

¡Javascript!

guayverd beta hub

Agenda del día



01

APIs

Definición.

Instalar Thunder Client.

API REST

Definición.

Recursos.

Verbos.

Body.

API FETCH

Definición.

GET

POST

MOCKAPI.

02

Ejercitación

Hacer un fetch de una orden de compra.

.



daily

¿Cómo venimos?

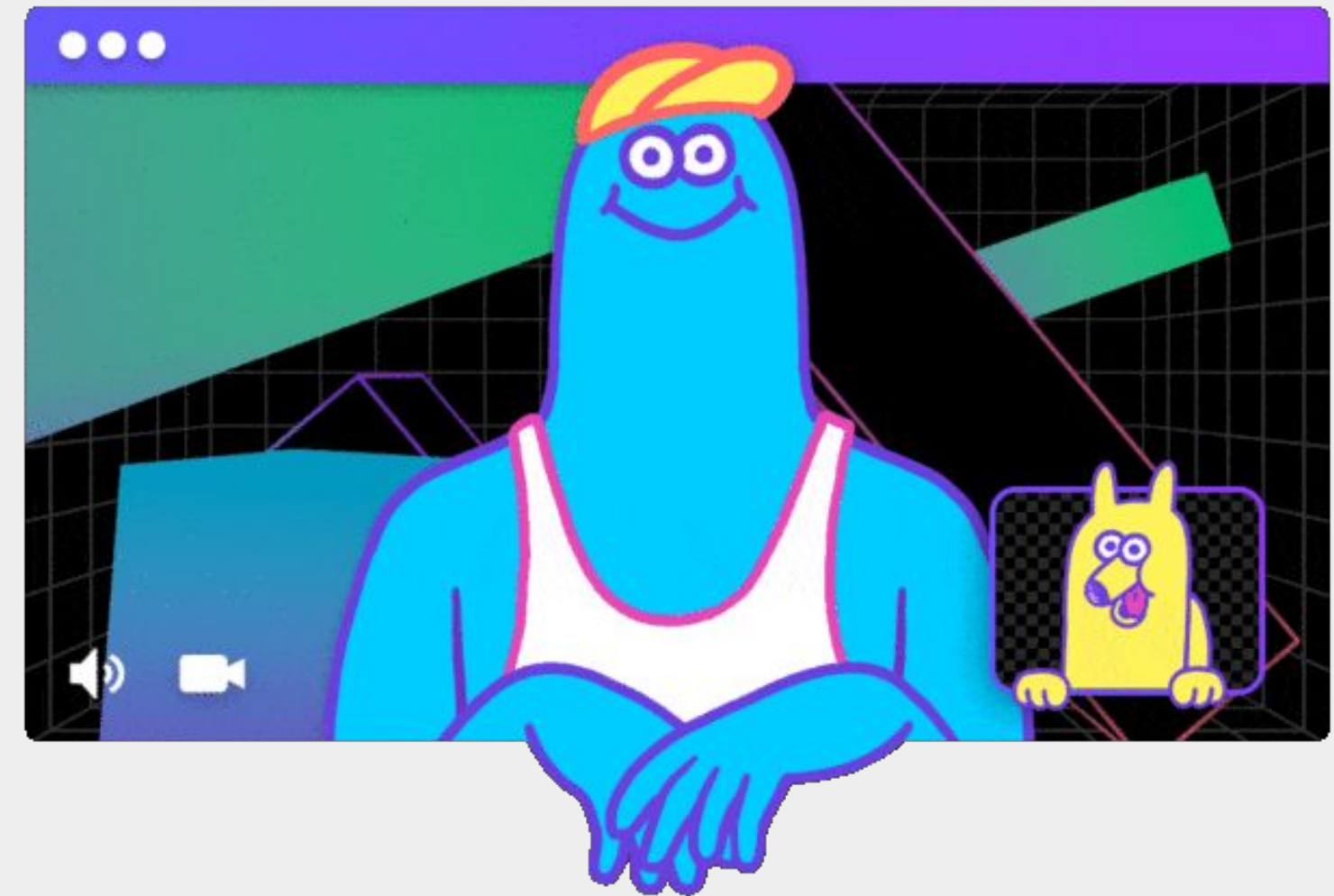
¿Algo nos bloquea?

¿Cómo seguimos?



API

Interfaz de Programación de Aplicaciones




SPRINT 3


API



Una API (Interfaz de Programación de Aplicaciones) es una implementación de reglas y protocolos que permite la comunicación entre programas o consumir servicios.

Ejemplo

 Home About API v2 GraphQL v1beta



The RESTful Pokémon API


Serving over **2.5 billion** API calls each month!

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

[Check out the docs!](#)

Try it now!

