website focusing on computer science and entrepreneurship. It follows the "anything that gratifies one's intellectual curiosity" slogan. If you haven't been on Hacker News before, feel free to pause this guide and stop by to get a feel of what we're going to work with. (Source: Wiki)

Types of Content

Content posted on Hacker News can be put into one of the following categories:

- Stories
- Jobs
- Comments
- Ask Hacker News

General Introduction to APIs

- Disclaimer (You can skip this if you have worked with APIs before)
- You talk with web APIs using HTTP clients. The windows command-line comes with curl built-in, but most popular programming languages have plug-and-play HTTP client libraries.
- Explain in quick sentences what APIs generally are
- Now a quick intro to Web APIs and the Restful paradigm

  NOTE
  If you know what APIs are and have worked with APIs in the web context, you can skip this part.

API stands for `Application Programming Interface`. Just by digesting the words of the acronym we can derive, that: - Something communicates with something else(`Interface`) - The context lies in the domain of software development(`Application`) - We write code in order to use APIs(`Programming`)

An API is an abstraction layer between some part of software and some other part of software, simplifying interaction between the two.

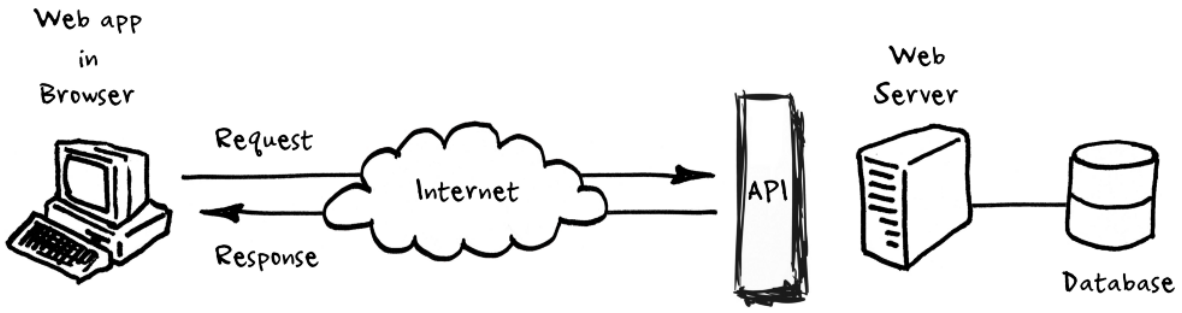
Minimal Example

One common type of API are libraries. You use libraries to abstract complexity away from complicated things, for example `dates`, or handling `User I/O` or more high-level things such as Data Science (pandas), utility libraries(libavutil) or even a library for speech recognition.

```
import speech_recognition as sr

r = sr.Recognizer()

with sr.AudioFile('I-dont-know.wav') as source:

    audio_text = r.listen(source)

    try:

        # using google speech recognition
        text = r.recognize_google(audio_text)
        print('Converting audio transcripts into text ...')
        print(text)

    except:
         print('Sorry.. run again...')
```

The example above is written in Python but the principle of abstraction applies to any library. So, libraries are one kind of API, what else is out there? Glad you asked…

Web Based APIs



Source

Since the world wide web has started to conquer the world since its modest beginning in March 12th, 1989 when Tim Berners-Lee submitted his memorandum titled “Information Management: A Proposal” to the management at CERN.

