



Fundamentos » APIs:

» ¡Hola a todos/as!

Empecemos el curso



OBJETIVOS	1	2	PROGRAMACIÓN DIDÁCTICA
MÓDULO 1. FUNDAMENTOS DE APIs	3	4	MÓDULO 2. FRAMEWORK DE DESARROLLO DE APIs EN PYTHON
EVALUACIÓN CALIDAD	5	6	CONTACTO

Si tienes cualquier consulta o necesitas contactar con el/la docente, escríbenos a tutoria@idexaformacion.com

» Objetivos



La presente formación persigue que el alumnado, al finalizar la formación, sea capaz de:

- » Dominar el desarrollo avanzado de APIs, incluyendo seguridad, autenticación y manejo de solicitudes complejas.
- » Aplicar técnicas avanzadas de prácticas y pruebas para garantizar la calidad y el rendimiento de las APIs desarrolladas.
- » Mejorar su rendimiento laboral por la aplicación de los conocimientos y técnicas adquiridas en este curso.

» Programación didáctica



MÓDULO 1. FUNDAMENTOS DE APIs

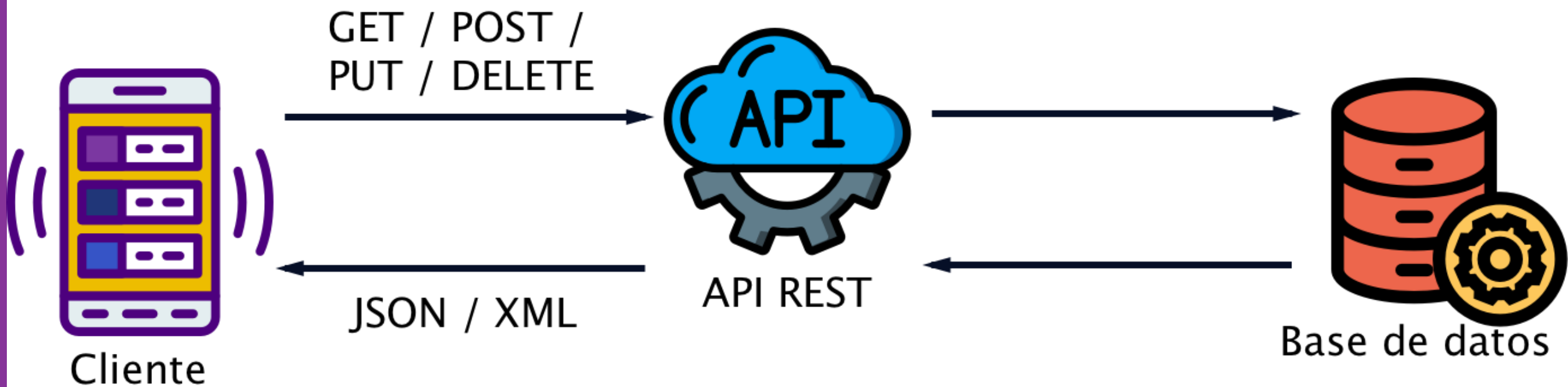
- » Entender qué es una API y cómo funciona.
- » Conocimiento básico sobre RESTful APIs.
- » Comprender los verbos HTTP (GET, POST, PUT, DELETE).

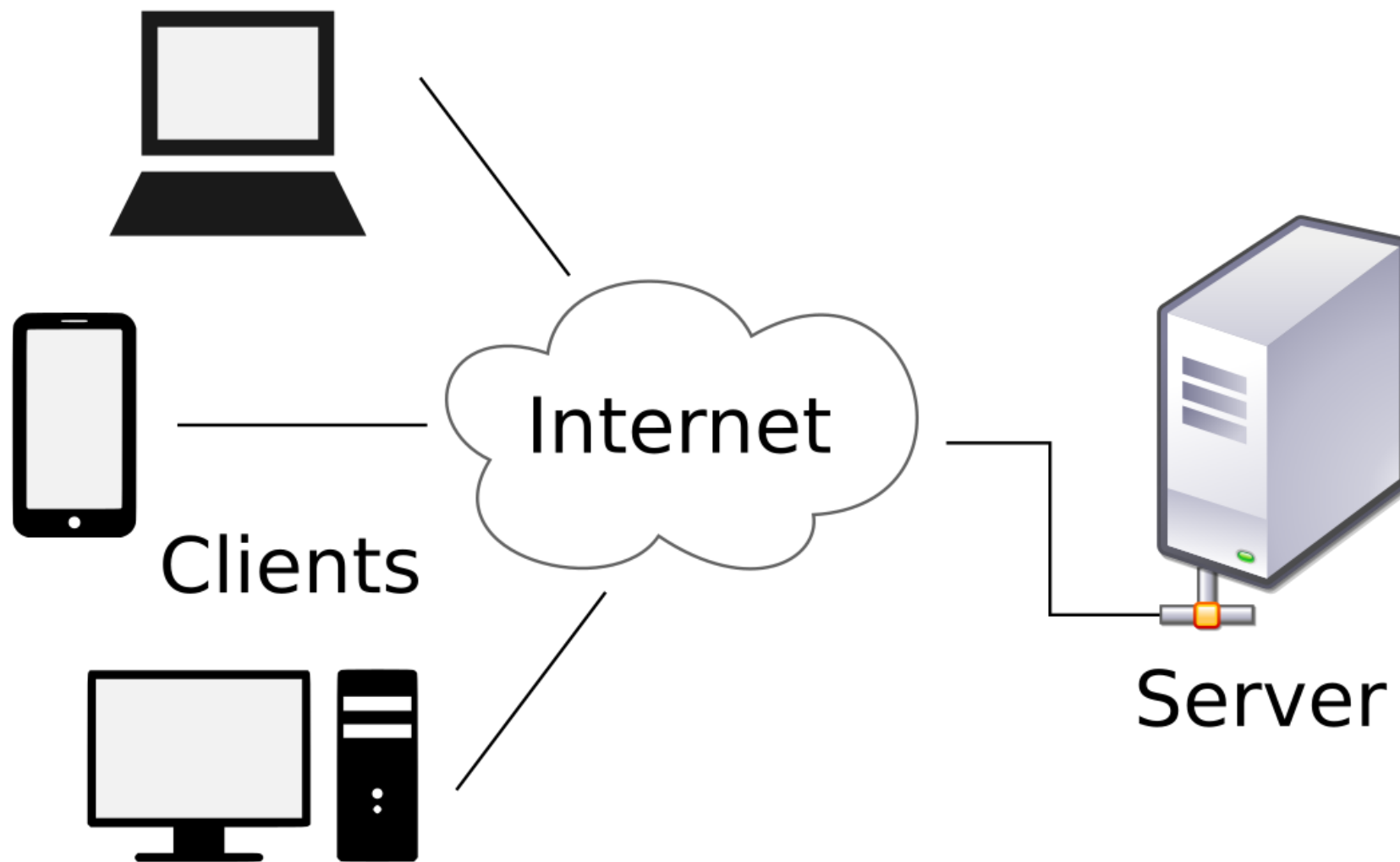
MÓDULO 2. FRAMEWORK DE DESARROLLO DE APIs EN PYTHON

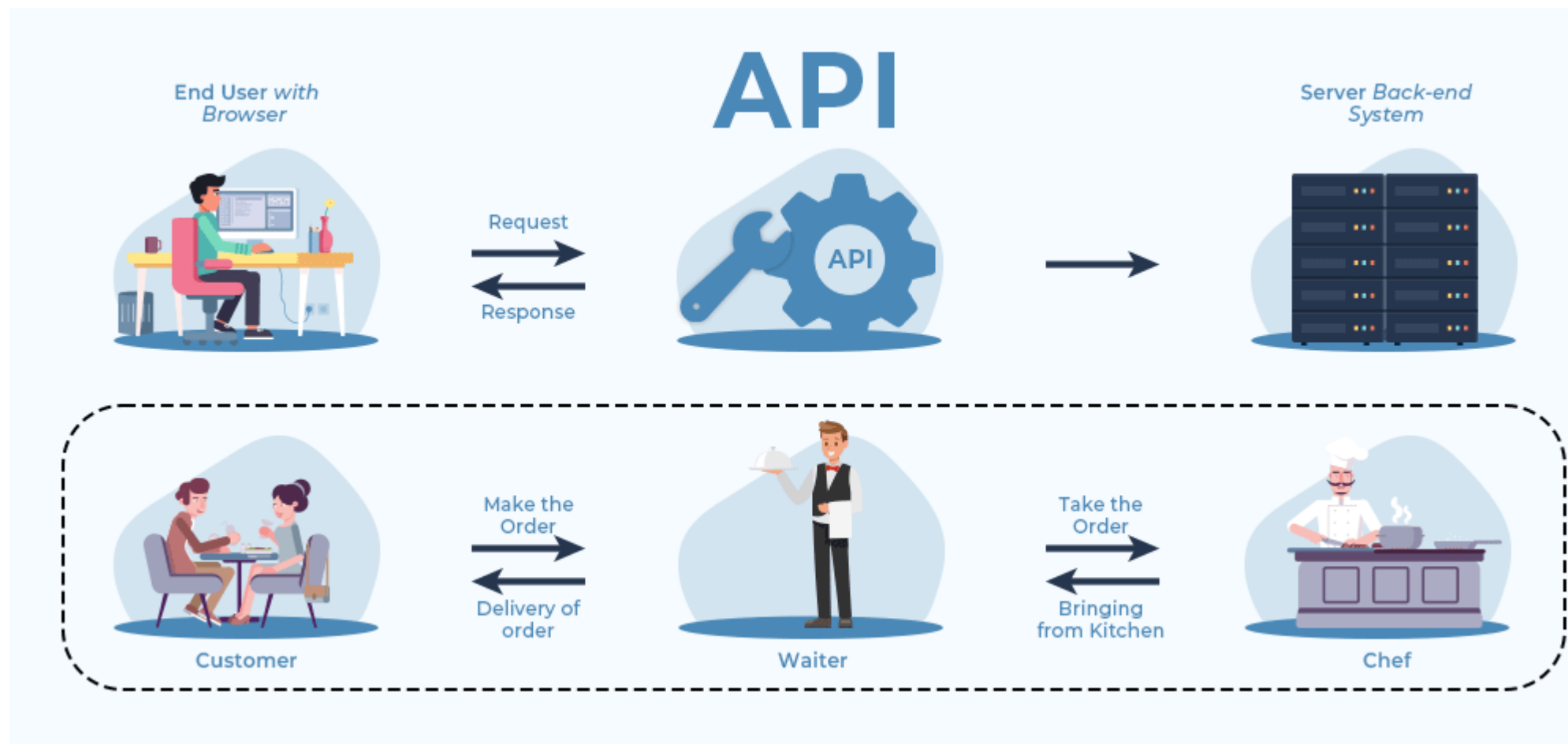
- » Aprender un framework de desarrollo de APIs en Python, como Flask o Django.
- » Familiarizarse con la estructura básica de un proyecto de API en el framework elegido.
- » Crear endpoints en la API para manejar solicitudes de clientes.



