APIs

<https://www.udemy.com/course/the-complete-web-development-bootcamp/learn/lecture/38912006#notes>

Application Programing Interface A set of rules and protocols that define how different software can interact with each other.

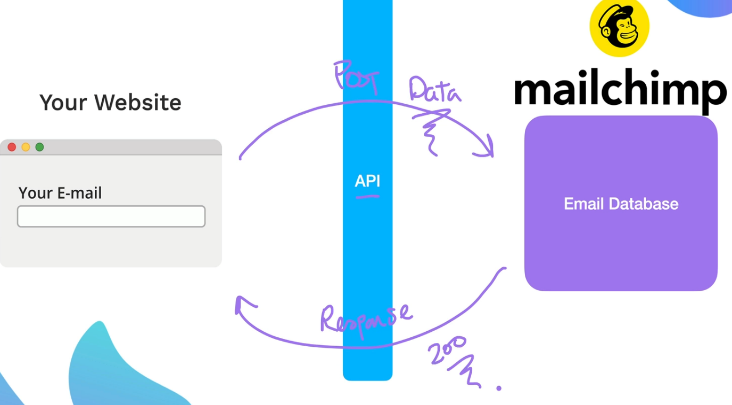
Bridges communication between pieces of software – how to interact and what to expect back

GET

A diagram of a software development process

AI-generated content may be incorrect.

POST



A screenshot of a computer

AI-generated content may be incorrect.

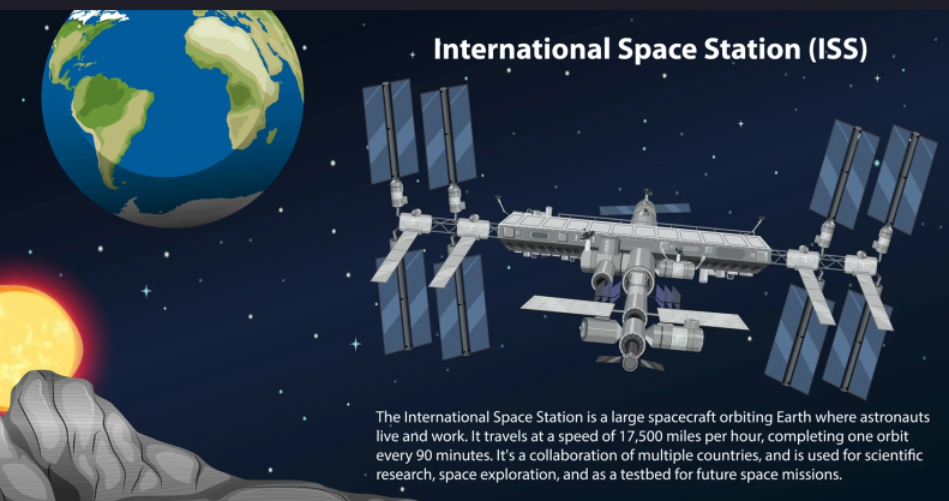
There are different architecual styles of APIs 

REST is most popular in web dev

Rest full apis follow HTTP protocol



**Using Postman to interact with an API** – Where is the international space station



URL to access, what kind of response can be expected- get request to the URL provided

GET request to API, get back lat,long can then put into google maps to see



<https://wheretheiss.at/w/developer>

A screenshot of a computer

AI-generated content may be incorrect.

Structuring API requests

A diagram of a private api

AI-generated content may be incorrect.

Use someone elses

A diagram of a process

AI-generated content may be incorrect.

<https://bored-api.appbrewery.com/>

Base URL , then an endpoint for different purposes-

A close-up of arrows

AI-generated content may be incorrect.

Endpoints are basically routes

Query Parameters – add ? to indicate it’s a query followed by a key value pair separated by an equal sign

A close-up of a website

AI-generated content may be incorrect.