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

 Submit

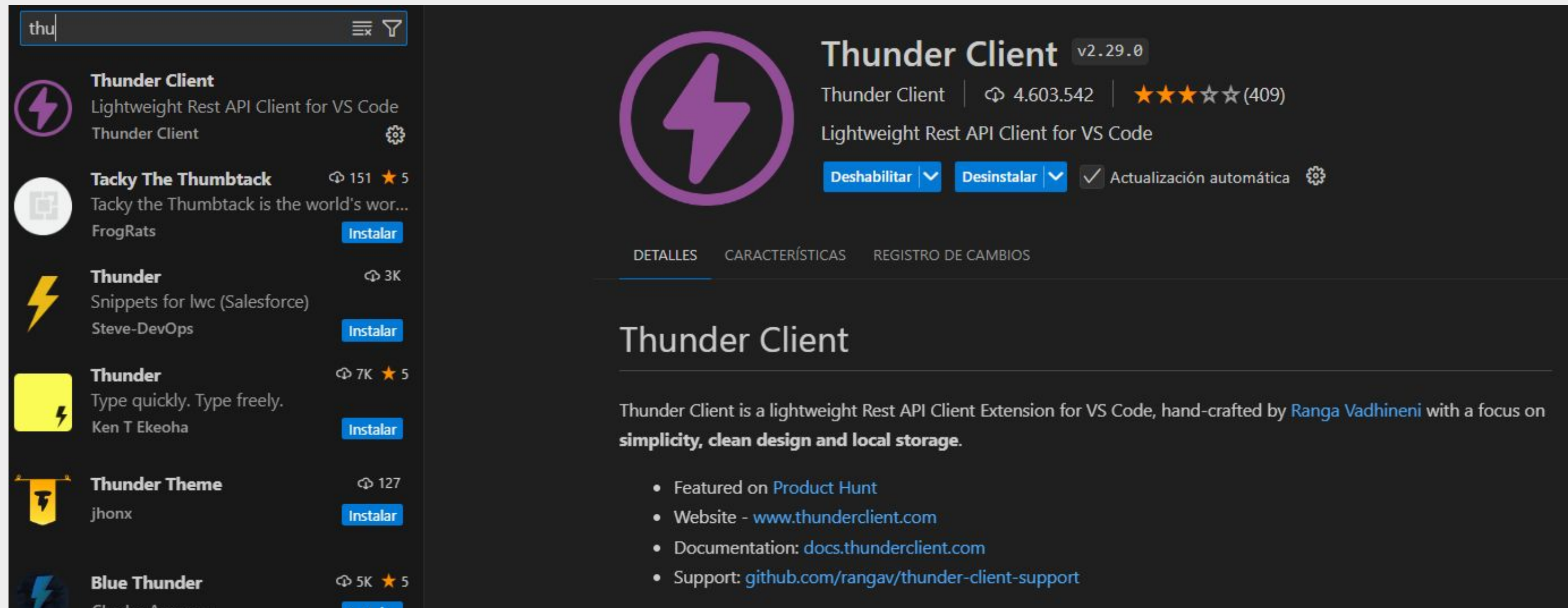
Need a hint? Try [pokemon/ditto](#), [pokemon-species/aegislash](#), [type/3](#), [ability/battle-armor](#), or [pokemon?limit=100000&offset=0](#).

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

Resource for ditto

```
▼ abilities: [] 2 items
  ▼ 0: {} 3 keys
    ▼ ability: {} 2 keys
      name: "limber"
      url: "https://pokeapi.co/api/v2/ability/7/"
```


Instalar Thunder Client en VSCode



The screenshot shows the VS Code extension marketplace interface. On the left, a search bar contains the text 'thu'. Below it, a list of extensions is displayed, including 'Thunder Client', 'Tacky The Thumbtack', 'Thunder', 'Thunder', 'Thunder Theme', and 'Blue Thunder'. The 'Thunder Client' extension is highlighted. The main panel on the right shows the details for the 'Thunder Client' extension, version 2.29.0. It includes a purple lightning bolt icon, the extension name, version, and a star rating of 4.603.542 (409). Below this, there are buttons for 'Deshabilitar', 'Desinstalar', and 'Actualización automática'. The 'DETALLES' tab is selected, showing the extension's description: 'Thunder Client is a lightweight Rest API Client Extension for VS Code, hand-crafted by Ranga Vadhineni with a focus on simplicity, clean design and local storage.' A list of links is provided at the bottom, including 'Featured on Product Hunt', 'Website - www.thunderclient.com', 'Documentation: docs.thunderclient.com', and 'Support: github.com/rangav/thunder-client-support'.

th

Thunder Client v2.29.0
Lightweight Rest API Client for VS Code
Thunder Client

Tacky The Thumbtack 151 ★ 5
Tacky the Thumbtack is the world's wor...
FrogRats **Instalar**

