

---

---

# Everybody wants an API

A rambling on API creation by  
Antonio Ortega Jr

---

# I am Antonio.

I want to consume APIs.

I don't want to spend a long time on things.

Now I want to provide an API to others.

## Also

I like turtles.



---

# What's an API?

API stand for Application Programmer Interface.

APIs allow developers to work with data in a simple way.

These days RESTful APIs are the coolest.

---

---

# What's REST?

Representational State Transfer.

Not quite a standard. More of a philosophy.

RESTful APIs are the coolest.

---

# Lots of APIs already exist

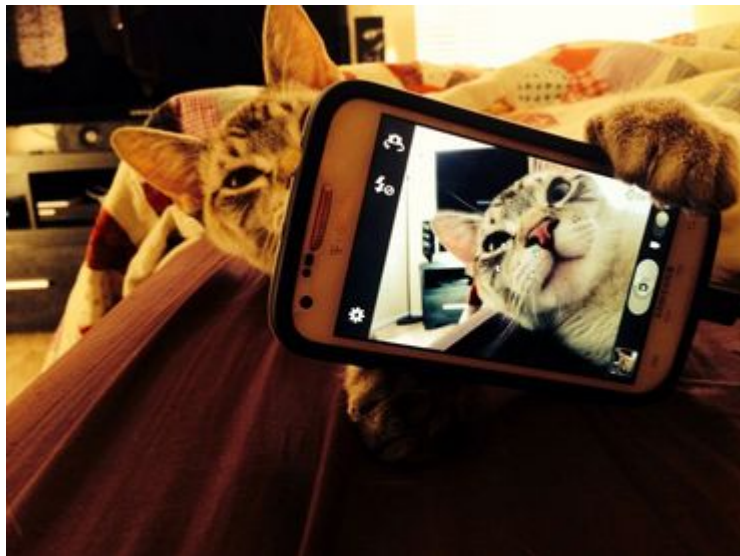
Do we really need more public APIs? Many of us already consume data from a service(s). Why do we need to make more about every little thing?

Mostly because companies will want others to push their own data further and leverage developers.

Additionally the personal API is a thing now.



# The Personal API?



Code: 200

```
{  
  "my_api": "online ",  
  "cats": "y",  
  "selfies": "mostly"  
}
```

## Antonio “Personal API” docs:

Auth may vary from endpoint to endpoint

**GitHub:** GET `https://api.github.com/users/antonioortegajr`

**Twitter:** GET `https://api.twitter.com/1.1/statuses/user_timeline.json`

**Untapped:** GET `https://api.untapped.com/v4/user/checkin/antonioortegajr`

This collection of endpoints provides data on the personal activities of me, Antonio.

Although clunky, if someone was looking to consume personal data about me this could be the functional docs of a personal aggregate API.

## api.naveen – a personal api

### The intro

Authenticate and pull from this API to access my real-time personal statistics.

### The access

You'll need to sign in, give me your Twitter username and get a token in order to call the functions here (poor man's piggyback-off-someone-else authentication™). I won't do any magic with your account, I just aim to keep track of access statistics.

[Sign in with Twitter](#) to get your access token.

Right. Pop your access token ('boofar') into the request like so:

```
http://api.naveen.com/v0/sleep?at=boofar
```

### The functions

/v0/sleep - my sleeps (and sometimes naps)

```
[
  {
    "start_date": "2013-05-27T22:52:56",
    "end_date": "2013-05-28T08:15:00",
    "id": "51a41c08b80df90002da515a"
  },
  {
    "note": "an early rise",
    "start_date": "2013-05-28T21:24:13",
    "end_date": "2013-05-29T06:30:00",
    "id": "51a558c682def70002a7f3b2"
  },
  {
    "start_date": "2013-05-29T00:49:44",
    "end_date": "2013-05-29T07:30:00",
    "id": "51a58912a14df6000216dd5d"
  },
  ...
]
```

/v0/weight - my weight (in kg)

```
[
  {
    "value": 67.25,
    "date": "2013-04-29T13:56:34",
    "id": "5180330a36ee3f0009088027"
  },
  {
    "value": 67.5,
```



# From API Consumer

I provide my auth

I want a format that I <3

I need documentation

I need consistency

I need warning of changes

# To API Provider

I issue auth creds to keep data secure

I provide JSON

I document my all endpoints

I respect the “API contract”

I use Semantic Versioning

I <3 my consumers

---

---

---

# The “API Contract”

This is an unspoken commitment to consistency and usability of the resources you provide with respect to the developers that are building on your API.

This includes, but is not limited to, security, definition, http response codes, output changes, documentation, API uptime and valid formatting of returns.

---

# Definition and Documentation

Versioning allows me to know when your API Definition changes. Semantic versioning is the norm.

Given a version number MAJOR.MINOR.PATCH, increment the:

1. MAJOR version when you make incompatible API changes
2. MINOR version when you add functionality in a backwards-compatible manner
3. PATCH version when you make backwards-compatible bug fixes

Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format.

Providing an API to the public requires Documentation. The more the better. Including a test console is the best.

There are plenty of API doc frameworks. Even auto gen. Your API code IS the documentation.

Swagger is my fav.



# Live the API Contract

Your consumers need you to respect the API contract.



## Tip

Those that provide the data are looking for others to build on it. So let's all play nice.

A close-up photograph of a man's face, showing a pained or frustrated expression with visible sweat or tears on his forehead. The image is heavily tinted with a blue color. In the background, other people are blurred, suggesting a crowded setting.

