

# Peticiones

**DEV.F**  
DESARROLLAMOS(PERSONAS);

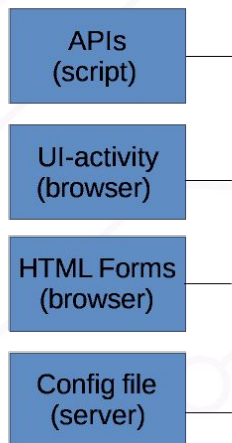
dev

# Mensajes HTTP

Los mensajes HTTP, son los medios por los cuales se intercambian datos entre servidores y clientes.

Hay dos tipos de mensajes: **peticiones**, enviadas por el cliente al servidor, para pedir el inicio de una acción; y **respuestas**, que son la respuesta del servidor.

Activity initiation



Translation  
into HTTP

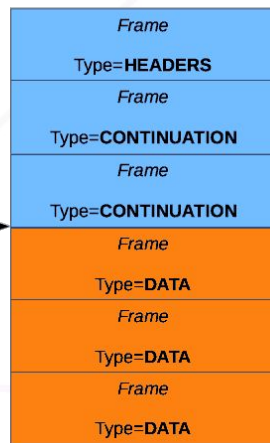
HTTP/1.x message

```
PUT /create_page HTTP/1.1
Host: localhost:8000
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: text/html
Content-Length: 345
```

```
Body line 1
Body line 2
...
```

Binary  
framing

HTTP/2 stream  
(composed of frames)



# Mensajes HTTP

La línea de inicio y las cabeceras HTTP, del mensaje, son conocidas como la cabeza de la peticiones, mientras que su contenido en datos se conoce como el cuerpo del mensaje.

## Petición

Verbo      Recurso      Versión

↑      ↑      ↑

```
GET /index.html HTTP/1.1
Host: wikipedia.org
Accept: text/html
```

## Respuesta

Versión      Código respuesta

↑      ↑

```
HTTP/1.1 200 OK
Server: wikipedia.org
Content-Type: text/html
Content-Lenght: 2026

<html>
...
</html>
```

Primera línea

Encabezados

Cuerpo

# Verbos HTTP

**DEV.F**  
DESARROLLAMOS(PERSONAS);

dev



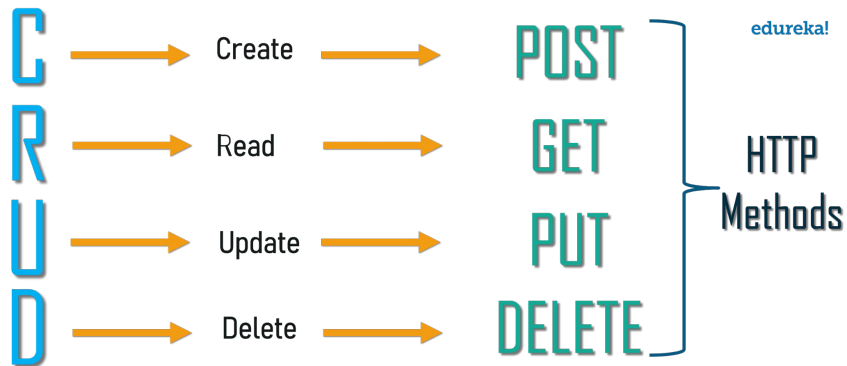
# VERBOS HTTP

La primera línea de un mensaje de petición empieza con un verbo (también se le conoce como método). **Los verbos definen la acción que se quiere realizar sobre el recurso.**

Los verbos más comunes son:

- **GET:** Solicitar un recurso.
- **POST:** Publicar un recurso.
- **PUT:** Reemplazar un recurso.
- **DELETE:** Eliminar un recurso.
- **PATCH:** Actualizar un recurso

Nota: Cuando ingresas a una página desde un navegador, por debajo el navegador envía un mensaje GET, lo mismo cuando oprimos un vínculo a otra página.



# Códigos de Respuesta

**DEV.F**  
DESARROLLAMOS(PERSONAS);

dev



## Response Codes

La primera línea de un mensaje de respuesta tiene un código de 3 dígitos que le indica al cliente cómo interpretar la respuesta.

Los códigos de respuesta se dividen en cinco categorías dependiendo del dígito con el que inician:

- 1XX: Información
- 2XX: Éxito
- 3XX: Redirección
- 4XX: Error en el cliente
- 5XX: Error en el servidor

¿Recuerdas el famoso error 404?

# HTTP Cats

<https://http.cat/>

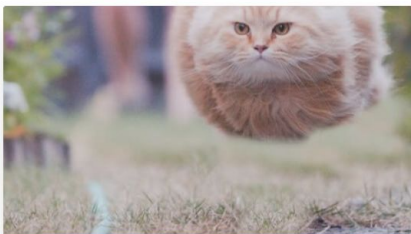


## Usage:

```
https://http.cat/[status_code]
```



**Note:** If you need an extension at the end of the URL just add .jpg.



100

Continue



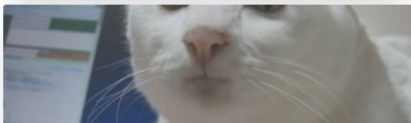
101

Switching Protocols

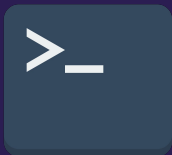


102

Processing







# DEV.F

## **PRO TIP**

*Los códigos de respuesta son una convención no una regla.*

*Cada desarrollador puede asignar el código de respuesta que desee a cada petición, pero se recomienda seguir las convenciones (lo más posible)*

# API Rest

**DEV.F**  
DESARROLLAMOS(PERSONAS);

dev

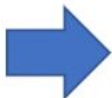
# ¿Qué es Rest?

REST es una interfaz para conectar varios sistemas basados en el protocolo HTTP y nos sirve para obtener, generar datos y operaciones, devolviendo esos datos en formatos muy específicos, como XML y JSON.

- Protocolo cliente/servidor "SIN ESTADO"
- Utiliza verbos http
  - GET (leer)
  - POST (crear)
  - PUT (editar)
  - DELETE (borrar)
- Devuelve
  - JSON
  - XML



HTTP STATUS  
200 – OK  
201 – CREATED  
403 – FORBIDDEN  
404 – NOT FOUND



Ejemplos:

```
GET    /rest/api/pedido/{id}
DELETE /rest/api/pedido/{id}
POST   /rest/api/pedido
PUT     /rest/api/pedido/{id}/producto/{idProducto}
```



Devuelve  
JSON/XML

```
{
  "nombre": "Pepito",
  "coches" : ["Ford", "Fiat", "Audi"]
}
```

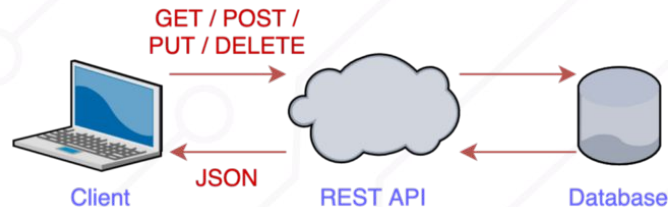
# API + Rest = API Rest

## API

**API** utiliza peticiones HTTP responsables de las operaciones básicas necesarias para la manipulación de datos (GET, POST, etc).

## REST

**Rest**, es un conjunto de restricciones que se utilizan para que las solicitudes HTTP cumplan con las directrices definidas en la arquitectura: cliente servidor, sin estado, con cache, etc.



## API REST

API Rest es el conjunto de buenas prácticas utilizadas en las peticiones HTTP realizadas por una API en una aplicación web.

Es decir, cuando se habla de API Rest, significa utilizar una API para acceder a aplicaciones back-end, de manera que esa comunicación se realice con los estándares definidos por el estilo de arquitectura Rest.

# Nombrando Endpoints en API Rest

`/users` // lista todos los usuarios

`/users/123` // lista a un usuario en específico

`/users/123/orders` // lista los pedidos de un usuario específico

`/users/123/orders/0001` // lista una orden especifica de un usuario específico

# Endpoints y Verbos Http

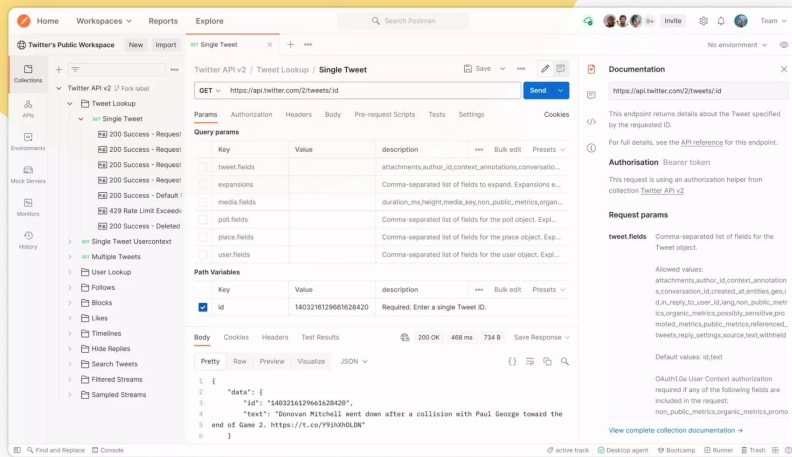
Un mismo Endpoint, hara diferentes acciones dependiendo el verbo http usado para accederlo

Recuros / Estado	POST	GET	PUT	DELETE
<b>/casas</b>	Crea una casa	Devuelve una lista de todos las casas	Modifica casas	Borra todas las casas
<b>/casas/123</b>	error	Devuelve los detalles de la casa <b>123</b>	Modifica la casa	Borra la casa

# Accediendo a API Rest

**DEV.F**  
DESARROLLAMOS(PERSONAS);

dev



# Postman

Postman nace como una herramienta que principalmente nos permite crear peticiones sobre APIs de una forma muy sencilla y poder, de esta manera, probar las APIs.

El interés fundamental de Postman es que lo utilizemos como una herramienta para hacer peticiones a APIs y generar colecciones de peticiones que nos permitan probarlas de una manera rápida y sencilla.



<https://www.postman.com/>

DEV.F



## Download Insomnia

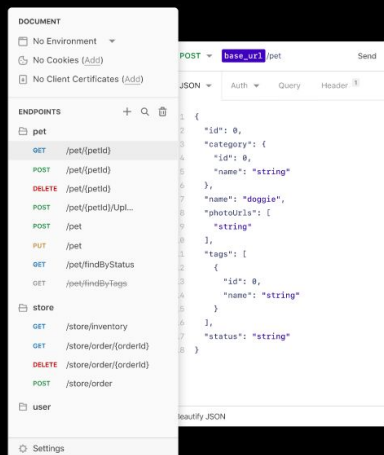
Start building, designing, testing better APIs through spec-first development driven by an APIOps CI/CD pipelines.

Download Insomnia for Windows

By downloading and using Insomnia, I agree to the Privacy Policy and Terms.

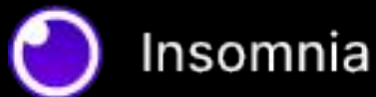
What's New? Changelog

Not your OS? Download for MacOS / Ubuntu or See all downloads.



# Insomnia

Es similar a Postman, con una interfaz un poco más sencilla.



<https://insomnia.rest/>

# APIs para probar

**DEV.F**  
DESARROLLAMOS(PERSONAS);

dev

PokeAPI proudly announces Beta support for GraphQL. Access our free console at [beta.pokeapi.co/graphql/console](https://beta.pokeapi.co/graphql/console) and take a look at the [documentation](#) ✕

Sword and Shield data might be inaccurate and lacking in various aspects due to the fact that it is not taken directly from Nintendo's Pokemon ROMs. If you spot any mistake, please report it [here](#) ✕



The RESTful Pokémon API

Serving over 60,000,000 API calls each month!

All the Pokémon data you'll ever need in one place,  
easily accessible through a modern RESTful API.

[Check out the docs!](#)

Try it now!

<https://pokeapi.co/api/v2/> pokemon/ditto

Submit

Need a hint? Try [pokemon/ditto](#), [pokemon/1](#), [type/3](#), [ability/4](#), or [pokemon?limit=100&offset=200](#).

Direct link to results: <https://pokeapi.co/api/v2/pokemon/ditto>

Resource for ditto

```
▼ abilities: [] 2 items
▼ 0: {} 3 keys
  ▼ ability: {} 2 keys
    name: "lieber"
    url: "https://pokeapi.co/api/v2/ability/7/"
    is_hidden: false
    slot: 1
  ▼ 1: {} 3 keys
    ▼ ability: {} 2 keys
      name: "imposter"
      url: "https://pokeapi.co/api/v2/ability/150/"
      is_hidden: true
      slot: 3
base_experience: 101
```

<https://pokeapi.co/>

# PokéAPI

# SWAPI

The Star Wars API

(what happened to swapi.co?)

All the Star Wars data you've ever wanted:

Planets, Spaceships, Vehicles, People, Films and Species

From all SEVEN Star Wars films

Now with The Force Awakens data!

Try it now!

<https://swapi.dev/api/>

request

Need a hint? try [people/1](#)/or [planets/3](#)/or [starships/9](#)

Result:

```
{
  "name": "Luke Skywalker",
  "height": "172",
  "mass": "77",
  "hair_color": "blond",
  "skin_color": "fair",
  "eye_color": "blue",
  "birth_year": "19BBY",
  "gender": "male",
  "homeworld": "https://swapi.dev/api/planets/1/",
  "films": [
    "https://swapi.dev/api/films/2/",
    "https://swapi.dev/api/films/6/",
    "https://swapi.dev/api/films/3/",
    "https://swapi.dev/api/films/1/",
    "https://swapi.dev/api/films/7/"
  ],
  "species": [
```

# SWAPI

(Star Wars)

<https://swapi.dev/>