MÓDULO 1. FUNDAMENTOS DE APIs



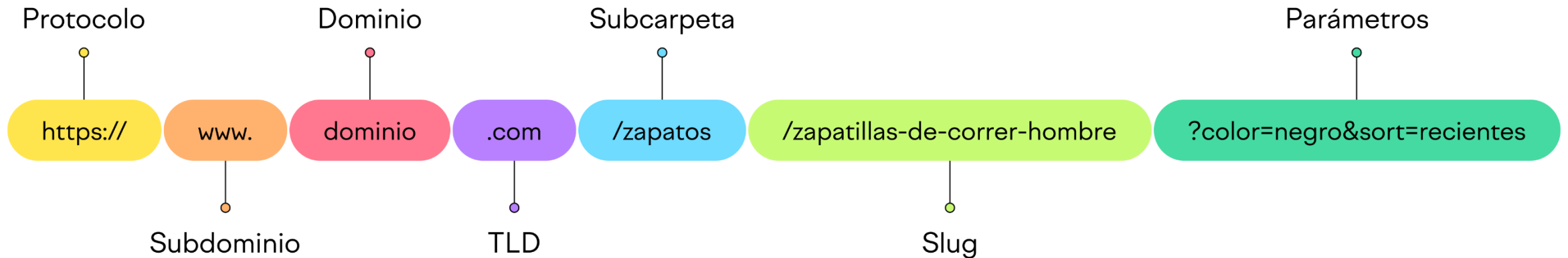


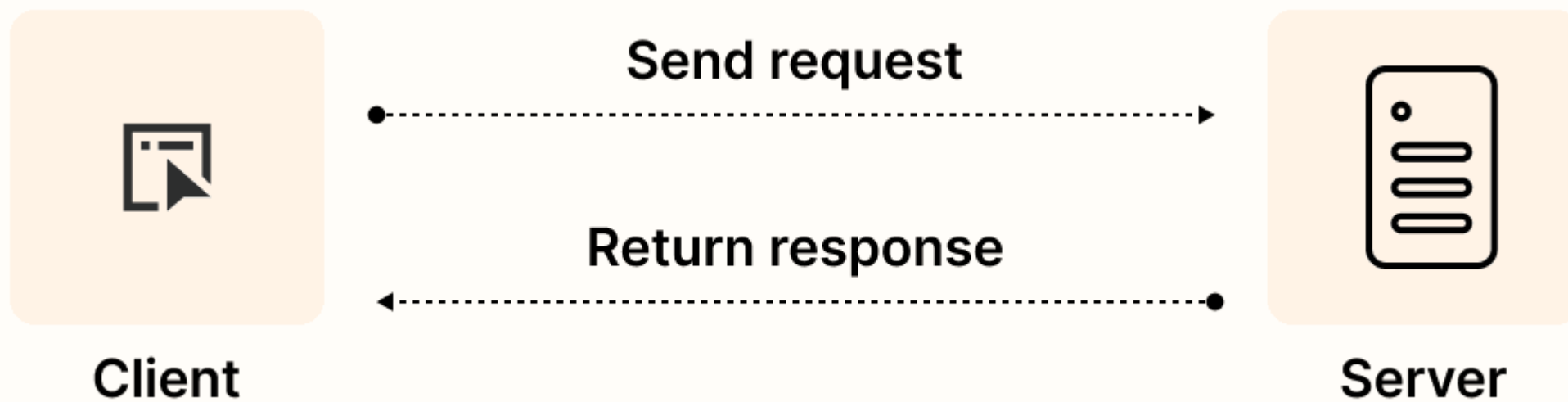






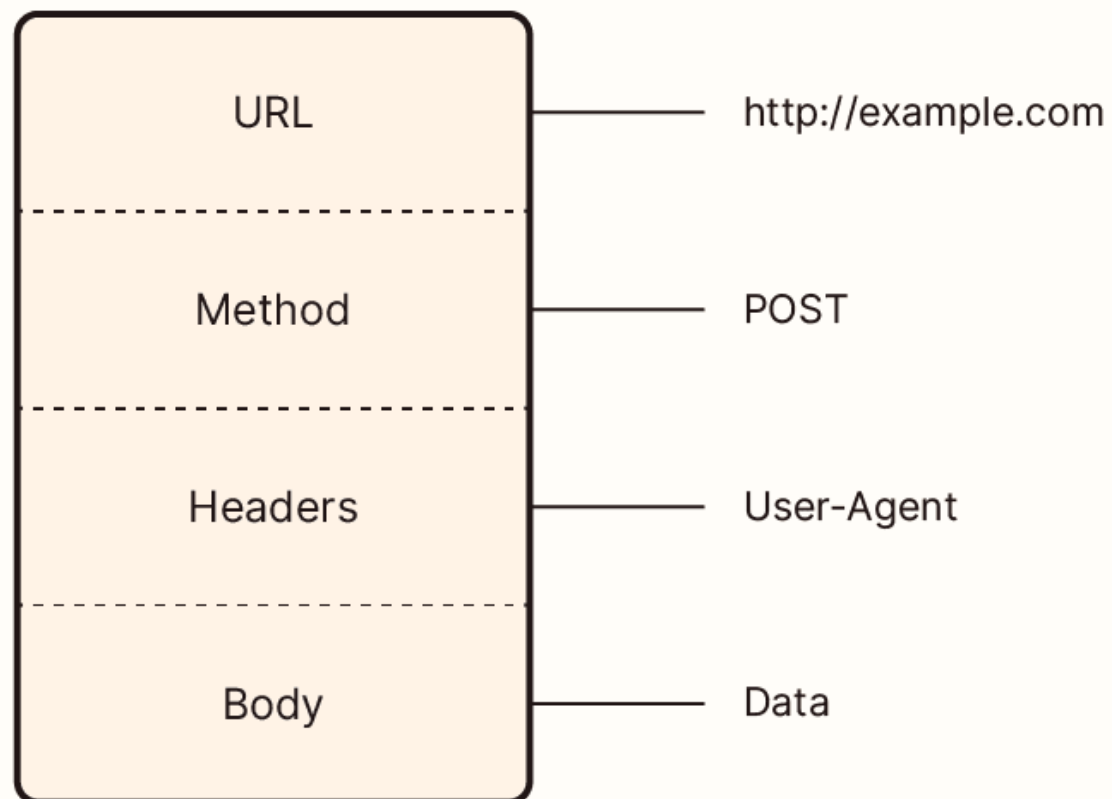
Partes de la estructura de una URL







Valid HTTP request





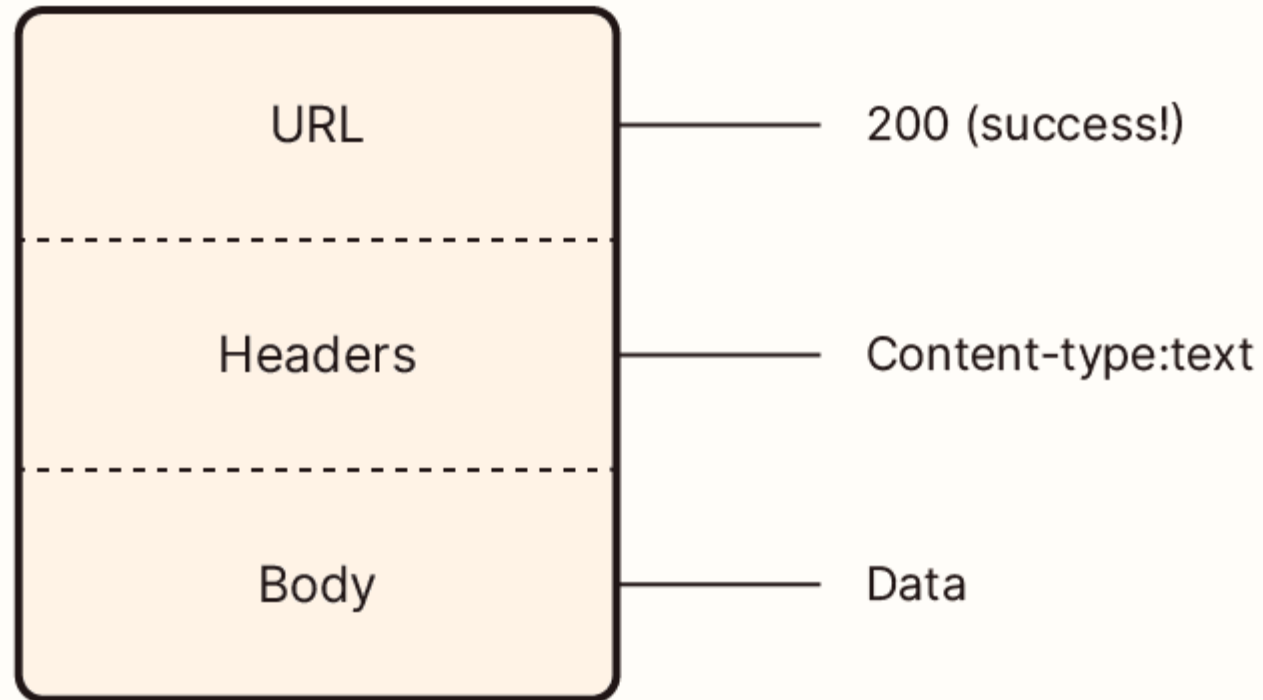
Not found

The requested URL / was not found on this server.

Apache/2.2.9 (Ubuntu) PHP/5.2.6-2ubuntu4 with Suhosin-Patch Server



Valid HTTP response





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`.

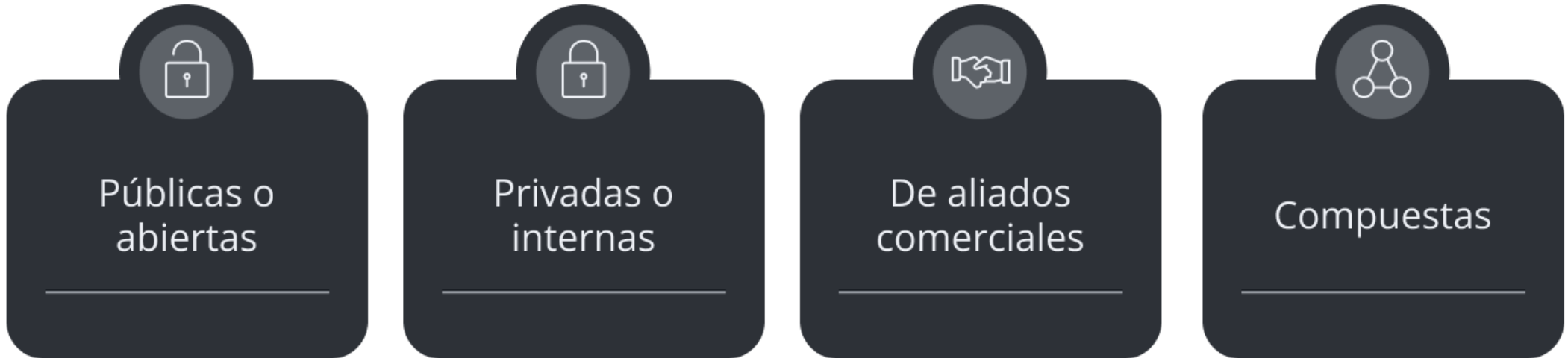
Direct link to results: <https://pokeapi.co/api/v2/pokemon/ditto>

Resource for ditto



Tipos de API

(Application Programming Interface)



S Y D L E

Datos estructurados



Lo que encuentras en una base de datos (usualmente)

Datos No estructurados



Lo que tu encuentras fuera de la base (texto, imagen, audio, video)



```
{  
  "crust": "original",  
  "toppings": ["cheese", "pepperoni", "garlic"],  
  "status": "cooking"  
}
```



```
{  
  "crust": "original",  
  "toppings": ["cheese", "pepperoni", "garlic"],  
  "status": "cooking",  
  "customer": {  
    "name": "Brian",  
    "phone": "573-111-1111"  
  }  
}
```



EJERCICIO

- Imagina que estás creando una aplicación para gestionar recetas de cocina. Quieres organizar la información de las recetas en formato JSON para que sea fácil de manejar y buscar.
- Estructura la información en un archivo JSON. Cada receta debe tener los siguientes campos:
 - Nombre de la receta
 - Ingredientes (una lista de ingredientes con sus cantidades)
 - Pasos de preparación (una lista de pasos detallados para preparar la receta)
 - Tiempo de preparación
 - Categoría (por ejemplo, entrante, plato principal, postre, etc.)



```
<order>
  <crust>original</crust>
  <toppings>
    <topping>cheese</topping>
    <topping>pepperoni</topping>
    <topping>garlic</topping>
  </toppings>
  <status>cooking</status>
</order>
```



EJERCICIO

- En este ejercicio, crearás un archivo XML para almacenar información sobre libros en una biblioteca. Además de los detalles básicos del libro, también incluirás información adicional, como una lista de autores, etiquetas y una descripción detallada.
- Cada libro debe tener los siguientes elementos:
 - <libro>: Contenedor principal para cada libro.
 - <titulo>: Título del libro.
 - <autores>: Lista de autores del libro.
 - <autor>: Nombre de un autor.
 - <genero>: Género literario al que pertenece el libro.
 - <anio_publicacion>: Año de publicación del libro.
 - <num_paginas>: Número de páginas del libro.
 - <etiquetas>: Lista de etiquetas que describen el contenido del libro.
 - <etiqueta>: Nombre de una etiqueta.
 - <descripcion>: Descripción detallada del libro.



```
1 <!DOCTYPE html>
2 <html>
3
4 <head>
5   <title>The Cat said No!</title>
6   <link href="https://fonts.googleapis.com/css?family=Lato:400,700,900" rel="stylesheet">
7   <link rel="stylesheet" href="http://localhost:3000/static/stylesheets/style.css" integrity="sha384-BVYiISiFekKl" rel="stylesheet">
8   <link rel="stylesheet" href="http://localhost:3000/static/stylesheets/style.css" integrity="sha384-BVYiISiFekKl" rel="stylesheet">
9   <link rel="stylesheet" href="http://localhost:3000/static/stylesheets/style.css" integrity="sha384-BVYiISiFekKl" rel="stylesheet">
10 </head>
11
12 <body class="preload">
13   <div class="centered">
14     <button type="button" class="jumbotron" id="togglePaw">
15       <div class="handle"></div>
16     </button>
17     <div class="catpaw-container">
18       
19     </div>
20     <div>
21       <h1 style="text-align:center">The Cat said No!</h1>
22     </div>
23     <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
24     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/js/bootstrap.min.js"></script>
25     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/js/bootstrap.min.js"></script>
26     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/js/bootstrap.min.js"></script>
27     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/js/bootstrap.min.js"></script>
28     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/js/bootstrap.min.js"></script>
29     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/js/bootstrap.min.js"></script>
30     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/js/bootstrap.min.js"></script>
31     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/js/bootstrap.min.js"></script>
32     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/js/bootstrap.min.js"></script>
33     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/js/bootstrap.min.js"></script>
34     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/js/bootstrap.min.js"></script>
35     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/js/bootstrap.min.js"></script>
36     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/js/bootstrap.min.js"></script>
37     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/js/bootstrap.min.js"></script>
38   </div>
39 </body>
```

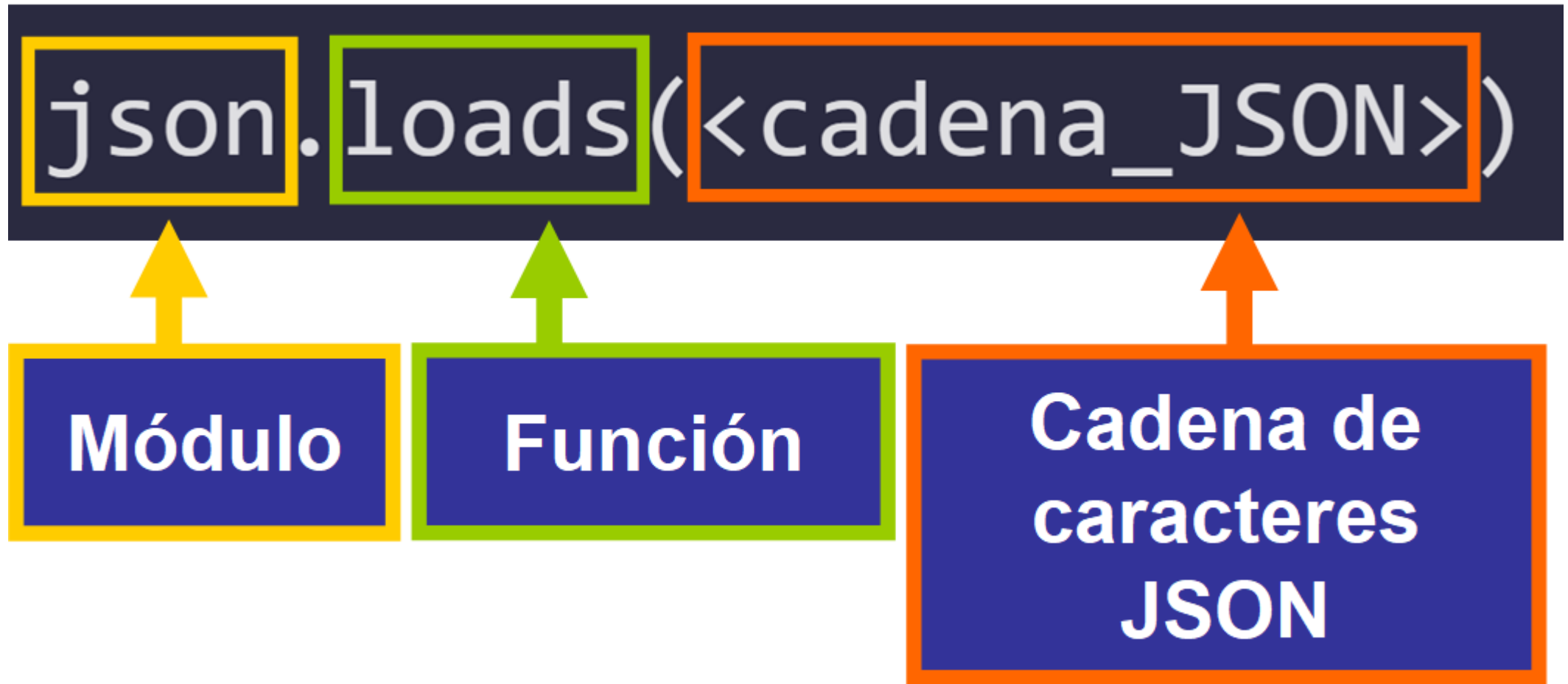
```
1 body {
2   font-family: 'Montserrat', 'Lato', 'Oswald', 'Helvetica Neue', 'Helvetica', 'Arial', sans-serif;
3   color: #6b7381;
4   background: #fff;
5   -webkit-touch-callout: none;
6   -webkit-user-select: none;
7   -khtml-user-select: none;
8   -moz-user-select: none;
9   -ms-user-select: none;
10  user-select: none;
11  transition: background-color 0.25s;
12 }
13
14 .jumbotron {
15   background: #6b7381;
16   color: #bd1c18;
17 }
18 .jumbotron h1 {
19   color: #fff;
20 }
21 .example {
22   margin: 4rem auto;
23 }
24 .example > .row {
25   margin-top: 2rem;
26   height: 5rem;
27   vertical-align: middle;
28   text-align: center;
29   border: 1px solid #d9d9d9;
30 }
31 .example > .row:first-of-type {
32   border: none;
33   height: auto;
34   text-align: left;
35 }
36 .example h3 {
37   font-weight: 400;
```

```
1 pawToggled = false;
2 var myTimeout;
3
4 function callbackToggle() {
5   return function () {
6     if (pawToggled) {
7       document.getElementById('togglePaw').classList.add('active');
8     }
9   }
10 }
11
12 function togglePaw() {
13   if (!pawToggled) {
14     // Runs when we toggle the button
15     document.getElementById('togglePaw').classList.add('active');
16     myTimeout = setTimeout(callbackToggle, 1000);
17   } else {
18     document.getElementById('togglePaw').classList.remove('active');
19     clearTimeout(myTimeout);
20   }
21   pawToggled = !pawToggled;
22 }
23
24 
```



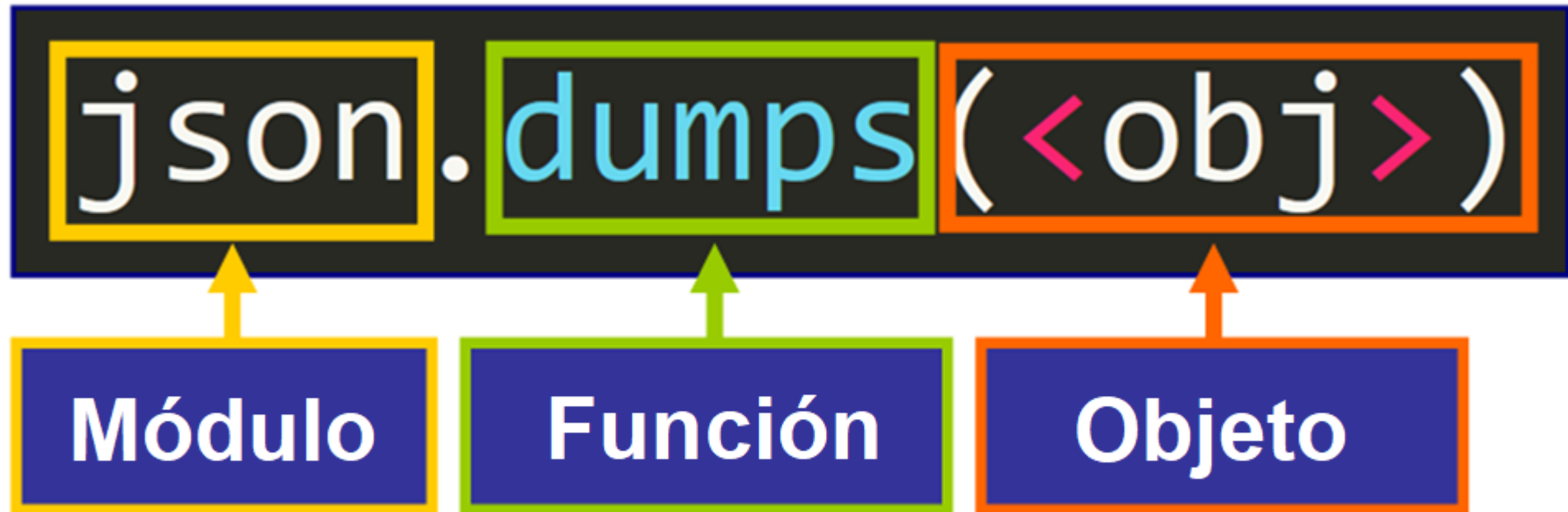
import json







JSON	Python
object	dict
array	list
string	str
número (int)	int
número (real)	float
true	True
false	False
null	None





Python	JSON
dict	object
list, tuple	array
str	string
int, float, Enums derivadas de int o float	number
True	true
False	false
None	null



```
json.dumps(<obj>, indent=<espacios>)
```

**Número de
espacios para
indentación**



```
json.dumps(<obj>, sort_keys=True)
```

Ordenar claves
alfabéticamente (o no).



Abrir ordenes.json en modo de lectura

Objeto archivo

```
with open("ordenes.json") as archivo:  
    datos = json.load(archivo)
```

Leer el archivo JSON y crear un diccionario



Abrir orders.json en modo de escritura

Objeto archivo

```
with open("ordenes.json", "w") as archivo:  
    # Agregar informacion al archivo JSON
```



```
json.dump(<obj>, <archivo>)
```

**Objeto que será
guardado en formato
JSON**

**Archivo donde
será guardado**



	load()	loads()
Propósito	Crear un objeto de Python a partir de un archivo JSON	Crear un objeto de Python a partir de una cadena de caracteres
Argumento	Archivo JSON	Cadena de caracteres
Valor retornado	Objeto de Python	Objeto de Python



	dump()	dumps()
Propósito	Escribir un objeto en formato JSON a un archivo	Obtener una cadena de caracteres JSON a partir de un objeto
Argumento	Objeto + Archivo	Objeto
Valor retornado	None	Cadena de caracteres



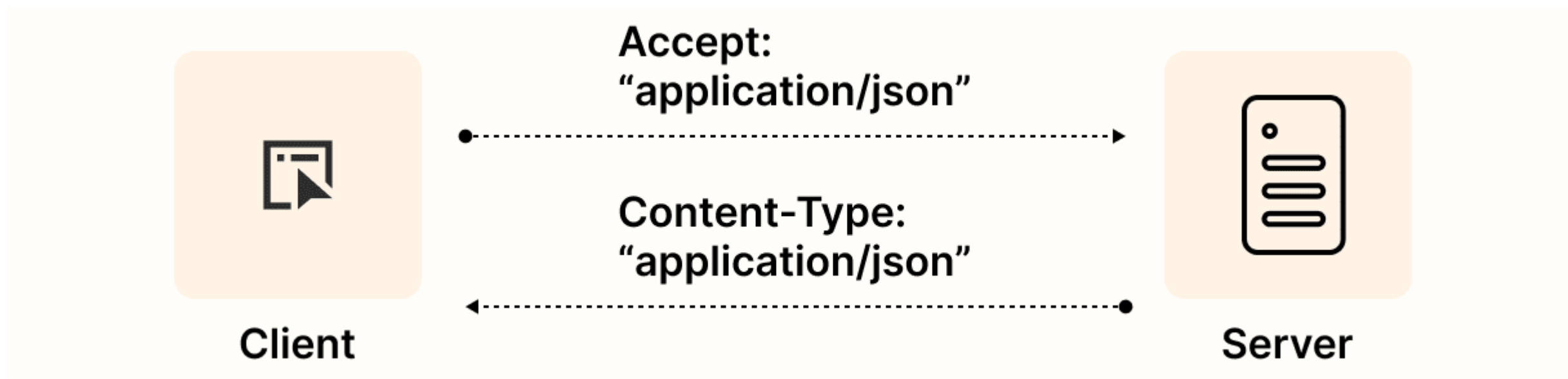
EJERCICIO

- Escribe distintas funciones en python que lean el fichero json libreria.json con datos de nuestra librería y muestre la siguiente información:
- ¿Cuántos libros hay en la librería?
- Recibe un límite inferior y superior para el precio y muestra todos los libros cuyo precio esta en ese intervalo.
- Recibe una cadena por teclado, y muestra el título y el año de publicación de los libros cuyo título empieza por la cadena introducida.
- Devuelve todos los títulos de los libros con la lista de sus autores.




EJERCICIO

- A partir del fichero JSON movies.json obtener la siguiente información:
- Listar información: Listar el título, año y duración de todas las películas.
- Contar información: Mostrar los títulos de las películas y el número de actores/actrices que tiene cada una.
- Buscar o filtrar información: Mostrar las películas que contengan en la sinopsis dos palabras dadas.
- Buscar información relacionada: Mostrar las películas en las que ha trabajado un actor dado.
- Ejercicio libre: Mostrar el título y la url del póster de las tres películas con una media de puntuaciones más alta y lanzadas entre dos fechas dadas.





Basic authentication
vs. API key authentication



Basic authentication

✓

Encodes username and password

✓

Widely compatible across browsers

✓

Easy to implement with low processing overhead

✗

Doesn't support nuanced role permissions

✗

Vulnerable to attacks, even over HTTPS

✗

Transmits usernames and passwords with each request

API key authentication

✓

Requires a unique key for API access

✓

Allows for granular permissions

✓

Allows users access without having to share passwords

✗

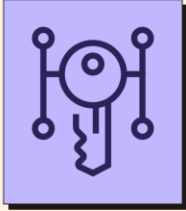
Still susceptible to unauthorized use

✗

Difficult to change

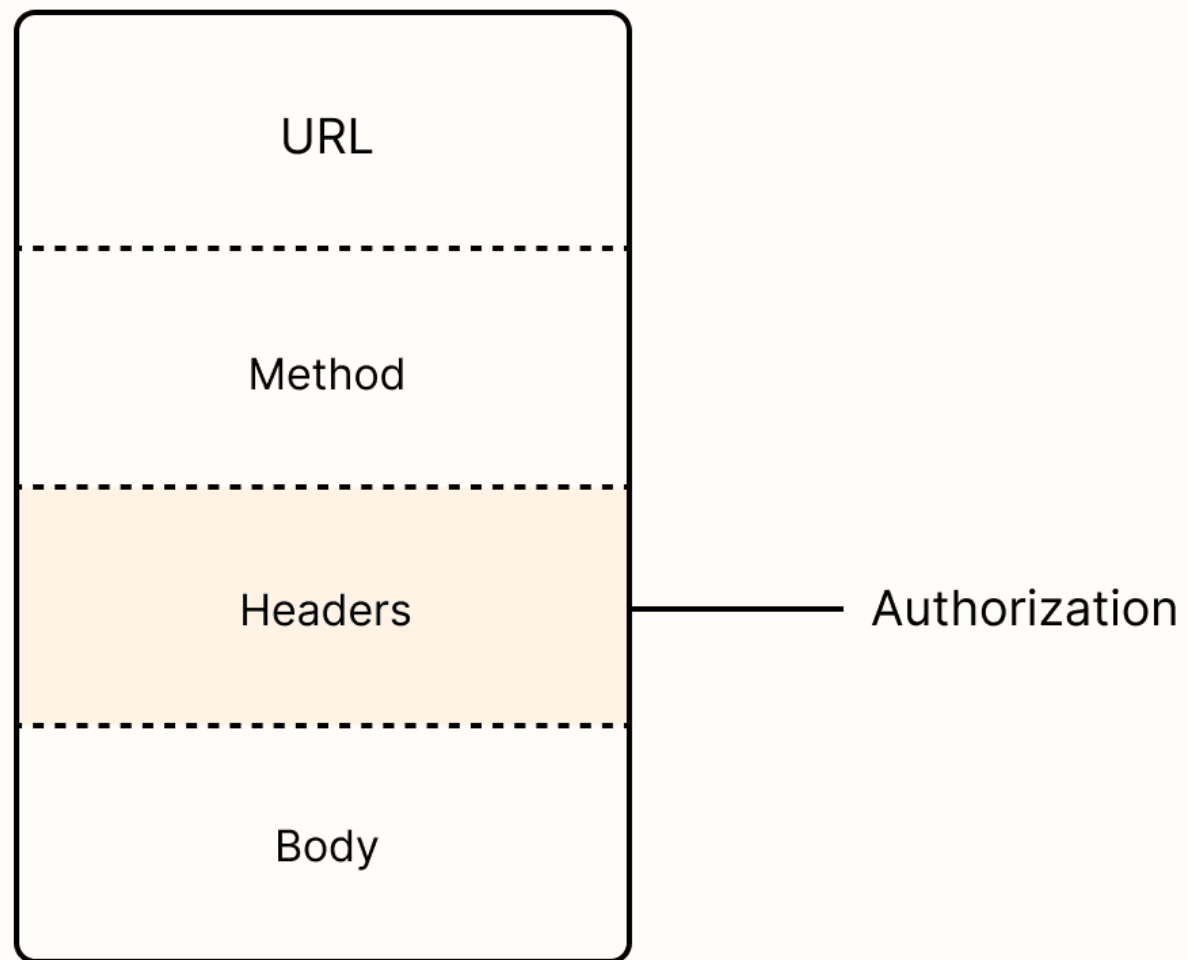
✗

Associated with specific applications, not individual users



MATERIAL CONSULTA | FUNDAMENTOS APIs

DOCENTE | JORGE LÓPEZ



Request



SMTP & API

[Generate a new API key](#)

SMTP API Keys

Your API Keys

Version	API Key	Name	Created on
<input type="checkbox"/> v3	*****Xo4rJT	prestashop 1.7	April 4, 2023 12:48 PM
<input type="checkbox"/> v3	*****QnbN3k	WordPress	



AUTHENTICATION VS. AUTHORIZATION

First Step:
AUTHENTICATION

“WHO ARE YOU?”



Second Step:
AUTHORIZATION

“WHAT
PERMISSIONS DO
YOU HAVE?”





Client application



Resource owner



Authorization server



Resource server



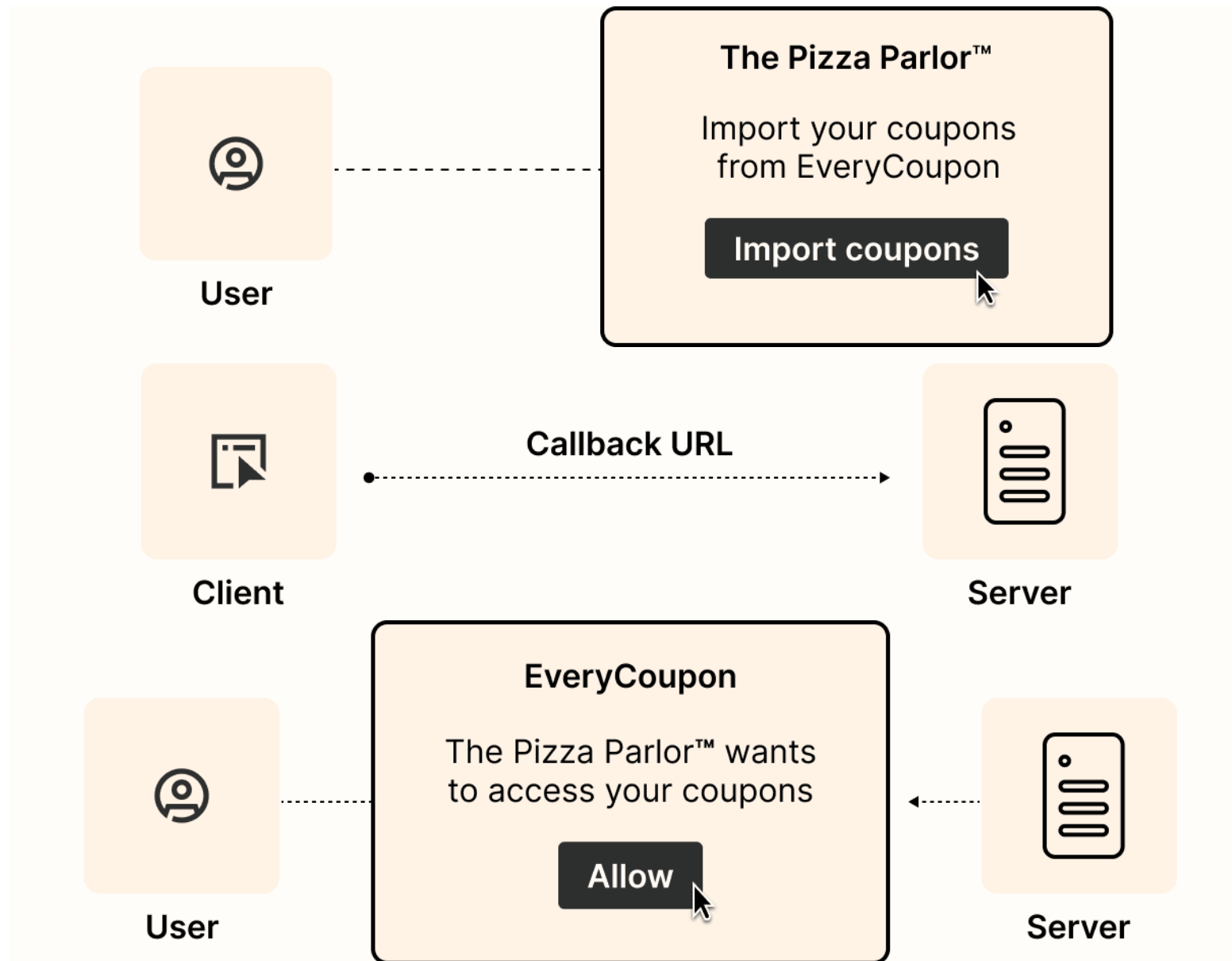
Grant

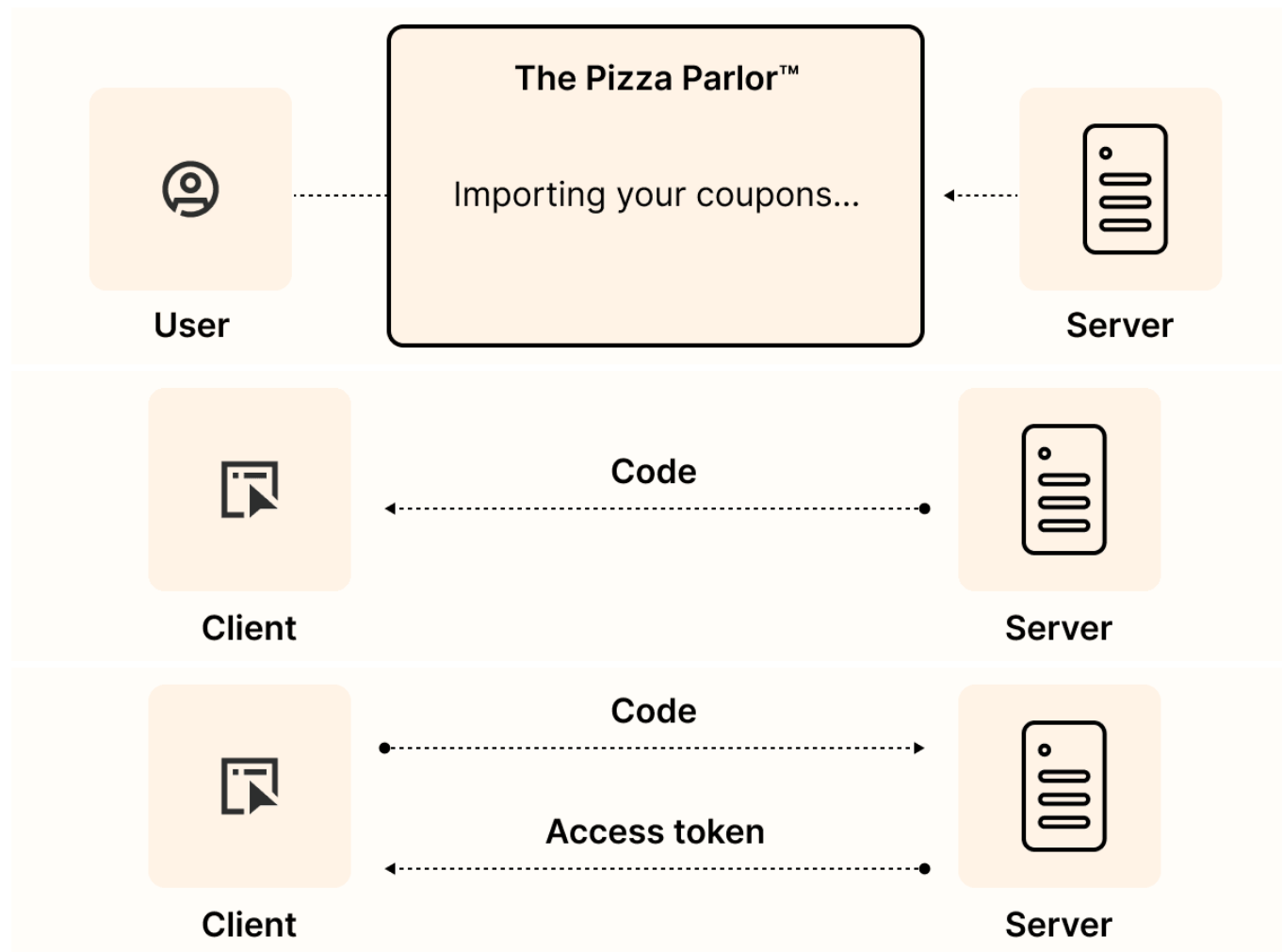
Grant

Access token

Access token

Protected resource

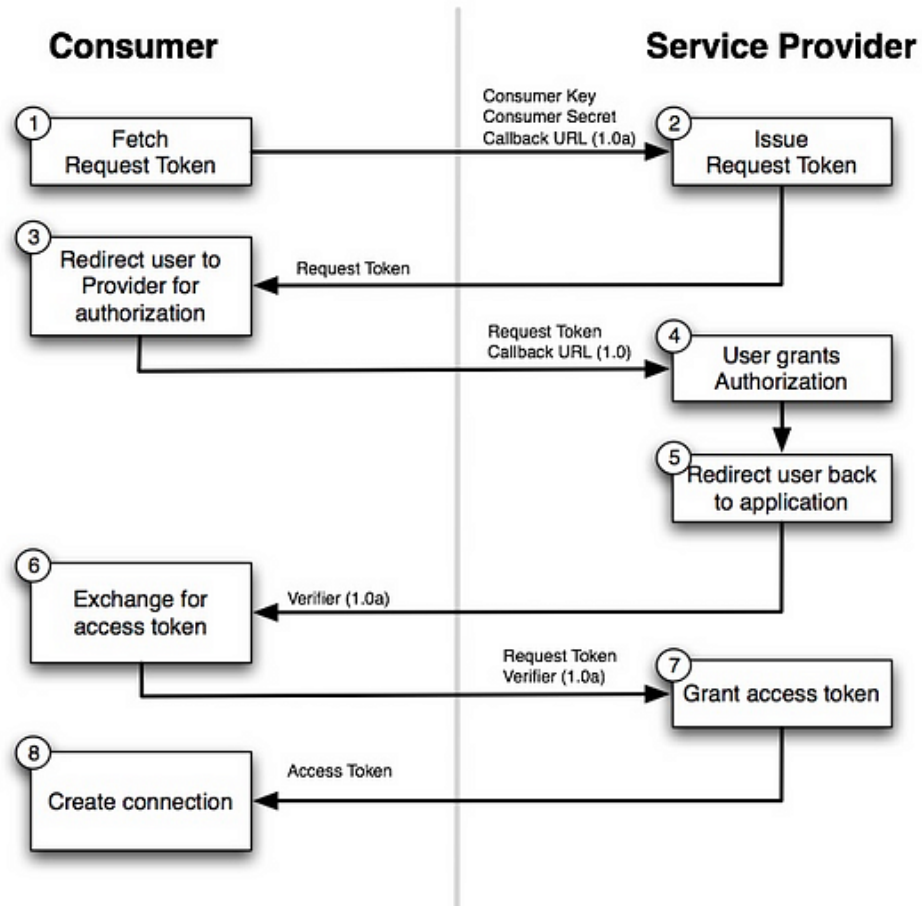




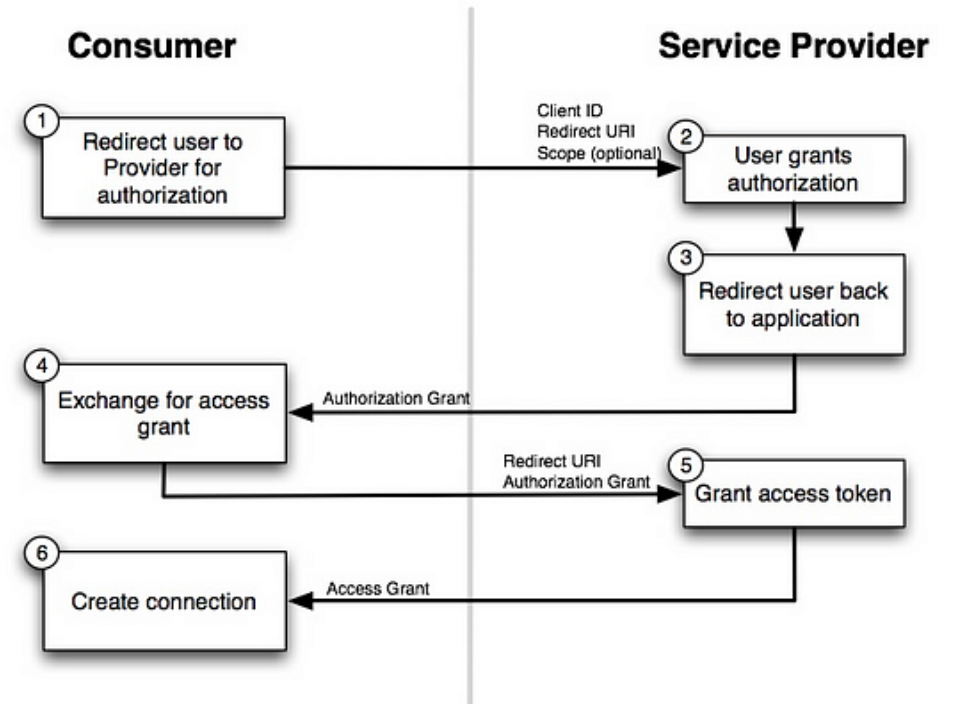




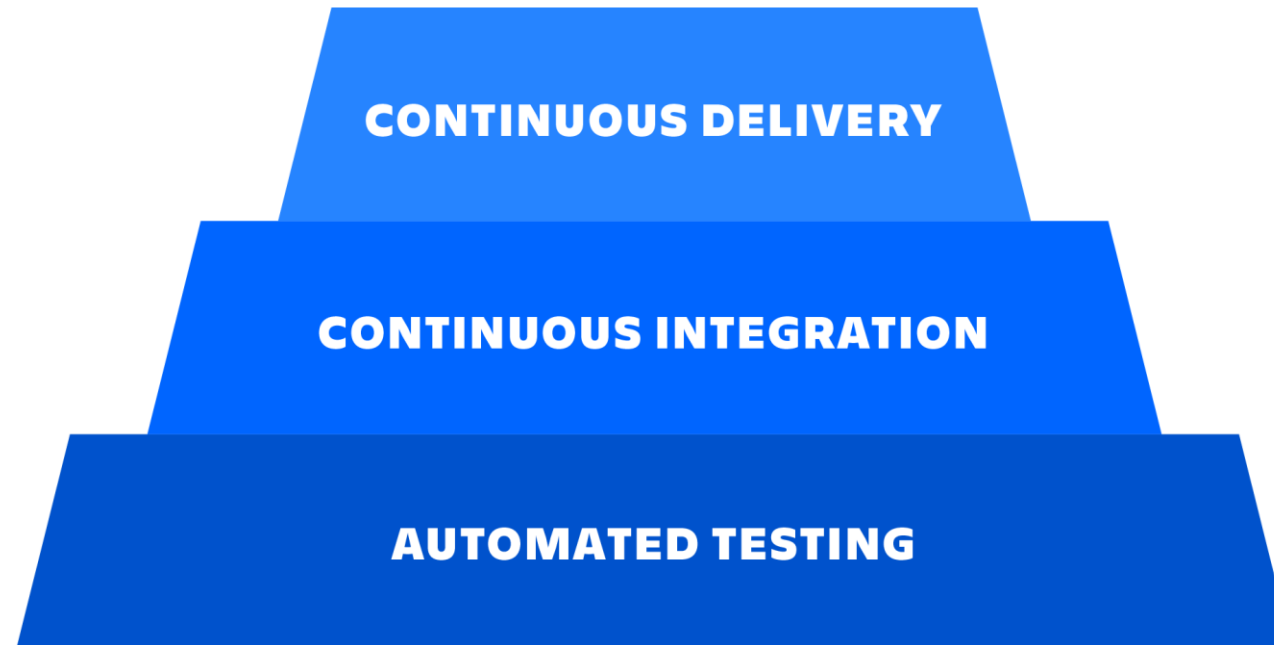
OAuth 1.0

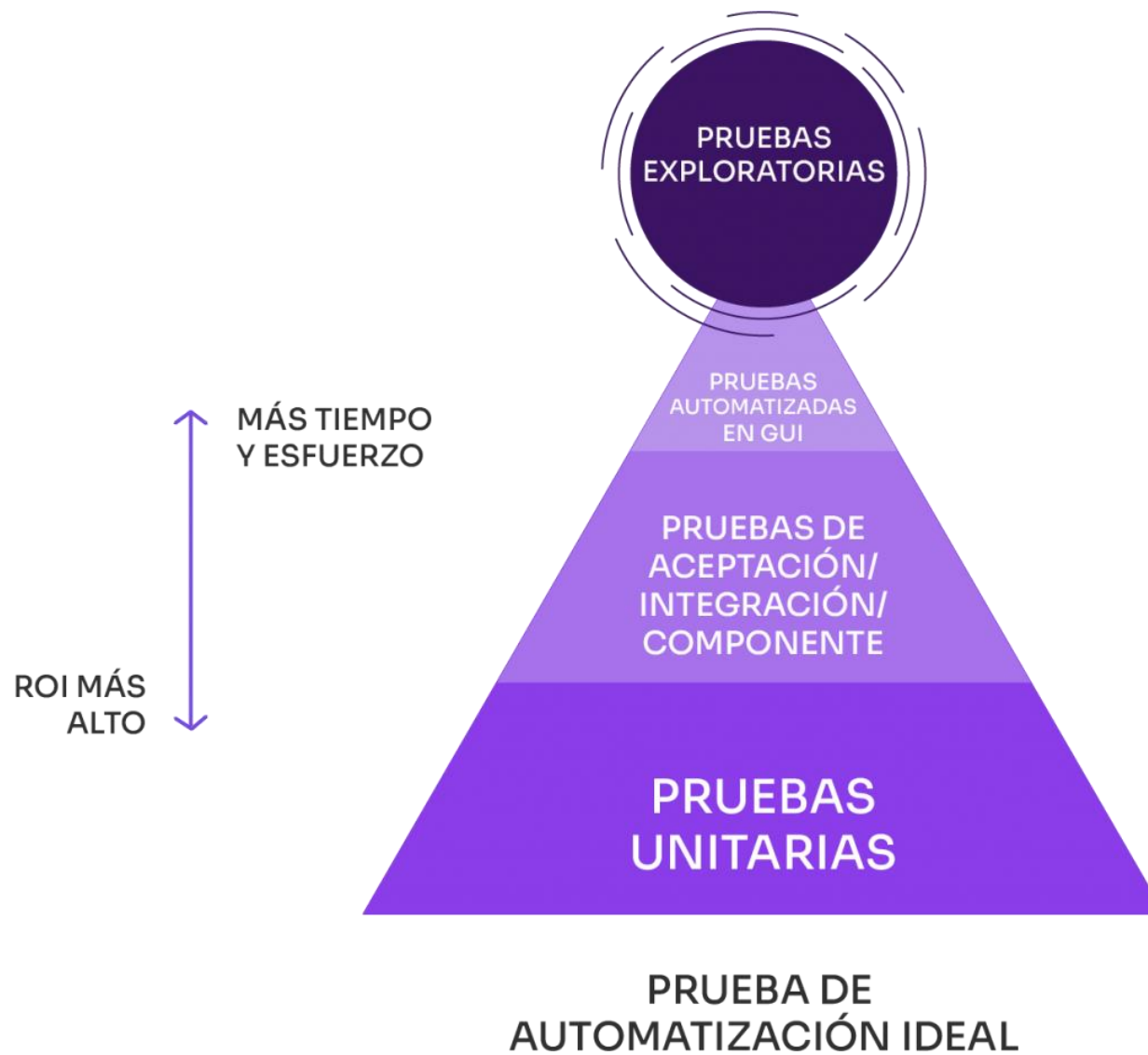


OAuth 2.0











Tipos de Pruebas API



Funcionales



Seguridad



Rendimiento



Integración



Documentación



HomeWorkspacesAPI NetworkReportsExplore

Search Postman

InviteSettingsNotificationsHelpUpgrade

My WorkspaceNewImport

Collections

APIs

Environments

Mock Servers

Monitors

Flows

History

Media Authentication API

POST Authentication

POST Access Token

GET User's Page

GET Instagram Account

GET New Request

Sports team fixturesSports Arena

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

GETUntitled Request

No Environment


Save

Send

ParamsAuthorizationHeaders (5)BodyPre-request ScriptTestsSettingsCookies

KEY	VALUE	DESCRIPTION	Bulk Edit
Key	Value	Description	

Response






Enter the URL and click Send to get a response

Console

BootcampAuto-select agentRunnerTrashHelp



History		Collections
All	Me	Team
		 
		Regression_test ***
POST		1_Incorrect_login_admin
POST		2_Correct_login_admin
POST		3_Create_employee2
GET		4_Get_employee2
PUT		5_Modify_employee2
GET		6_Get_employees
DEL		7_Delete_employee2
GET		8_Get_employee2
GET		9_Get_employees
POST		10_Create_service1
PUT		11_Modify_service1
GET		12_Get_service1
GET		13_Get_services
POST		14_Create_client1
PUT		15_Modify_client1
GET		16_Get_client1
POST		17_Create_appointment1

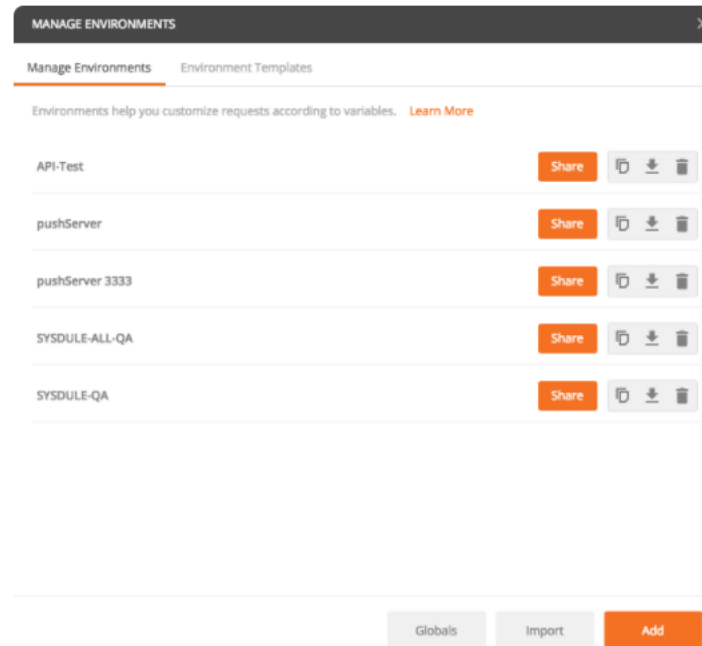


Figura 3. Gestión de entornos.





Postman

File Edit View Help

New Import Runner

5 Invite 6

7 History 8 Collections

9 GET Untitled Request + ...

10 GET 11 Enter request URL 12 Send Save

13 Params 14 Authorization 15 Headers 16 Body 17 Pre-request Script 18 Tests

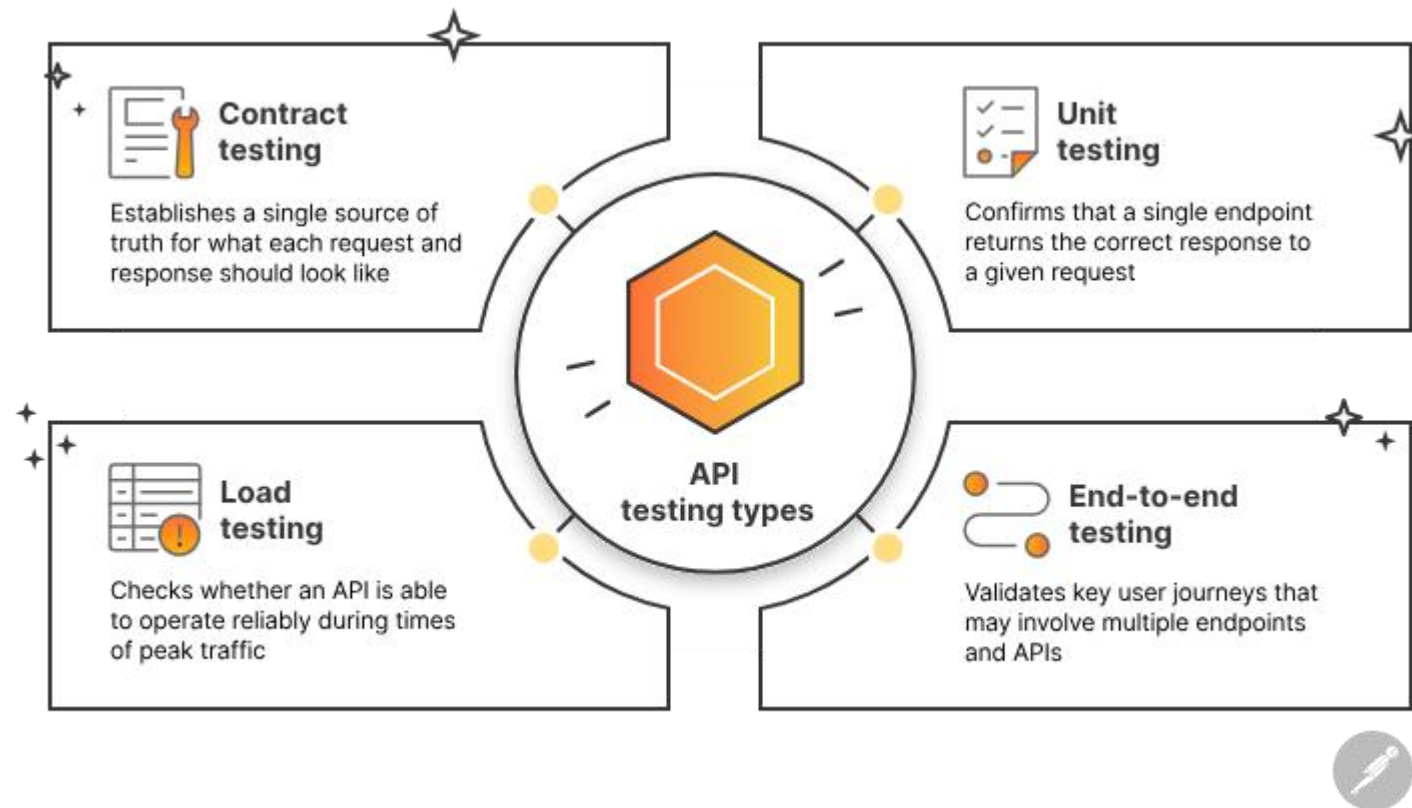
KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Response

Hit the Send button to get a response.

Create a collection







HTTP verb	Endpoint	Action
GET	/orders	List existing orders
POST	/orders	Place a new order
GET	/orders/1	Get details for order #1
GET	/orders/2	Get details for order #2
PUT	/orders/1	Update order #1
DELETE	/orders/1	Cancel order #1



URI
<code>/customers</code>
<code>/customers/16/phone-number</code>
<code>/customers/16/address/home</code>
<code>/users/{userId}</code>
<code>/Customers</code>
<code>/generalMembers</code>
<code>/MenuItems</code>
<code>/GeneralMembers</code>
<code>/customers/16/tel-no</code>
<code>/customers/16/phone_number</code>
<code>/customers/16/phonenummer</code>
<code>/users/{user-id}</code>



Una tienda puede tener clientes que hayan realizado muchos pedidos y cada uno de estos pedidos puede tener direcciones de entrega, elementos de menú y facturas.

URI	Estado
/store/customers/{customerId}/orders	Bueno
/store/orders/{orderId}	Bueno
/store/orders/{orderId}/menu-items	Bueno

Del mismo modo, una biblioteca puede tener libros de muchos autores. Cada uno de estos libros tiene un número ISBN.

URI	Estado
/library/authors/books	Bueno
/library/book/{bookId}/isbn	Bueno



URI
<code>/orders</code>
<code>/users/{userId}</code>
<code>/order</code>
<code>/getOrder</code>
<code>/getUser/{userId}</code>



Siempre debe evitar los caracteres especiales en los puntos finales de su API. Pueden resultar confusos y técnicamente complejos para sus usuarios. Considere los siguientes malos ejemplos.

URI	Estado	Por qué
/users/12 23 23/address	Malo	Carácter especial
/orders/16/menu^items	Malo	Carácter especial ^

Si su API puede aceptar varios identificadores de usuario, deben separarse mediante una coma, como se muestra a continuación.

URI	Estado	Por qué
/users/12,23,23/address	Bueno	Utiliza una coma para la separación



Siempre debe evitar las extensiones de archivo en los nombres de sus API. Por ejemplo, si su API puede entregar una salida tanto en formato JSON como XML, nunca debería tener este aspecto.

URI	Estado	Por qué
<code>/sports/basketball/teams/{teamId}.json</code>	Malo	Extensión de archivo al final
<code>/sports/basketball/teams/{teamId}.xml</code>	Malo	Extensión de archivo al final

En su lugar, su cliente debería poder indicar su formato esperado en una cadena de consulta, como ésta

URI	Estado	Por qué
<code>/sports/basketball/teams/{teamId}?format=json</code>	Bien	Sin extensión de archivo
<code>/sports/basketball/teams/{teamId}?format=xml</code>	Bueno	Sin extensión de archivo



URI
<code>/users/{userId}/locations</code>
<code>/users/{userId}/locations? country=USA</code>
<code>/articles?per-page=10&page=2</code>
<code>/users/{userId}/locations/USA</code>
<code>/articles/page/2/items-per-page/10</code>



Cuando comparta su punto final de API con otros miembros de su equipo, o en público, evite utilizar una barra al final de sus puntos finales de API. Considere los siguientes ejemplos.

URI	Estado	Por qué
/users/{userId}	Bueno	Sin barra al final
/articles/{articleId}/author	Bueno	Sin barra al final
/users/{userId}/	Malo	Barra al final
/articles/{articleId}/author/	Malo	Barra oblicua final





EJERCICIO

- Utilizando la biblioteca requests de Python, crea un programa que interactúe con la API JSONPlaceholder para realizar las siguientes operaciones HTTP:
- GET: Obtener una lista de todos los posts.
- POST: Crear un nuevo post con un título, cuerpo y ID de usuario proporcionados por el usuario.
- PUT: Actualizar un post existente con un nuevo título, cuerpo y ID de usuario proporcionados por el usuario.
- PATCH: Actualizar parcialmente un post existente cambiando solo su título.
- DELETE: Borrar un post existente.
- Asegúrate de manejar las respuestas de la API de manera adecuada y de imprimir los resultados de cada operación para verificar su éxito o cualquier error que pueda surgir.





EJERCICIO

- Objetivo: Crear un programa en Python que interactúe con una API pública para obtener información sobre el clima de una ciudad específica.
- Descripción: Utilizaremos la biblioteca requests de Python para enviar solicitudes HTTP a una API pública que proporciona datos meteorológicos. La API que utilizaremos es OpenWeatherMap, que ofrece datos meteorológicos actuales y pronósticos para ciudades de todo el mundo.
- Obtención de una clave de API:
 - Regístrate en el sitio web de OpenWeatherMap para obtener una clave de API gratuita. Esta clave de API es necesaria para acceder a los datos de la API del clima.
 - Una vez que hayas obtenido tu clave de API, guárdala en un lugar seguro. La necesitarás para hacer solicitudes a la API del clima.
- Implementación del programa:
 - Crea un script en Python que solicite al usuario el nombre de una ciudad.
 - Utiliza la biblioteca requests para enviar una solicitud GET a la API del clima de OpenWeatherMap, proporcionando el nombre de la ciudad y tu clave de API en la URL.
 - Procesa la respuesta de la API y extrae la información relevante, como la temperatura actual y la descripción del clima.
 - Imprime la información del clima en la consola, incluyendo la descripción del clima y la temperatura en grados Celsius.



django

 Flask

VS

 FastAPI







EJERCICIO

- Definir la ruta para eliminar empleados:
 - Crear una ruta en la API de Flask que escuche las solicitudes DELETE en la URL `/employees/<int:id>`, donde `<int:id>` representa el ID del empleado a eliminar.
 - La función asociada debe aceptar el parámetro `id`, que será el ID del empleado a eliminar.
- Implementar la lógica de eliminación:
 - Utilizar una función para eliminar al empleado con el ID especificado de la lista de empleados existente.
 - Si el empleado no existe en la lista, devolver un mensaje de error con un código de estado 404 (Not Found).
 - Si se elimina correctamente, devolver los detalles del empleado eliminado con un código de estado 200 (OK).
- Actualizar la lista de empleados:
 - Utilizar una lista global llamada `employees` que contenga los detalles de los empleados.
 - Actualizar la lista después de eliminar un empleado para reflejar los cambios.



Types of real-time APIs



Push APIs

Rather than continually feeding updates to a client, these send them only when the client requests them



Streaming APIs

These feed data to clients continuously without requiring them to request it

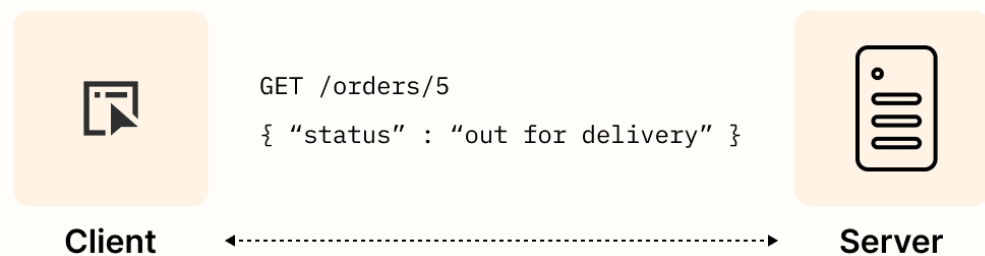
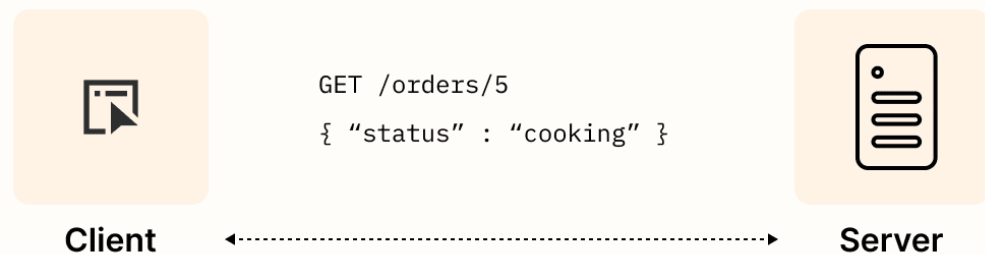
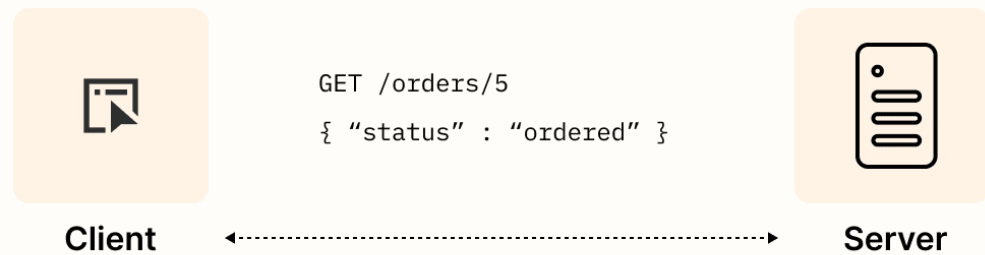


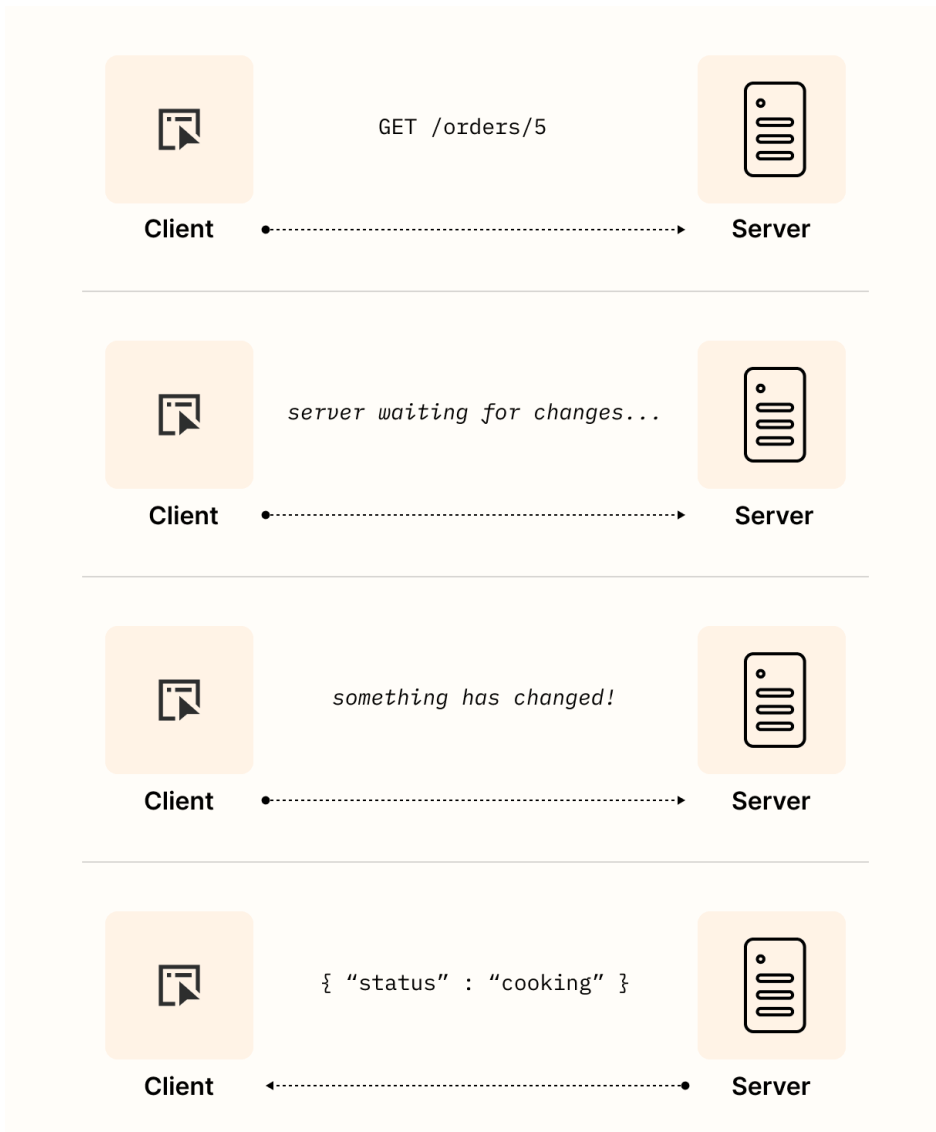
Event-driven APIs

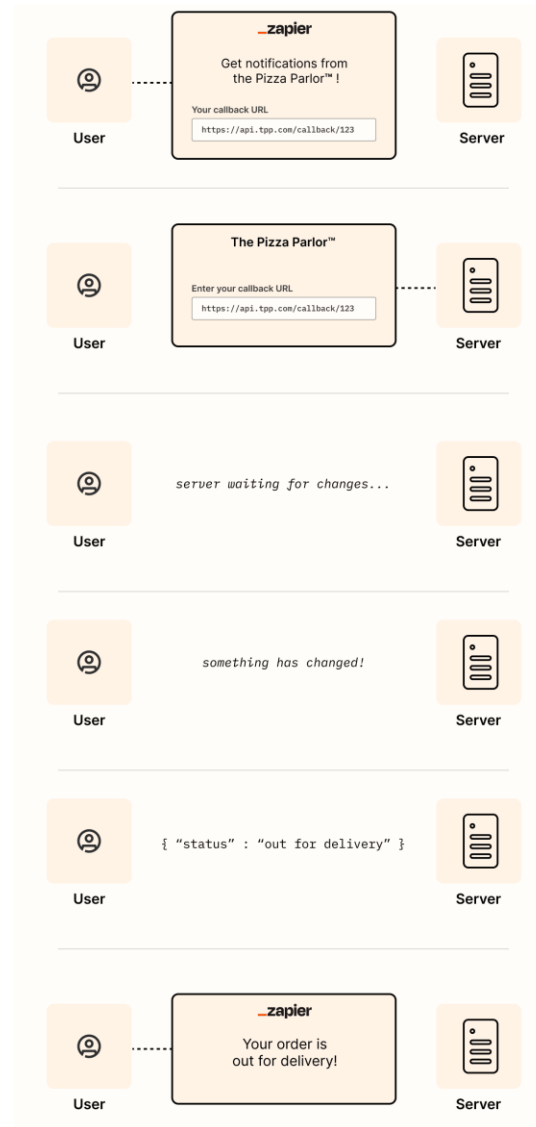
These APIs wait for predetermined events to trigger communication



Server









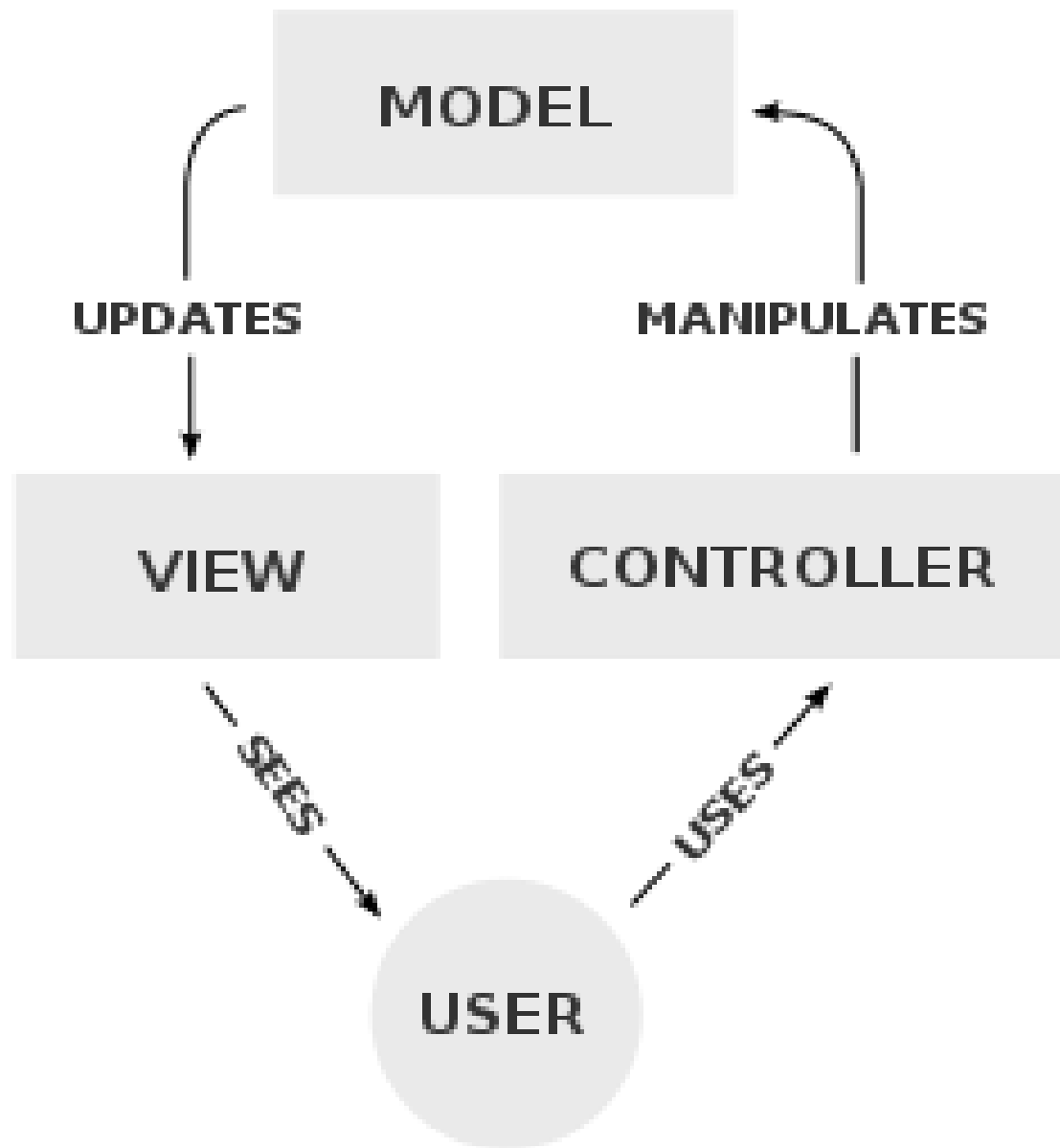


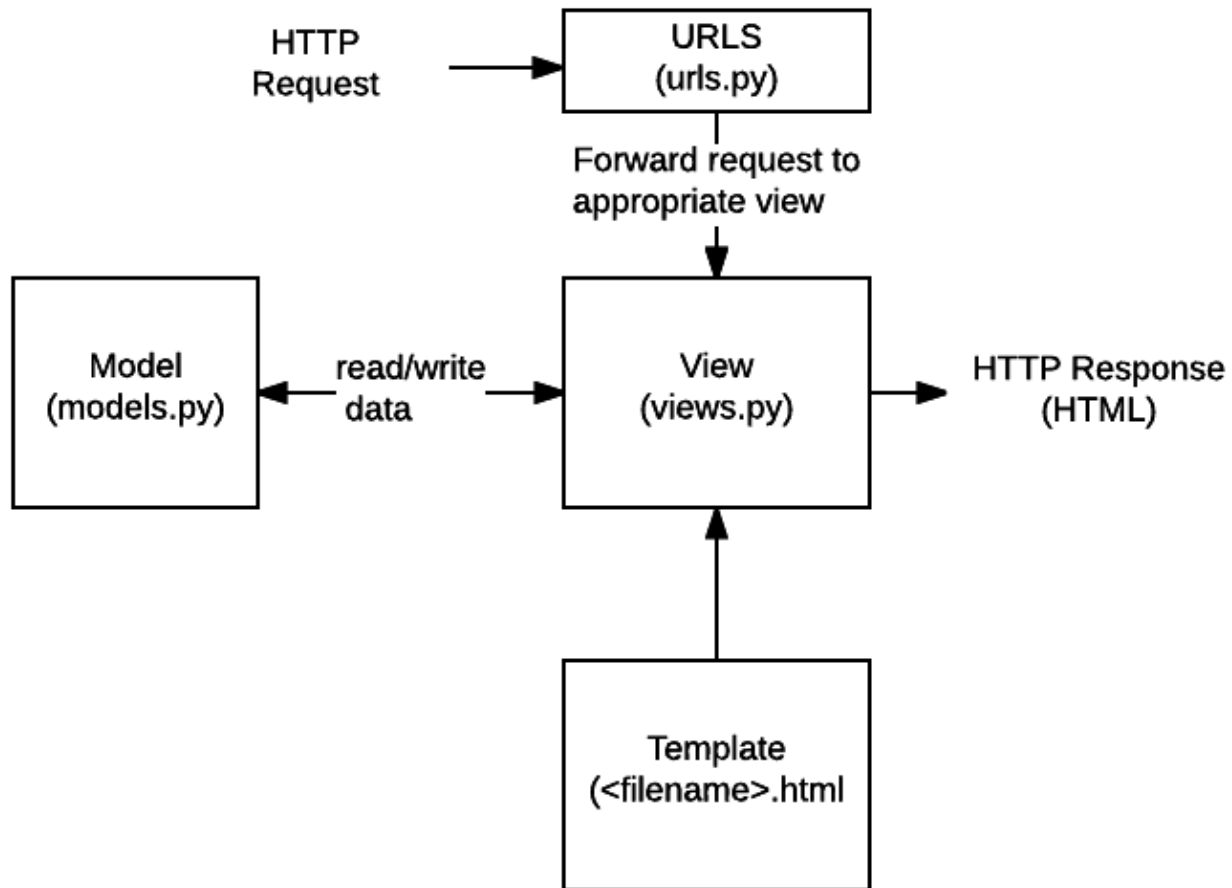


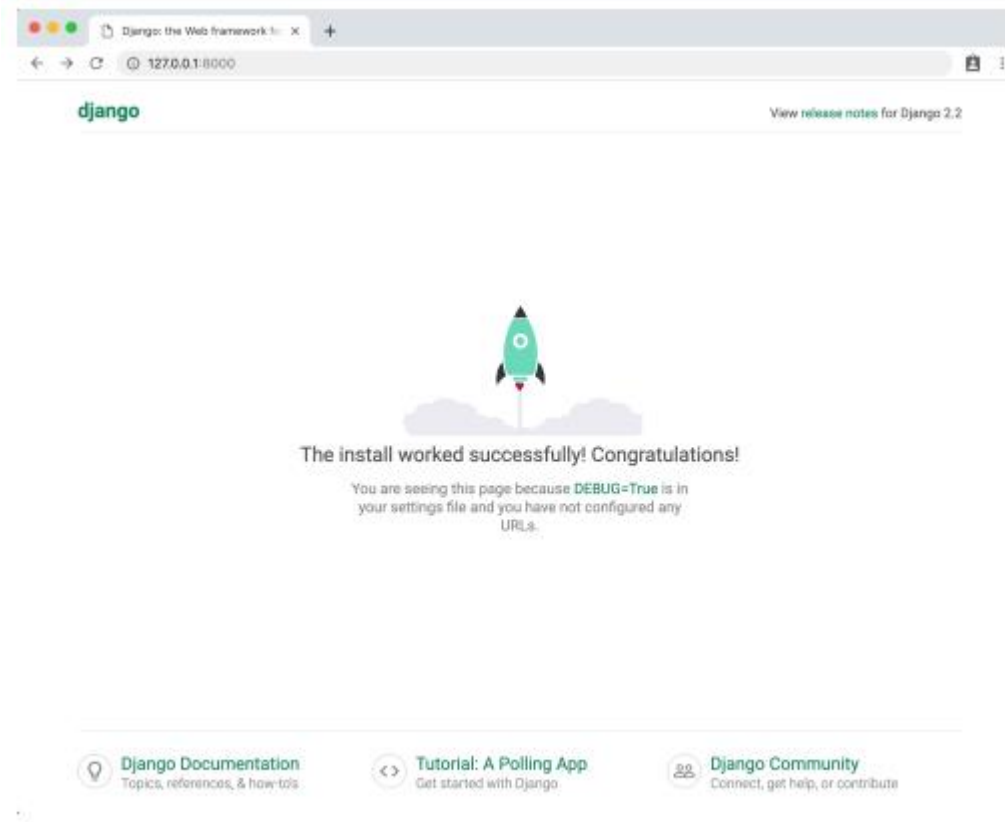
MÓDULO 2. FRAMEWORK DE DESARROLLO DE APIs EN PYTHON



django







Django welcome page



```
from django.db import models

class Book(models.Model):
    title = models.CharField(max_length=250)
    subtitle = models.CharField(max_length=250)
    author = models.CharField(max_length=100)
    isbn = models.CharField(max_length=13)

    def __str__(self):
        return self.title
```




django

REST

framework

» Tips

- » Una API (Interfaz de Programación de Aplicaciones) es un conjunto de reglas y protocolos que permite a diferentes aplicaciones comunicarse entre sí. Funciona proporcionando puntos de acceso definidos (endpoints) a través de los cuales los sistemas pueden intercambiar datos y funcionalidades.
- » REST (Representational State Transfer) es un estilo de arquitectura para diseñar servicios web que utiliza los métodos estándar de HTTP para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en recursos. Las APIs RESTful se basan en este principio para proporcionar una interfaz uniforme y escalable.
- » La librería json en Python permite la serialización y deserialización de datos en formato JSON, comúnmente utilizado para intercambiar datos entre sistemas. La librería requests simplifica el envío de solicitudes HTTP desde Python, permitiendo interactuar fácilmente con APIs externas.
- » La autenticación verifica la identidad del usuario que intenta acceder a un recurso, mientras que la autorización determina si ese usuario tiene permisos para realizar una determinada acción en dicho recurso.
- » Django es un framework web de alto nivel y de código abierto escrito en Python que facilita el desarrollo rápido y limpio de aplicaciones web, incluyendo la creación de APIs.
- » En Django, un proyecto se compone de una o más aplicaciones, y cada aplicación puede tener sus propios modelos, vistas y URLconf. Los endpoints en Django se crean definiendo vistas (funciones o clases) que manejan solicitudes HTTP específicas y devuelven respuestas adecuadas.



PROYECTO FINAL

- Crear una API utilizando Django Rest Framework que gestione dos modelos: Task y User, con endpoints para realizar operaciones CRUD en ambos modelos. La API debe permitir la creación, lectura, actualización y eliminación de tareas (Task), así como la gestión de usuarios (User). Añade documentación, tantos endpoints como creas y si tienes tiempo permisos.



EVALUACIÓN DE CALIDAD

<https://diga.idexaformacion.com/test/1322>



Si tienes alguna duda o consulta
escríbenos a:

tutoria@idexaformacion.com

y recuerda que puedes seguirnos
en:

