

# API TESTING

TESTING BOOTCAMP

# PROYECTO API

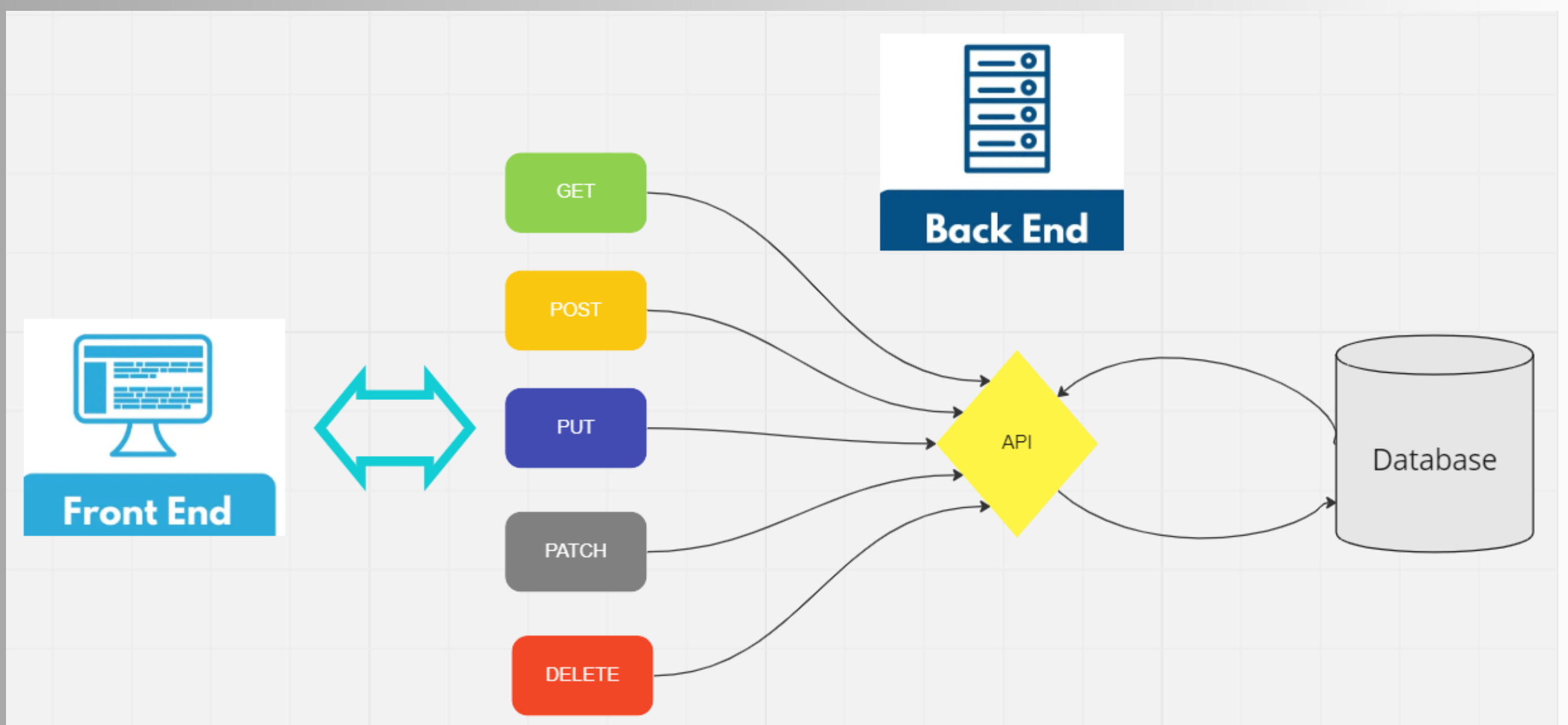
## Objetivo

Entender estructura de JSON.

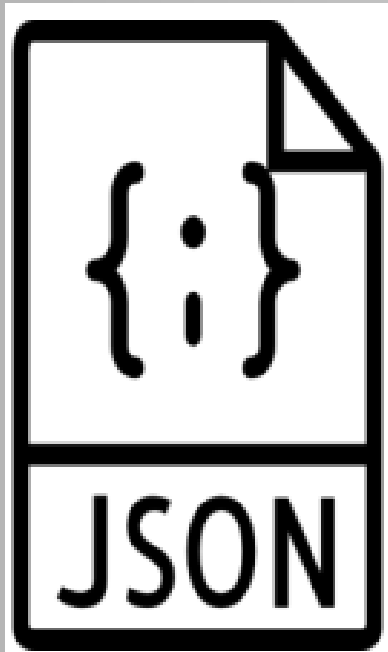
Comprender métodos de request en APIs.

Realizaremos peticiones https. Los métodos para utilizar serán:

- GET
- POST
- DELETE
- UPDATE



# JSON



## ¿Qué es? ¿Qué estructura y datos tiene?

JSON: Se trata de un formato para guardar e intercambiar información que cualquier persona pueda leer. Los archivos JSON contienen solo texto y usan la extensión .json. Se utiliza principalmente para transferir datos entre un servidor y un cliente. Sus siglas en inglés significan JavaScript Object Notation.

### Estructura básica:

`{"propiedad" = "valor"}`

Propiedad: "nombre", "edad", "nombre\_apellido", "numeroTelefono", etc.

Valor: ¿Qué valores puedo poner?

- Strings: conjunto de caracteres dentro de comilla doble .
- Número: numero entero o punto flotante.
- Booleano: verdadero o falso.
- Nulo: null, muestra que no hay información.
- Array: arreglo, colección de valores, entre corchetes y separados por coma.

Nota: Cada propiedad es separada con coma, salvo la última que no lo necesita.

# JSON

## Ejemplo:

```
{
  "Empresa": "MindHub",
  "Id": 6,
  "Cuil": false,
  "Adress": null,
  "Cursos": [
    "MERN Fullstack Mobile Apps",
    "Java Full Stack"
  ]
}
```

## Ejercicio 1:

Crear un JSON con propiedad: nombre, apellido, edad y si tenes mascota.

El json debe contar con un arreglo y alguna propiedad nula.

Debes validar el JSON en el siguiente enlace:

<https://jsonformatter.curiousconcept.com>

Realizar un screenshot a dicha validación donde se observe el json y la comprobación de validado.

# API + TOKEN

Mediante estos ejercicios podrás realizar las peticiones al sitio mostrado. Se proporcionará TOKEN, urls, y códigos de respuesta posibles Así como los scripts q debemos utilizar en el body.

## Ejercicio 2:

### WORKSPACE

- Crea un Workspace llamado “GoRest”.
- Coloca en el summary : Workspace API test.
- Visibility; Personal

#### Create workspace

Name

Summary

Add a brief summary about this workspace.

Visibility

Determines who can access this workspace.

- ☒ Personal  
Only you can access
- ☐ Private  
Only invited team members can access
- ☐ Team  
All team members can access
- ☐ Partner NEW  
Only invited partners and team members can access
- ☐ Public  
Everyone can view

Create Workspace

Cancel

# API + TOKEN

## Ejercicio 2:

### WORKSPACE

- Crea un Workspace llamado “Testing QA”.
- Coloca en el summary : Workspace API test.
- Visibility; Personal

#### Create workspace

Name

Summary

Add a brief summary about this workspace.

Visibility

Determines who can access this workspace.

- ☒ Personal  
Only you can access
- ☐ Private  
Only invited team members can access
- ☐ Team  
All team members can access
- ☐ Partner NEW  
Only invited partners and team members can access
- ☐ Public  
Everyone can view

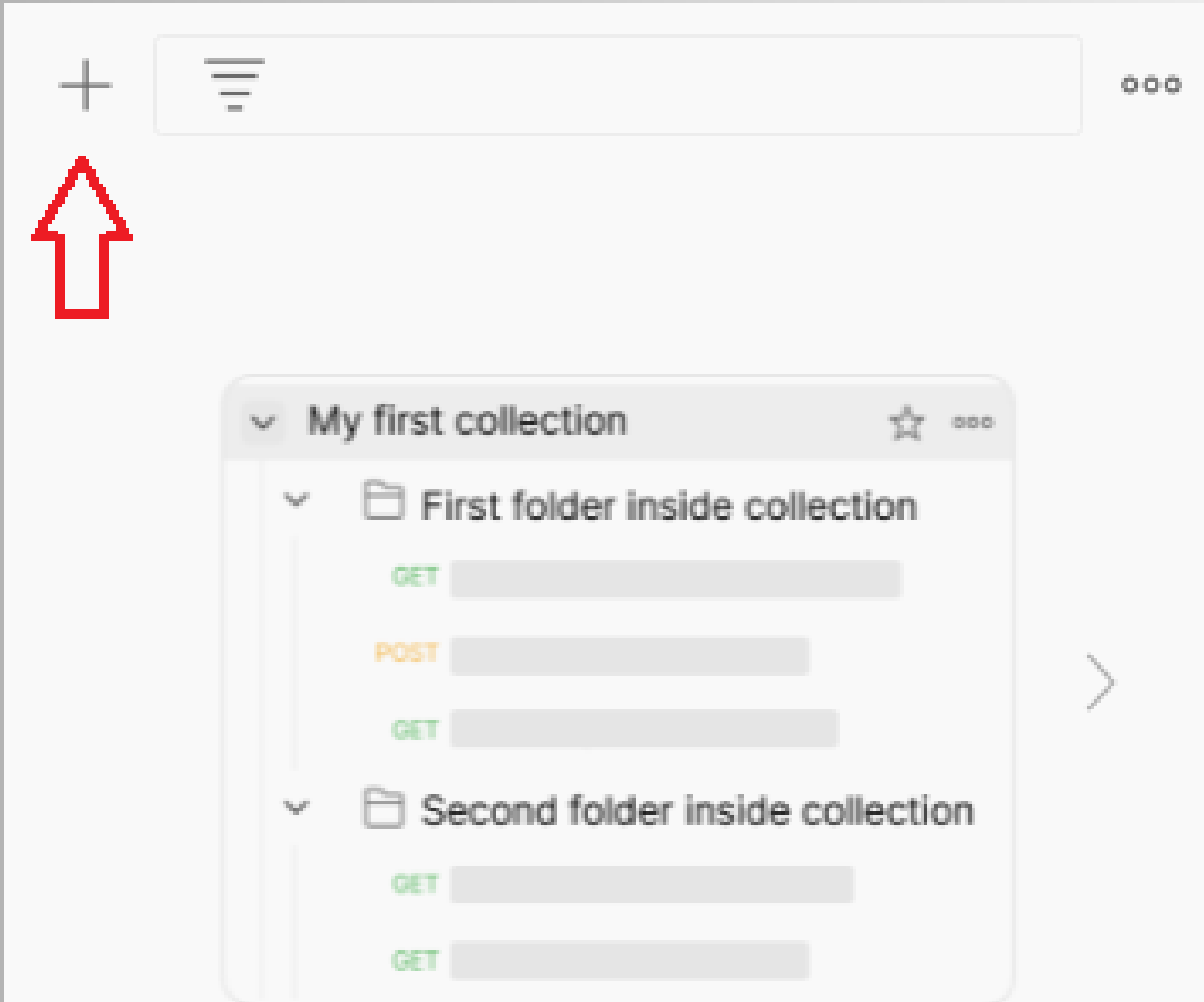
Create Workspace

Cancel

# API + TOKEN

## COLECCIÓN

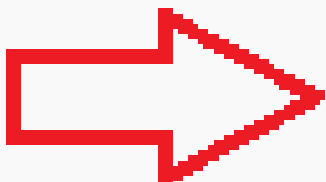
- Crea una colección llamada “GoRest”.



The screenshot shows the Postman interface. At the top left, there is a '+' icon for creating a new collection, which is highlighted by a red arrow. Below it, a collection named 'My first collection' is displayed, containing two folders: 'First folder inside collection' and 'Second folder inside collection'. Each folder contains several requests with methods like GET and POST. A red arrow points to the 'Create Collection' button at the bottom.

**Create a collection for your requests**

A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.

 [Create Collection](#)

or

[Use a Template](#)

# API + TOKEN

## REQUEST

- Crea las request correspondientes para cada método.
- Utiliza el Bearer Token en los métodos POST, PUT, PATCH y DELETE:

428a76866874be289dfe43958d71a6bf2431460f496af449c62ed3d31dc2f138

- Utiliza la siguiente estructura como json body:

```
{
  "name": "Tenali Ramakrishna",
  "gender": "male",
  "email": "tenali.ramakrishna@15ce.com",
  "status": "active"
}
```

## ENDPOINTS

TipodePetición	Endpoint
POST	https://gorest.co.in/public/v2/users
GET	https://gorest.co.in/public/v2/users
PUT / PATCH	https://gorest.co.in/public/v2/users/{{userID}}
DELETE	https://gorest.co.in/public/v2/users/{{ userID}}



# API + TOKEN

GoRest	
GET	Get users
POST	Post user
PUT	Put/Patch user
DEL	Delete user

## TESTS AUTOMATION

- Crea los tests correspondientes para cada método.
- Ejecuta la colección completa.

Params	Authorization	Headers (6)	Body	Pre-request Script	Tests ●	Settings
1	pm.test("Status code is 200 OK", function () {					
2	pm.response.to.have.status(200);					
3	});					

All Tests	Passed (3)	Failed (1)	Skipped (0)
Iteration 1			
GET Get users			
https://gorest.co.in/public/v2/users			
PASS	Status code is 200 OK		
POST Post user			
https://gorest.co.in/public/v2/users			
PASS	Status code is 201 created		
PUT Put/Patch user			
https://gorest.co.in/public/v2/users/469522			
PASS	Status code is 200, user update		
DELETE Delete user			
https://gorest.co.in/public/v2/users/463538			
FAIL	Status code is 204, delete   AssertionError: expected response to have status code 204 but got 404		