# An API barrier left me feeling lonely and hurt.

A small icon of a piece of torn, textured paper.

## Remember

Every barrier I come across in discovering your API is another reason for me not to use your API.



### Error Status Codes

HTTP Status Code	Reason
409	Limit greater than 100.
409	Limit invalid or below 1.
409	Invalid or unrecognized parameter.
409	Empty parameter.
409	Invalid or unrecognized ordering parameter.
409	Too many values sent to a multi-value list filter.
409	Invalid value passed to filter.

Try it out!

Marvel API Docs

One error status code???



Untappd to me

1/28/15

UNTAPPD  
DRINK SOCIALLY

Hello Antonio,

As an Untappd API developer, we wanted to inform you that you will need to update your application to use our secure, HTTPS API endpoint (<https://api.untappd.com>). We will be discontinuing our standard **HTTP API endpoint on 2/1/15**. This will require you to update your code for your application to change the <http://> protocol to <https://>.

After 2/1/15, your API key will no longer work over **HTTP** and return a **500 HTTP error**. The security of the Untappd community is extremely important to us and this change will help keep things safer.

We recommend that you join our API Developer Group at <http://untpd.it/apigroup> for up the information about the API.

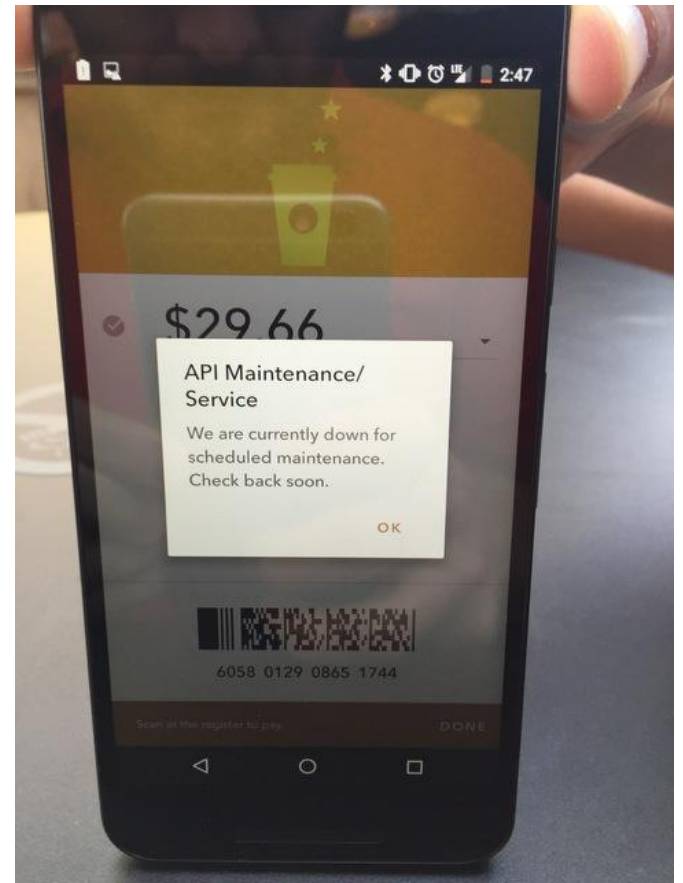
If you have any questions or concerns, please visit our API Developer group or <http://help.untappd.com>.

Thanks,  
The Untappd Team



Reply

Just now you add SSL??



Not living this API Contract can have pretty big impact

---



# 1. Micro Frameworks

**Why not a framework?** RESTful and simple.

→ **Promotes standardization**

Still free to mess it up

→ **Micro**

Use only what you need.

→ **Simple**

Provide a simple, consumable API quickly. Very, very quickly.



---

# What micro frameworks to use?

## What language?



### Tip

Remember that some of your consumers will likely be calling your API in many languages.

So your docs should be agnostic.

---

# Just one!

# The one that is easy for you.



## Tip

You have to maintain this API, so unless there is a great reason to leave your comfort zone, why do it?

Why not something with good docs and plenty of tutorials?

Let's offer an API using  
micro frameworks in

**three**

**LANGUAGES** Node.

js, PHP, and Python.

Just one GET route.

No database or auth.

Simple.



**Tip**

There are plenty of other options.

These three were chosen because I like them. Just how I like Ghidorah.

Let's just return some static json like

```
{"trap": "y"}
```





## 2. Frameworks

Using each of the following popular micro frameworks let's set up a route.

→ **Express**

Node package

→ **Flask**

Built by the folks that run Twilio

→ **Lumen**

From the Laravel gang

→ **And WordPress ... ?!?!?**

Yuup. That is a thing now too.



### Tip

You can use Lumen now and plug in right back into Laravel later!

## Lumen

The Laravel micro framework.

Fast and extensible Lumen does things mostly the Laravel way, but without all of the Laravel things.



# Express package.

Minimalist Node package.

Simple.

Node (yay JS!).



# Express

**Fun**

JS all the things



## Flask

Micro framework from Twilio.

Built by the consumers!

