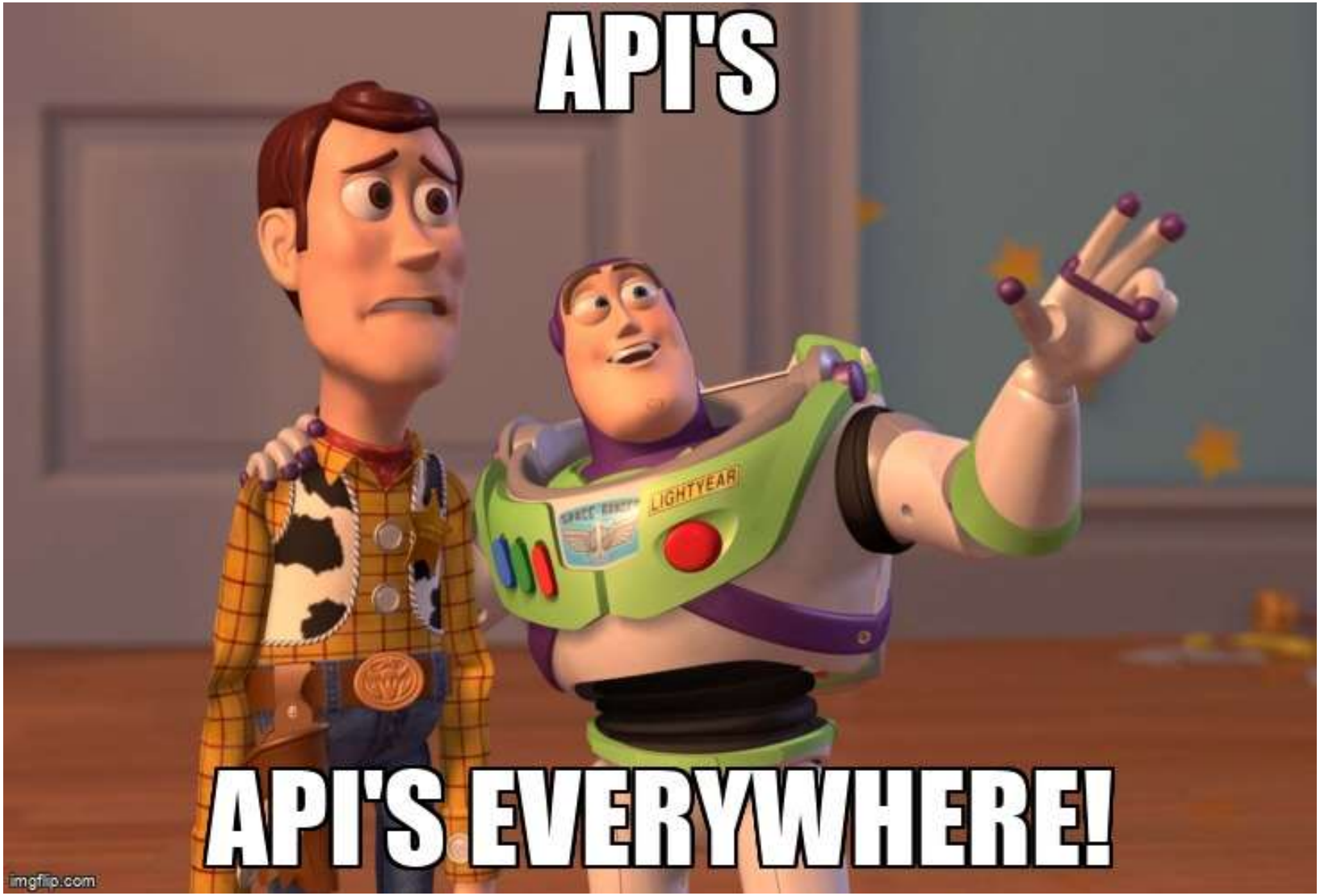
API

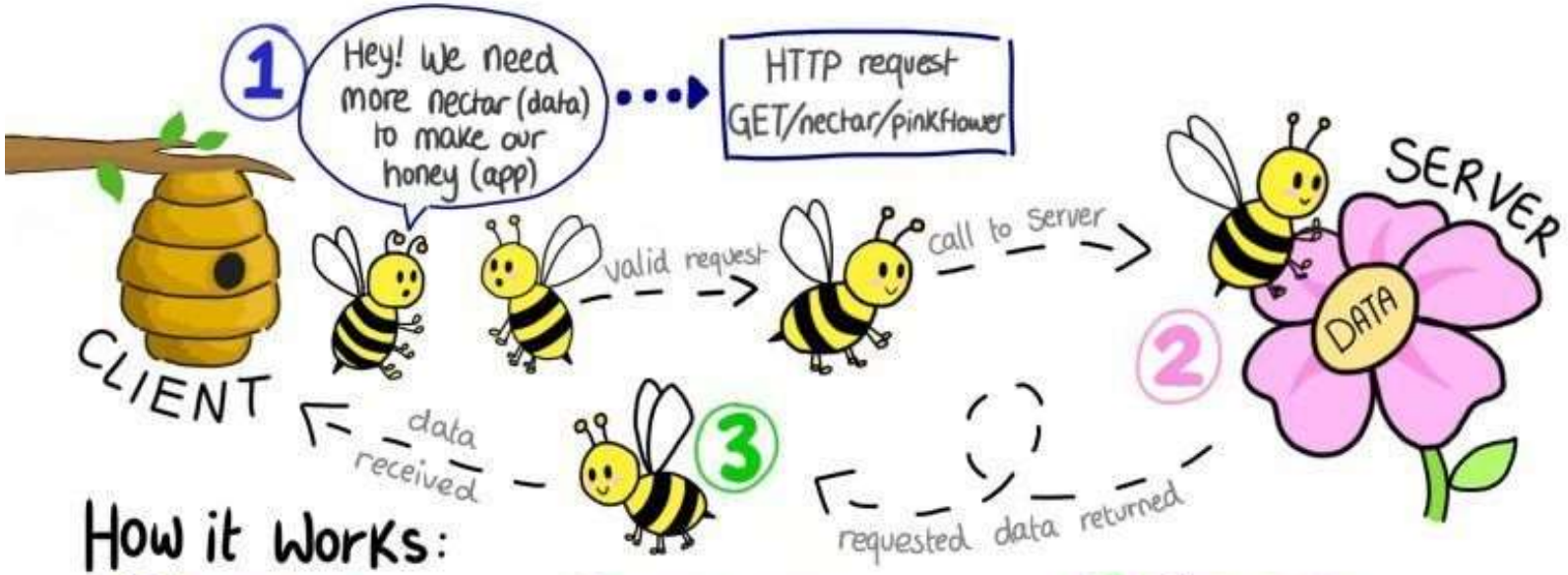

API'S
LIGHTYEAR
API'S EVERYWHERE!
imgflip.com


RITIVO
L'APE NSATRICE
APEROL
1919
APE COR
zzzzzzzzz...
L'APE NNICHELLA
TORINO

What is an API?