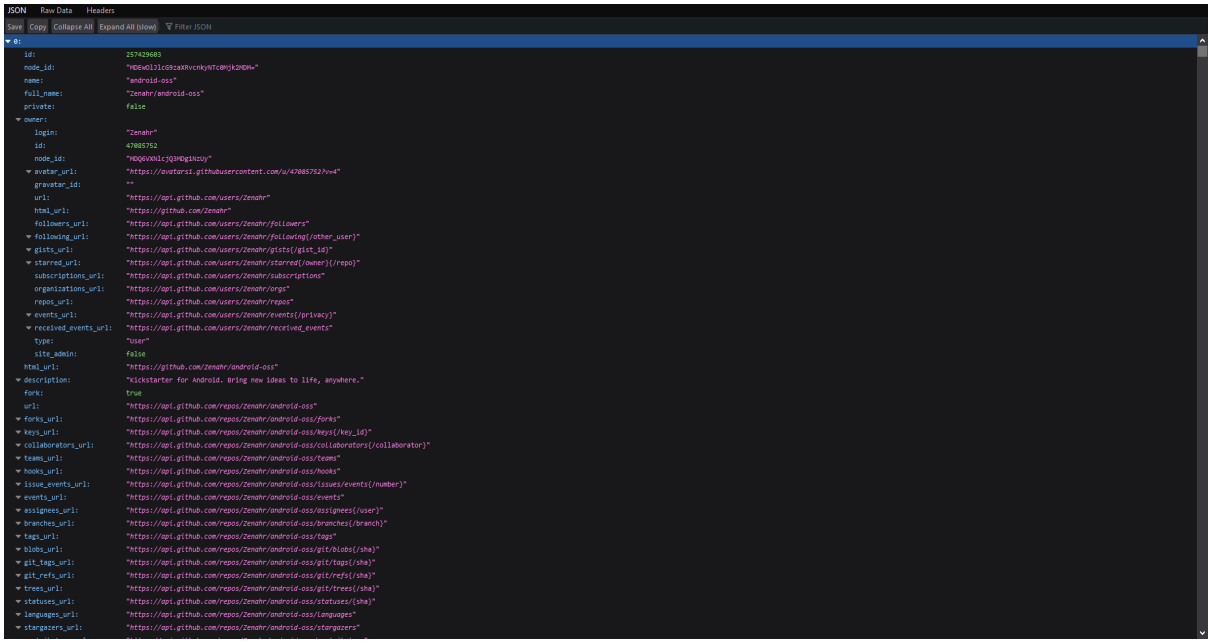
Web APIs are responsible for transferring data from the server to the client. Several protocols have been developed for that kind of communication today known as HTTP (Hypertext Transfer Protocol).

A Short Experiment

Visit following URL with your web browser:

`https://api.github.com/users/Zenahr/repos`

If you’re using Firefox as your web browser, it should look something like this:



The web browser has hit an API endpoint and represents the structured data (JSON) for us.

What just happened?

We made an *HTTP request* to the *repos* endpoint of the *GitHub API* and got a bunch of JSON objects back as a *response*. JSON is one common way of transportation via the web. JSON stands for `Javascript Object Notation`. I won't go through the details of it here The only thing we need to be aware of is that the data we get back is structured data and the container used to structure the data is JSON. Read more about JSON and its intricacies here. It's worth it if you're planning on working in web-development regardless of whether you specialise in front-end or back-end development.

Choosing a HTTP Client

We need a meaningful way of representing data received from API endpoints. This is where HTTP clients come into play. Remember, APIs use the HTTP protocol for transportation.

If you'd like to very quickly test something out you could always use curl.

open your terminal(Linux) or command-line(Windows) and enter the following line of code:

```
curl https://api.github.com/users/Zenahr/repos
```

Yes, it's the same HTTP request we sent before, but this time we used our terminal/command-line to receive the payload. Payload is the actual content of the HTTP response without accounting for metadata and other things that go along with an HTTP response. In other words: If everything is working fine you usually only care and work with the payload of HTTP responses.

If you plan on developing, testing and documenting APIs regularly make sure to check out Postman. It's free and open-source. We will be using Python and the `requests` library in this tutorial though.

Exploring The API

The Hacker News API is public. This means it is free to use. Also, you don't need an API key to access it.

The API is a collection of HTTP RPC-style methods using following URL building principle:
`https://hacker-news.firebaseio.com/v0/METHOD_FAMILY.method` example:
`https://hacker-news.firebaseio.com/v0/beststories.json`

> For anyone interested in reading more about the differences between REST and RPC paradigms I recommend reading this article by Phil Sturgeon and this article by RapidAPI.