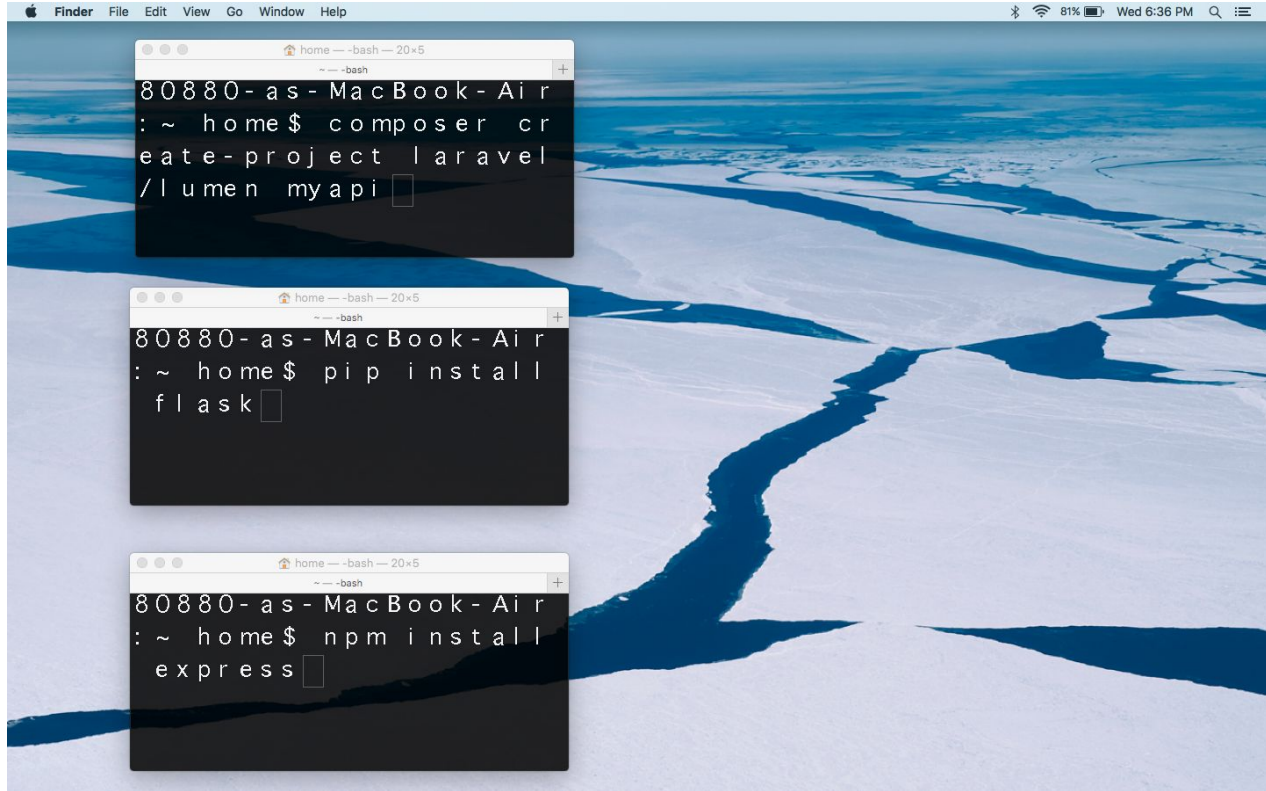
<3

Twilio uses Flask.

