

# APIs Design

Pablo Trinidad - @PabloTrinidadPa

p@pablotrinidad.me

# Hello



# Prerequisites

# Developers!

You'll be designing an interface for *“programmers”*

# Perspective

Perspective of an API user over an API designer

# Goal

Start asking – what are we trying to achieve?

A close-up photograph of a baby's face, looking slightly to the left. The baby has light brown hair and is wearing a green bib. The background is a soft, out-of-focus grey. The word "Success" is overlaid in large white letters.

# Success

*“Is measured by how quickly developers can get up to speed and start enjoying success using your API.”*

# REST

Software Architectural Style –  
A.K.A The internet now a days



# Resources

Use nouns not verbs!

/getAllMovies

/getMovies/<id>

/addMovie

/deleteMovie

/updateMovie/<id>

/getAllMovies

/getMovies/<id>

/addMovie

/deleteMovie

/updateMovieTitle/<id>

/updateMovieDirector/<id>

/updateMovieReleaseDate/<id>

/updateMovieDirectors/<id>

...



/movies

/movies/<id>

# HTTP Verbs

GET, POST, PUT, PATCH, DELETE

**GET** /movies *list movies*

**POST** /movies *creates a movie*

**PUT** /movies/<id> *updates movie*

**PATCH** /movies/<id> *partially updates a movie*

**DELETE** /movies/<id> *removes a movie*



Chuck Norris



# Plurals

`/movies` instead of `/movie`

# Relationships

Be careful with nested  
relationships

# Relationships

If a resource can only exist within another resource:

**GET** /team/<id>/players *lists all players of an specific team*

**GET** /team/<id>/players/<id> *retrieves player of a team*

**POST** /team/<id>/players *creates a new player in a team*

**PUT** /team/<id>/players/<id> *updates player of a team*

**DELETE** /team/<id>/players/<id> *removes a player from a team*

# Use `?`

**GET** /user?age=18&city=SF0

# Filtering

Filter by fields, i.e:

**GET** /movies/?status=released

# Sorting

**GET** /movies/?sort=-release\_date,updated\_at

# Searching

**GET** /movies/?q=SOME\_AWESOME\_QUERY\_TEXT

# Aliases

For common queries, i.e:

**GET** /movies/released



# Fields

Limiting the fields the API response contains.

**GET** /movies?fields=id,directors,status

# Pagination

Limit the number of results per request

# Pagination

```
GET /movies - 200 OK
{
  "status": 200,
  "results": [],
  "next": "/movies?page=4",
  "previous": null,
}
```

# Pagination

```
GET /movies?page=3 - 200 OK
{
  "status": 200,
  "results": [],
  "next": "/movies?page=4",
  "previous": "/movies?page=2",
}
```

# Pagination with Metadata

```
GET /movies?page=3 - 200 OK
{
  "status": 200,
  "results": [],
  "total": 94658,
  "next": "/movies?page=4",
  "previous": "/movies?page=2",
}
```

# HTTP *Status* codes

# HTTP Status codes

**Use at least 3:**

- 200 – Ok
- 400 – Bad request
- 500 – Internal Server Error

# HTTP Status codes

**Be gentle:**

- 201 – Created
- 304 – Not modified
- 404 – Not found
- 401 – Unauthorized
- 403 – Forbidden



# Return resource

Updates and creation should return a resource representation

```
POST /movies - 201 Created
{
  "id": 1,
  "title": "Inception",
  "status": "released"
}
```

# Handle errors



# Simple messages

```
{  
  "code": 845,  
  "message": "Invalid access token",  
  "description": "The provided access token has  
                  expired"  
}
```

# Validation messages

```
{  
  "code": 438,  
  "message": "Validation failed",  
  "errors": [  
    {  
      "code": 123,  
      "field": "title",  
      "message": "title length must be less than 50"  
    }...  
  ]  
}
```

# Versioning

Specify the version on the URL, i.e:

**GET** /v1/movies

# Subdomains

Consolidate API under one subdomain i.e:

**api**.gatos.io

# Stability & Consistency

# JSON First

Set type on URL whenever its posible, i.e:

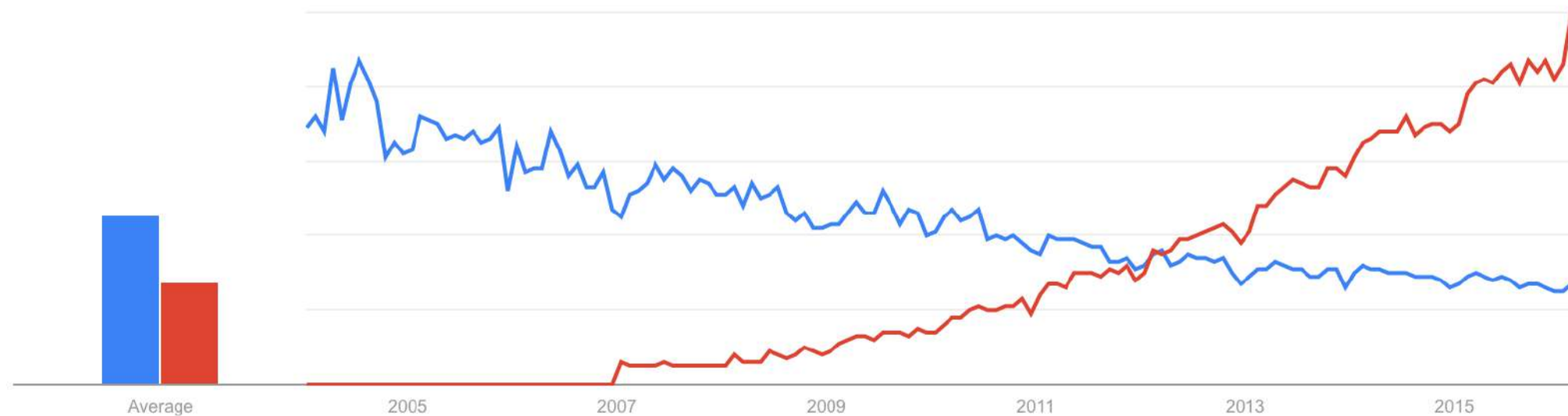
**GET** /movies?type=json



# JSON First

Interest over time ?

News headlines ? Forecast ?



</>

**camelCase**

**VS**

**snake\_case**

# camelCase **VS** snake\_case

snake\_case is **20%** easier to read than camelCase!

# Authentication

RESTful APIs should be stateless. Access  
Tokens are always a good idea ;)

# Browser

Browser explorable APIs are F\*\*king  
**AWESOME!**



**Document *your* API!**

# Document your API!



# Docs!

*“One thing you hate more than having to write documentation is having to try to use an undocumented API”*



# Pro Tips

4GIFs.com

# SSL

Everything



# Cache

**Everything**

*Hey, Don't even worry about it.*



A man in a grey suit and tie is smiling and looking towards a woman with blonde hair. The background is a dimly lit room with a window and a lamp.

# validate

Everything

S

A close-up photograph of a man with a beard and mustache, wearing a dark blue sweater, sleeping peacefully on a light-colored couch. His eyes are closed, and he has a slight smile. His hands are clasped near his face. The background is softly blurred, showing more of the couch and pillows. Overlaid on the right side of the image is the text 'Protect Against CSRF' in yellow and white.

**Protect Against**  
**CSRF**





Limit the

# Requests



Complement with a  
**SDK**



# Gracias

## APIs Design

Pablo Trinidad – @PabloTrinidadPa