@Rapid_API

An application programming interface allows two programs to communicate. On the web, APIs sit between an application and a web server, and facilitate the transfer of data.

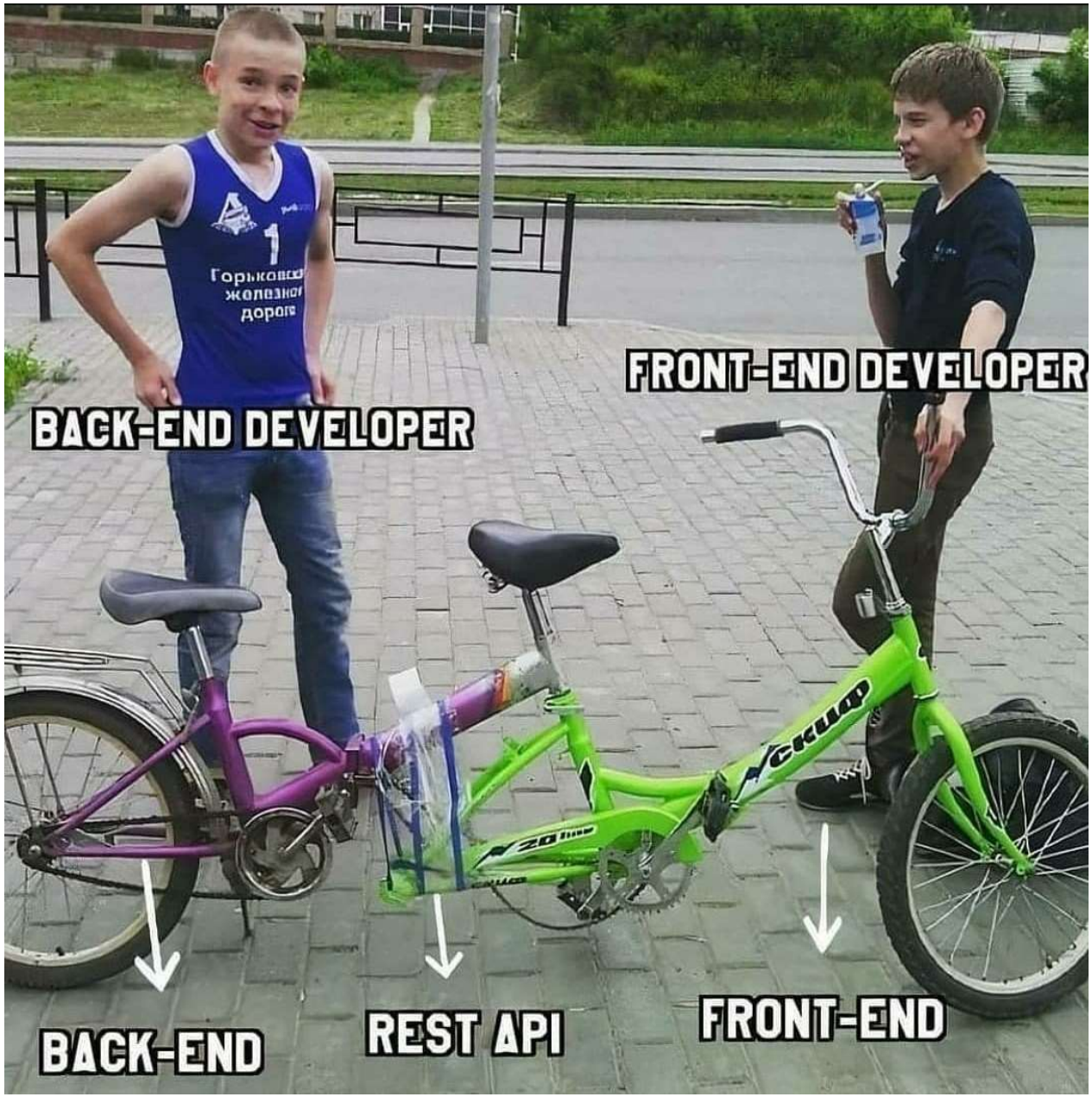


How it Works:

1 Request
API call is initiated by by the client application via a HTTP request

2 Receive
Our worker bee acts as an API, going to a flower (server) to collect nectar (data)

3 Response
The API transfers the requested data back to the requesting application, usually in JSON format


BACK-END DEVELOPER
FRONT-END DEVELOPER
BACK-END
REST API
FRONT-END

Web API - REST, HTTP, CODICI DI STATO

Facciamo un po di chiarezza!

Il termine **REST** è acronimo di **Representational State Transfer**.

Appare per la prima volta nella tesi di dottorato dell'informatico statunitense Roy Fielding.

Web API - REST, HTTP, CODICI DI STATO

Con **REST**, Fielding descrive un **approccio architetturale** alla creazione di Web API basate sul protocollo HTTP, che tenga conto delle seguenti caratteristiche:

- Risorse accessibili tramite Endpoint URL
- Utilizzo del formato JSON o XML
- Senza stato (stateless, come HTTP)
- Impiego dei principali Metodi HTTP (GET, POST, PUT, DELETE...)

HTTP = HyperText Transfer Protocol

"protocollo di trasferimento di un ipertesto"

HTTP è il protocollo usato come principale sistema per la trasmissione d'informazioni sul Web, ovvero in una tipica architettura client/server, basato su un sistema di richieste e risposte.

Questo scambio di messaggi avviene tipicamente tra un browser che accede a un web server, o un'app client che accede a un'API.

Ciclo Richiesta/Risposta HTTP

Il messaggio di richiesta è composto di quattro parti:

- Riga di Richiesta (descrizione della richiesta da implementare)
- Sezione Header (eventuali informazioni aggiuntive)
- Riga Vuota (indica che le meta-informazioni sono state inviate)
- Body (corpo del messaggio; opzionale)

Abbiamo visto come una Web API esponga le funzionalità della nostra Web App affinché possano essere usate da altre App Client o da altre parti della stessa Web App, tramite endpoints URL.

Spesso a queste funzionalità corrispondono anche differenti tipi di azioni, come ad esempio **ottenere informazioni** su uno specifico tweet oppure **creare** un nuovo tweet a partire da dati da noi passati all'API (formato JSON)

Allo stesso modo potremmo voler **aggiornare** le informazioni di un evento salvato su un'app calendario, o **cancellare** un appuntamento.