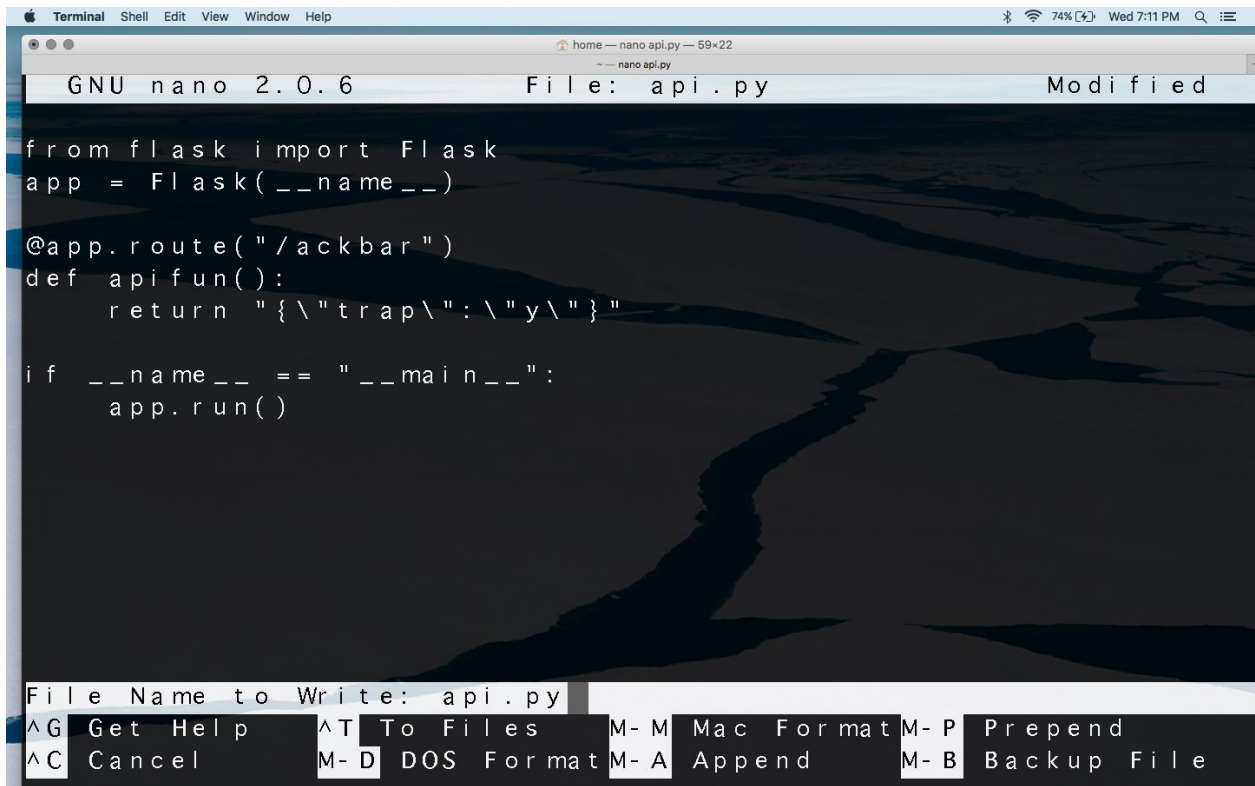




# Install with Dependency managers



# Flask code needed for first route



```
from flask import Flask
app = Flask(__name__)

@app.route("/ackbar")
def api fun():
    return {"trap": "y"}

if __name__ == "__main__":
    app.run()
```

File Name to Write: api.py

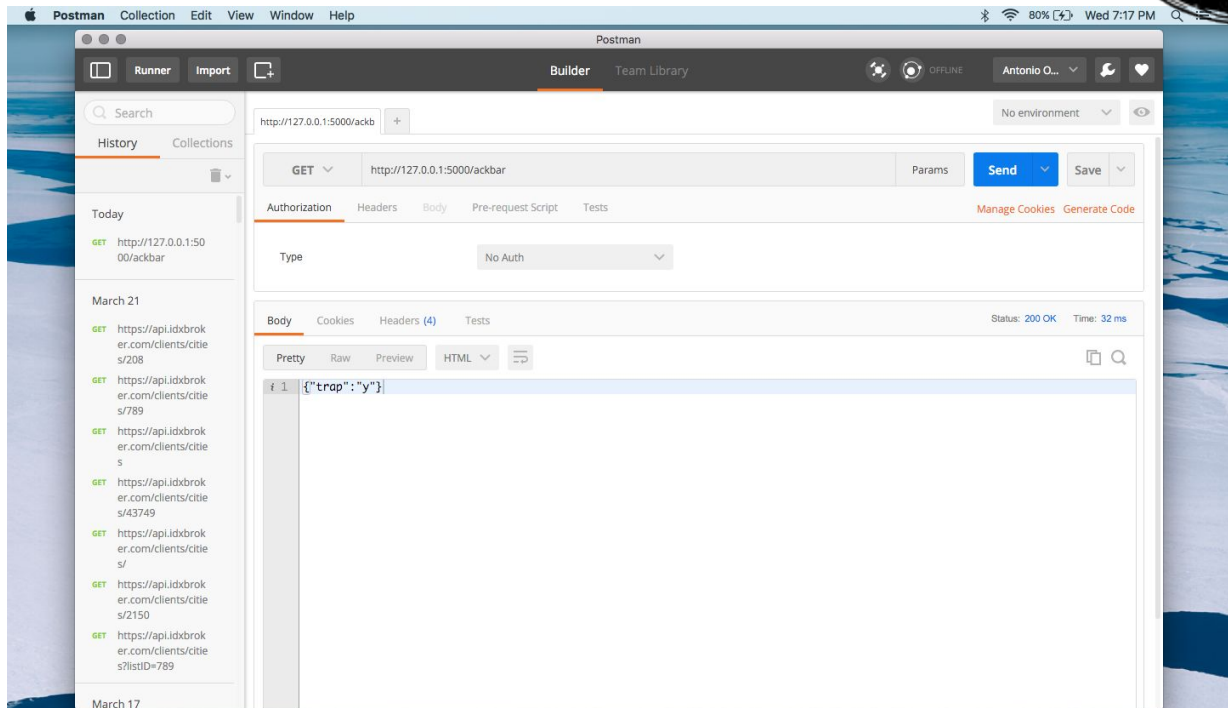
^G Get Help    ^T To Files    M-M Mac Format    M-P Prepend  
^C Cancel    M-D DOS Format    M-A Append    M-B Backup File