Thunder 3K
Snippets for lwc (Salesforce)
Steve-DevOps **Instalar**

Thunder 7K ★ 5
Type quickly. Type freely.
Ken T Ekeoha **Instalar**

Thunder Theme 127
jhonx **Instalar**

Blue Thunder 5K ★ 5
Charles Assunção **Instalar**

Thunder Client v2.29.0
Thunder Client | 4.603.542 | ★★★★★ (409)
Lightweight Rest API Client for VS Code
Deshabilitar **Desinstalar** ☒ Actualización automática

DETALLES CARACTERÍSTICAS REGISTRO DE CAMBIOS

Thunder Client

Thunder Client is a lightweight Rest API Client Extension for VS Code, hand-crafted by [Ranga Vadhineni](#) with a focus on **simplicity, clean design and local storage.**

- Featured on [Product Hunt](#)
- Website - www.thunderclient.com
- Documentation: docs.thunderclient.com
- Support: github.com/rangav/thunder-client-support

Servidores



Clientes



API REST

Representational State Transfer



The logo features the words "SPECIAL" and "DELIVERY" in a bold, yellow, hand-drawn font with black outlines. The text is arranged in two lines, with "SPECIAL" on top and "DELIVERY" below it. The background is a solid light blue rectangle.

API REST

Es una arquitectura para diseñar servicios web que permiten comunicar datos o información entre clientes y servidores.

Está compuesto por:

- Recursos
- Métodos o verbos
- Parámetros
- Headers
- Body

Recursos

Nos comunicamos con una API a través de una URL y obtenemos los recursos representados como datos utilizables.

The screenshot shows the Thunder Client interface within Visual Studio Code. The top toolbar includes 'New Request', 'Activity', 'Collections', and 'Env'. The left sidebar displays a list of recent requests, with 'GET google.com' selected. The main panel shows the details of this request: a GET method to 'https://google.com' with a status of '200 OK', size of '20.87 KB', and time of '117 ms'. The 'Response' tab is active, displaying the HTML content of the page. A red label 'DATOS RECIBIDOS' is overlaid on the response content.

URL

Status: 200 OK Size: 20.87 KB Time: 117 ms

Response

```
1 <!doctype html><html itemscope itemtype="http://schema.org
2 /WebPage"
3 lang="es-419"><head><meta content="text/html; charset=UTF-8"
4 http-equiv="Content-Type"><meta
5 content="/images/branding/googleg/1x
6 /googleg_standard_color_128dp.png"
itemprop="image"><title>Google</title><script
nonce="Is88CtFRhsSngMl- iee7w">(function(){var _g={kEI
: 'b44XZB...3700314,1070
,538661,2872,2891,89155,203622,6700,124313,2006,8155
,23351,22435,9779,45601,17057,76208,15816,1804,26407
,686,19989,1635,13492,15784,5230281,9477,1021,5990731
,2841781,1162,118,155,5,7,7439765,20556493,43887,3
,1603,3,2124363,23029351,12799,28458,72023,19294,3329
,7552,7612,8182,49429,21674,6750,155,1751,5,728,13504
,7735,9140,744,2,2,3851,328,543,2674,4,1238,1766,20601
,2,4861,2526,5634,687,10633,5631,3069,4737,2236,2784
,764,1346,5402,3273,3112,1859,57,352,1860,2,9,6850
```


Métodos o verbos: GET

Permiten comunicar distintas intenciones al servidor. El más usado es GET, que trae directamente un recurso. Lo usamos comúnmente en la barra de direcciones del navegador.

The screenshot displays the Thunder Client interface within Visual Studio Code. The top toolbar includes a 'New Request' button. The main workspace is divided into three panels. The left panel, titled 'Activity', lists several GET requests, with 'google.com' selected. The middle panel, titled 'Query', shows the details of the selected request, including the method 'GET' and the URL 'https://google.com'. The right panel, titled 'Response', displays the status '200 OK', size '20.87 KB', and time '117 ms'. Below this, the response body is shown as HTML code.

VERBO

GET `https://google.com` Send

Query Parameters

parameter	value
-----------	-------

Status: 200 OK Size: 20.87 KB Time: 117 ms

Response

```
1 <!doctype html><html itemscope itemtype="http://schema.org
2 /WebPage"
3 lang="es-419"><head><meta content="text/html; charset=UTF-8"
4 http-equiv="Content-Type"><meta
5 content="/images/branding/googleg/1x
6 /googleg_standard_color_128dp.png"
itemprop="image"><title>Google</title><script
nonce="Is88CtFRhsSpgML-_iee7w">(function(){var _g={kEI
:'b44XZ8zwHs7L1sQPkcqZ8Q8',kEXPI:'0,3700314,1070
,538661,2872,2891,89155,203622,6700,124313,2006,8155
,23351,22435,9779,45601,17057,76208,15816,1804,26407
,686,19989,1635,13492,15784,5230281,9477,1021,5990731
,2841781,1162,118,155,5,7,7439765,20556493,43887,3
,1603,3,2124363,23029351,12799,28458,72023,19294,3329
,7552,7612,8182,49429,21674,6750,155,1751,5,728,13504
,7735,9140,744,2,2,3851,328,543,2674,4,1238,1766,20601
,2,4861,2526,5634,687,10633,5631,3069,4737,2236,2784
,764,1346,5402,3273,3112,1859,57,352,1860,2,9,6850
```

Métodos o verbos: POST

Permite enviar información al servidor para crear un recurso en el servidor. Enviamos la información en formato JSON y usamos el campo body.