The current API does not follow the REST paradigm. Everything is an item. There are no semantic endpoints such as `api/stories`, or `api/jobs` except `api/users`. The only way to only get one type of item is by using the following

Method	Description	Endpoint
`topstories.json`	Get 500 top stories	https://hacker-news.firebaseio.com/vo/topstories.json
`newstories.json`	Get 500 latest stories (Also contains jobs)	https://hacker-news.firebaseio.com/vo/newstories.json
`beststories.json`	Get 500 best voted stories sorted by votes in descending order	https://hacker-news.firebaseio.com/vo/beststories.json
`askstories.json`	Get latest Ask Stories	https://hacker-news.firebaseio.com/vo/askstories.json

Method	Description	Endpoint
`showstories.json`	Get latest Show Stories	https://hacker-news.firebaseio.com/vo/showstories.json
`jobstories.json`	Get latest Job Stories	https://hacker-news.firebaseio.com/vo/jobstories.json
`updates.json`	Get latest items that have been updated (including profiles)	https://hacker-news.firebaseio.com/vo/updates.json

(Feel free to open the links using your browser or `curl` and take a look look at what data you receive)

Next let's familiarize ourselves with the API by querying some data.

Querying Data

Imagine writing a mobile App (in Flutter, React Native or whatever you're comfortable with) and imagine you've got a section which should list the 50 most upvoted stories on HN.

We will work with Python because it's easy to prototype in and it features some neat functions right out-of-the-box for data manipulation.

> **ATTENTION**
> Make sure to have the requests library installed to follow along.

Let's make a simple GET request via the requests library to the `besttories.json` method

```
import requests

url = "https://hacker-news.firebaseio.com/v0/beststories.json"

response = requests.get(url).json()
```

The `https://hacker-news.firebaseio.com/v0/beststories.json` returns the 200 most voted stories in descending order, meaning the first result is the story with the highest number of votes.

Let's verify that by printing out the length of the result (the response we get is a list of IDs)

```
print(len(response))
```

```
>>> 200
```

Since we only care about the first 50 entries in this list, we can strip the list like so

```
response = response[:50]
print(len(response))
```

```
>>> 50
```

We could write another function which would then create direct links to the different posts like so

```
def id_to_link(id):
    return "https://news.ycombinator.com/item?id=" + str(id)

def id_list_to_link_list(id_list):
    return [id_to_link(id) for id in id_list]
```

```
link_list = id_list_to_link_list(response)
print("Links:", link_list)
```

The complete and abbreviated code for this looks like this:

> **NOTE**
> I have added more descriptive print functions.

```
import requests

url = "https://hacker-news.firebaseio.com/v0/beststories.json"
response = requests.get(url).json()[:50]
print("Amount of results:", len(response))
print("IDs:", response)

def id_to_link(id):
    return "https://news.ycombinator.com/item?id=" + str(id)

def id_list_to_link_list(id_list):
    return [id_to_link(id) for id in id_list]

link_list = id_list_to_link_list(response)
print("Links:", link_list)

>>> Amount of results: 50
>>> IDs: [24009177, 24030969, 24022751, 24042305, ..., 24032136]
>>> Links: ['https://news.ycombinator.com/item?id=24009177', ...]
```

We could now plug this code into our Flutter app and feed it to our frontend by creating an API of our own which in turn talks to the HackerNews API. Pretty meta, right?

HackerNews API Limitations

The current API is limited in its functionality. It supports only ¼ of standard CRUD functionality. This means one can't Create(C), Update(U) or Delete(C) but only Read(R) database entries. It also does not support pagination.

Recap

Let's recap what we've learned today:

- What APIs are
- How Web APIs work
- How to quickly test endpoints using `curl`
- How to use Python to manipulate simple data
- What HTTP Requests are (only scratched the surface on that one)

You can find the code along with this guide in Markdown format and here and an unofficial API Reference here in case you'd like to use this as a resource for further learning or to build the next Hacker News Reader App