Separated by ampersand

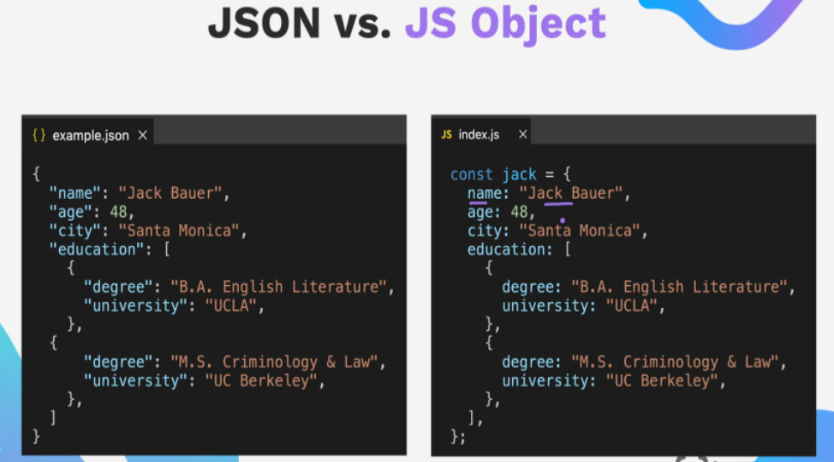


Path parameters

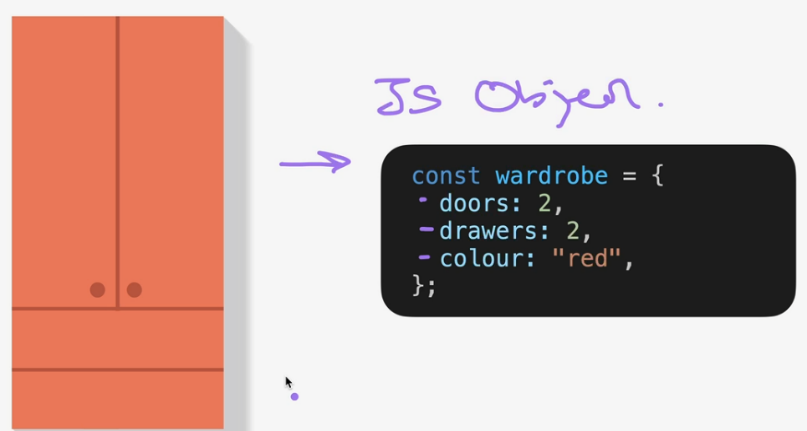
A white board with purple text

AI-generated content may be incorrect.

**Looking at JSON – key values in JSON keys are serialized into strings to transport over the web**



In Javascript – big object is ok



To transport over the web – compress to JSON for transport like ikea – it’s a string, it is serialized

A close-up of a computer mouse

AI-generated content may be incorrect.

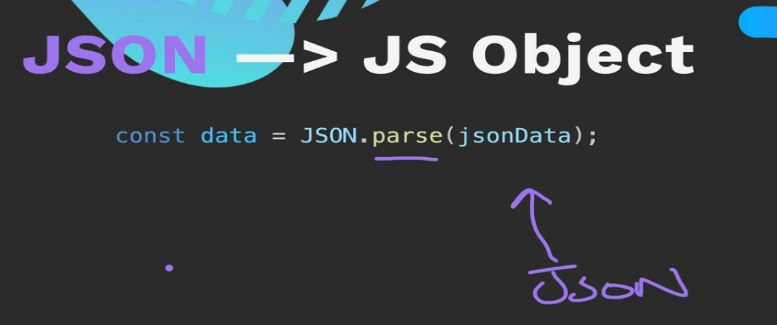
A Json visualizer on the web – makes the flatpack viewable for humans

<https://jsonviewer.stack.hu/>

To get a javascirpt object viewable as a json need to serialize it, or “stringify” here data is the JS object