In REST per convenzione il metodo HTTP utilizzato nell'effettuare una richiesta a un Endpoint dell'API corrisponde alla tipologia di azione che vogliamo compiere:

- GET: recuperare una risorsa
- POST: creare una nuova risorsa
- PUT / PATCH: aggiornare una risorsa
- DELETE: cancellare una risorsa

Com'è fatto un messaggio di risposta?

Il messaggio di risposta è molto simile a quello di richiesta:

- Riga di Stato (codice di stato della richiesta - success or failure)
- Sezione Header (eventuali informazioni aggiuntive)
- Riga Vuota (indica che le meta-informazioni sono state inviate)
- Body (opzionale)

HTTP/1.1 200 OK

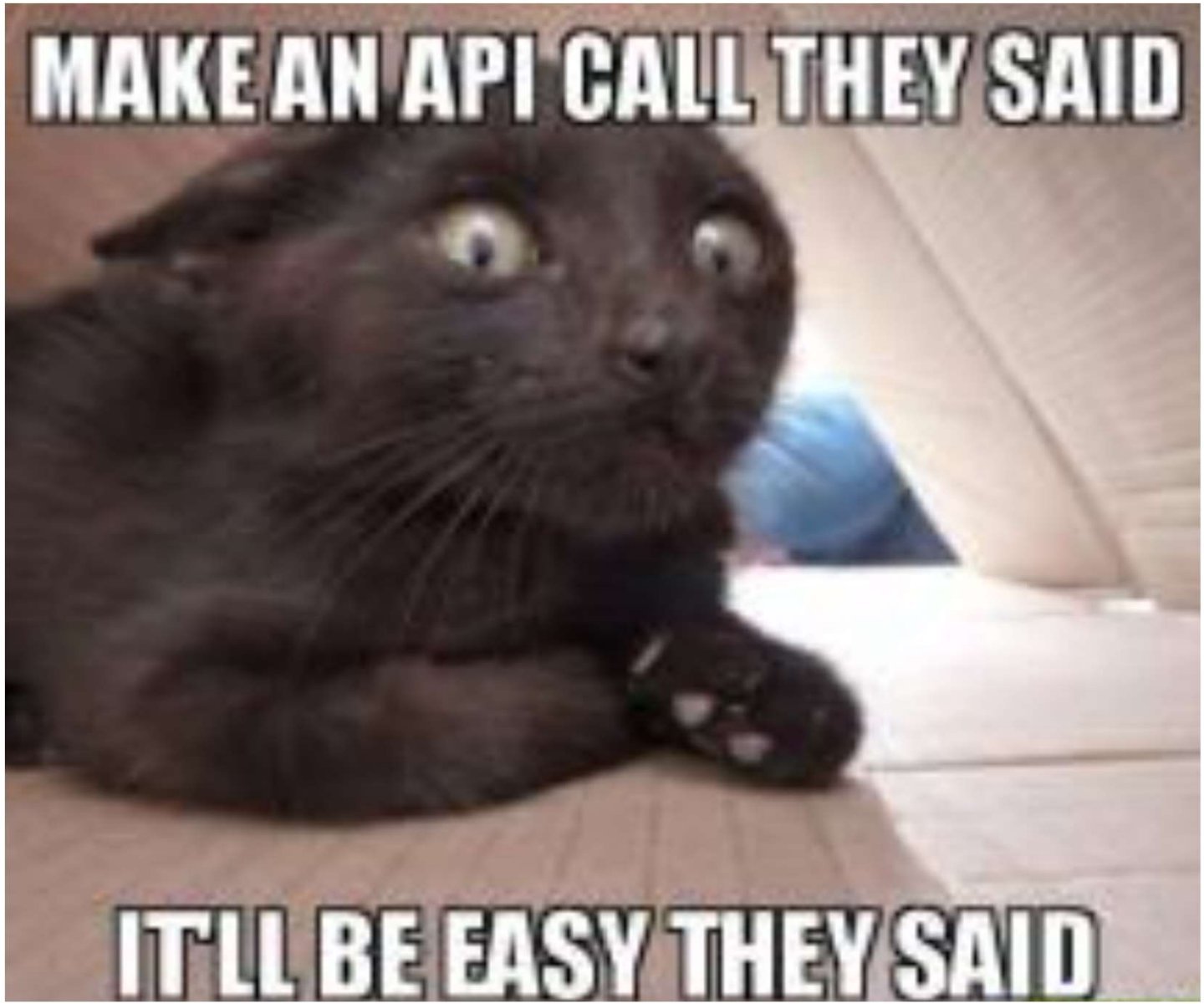
Esempi di **Codici di Stato** in un Messaggio di Risposta:

- 200 OK
- 201 Created
- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found
- 405 Method Not Allowed
- (...)

- 1xx - Informational
- 2xx - Success
- 3xx - Redirezione
- 4xx - Client Error
- 5xx - Server Error

In sintesi

Endpoint URL	Metodo HTTP	Esito Atteso	Status Code
/products/	GET	Elenco di tutti i prodotti (JSON)	200 OK
/products/17/ (questo prodotto esiste)	GET	Dettagli del prodotto con pk 17 (JSON)	200 OK
/products/21/ (questo prodotto non esiste)	GET	Nessun prodotto e messaggio d'errore	404 Not Found
/products/	POST	Creazione di un nuovo prodotto	201 Created
/products/15/	PUT	Aggiornamento del prodotto con pk 15	200 OK
/products/15/	DELETE	Cancellazione del prodotto con pk 15	204 No Content


MAKE AN API CALL THEY SAID
IT'LL BE EASY THEY SAID

HTTP Status Codes

5 Status Codes

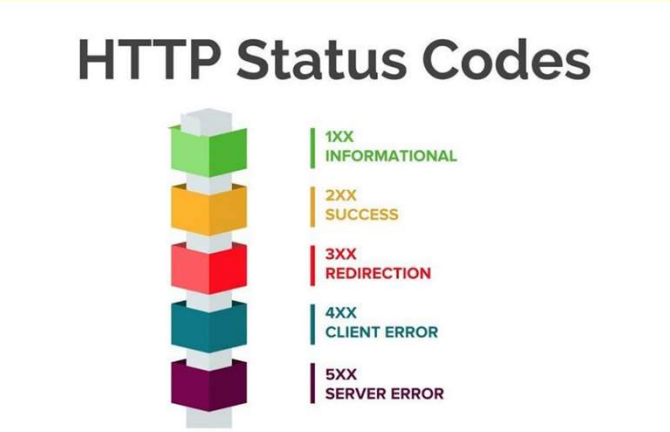
1. 1xx
Informational.

2. 2xx
Success.

3. 3xx
Redirection.

4. 4xx
Client Errors.