# Time to first route: under 5 min

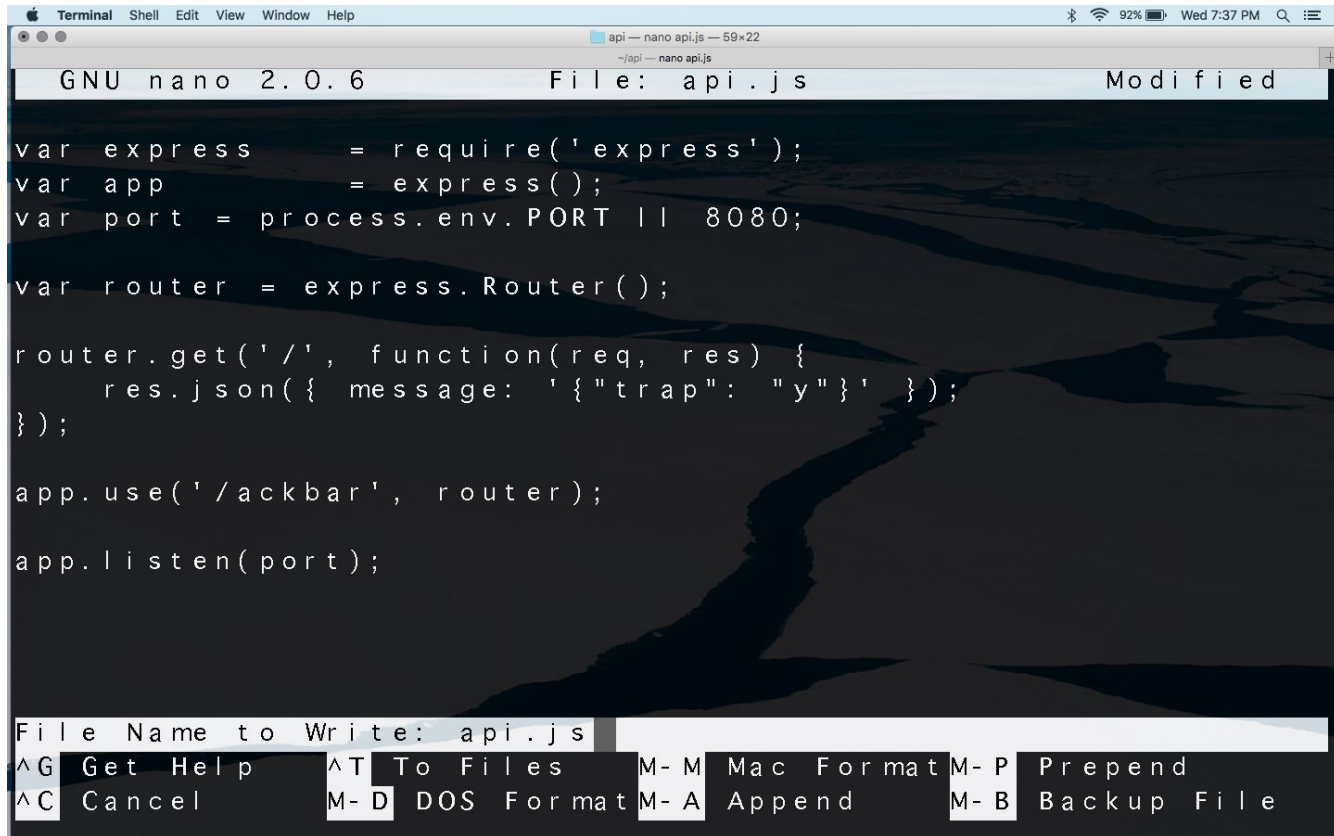


# Flask

web development,  
one drop at a time



# Express code needed for first route



```
var express    = require('express');
var app        = express();
var port       = process.env.PORT || 8080;
var router     = express.Router();

router.get('/', function(req, res) {
  res.json({ message: '{"trap": "y"}' });
});

app.use('/ackbar', router);

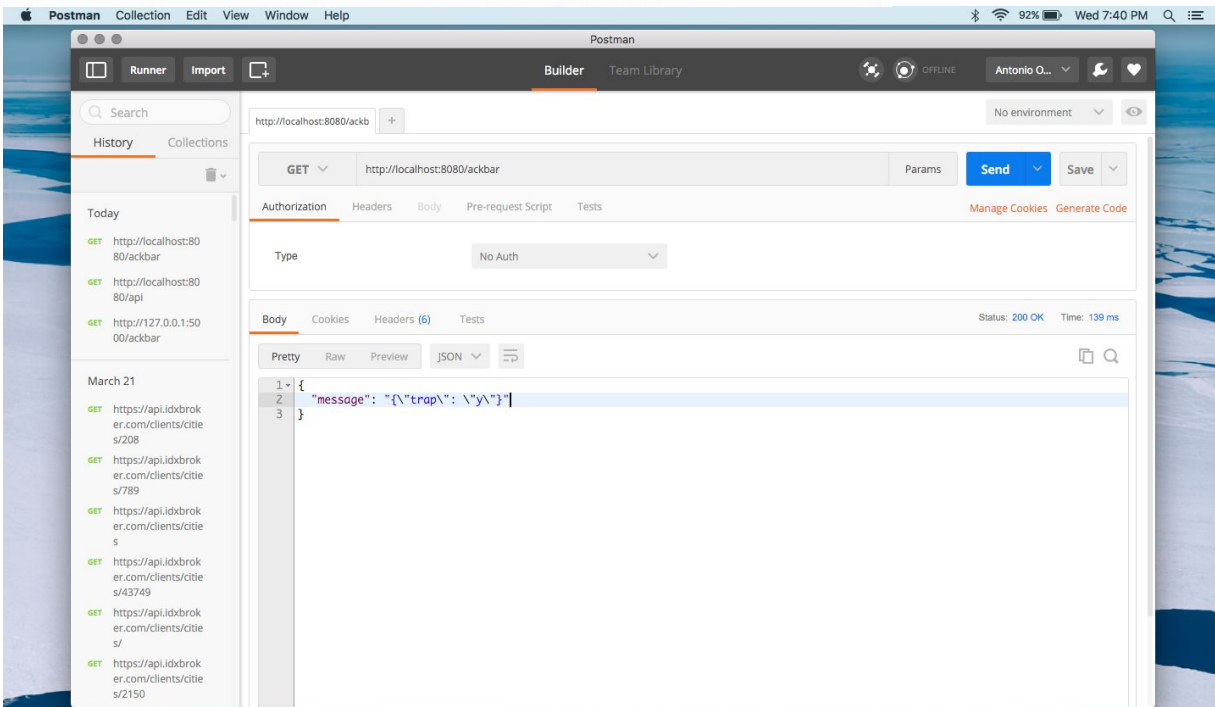
app.listen(port);
```