To unpack use parse



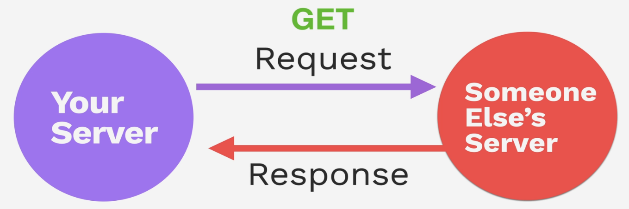
Post request

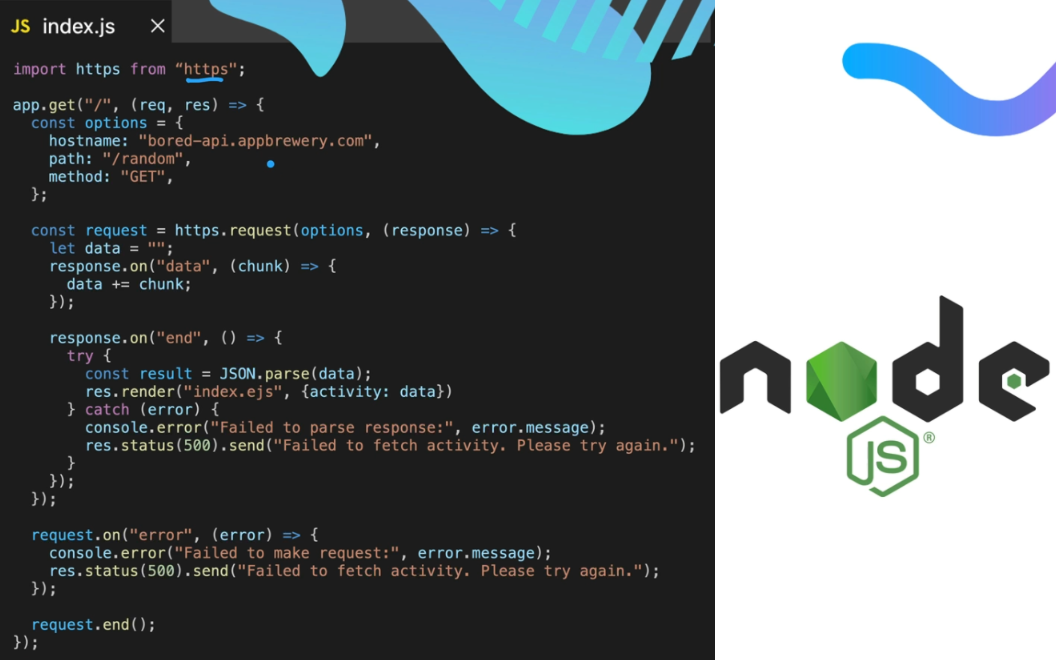
Body parser give access to request.body and what is sent over is what is clicked in based on the value.

Parse the JSON to make it into a JS object

Then its like an array

228 – Making server side API request with Axios

There is making API request from browser, here using server node and express

This can be done using the HTTPS native module from node 

Axios makes it much more simple – ( fetch is available natively )

Async / await , no need to use json.parse() with axios the response is already a json object

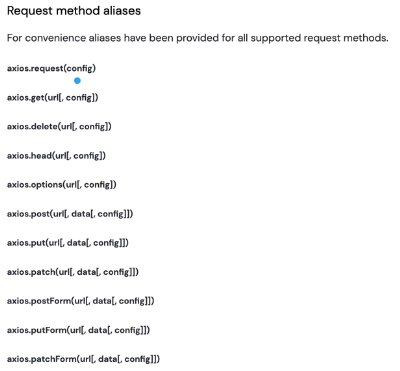


A screenshot of a computer screen

AI-generated content may be incorrect.

NPM axios

Axios has aliases for request methods



Axios.zip file – find something to do web site

Get request to get random activity from website to populate card

Make a post request to find activities with certain types like education and for X amount of people

If no activities found that match, send message back to user

Authentication – may not want all to be able to see or control resources that are in the API

Protect with authentication – don’t allow access and send 401 error message

4 tiers of authentication :

Purple text on a black background

AI-generated content may be incorrect.

0 is no authentication but putting a rate limit on the site basedo n IP address and requ per min.

A yellow alarm clock with white hands

AI-generated content may be incorrect.

1 Basic Authentication **sent in header** > match up with API

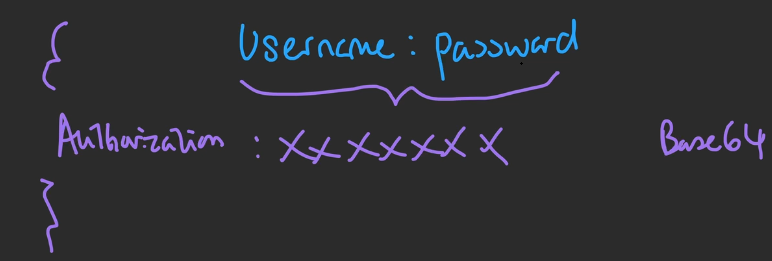
Authentication header

A screenshot of a computer

AI-generated content may be incorrect.

Basic Authentication – user name and password. HTTPS makes it a bit more secure but it is still vulnerable.

username:password is converted into a string, using base 84 encoding



Docs for API - <https://secrets-api.appbrewery.com/>

API\_URL = "https://secrets-api.appbrewery.com/";

Make post request to the register route, in body send username and password

A screenshot of a web page

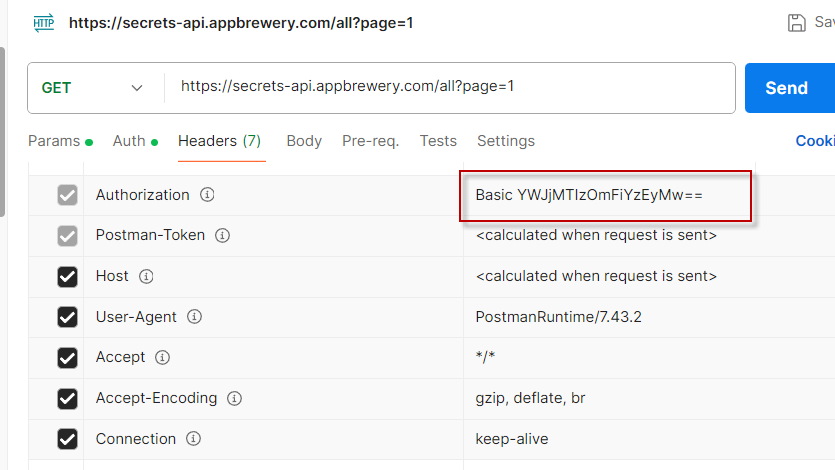
AI-generated content may be incorrect.

Then make the get request , use basic auth and put in the username and pw just registered

A screenshot of a computer

AI-generated content may be incorrect.

Can have a look at the headers tab to see what postman sent over as the BASE64 encoded username and password



Axios does this encoding out of the box.

API Key Auth

API Key Authorization vs Authentication

**Authorization allows you to use the API, Authentication identifies you as a user**

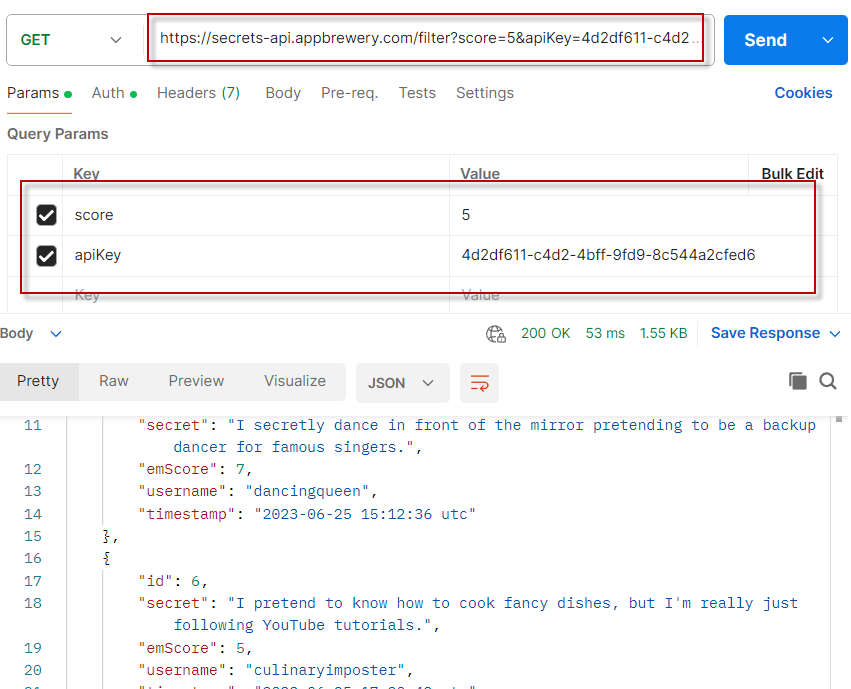
The Authorization Key can be used to track the usage of the API by your app, vs identifying the user.

Hit the end point to get the key

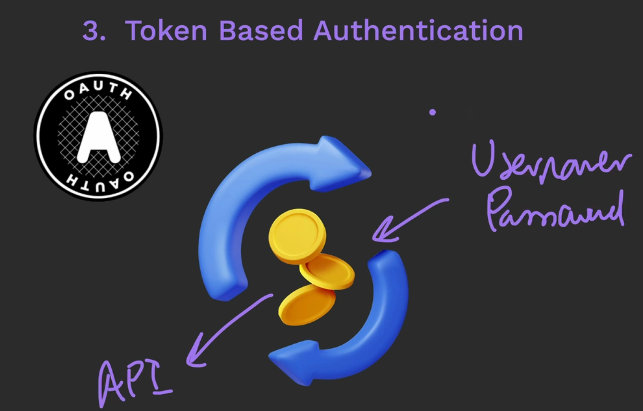
A screenshot of a computer

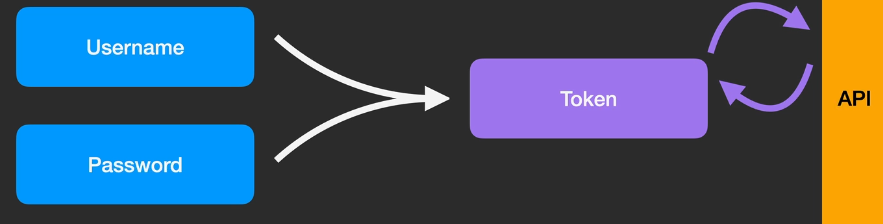
AI-generated content may be incorrect.

Using get filter with API key



Token Based Authentication – user logs in with user name and password, then a token gets generated. That token is used for API auth, not the user name and password directly. OAuth is the industry standard for doing token based authentication



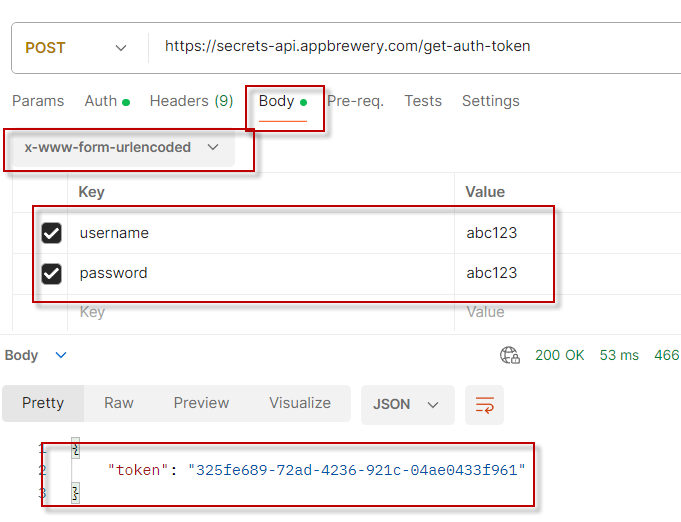


Google handles the security

A screenshot of a computer

AI-generated content may be incorrect.

**Bearer Token Authentication** – user name password plus token from the secrets API token endpoint. Token is used as a proxy for interacting with API as that user.



After getting the token put in under auth as Bearer token

A screenshot of a computer

AI-generated content may be incorrect.

Note the provided code did not work

REST APIs

A close-up of a web page

AI-generated content may be incorrect.

Async await GET

A computer screen with text and numbers

AI-generated content may be incorrect.

Changing using .then

The .thens wait until the proceeding is done before going on

A screen shot of a computer

AI-generated content may be incorrect.

**Using async await is the more modern and preferred way to handle**

The post request has 3 things that can be passed into the post request – here data is the body

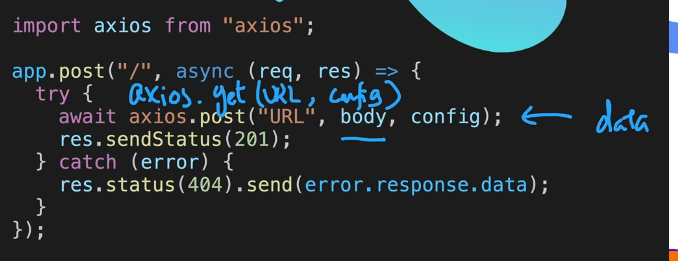
Often the body of the URL encoded coming from the form. In the config one can add headers(basic auth)

The config can be used as a third option

<https://axios-http.com/docs/req_config>

POST

POST **axios.post(url[, data[, config]])**



PATCH **axios.patch(url[, data[, config]])**

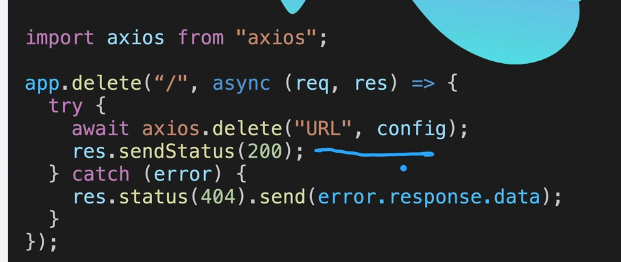


PUT **axios.put(url[, data[, config]])**

**A screen shot of a computer code

AI-generated content may be incorrect.**

DELETE **axios.delete(url[, config])**



Use a Public API

* Overview
* In this project, students are expected to build a website using the Express/Node.js platform, with the Axios HTTP client, that integrates a chosen public API from the given list: [**Public API Lists**](https://github.com/appbrewery/public-api-lists). The website should interact with the chosen API, retrieve data, and present it in a user-friendly manner.
* Objectives
* Develop an understanding of how to integrate public APIs into web projects.
* Gain practical experience using Express/Node.js for server-side programming.
* Enhance understanding of client-server communication using Axios.
* Demonstrate ability to manipulate, present, and work with data retrieved from APIs.
* Example Ideas
* Use the [JokeAPI](https://sv443.net/jokeapi/v2/" \t "_blank) to Create a website that gives the user a joke based on their name.
* Use the [OpenWeatherMap API](https://openweathermap.org/api/one-call-3" \t "_blank) to build a website that tells a user if it will rain tomorrow in their location of choice.
* Use the [Blockchain API](https://api.blockchain.com/v3/#/unauthenticated/getTickerBySymbol) to check the price of a cryptocurrency for the user.
* Use the [CocktailDB API](https://www.thecocktaildb.com/api.php" \t "_blank) to make a website that gives the user a random cocktail recipe with images of the cocktail.
* Use the [Open UV API](https://www.openuv.io/) to make a website based on your home location that tells you if you need to apply sunscreen today.
* Bonus points for choosing your own API and creating something of your own. Sometimes, you'll decide on an API and then realise that it doesn't work, that's all a part of the process, just move on to the next one and try again!
* Requirements
* 1. API Choice
* Browse through the [provided list](https://github.com/appbrewery/public-api-lists) and choose an API of interest. This choice should be guided by the potential to retrieve, manipulate, and present data in a meaningful and interactive way. I recommend choosing an API that does not require authentication and is CORS enabled. ([What is CORS?](https://medium.com/@electra_chong/what-is-cors-what-is-it-used-for-308cafa4df1a))
* 2. Project Planning
* Think through your project, researching the chosen API, its features, what data it will provide, and how it will be used in your web application.
* 3. Project Setup
* Set up a new Node.js project using Express.js.
* Include Axios for making HTTP requests.
* Include EJS for templating.
* Ensure that the project has a structured directory and file organization.
* 4. API Integration
* Implement at least a GET endpoint to interact with your chosen API.
* Use Axios to send HTTP requests to the API and handle responses.
* 5. Data Presentation
* Design the application to present the retrieved data in a user-friendly way. Use appropriate HTML, CSS, and a templating engine like EJS.
* 6. Error Handling
* Ensure that error handling is in place for both your application and any API requests. You can console log any errors, but you can also give users any user-relevant errors.
* 7. Documentation
* Include comments throughout your code to explain your logic.
* 8. Code Sharing
* Use what you have learnt about GitHub to commit and push your project to GitHub so that you can share it with other students in the Q&A area, I'd love to see what you've build too! You can tweet at me @yu\_angela
* Include a Readme.md file that explains how to start your server, what commands are needed to run your code. e.g. npm i  and then nodemon index.js
* Recommended Resources
* Express.js: [**Getting Started Guide**](https://expressjs.com/en/starter/installing.html)
* Node.js: [**Documentation**](https://nodejs.org/en/docs/)
* Axios: [**Documentation**](https://axios-http.com/docs/intro)
* Public APIs: [**API List**](https://github.com/appbrewery/public-api-lists)
* More free APIs: <https://rapidapi.com/collection/list-of-free-apis>

Some interesting looking APIs I found

Recipes

<https://spoonacular.com/food-api>

Nutritional database

<https://developer.nutritionix.com/>

Open FDA.gov

<https://open.fda.gov/>