5. 5xx
Server Errors.



API Design

1. REST
Representational State Transfer, a design pattern for building web services.

2. SOAP
Simple Object Access Protocol, a messaging protocol for exchanging structured data.

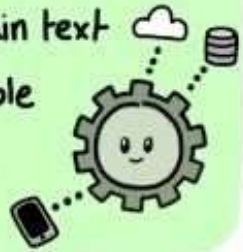
3. GraphQL
A query language and runtime for building APIs.

4. API Gateway
A service that manages, protects, and scales APIs.

API Architecture Types

REST (Representational State Transfer)

- Follows six REST architectural constraints
- Can use JSON, XML, HTML, or plain text
- Flexible, lightweight, and scalable
- Most used API format on the Web
- Uses HTTP



GraphQL

- A query language for APIs
- Uses a Schema to describe data
- Functions using queries and mutations
- Uses a single endpoint to fetch specific data
- Used in apps requiring low bandwidth

SOAP (Simple Object Access Protocol)

- Strictly defined messaging framework that relies on XML
- Protocol independent
- Secure and extensible
- Used in secure enterprise environments

RPC (Remote procedure Call)

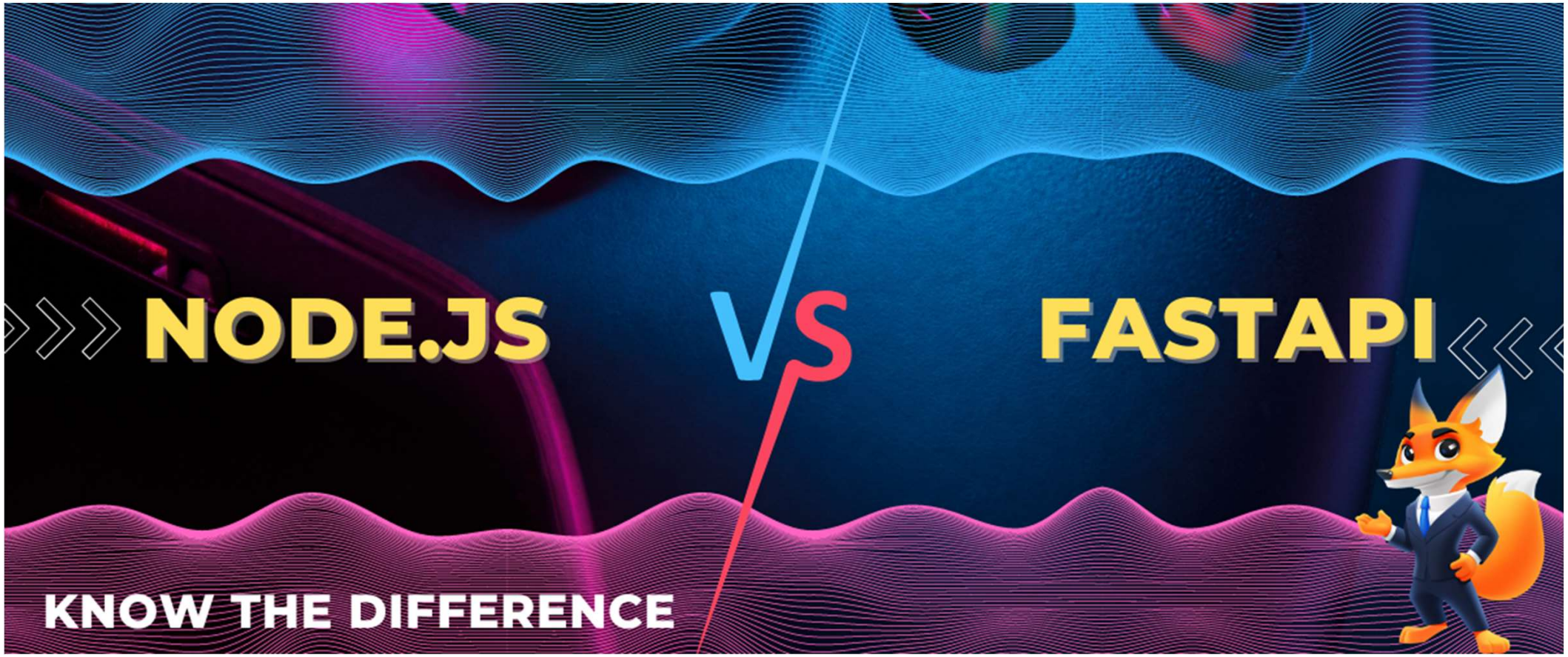
- Action-based procedure great for command-based systems
- Uses only HTTP GET and POST
- Has lightweight payloads that allow for high performance
- Used for distributed systems

Apache KafKa

- Used for live event streaming
- Communicates over TCP protocol
- Can publish, store, and process data as it occurs
- Captures and delivers real-time data e.g. Stock markets

SWAGGER, OPENAPI, REQUEST METHODS & STATUS CODES

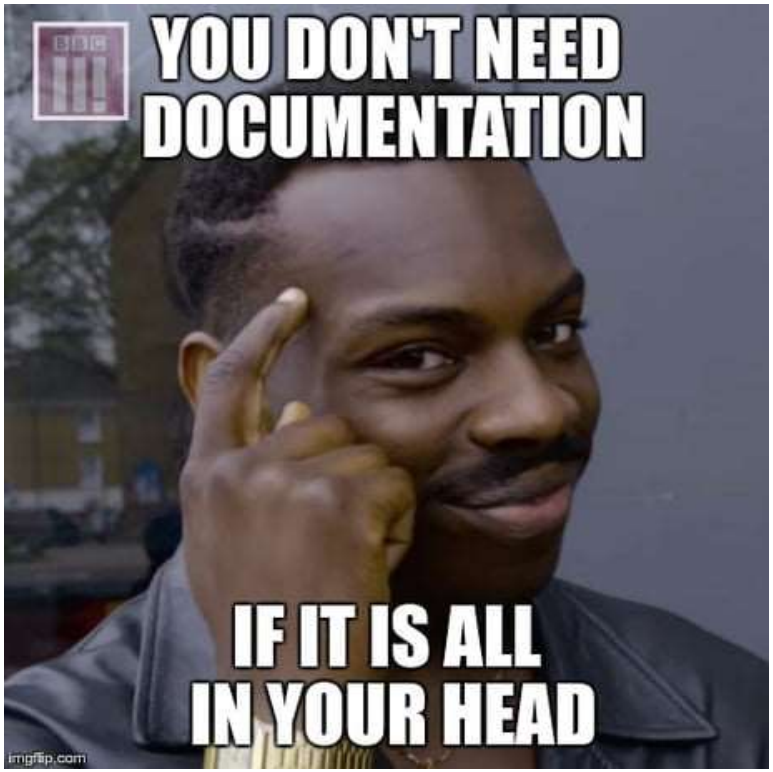
FastAPI



https://fr.hostadvice.com/blog/web-hosting/node-js/fastapi-vs-nodejs

OPENAPI SPECIFICATION (OAS):