File Name to Write: api.js

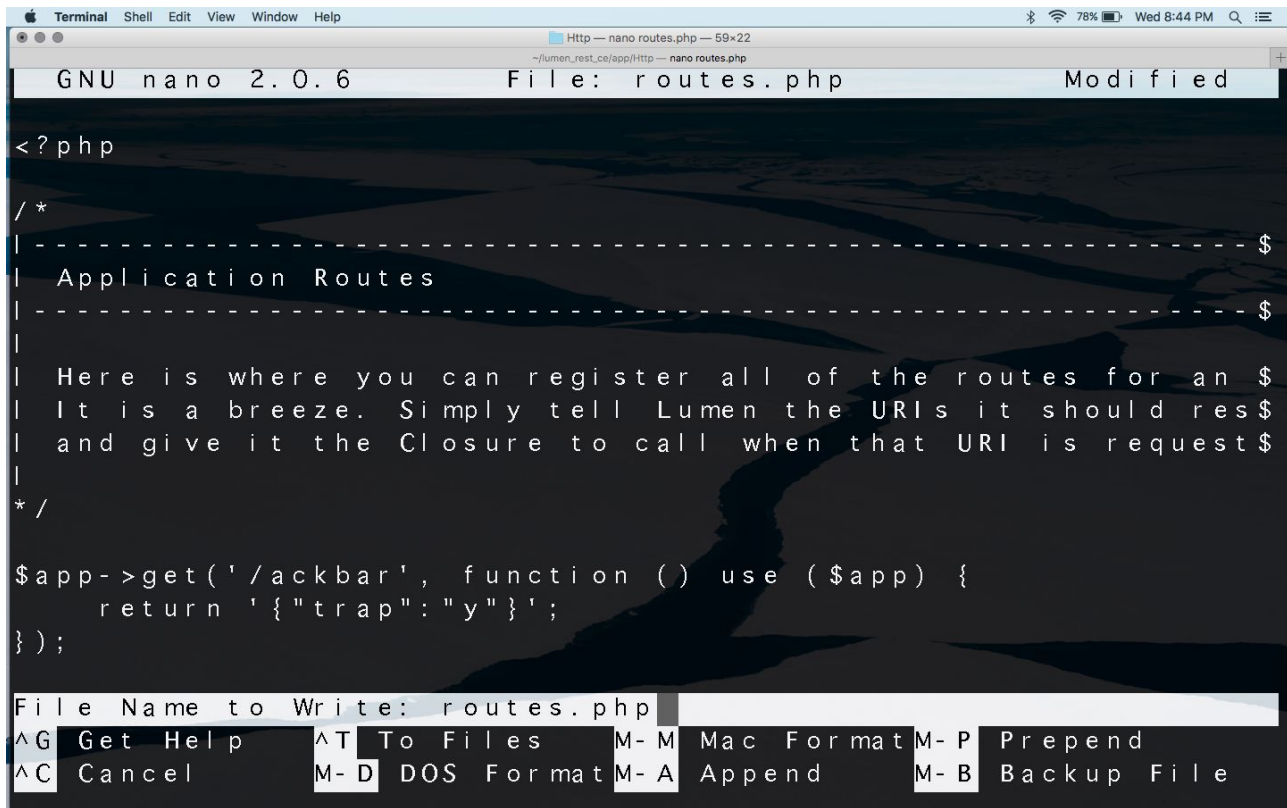
<b>^G</b> Get Help	<b>^T</b> To Files	<b>M-M</b> Mac Format	<b>M-P</b> Prepend
<b>^C</b> Cancel	<b>M-D</b> DOS Format	<b>M-A</b> Append	<b>M-B</b> Backup File

# Express

**Time to first route:  
under 10 min**



# Code needed for first route



```
GNU nano 2.0.6 File: routes.php Modified

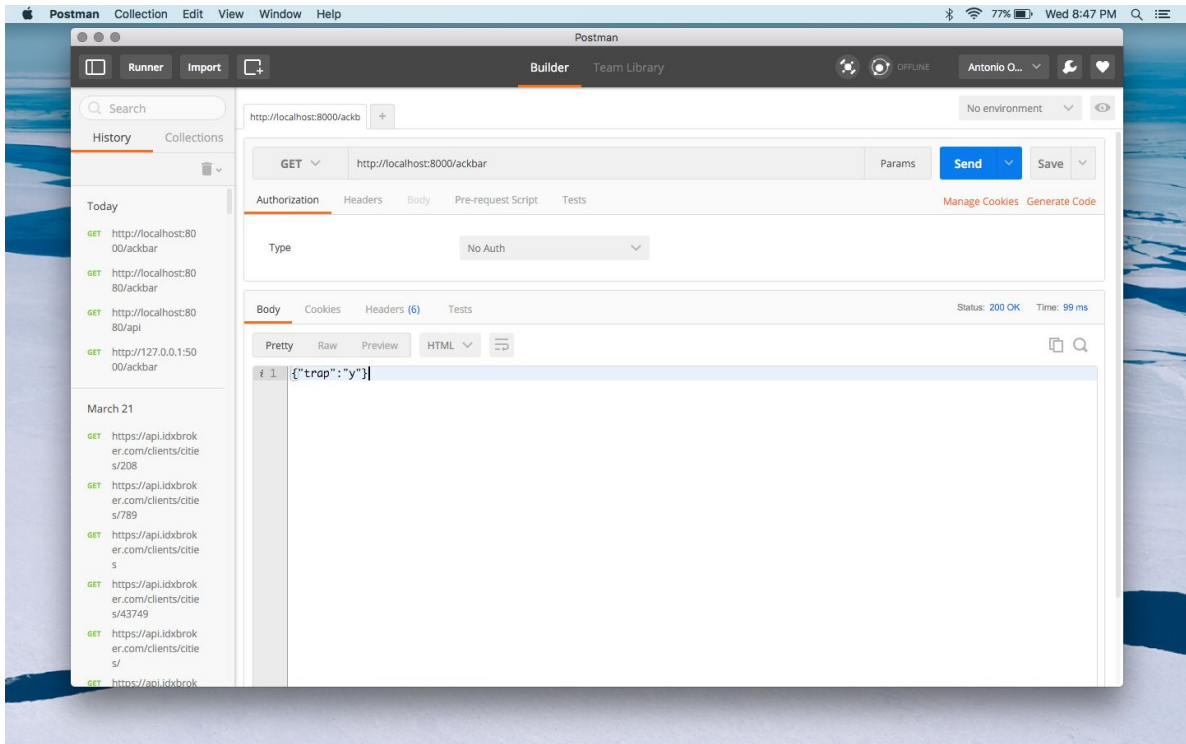
<?php

/*
|-----$
| Application Routes
|-----$
|
| Here is where you can register all of the routes for an $
| It is a breeze. Simply tell Lumen the URIs it should res$
| and give it the Closure to call when that URI is request$
|
*/

$app->get('/ackbar', function () use ($app) {
    return '{"trap": "y"}';
});

File Name to Write: routes.php
^G Get Help      ^T To Files      M-M Mac Format   M-P Prepend
^C Cancel        M-D DOS Format   M-A Append      M-B Backup File
```

# Time to first route under 20 min



Lumen

---

# We are already a little RESTful!

RESTfulness from the framework

- Returning common web status codes.
  - We have controllers in place already to keep our routes organized
  - We can version easily
  - Using common web verbs
  - Framework code is well documented
  - There is community
  - Takes so little time to get started
-



**In the end I just want  
something delicious -**  
consistent and easy to consume.

Mostly I want a tall cool glass of JSON.

Now we can serve up data for others to  
enjoy!



**With easy-to-use  
and well  
documented API  
frameworks we  
can spend more  
time on API  
details.**

Decide on auth to use

Choose how to handle versioning

Organize the information in our  
JSON responses

Spend more time on documentation

Create example calls and returns

---

# Now devs love you

A background image of two sea turtles swimming in clear blue water. The turtle on the left is facing forward with a neutral expression. The turtle on the right is facing the camera with its mouth wide open, showing its teeth and tongue, as if it is shouting or calling out. Both turtles have green and brown patterned shells and heads.

APIs that devs can build on will, help your API community grow.

All devs including you will appreciate dev love.

**I like turtles!**

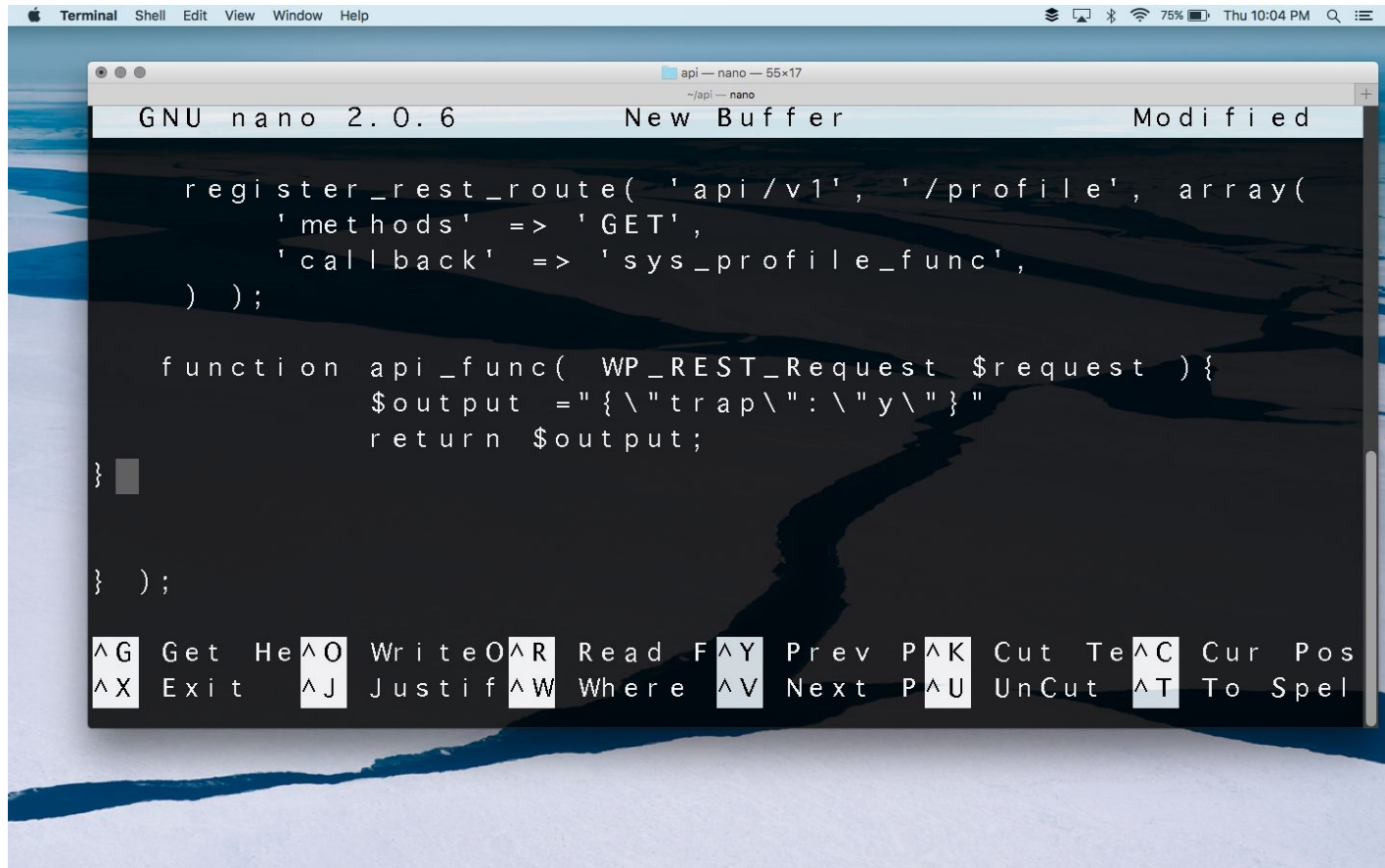
---

# WordPress API

Depending on who you ask  
WP is 26% to 33% of all websites.



# WP Plugin code needed for first route



```
GNU nano 2.0.6 New Buffer Modified

register_rest_route( 'api/v1', '/profile', array(
    'methods' => 'GET',
    'callback' => 'sys_profile_func',
) );

function api_func( WP_REST_Request $request ){
    $output = "{\"trap\": \"y\"}";
    return $output;
}

} );

^G Get Help ^O Write Out ^R Read From ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where ^V Next Page ^U UnCut ^T To Spell
```

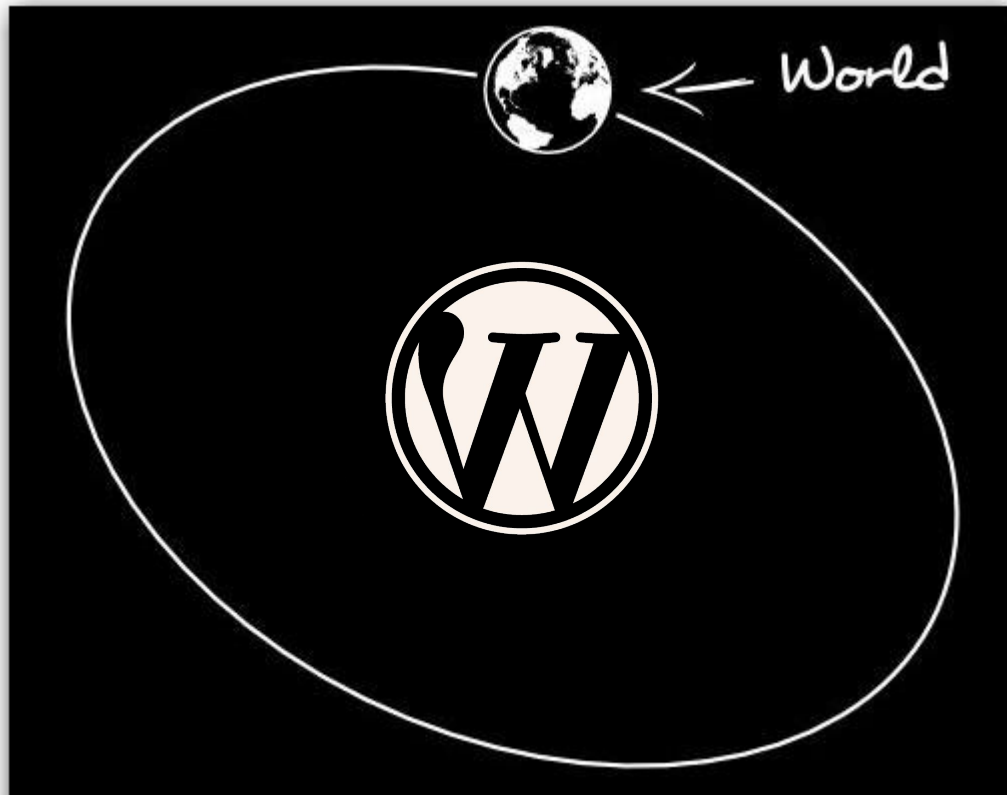
# Time to first route, more like an hour...

- You need to know how to make a plugin first
- The documentation isn't very well established
- Little community
- For all the hype, this is phase one.
- Not really RESTful yet

**Even the people I know that live in an entirely WP world have yet to touch it.**

Has the potential to do everything that WordPress has done for websites.

Interpret that how you like.





## 3. REST API Philosophy

APIs are more about delivering a good Dev Experience than great coding.

→ **Design**

Deliver data needed that others easily consume

→ **Configure**

Make changes to deliver more or more easily





# Delivering a good API will help the platform and your data.

## Making

What kind of data can you provide to the public?

What kind of data can you provide for yourself?





# Good luck!

Make your own APIs.

For your own use or public use.



**Send me a link to your public API!**

[antonioortegajr@gmail.com](mailto:antonioortegajr@gmail.com)

[twitter.com/antonioorteagjr](https://twitter.com/antonioorteagjr)

[GtiHub.com/antonioortegajr](https://github.com/antonioortegajr)

# Let's talk about it

**I want to  
provide an API!**

Tell everyone more.

**What about  
beyond setting  
up a route?**

That will depends on your  
goals. What are they?

**I used SOAP  
before...**

You miss it?

*Links and slides on GitHub:*

*<https://github.com/antonioortegajr/meet-up-api-from-zero-to-first-route>*