VERBO

POST ☐ <https://64557b55f803f34576439459.mockapi.io/cats> [Send](#)

Status: 201 Created Size: 140 Bytes Time: 831 ms

Query Headers ² Auth **Body ¹** Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content **BODY** Format

```
1 {
2   "name": "Wen",
3   "avatar": "https://ipfs.everipedia.org/ipfs
4     /QmTWLHFwyNvvbQBd9LfcbGiPu1rrWpX6rchEv3YiputpUY",
5   "stock": 94,
6   "category": {},
7   "id": "58"
}
```

Response Headers ¹³ Cookies Results Docs

```
1 {
2   "name": "Wen",
3   "avatar": "https://ipfs.everipedia.org/ipfs
4     /QmTWLHFwyNvvbQBd9LfcbGiPu1rrWpX6rchEv3YiputpUY",
5   "stock": 94,
6   "category": {},
7   "id": "58"
}
```




API FETCH

Navegador





API FETCH

Es una API de navegador que permite comunicarnos con APIs (servidores) desde el navegador.

De este modo integramos nuestras aplicaciones con datos de la **nube**.

Fetch funciona asincrónicamente, es una **promesa**, por lo cual, encadena los métodos: then, catch, finally.

GET

El verbo get del método fetch, es el verbo predeterminado, por eso no se declara. Fetch recibe como primer argumento la URL. En el primer then, transformamos lo que devuelve el servidor en un formato que Javascript puede entender. El segundo then obtiene la data.



```
fetch("https://pokeapi.co/api/v2/pokemon-species")  
  .then(response => response.json())  
  .then(data => console.log(data))
```

GET



En Dev Tools, en la pestaña Red (Network) podemos ver la petición.
El status 200 indica que la comunicación funcionó.

The screenshot shows the Chrome DevTools Network tab. The top bar includes tabs for Elements, Console, Sources, Network (selected), Performance, Application, Memory, Security, and Lighthouse. Below the tabs, there are icons for recording, pausing, and filtering, along with checkboxes for 'Preserve log' and 'Disable cache', and a 'No throttling' dropdown. A filter bar is set to 'Filter'. The main area displays a timeline of network requests. A single request is highlighted, showing a green bar indicating a successful status. Below the timeline, a table lists the request details:

Name	Status	Type	Initiator
pokemon-species	200	fetch	<u>cart.js:89</u>

GET

Si damos clic en la petición podremos ver información en varias pestañas.

Name	✕	Headers	Preview	Response	Initiator	Timing
 pokemon-species	▼ General					
	Request URL:		https://pokeapi.co/api/v2/pokemon-species			
	Request Method:		GET			
	Status Code:		 200 OK (from disk cache)			
	Remote Address:		172.67.195.193:443			
	Referrer Policy:		strict-origin-when-cross-origin			
	▼ Response Headers					
	Accept-Ranges:		bytes			
	Access-Control-Allow-Origin:		*			
	Alt-Svc:		h3=":443"; ma=86400			
	Cache-Control:		public, max-age=86400, s-maxage=86400			

GET

Si damos clic en la petición podremos ver información en varias pestañas.

The screenshot shows a web browser's developer tools interface. The 'Name' column on the left lists 'pokemon-species'. The 'Preview' tab is selected, displaying the JSON response of a GET request. The response includes a 'count' of 1025, a 'next' URL, a 'previous' URL (null), and a 'results' array containing 11 Pokemon species with their names and API URLs.

Name	Headers	Preview	Response	Initiator	Timing
pokemon-species		<pre>▼ {count: 1025, next: "https://pokeapi.co/api/v2/pokemon-species?offset=20&limit=20", previous: null,...} count: 1025 next: "https://pokeapi.co/api/v2/pokemon-species?offset=20&limit=20" previous: null ▼ results: [{name: "bulbasaur", url: "https://pokeapi.co/api/v2/pokemon-species/1/"},...] ▶ 0: {name: "bulbasaur", url: "https://pokeapi.co/api/v2/pokemon-species/1/"} ▶ 1: {name: "ivysaur", url: "https://pokeapi.co/api/v2/pokemon-species/2/"} ▶ 2: {name: "venusaur", url: "https://pokeapi.co/api/v2/pokemon-species/3/"} ▶ 3: {name: "charmander", url: "https://pokeapi.co/api/v2/pokemon-species/4/"} ▶ 4: {name: "charmeleon", url: "https://pokeapi.co/api/v2/pokemon-species/5/"} ▶ 5: {name: "charizard", url: "https://pokeapi.co/api/v2/pokemon-species/6/"} ▶ 6: {name: "squirtle", url: "https://pokeapi.co/api/v2/pokemon-species/7/"} ▶ 7: {name: "wartortle", url: "https://pokeapi.co/api/v2/pokemon-species/8/"} ▶ 8: {name: "blastoise", url: "https://pokeapi.co/api/v2/pokemon-species/9/"} ▶ 9: {name: "caterpie", url: "https://pokeapi.co/api/v2/pokemon-species/10/"} ▶ 10: {name: "metapod", url: "https://pokeapi.co/api/v2/pokemon-species/11/"}</pre>			

1 / 11 requests | 0 B / 4

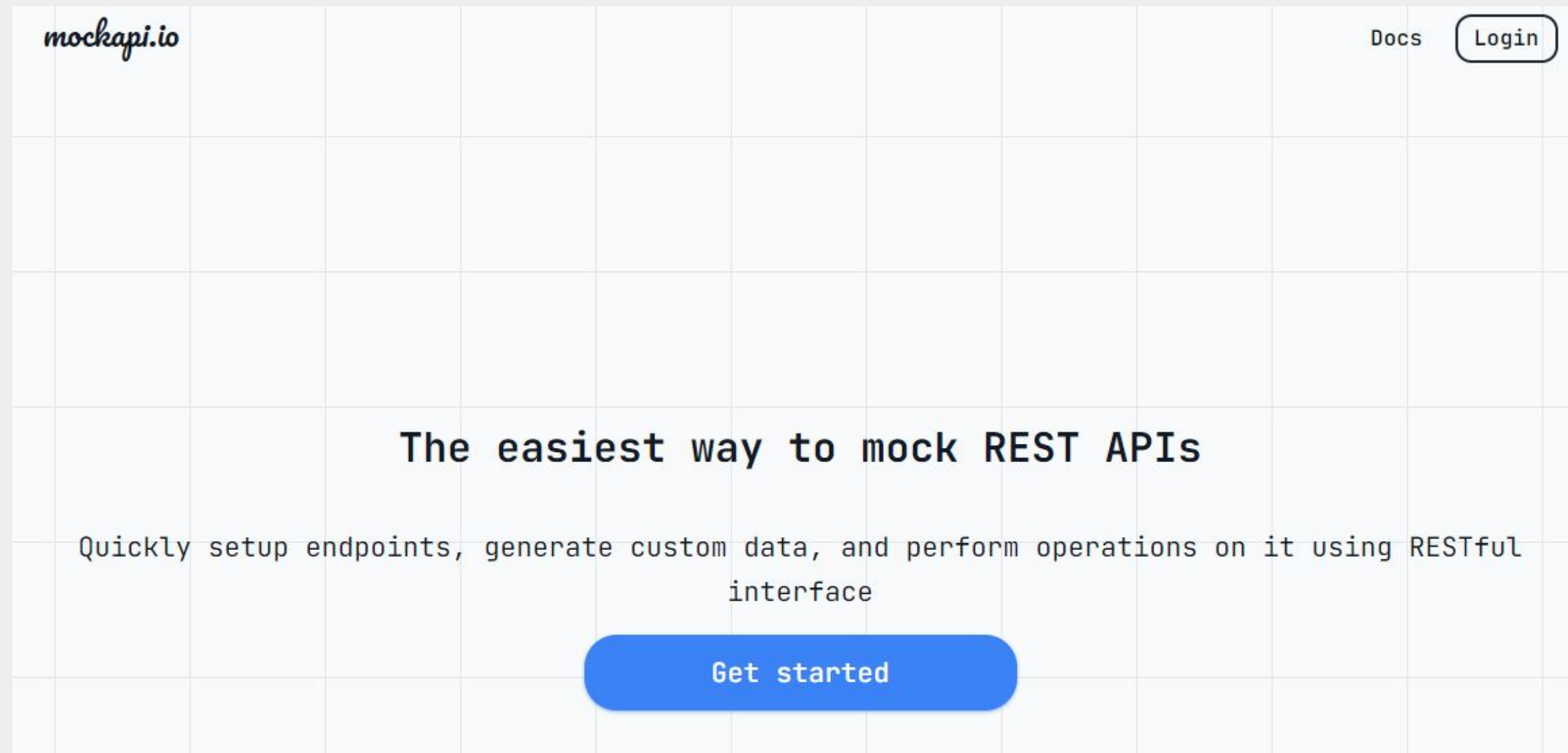
POST

En el segundo argumento del método fetch se incluye un objeto de opciones. El verbo POST lo especificamos con la clave method y el body con la data que queremos enviar stringificada.

```
const recurso = {  
  name: "Wen",  
  avatar: "https://ipfs.everipedia.org/ipfs/QmTWLHFwyNvvbQBd9LfcbGiPu1rrWpX6rchEv3YiputpUY",  
}  
  
fetch("https://64557b55f803f34576439459.mockapi.io/cats", {  
  method: "POST",  
  body: JSON.stringify(recurso),  
})  
  .then(response => response.json())  
  .then(data => console.log(data))
```

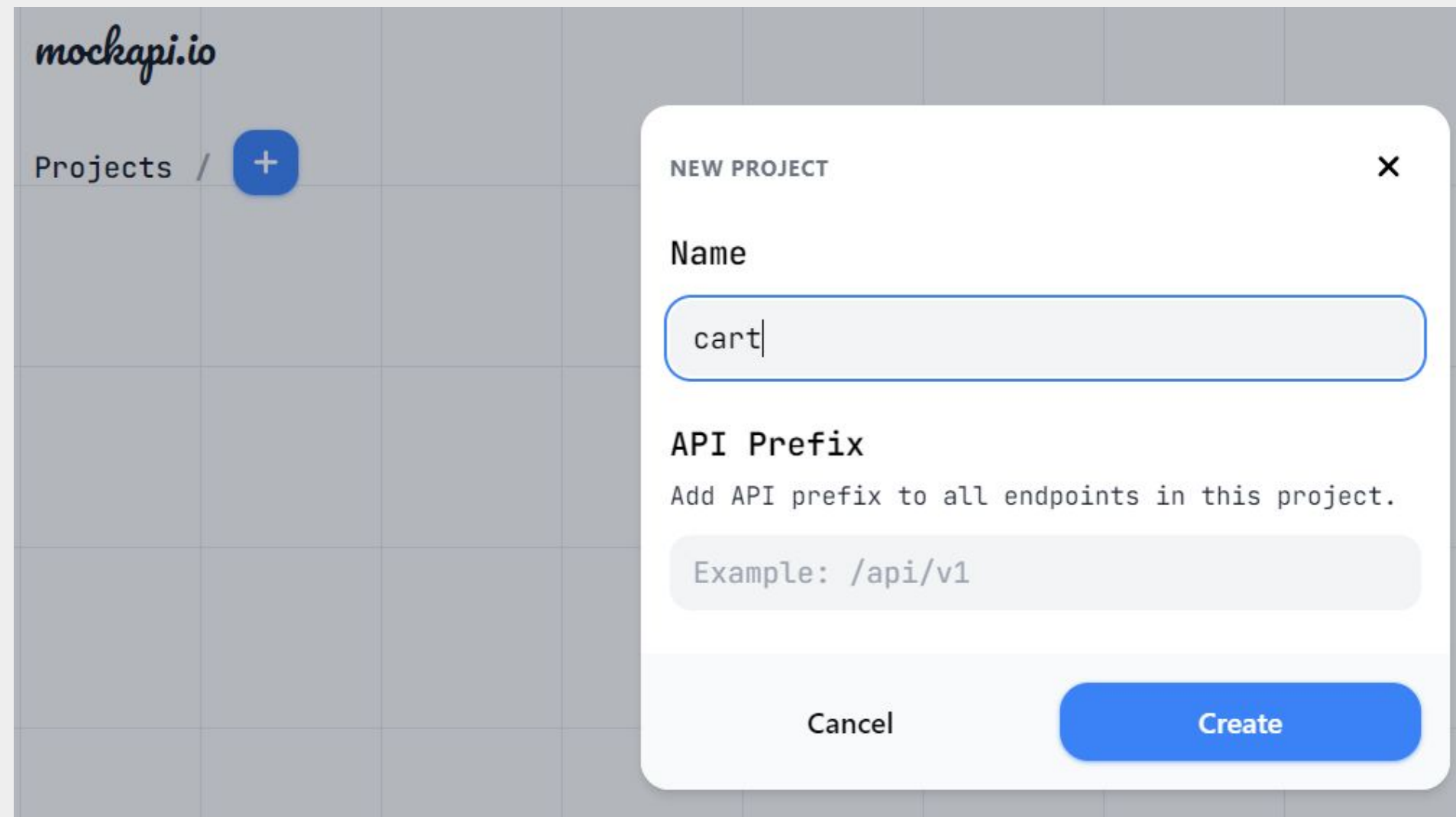
MOCKAPI

Creamos una API de prueba. Ver: <https://mockapi.io/projects>



MOCKAPI

Damos un clic al botón **+** y ponemos en name: cart.



The screenshot shows the MockAPI.io web interface. On the left, there is a sidebar with the logo 'mockapi.io' and a 'Projects /' section containing a blue circular button with a white plus sign. A modal window titled 'NEW PROJECT' is open on the right. It has a close button (X) in the top right corner. The modal contains two input fields: 'Name' with the text 'cart' and 'API Prefix' with a placeholder 'Example: /api/v1'. Below the inputs are two buttons: 'Cancel' and 'Create'.

mockapi.io

Projects / +

NEW PROJECT X

Name

cart

API Prefix

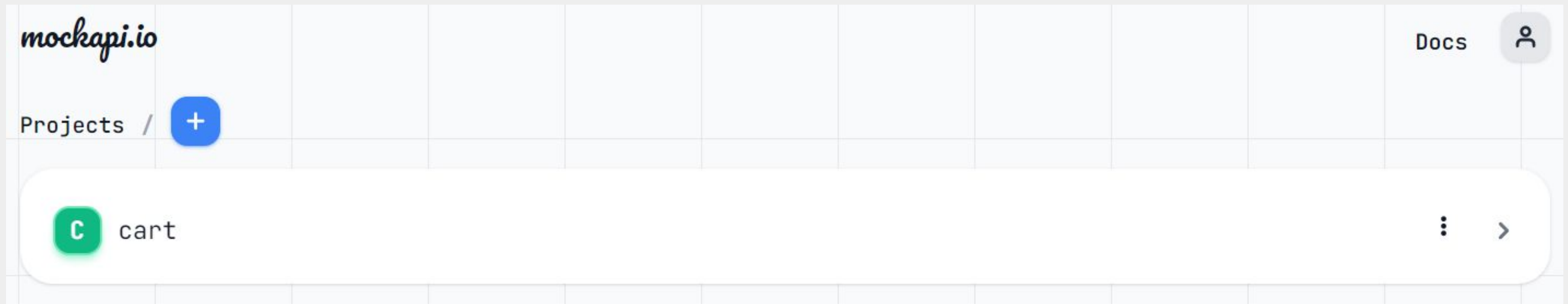
Add API prefix to all endpoints in this project.

Example: /api/v1

Cancel Create

MOCKAPI

Damos un clic a la flecha de la derecha.



MOCKAPI

Agregamos el campo items como Object y el campo user como string.

Schema
Define Resource schema, it will be used to generate mock data.

id	Object ID	
createdAt	Faker.js	date.recent
items	Object	
user	Faker.js	random.word

+ Add field

Faker.js ☒

String

Object template
To define more complex schema, you can use an object template. You can reference

Requires subscription [See plans](#)

MOCKAPI

Damos un clic en New Resource.

En Resource name ponemos: orders y removemos el campo name y avatar.

The screenshot shows the MockAPI interface with a 'NEW RESOURCE' modal open. The modal has a title bar with a close button. The 'Resource name' section has a text input with 'orders' and a description: 'Enter meaningful resource name, it will be used to generate API endpoints.' The 'Schema' section has a description: 'Define Resource schema, it will be used to generate mock data.' Below this is a table of fields:

id	Object ID	
createdAt	Faker.js	date.recent
name	Faker.js	name.fullName
avatar	Faker.js	image.avatar

There are 'X' icons next to the 'name' and 'avatar' rows, and a 'Remove field' button at the bottom right of the schema section.

MOCKAPI

Hacemos scroll y vemos los métodos. Finalmente damos clic en Update.

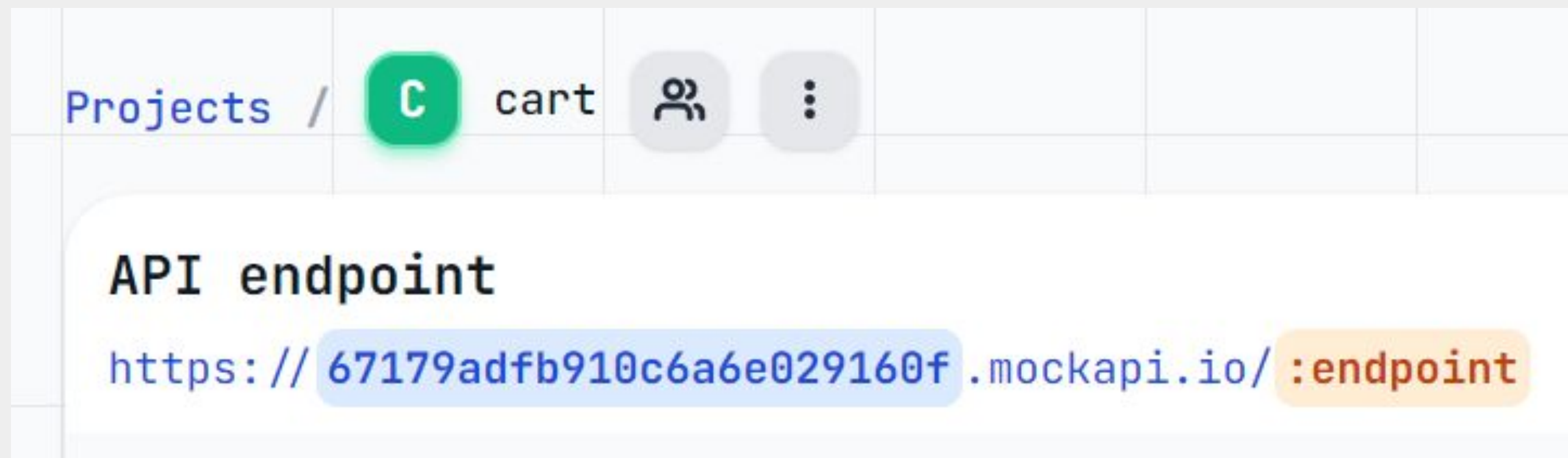
The screenshot shows the MockAPI interface with a list of four API methods. Each method has a toggle switch labeled 'On', the HTTP method and endpoint, and a text input field containing '\$mockData'. The methods are:

- GET /orders
- GET /orders/:id
- POST /orders
- PUT /orders/:id

At the bottom of the interface, there are two buttons: 'Close' and 'Update'.

MOCKAPI

Endpoints.



MOCKAPI GET

The screenshot displays a REST client interface with a GET request to `https://67179adfb910c6a6e029160f.mockapi.io/orders`. The response is a JSON array containing one object. The left pane shows the JSON content, and the right pane shows the formatted response with line numbers.

Request: GET `https://67179adfb910c6a6e029160f.mockapi.io/orders` Send

Response: Status: 200 OK Size: 528 Bytes Time: 678 ms

JSON Content:

```
1
```

Response (Formatted):

```
1  [
2    {
3      "createdAt": "2024-10-22T00:41:58.027Z",
4      "items": [
5        {
6          "id": 1,
7          "title": "Heroes of Telemark, The ",
8          "detail": "Nam ultrices, libero non mattis pulvinar,
          nulla pede ullamcorper augue, a suscipit nulla elit
          ac nulla. Sed vel enim sit amet nunc viverra dapibus.
          Nulla suscipit ligula in lacus.\n\nCurabitur at ipsum
          ac tellus semper interdum. Mauris ullamcorper purus
          sit amet nulla. Quisque arcu libero, rutrum ac,
          lobortis vel, dapibus at, diam.",
9          "price": 8,
10         "stock": 56,
11         "category": "Moderno",
12         "quantity": 2
13       }
14     ],
15     "user": "pato@duck.com",
16     "id": "1"
17   }
18 ]
```


MOCKAPI POST

The screenshot displays a REST client interface with a POST request to `https://67179adfb910c6a6e029160f.mockapi.io/orders`. The request body is a JSON object with the following structure:

```
{
  "user": "pato@duck.com",
  "items": [
    {
      "id": 1,
      "title": "Heroes of Telemark, The ",
      "detail": "Nam ultrices, libero non mattis pulvinar, nulla pede ullamcorper augue, a suscipit nulla elit ac nulla. Sed vel enim sit amet nunc viverra dapibus. Nulla suscipit ligula in lacus.\n\nCurabitur at ipsum ac tellus semper interdum. Mauris ullamcorper purus sit amet nulla. Quisque arcu libero, rutrum ac, lobortis vel, dapibus at, diam.",
      "price": 8,
      "stock": 56,
      "category": "Moderno",
      "quantity": 2
    }
  ]
}
```

The response status is **201 Created**, with a size of **526 Bytes** and a time of **707 ms**. The response body is a JSON object with the following structure:

```
{
  "createdAt": "2024-10-22T00:41:58.027Z",
  "items": [
    {
      "id": 1,
      "title": "Heroes of Telemark, The ",
      "detail": "Nam ultrices, libero non mattis pulvinar, nulla pede ullamcorper augue, a suscipit nulla elit ac nulla. Sed vel enim sit amet nunc viverra dapibus. Nulla suscipit ligula in lacus.\n\nCurabitur at ipsum ac tellus semper interdum. Mauris ullamcorper purus sit amet nulla. Quisque arcu libero, rutrum ac, lobortis vel, dapibus at, diam.",
      "price": 8,
      "stock": 56,
      "category": "Moderno",
      "quantity": 2
    }
  ],
  "user": "pato@duck.com",
  "id": "1"
}
```







¡Manos a la obra!

SPRINT 3

Fetch

Entregable



- Crear una mockapi por grupo.
- Al dar clic en el botón Checkout de la página cart enviar un fetch para crear un recurso.
- Si **funciona**, mostrar un SweetAlert que incluya el email del usuario y el número de orden creada. Vaciar el carrito.
- Si **falla** qué catch muestre un mensaje al usuario con SweetAlert.



retro

¿Cómo nos fué?

¿Qué cosas no quedaron claras y
necesitamos repasar la próxima?