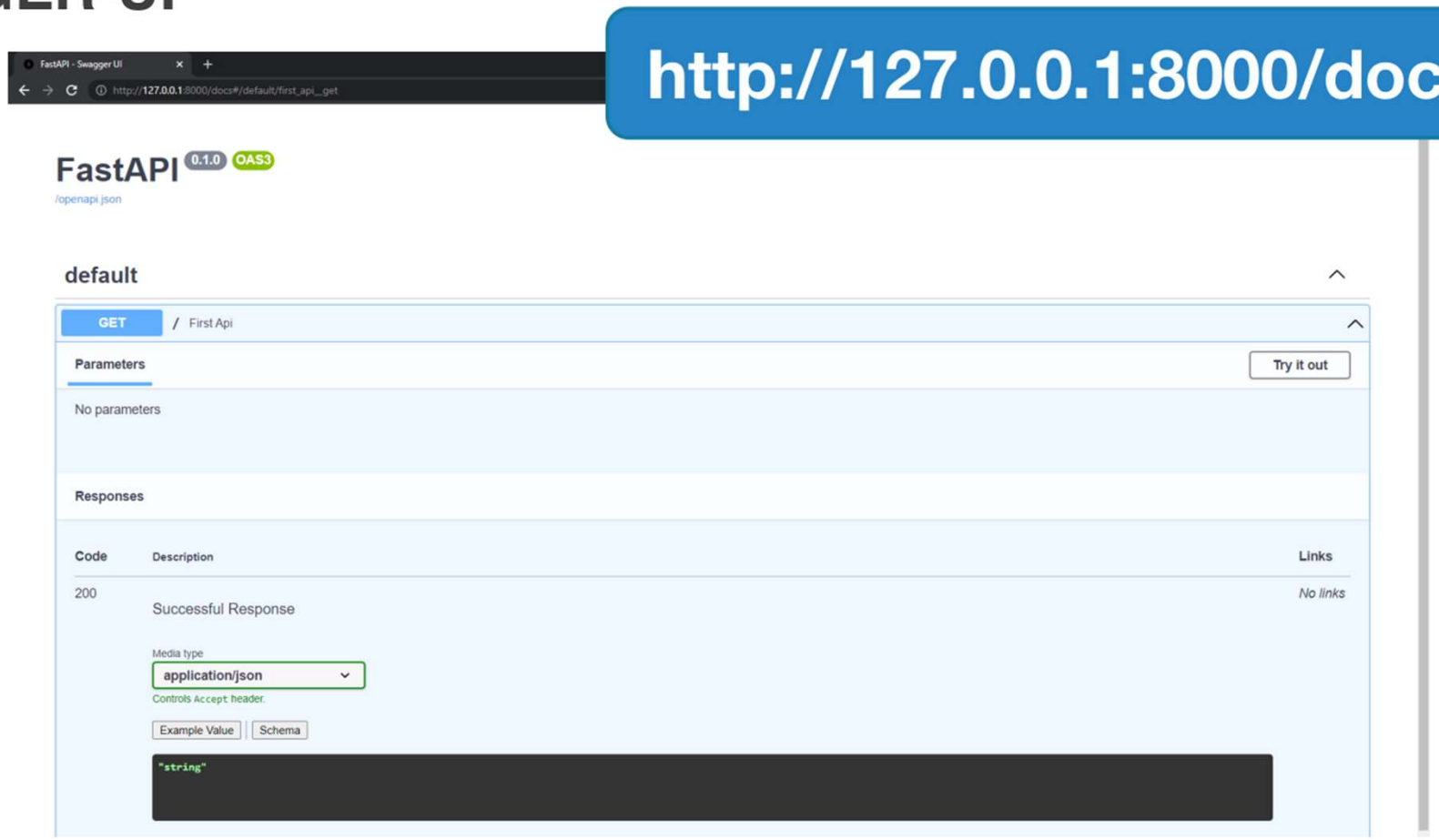
- OpenAPI Document Defines:
  - Schema
  - Data Format
  - Data Type
  - Path
  - Object
  - And much more



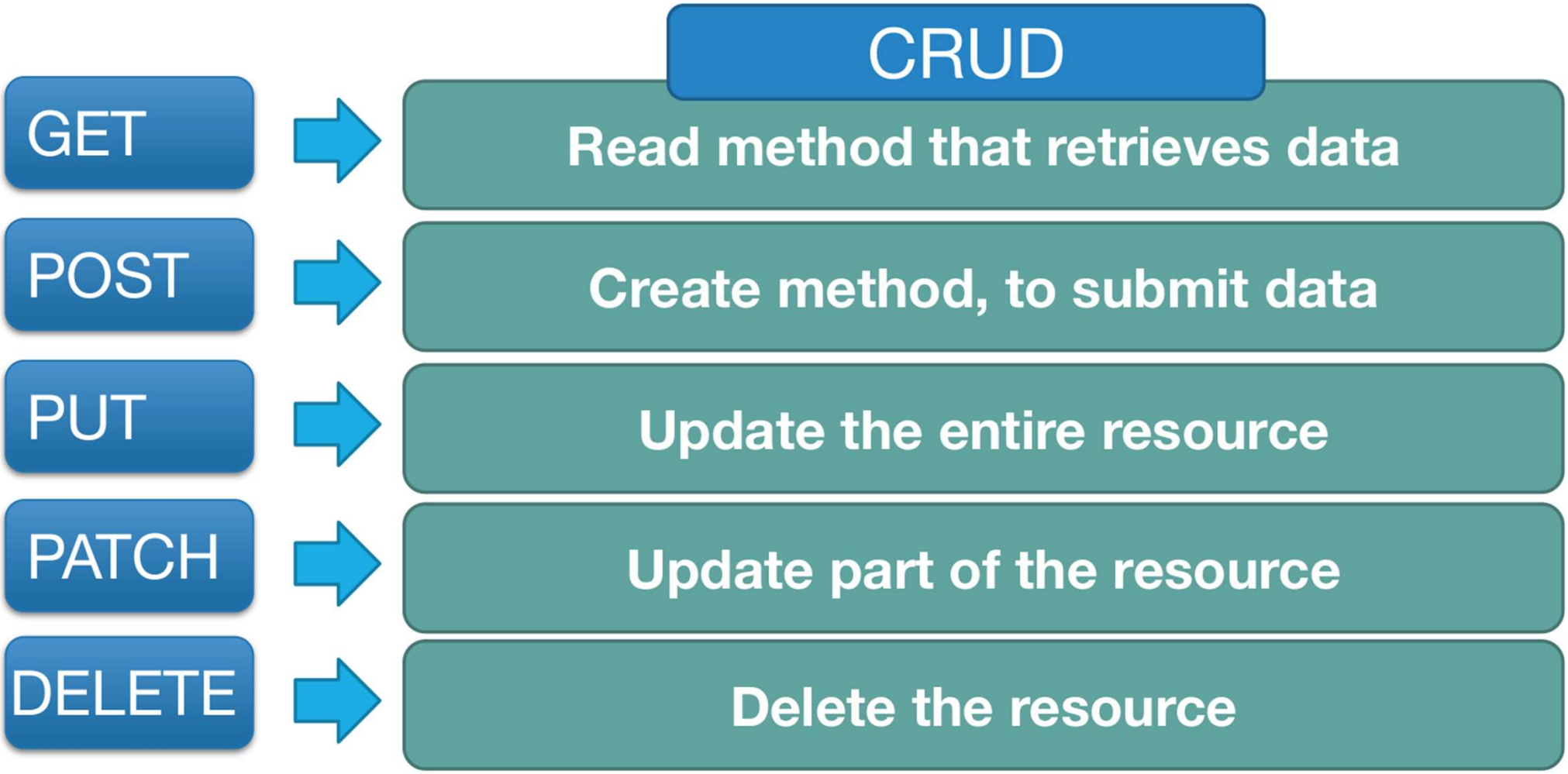
VIEW OPENAPI SCHEMA

- FastAPI generates the OpenAPI schema so you can view
- http://127.0.0.1:8000/openapi.json
- Helps the developer create RESTful APIs based on standards so individuals can use the APIs easily

SWAGGER-UI


FastAPI - Swagger UI
http://127.0.0.1:8000/docs#/default/first_api_get
http://127.0.0.1:8000/docs
FastAPI
0.1.0
OAS3
/openapi.json
default
GET
/ First Api
Parameters
Try it out
No parameters
Responses
Code
Description
Links
200
Successful Response
No links
Media type
application/json
Controls Accept header.
Example Value
Schema
"string"

FASTAPI USES HTTP REQUEST METHODS



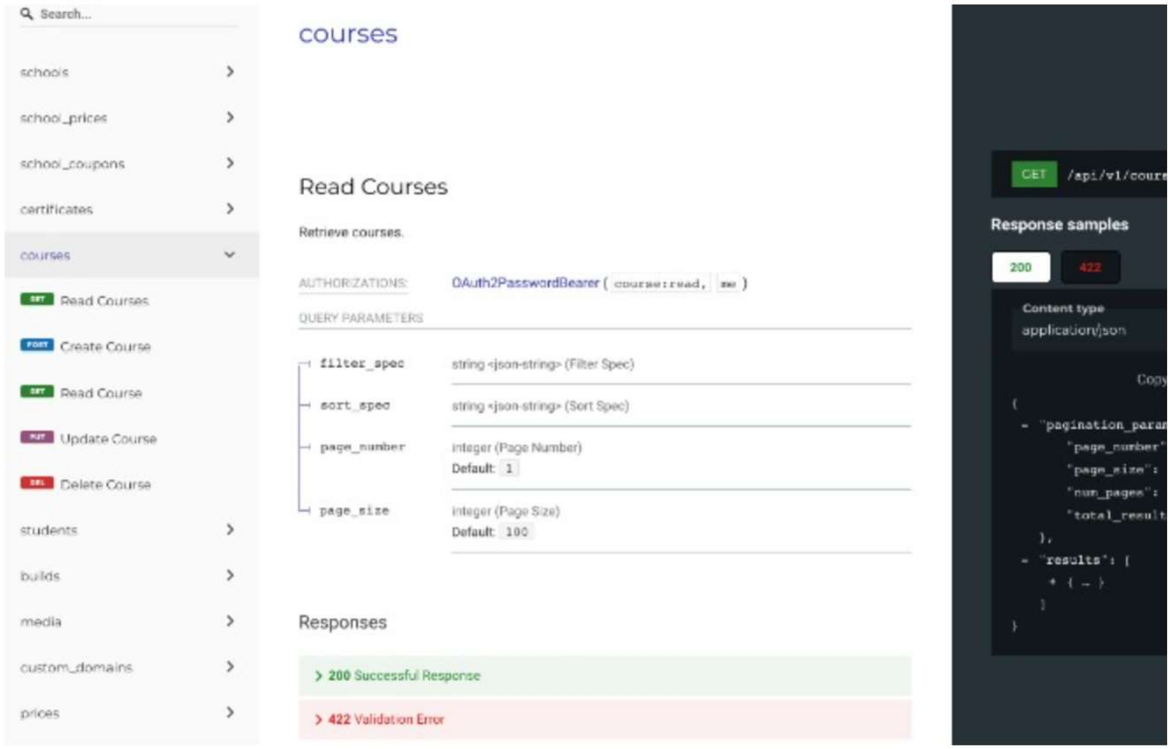
FastAPI 0.1.0 OAS3

/openapi.json

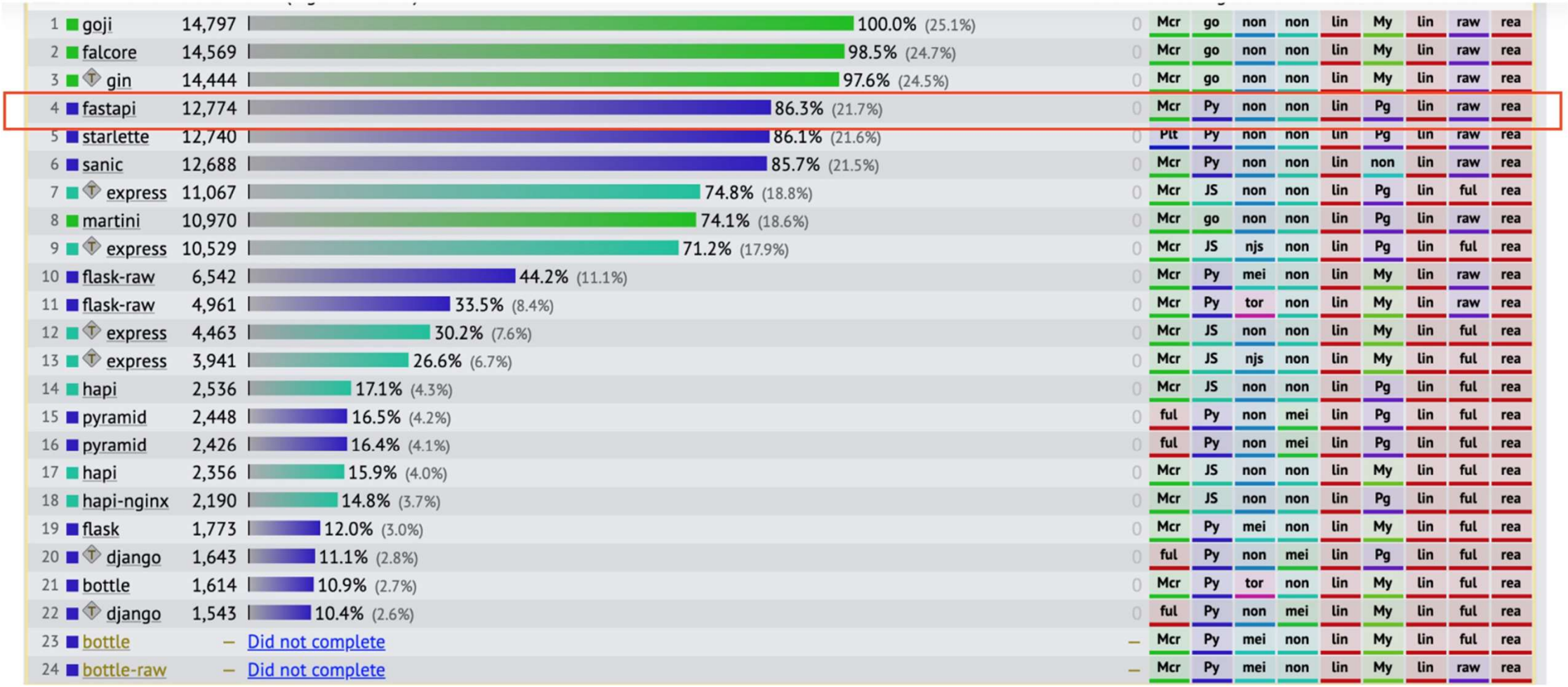
default

- GET /programming_languages List Pro
- POST /programming_languages Create
- GET /programming_languages/{programming_language_id}
- PUT /programming_languages/{programming_language_id}
- DELETE /programming_languages/{programming_language_id}

- ReDoc



PERFORMANCE

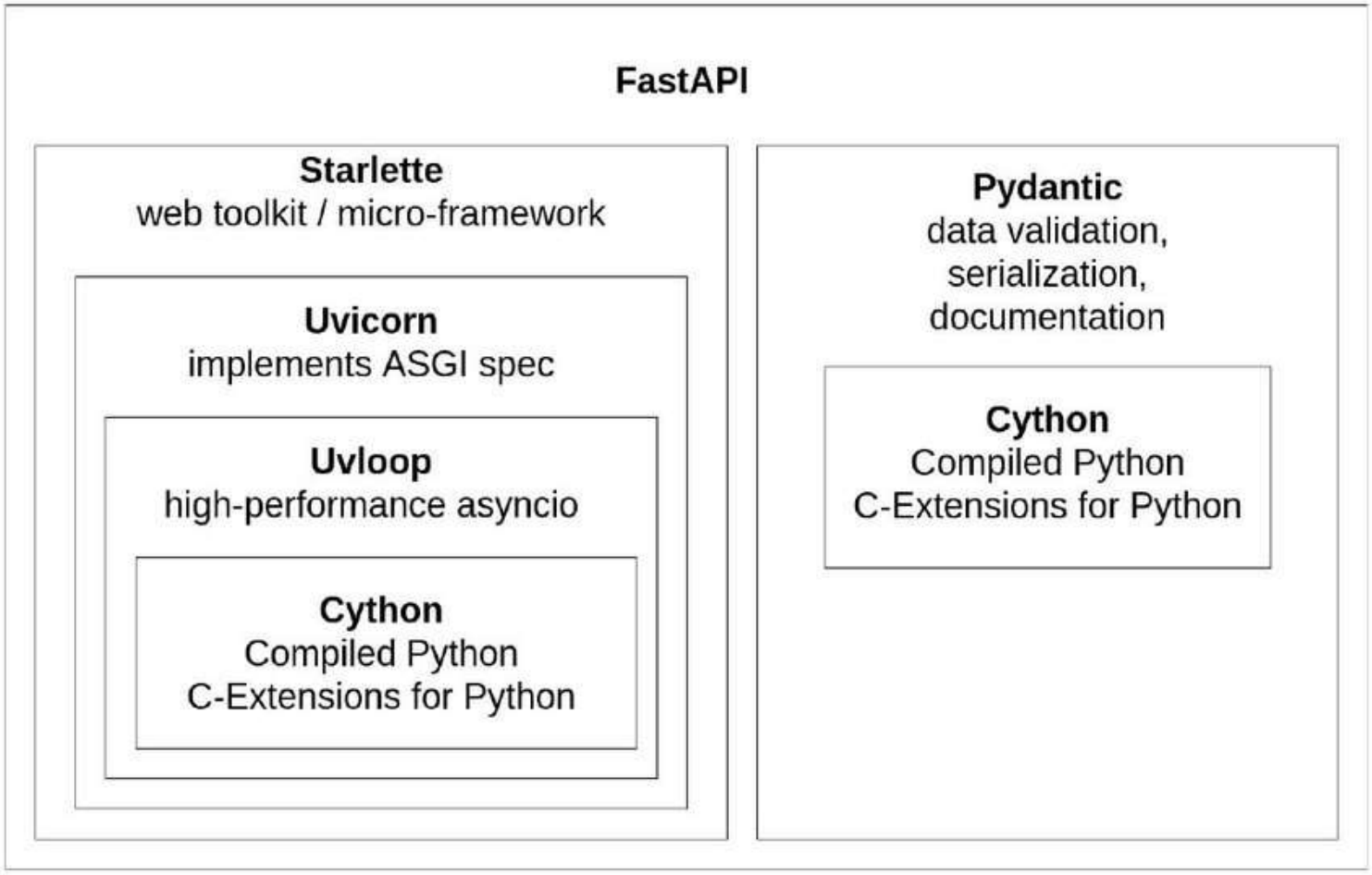


1	goji	14,797	100.0% (25.1%)	0	Mcr	go	non	non	lin	My	lin	raw	rea
2	falcore	14,569	98.5% (24.7%)	0	Mcr	go	non	non	lin	My	lin	raw	rea
3	gin	14,444	97.6% (24.5%)	0	Mcr	go	non	non	lin	My	lin	raw	rea
4	fastapi	12,774	86.3% (21.7%)	0	Mcr	Py	non	non	lin	Pg	lin	raw	rea
5	starlette	12,740	86.1% (21.6%)	0	Plt	Py	non	non	lin	Pg	lin	raw	rea
6	sanic	12,688	85.7% (21.5%)	0	Mcr	Py	non	non	lin	non	lin	raw	rea
7	express	11,067	74.8% (18.8%)	0	Mcr	JS	non	non	lin	Pg	lin	ful	rea
8	martini	10,970	74.1% (18.6%)	0	Mcr	go	non	non	lin	Pg	lin	raw	rea
9	express	10,529	71.2% (17.9%)	0	Mcr	JS	njs	non	lin	Pg	lin	ful	rea
10	flask-raw	6,542	44.2% (11.1%)	0	Mcr	Py	mei	non	lin	My	lin	raw	rea
11	flask-raw	4,961	33.5% (8.4%)	0	Mcr	Py	tor	non	lin	My	lin	raw	rea
12	express	4,463	30.2% (7.6%)	0	Mcr	JS	non	non	lin	My	lin	ful	rea
13	express	3,941	26.6% (6.7%)	0	Mcr	JS	njs	non	lin	My	lin	ful	rea
14	hapi	2,536	17.1% (4.3%)	0	Mcr	JS	non	non	lin	Pg	lin	ful	rea
15	pyramid	2,448	16.5% (4.2%)	0	ful	Py	non	mei	lin	Pg	lin	ful	rea
16	pyramid	2,426	16.4% (4.1%)	0	ful	Py	non	mei	lin	Pg	lin	ful	rea
17	hapi	2,356	15.9% (4.0%)	0	Mcr	JS	non	non	lin	My	lin	ful	rea
18	hapi-nginx	2,190	14.8% (3.7%)	0	Mcr	JS	non	non	lin	Pg	lin	ful	rea
19	flask	1,773	12.0% (3.0%)	0	Mcr	Py	mei	non	lin	My	lin	ful	rea
20	django	1,643	11.1% (2.8%)	0	ful	Py	non	mei	lin	Pg	lin	ful	rea
21	bottle	1,614	10.9% (2.7%)	0	Mcr	Py	tor	non	lin	My	lin	ful	rea
22	django	1,543	10.4% (2.6%)	0	ful	Py	non	mei	lin	My	lin	ful	rea
23	bottle	–	Did not complete	–	Mcr	Py	mei	non	lin	My	lin	ful	rea
24	bottle-raw	–	Did not complete	–	Mcr	Py	mei	non	lin	My	lin	raw	rea


uvicorn

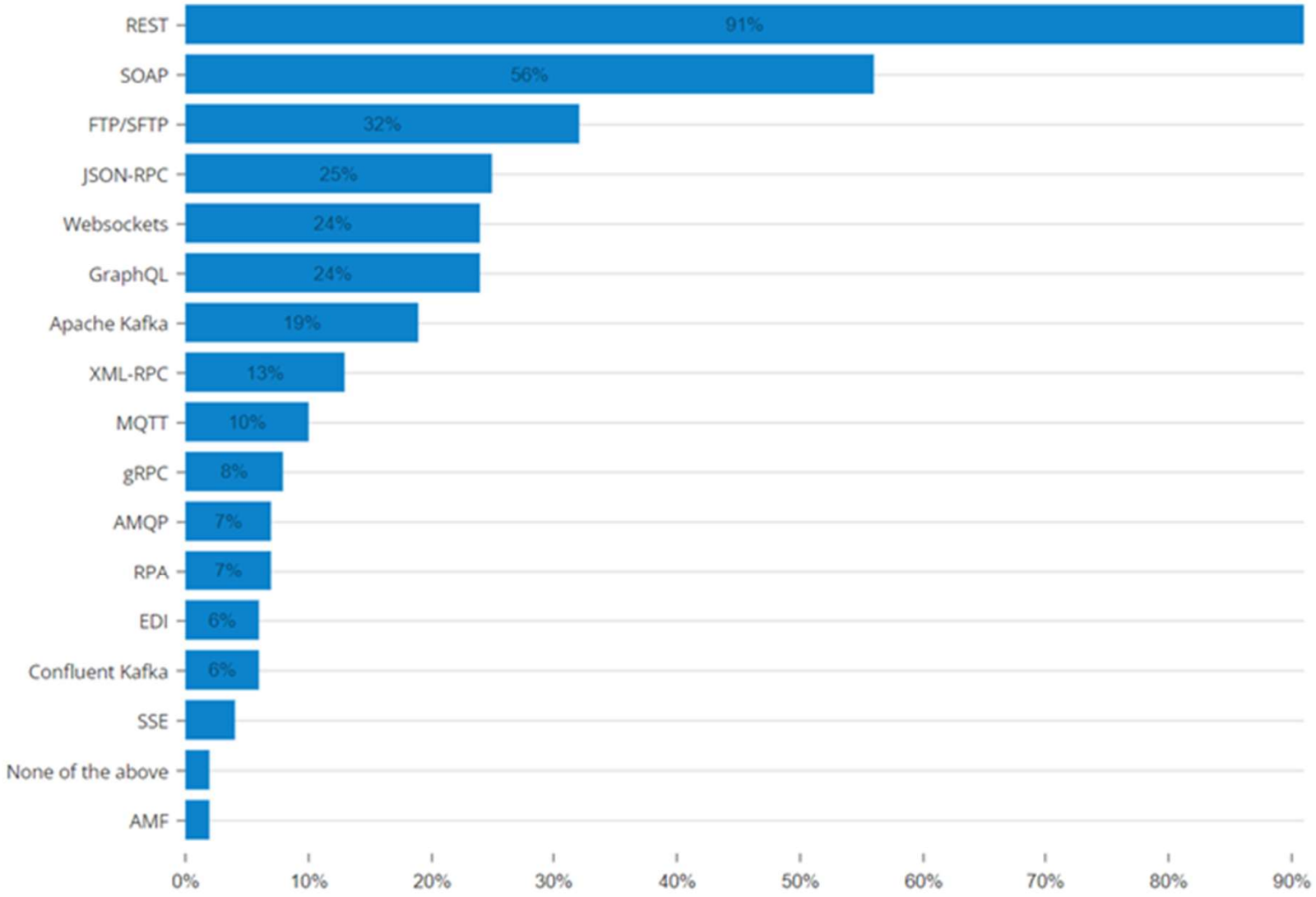

Starlette

Performance with FastAPI



@tiangolo

Does your organization use any of the following API protocols?




OH YOU CAN STORE AND ANALYZE JSON
GOOD FOR YOU


BRACE YOURSELF
JSON IS COMING


WHEN THIRD PARTY API
SENDS XML INSTEAD OF JSON

"So, you're a developer?That means you copy code from stack overflow right?"

Me: